Particle Filtering and Optimal Control for Vehicles and Robots

Berntorp, Karl

2014

Document Version:
Publisher's PDF, also known as Version of record

Particle Filtering and Optimal Control for Vehicles and Robots

KARL BERNTORP
DEPARTMENT OF AUTOMATIC CONTROL | LUND UNIVERSITY

Particle Filtering and Optimal Control for Vehicles and Robots

Karl Berntorp

LUND
UNIVERSITY

Department of Automatic Control

Cover photo from Tärnö, Hällaryds skärgård, Karlshamn

PhD Thesis
ISRN LUTFD2/TFRT--1101--SE
ISBN 978-91-7473-947-3 (print)
ISBN 978-91-7473-948-0 (web)
ISSN 0280–5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Media-Tryck.
Lund 2014

To Karl, Axel, Ellard

There is only one success—to be able to spend your life in your own way
— *CHRISTOPHER MORLEY, Where the Blue Begins (1922)*

Abstract

This thesis covers areas within estimation and optimal control of vehicles, in particular four-wheeled vehicles. One topic is how to handle delayed and out-of-sequence measurements (OOSMs) in tracking systems. The motivation for this is that with technological development and exploitation of more sensors in tracking systems, OOSMs gain more significance in various applications. The thesis derives a Bayesian formulation of the OOSM problem for nonlinear state-space models, when a linear, Gaussian substructure is present. This formulation is utilized when developing two particle-filter algorithms for the OOSM problem. The algorithms improve estimation accuracy and tracking robustness, compared with methods that do not utilize the linear substructure.

A second topic is sensor fusion for improved autonomy in vehicles. A novel approach to model-based joint wheel-slip and motion estimation of four-wheeled vehicles is developed. Unlike other approaches, the method explicitly models the nonlinear slip dynamics in the state and measurement equations. Excellent and consistent accuracy for all relevant states are reported, both during steady-state driving and aggressive maneuvering. The method applies to general classes of four-wheeled vehicles and it only assumes kinematic relationships.

Optimization-based control methods have found their way into automotive applications. Optimal control for vehicles typically results in control inputs that give aggressive maneuvering. Proper models are therefore crucial. An investigation on what impact different vehicle models and road surfaces have on the optimal trajectories in safety-critical maneuvers is presented. One conclusion is that the control-input behavior is highly sensitive to the choice of chassis and tire models. Another conclusion is that the optimal driving techniques are different depending on tire-road characteristics. The conclusions motivate the design of a novel, two-level hierarchical approach to optimal trajectory generation for wheeled vehicles. The first novelty is the use of a nonlinear vehicle model with tire modeling in the optimization problem at the high level. This provides for

better coupling with the low-level controller, which uses a nonlinear model predictive controller (MPC) for allocating the torques and steer angles to the wheels. This is combined with a linear MPC, which is used when the nonlinear MPC fails to converge in time.

The thesis also describes a hierarchical design flow for performing online, minimum-time trajectory generation for four-wheeled vehicles with independent steer and drive actuation, combined with real-time obstacle avoidance. The approach is based on convex optimization. It therefore allows fast computations, both for trajectory generation and online feedback-based obstacle avoidance. The proposed method is fully implemented on a pseudo-omnidirectional mobile platform and evaluated in experiments in a path-tracking scenario.

Acknowledgments

A wise man recently told me: the things you reject are usually rejected because you make an informed decision; the things you accept are usually accepted because you do not reject them. I can only agree. Back in 2006, Karl-Erik Årzén asked me if I wanted to teach the Department's basic course. Before this, controls was not part of my five-year plan. One thing led to another and with perspective, choosing to pursue a PhD at the Department is one of the best decisions I have made, for numerous reasons. I thank Karl-Erik for sending me that e-mail eight years ago. In the role as my main supervisor, I thank him for allowing me to choose my own problems to work with; taking the time to read, and suggest many improvements for, this Thesis and all papers; and for being a great guy.

I thank my second supervisor, Anders Robertsson, for always helping out. It does not matter if it is about getting experimental equipment to work in the lab, helping me cover for an overslept PhD student, or proofreading a manuscript at Copenhagen Airport in the middle of the night. He always finds time. Of course, I also appreciate the many comments on this Thesis.

Karl Johan Åström's door has always been open for me, and for that I am very grateful. I have had numerous discussions with him over these five years, including research, family, and future, and it is always a pleasure to hear his thoughts on things. Thank you, it means a lot to me.

Everyone in the administrative and technical staff, including former staff members, have helped me with many things the time I have been here. I am very thankful for that. In fact, *everyone* at the Department of Automatic Control have contributed in various aspects, not the least in providing a very nice working atmosphere. This includes the often amusing, but seemingly superfluous, lunch discussions.

Leif Andersson has been pivotal during the finalization of this Thesis. He has, for example, provided suggestions for how to correct numerous stylistic flaws.

My gratitude goes to Björn Olofsson, with whom I have collaborated over the last two years within ELLIIT. His thoroughness and interest in getting things to work in practice is much appreciated.

I acknowledge Lars Nielsen and Kristoffer Lundahl at the Vehicular Systems Division, Linköping University, for the cooperation within ELLIIT, and a recognition goes to Kristoffer Lundahl for providing me with datasets for verification of the algorithm in Chapter 9.

Anders Mannesson, Björn Olofsson, and Fredrik Magnusson have read parts of this Thesis. Thank you for the suggestions for improvements—for example, that I should include Mannesson plots.

One of the many course I have taken during the years is the Control System Synthesis course in 2010, taught by Bo Bernhardsson and Karl Johan Åström. Karl Johan's inspiring manners and Bo's ability to provide intuition were a great combination, and the flexible-servo tuning contest was really fun.

I thank my original family for all the obvious reasons, but also for all family stories, which ultimately made me choose Engineering Physics.

Jennifer, thank you for these last 11 years, for showing me what is important in life, for introducing me to Bigos, and for being such a wonderful and unselfish person. A day with you is never dull.

Axel, you have been an immense support for both me and Jennifer these last two years. You mean the world to me, every day. To (almost) quote Chesney Hawkes: You are the one and only.

Financial Support

I acknowledge the following for financial support: The Swedish Foundation for Strategic Research through the project ENGROSS, the Swedish Research Council through the LCCC Linnaeus Center, and the Strategic Research Area ELLIIT.

Contents

Nomenclature

The following pages give the notation, abbreviations, and the names of the algorithms that are used in the thesis. The aim has been to identify each entity with a unique variable, but sometimes the same notation means different things.

Symbol Descriptions

The table below summarizes the most frequently used notation in the thesis. Some of the symbols sometimes have additional indices referring to time or different wheels. This is not included here.

Notation	**Description**
$\mathbb{R}$	Real numbers
$\mathbb{R}^m$	Real-valued matrices of dimension $m \times 1$
$\mathbb{R}^{m \times n}$	Real-valued matrices of dimension $m \times n$
$\mathbf{0}_{m \times n}$	The zero matrix of dimension $m \times n$
$\mathbf{0}_m$	The zero vector of dimension $m \times 1$
$\boldsymbol{I}_{n \times n}$	The identity matrix of dimension $n \times n$
$\boldsymbol{I}_n$	The unity vector of dimension $n \times 1$
$\boldsymbol{x}$	State vector
x	Scalar state
$\dot{\boldsymbol{x}}$	Time derivative of $\boldsymbol{x}$
$\boldsymbol{x}^{\mathrm{T}}$	Transpose of $\boldsymbol{x}$
$\|\boldsymbol{x}\|$	Euclidean norm of $\boldsymbol{x}$
$\|\boldsymbol{x}\|^2_{\boldsymbol{Q}}$	$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}$
T_s	Sampling period
k	Time index
t_k	Time corresponding to k
$\mathrm{diag}(\boldsymbol{x})$	Diagonal matrix ($\boldsymbol{x}$ appropriately interpreted)

sign	Signum function

Notation	Description
Estimation	
$\boldsymbol{z}$	Linear part of state vector
$\boldsymbol{\eta}$	Nonlinear part of state vector
$\boldsymbol{y}$	Measurement vector
$\boldsymbol{w}$	Process noise vector
$\boldsymbol{Q}$	Process noise covariance matrix
$\boldsymbol{e}$	Measurement noise vector
$\boldsymbol{R}$	Measurement noise covariance matrix
$\boldsymbol{u}$	Input vector
$\boldsymbol{f}$, $\boldsymbol{g}$, $\boldsymbol{h}$	System vectors
$\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{C}$	System matrices
$\mathcal{E}(\boldsymbol{x})$	Expected value of $\boldsymbol{x}$
$\mathrm{P}(\boldsymbol{x})$	Probability of $\boldsymbol{x}$
p	Probability density function (PDF)
$p(\cdot\|\cdot)$	Conditional PDF
$\mathcal{N}$	Gaussian distribution
$\mathcal{N}(x\|\mu, \Upsilon)$	Gaussian conditional PDF
$\hat{\boldsymbol{x}}_{k\|T}$	Estimated state vector at time t_k given $\boldsymbol{y}_{0:T}$
$\boldsymbol{P}_{k\|T}$	Covariance matrix at time t_k given $\boldsymbol{y}_{0:T}$
$\sim$	Sampled from or distributed according to
$\propto$	Proportional to
w^i	Particle weight i
N	Number of particles
τ	OOSM time index
l	OOSM delay
t_k^a	Arrival time
$\mathcal{Z}_k$	Set of OOSMs generated in $[0, k]$
$\mathcal{Y}_k$	Set of ISMs generated in $[0, k]$

Notation	Description
Vehicle Variables	
CoG	Mass center or geometric center
X	Longitudinal coordinate axis
Y	Lateral coordinate axis
Z	Vertical coordinate axis
x	Longitudinal wheel-coordinate axis
y	Lateral wheel-coordinate axis

z	Vertical wheel-coordinate axis
I	Inertial frame
$\mathcal{W}$	Robot-fixed inertial frame
$\mathcal{V}$	Vehicle-fixed frame
$\mathcal{C}$	Chassis-fixed frame
$\mathcal{B}$	Body-fixed frame
$X^{\mathcal{V}}$	Longitudinal coordinate axis of frame $\mathcal{V}$
$\boldsymbol{p}$	Position vector
$\boldsymbol{v}$	Velocity vector
$\boldsymbol{a}$	Acceleration vector
$\boldsymbol{b}$	Bias vector
$\boldsymbol{v}^{\mathcal{V}}$	Velocity vector expressed in frame $\mathcal{V}$
v^X	Longitudinal component of velocity vector
β	Vehicle sideslip angle
$\boldsymbol{\xi}$	Spatial angular-velocity vector
$\mathbf{F}$	Total force vector acting on vehicle
$\mathbf{M}$	Total torque vector acting on vehicle
$\mathscr{F}$	Generalized torques
ϕ	Roll angle (rotation about chassis X-axis)
θ	Pitch angle (rotation about vehicle Y-axis)
ψ	Yaw angle (rotation about inertial Z-axis)
$\boldsymbol{\psi}$	Rotation angle vector
$\boldsymbol{R}^I_{\mathcal{V}}$	Rotation matrix from $\mathcal{V}$ to I
e	Deviation from mid-lane segment

Notation	**Description**
Wheel Variables	
$\boldsymbol{v}^w$	Wheel velocity
α	Wheel-slip angle
λ	Longitudinal wheel slip
ω	Wheel angular velocity
ϑ	Wheel drive angle
δ	Wheel steer angle
$\boldsymbol{\tau}$	Wheel-torque vector

Notation	**Description**
Vehicle Parameters	
m	Vehicle mass
I_{XX}	Vehicle moment of inertia about $X^{\mathcal{B}}$-axis
I_{YY}	Vehicle moment of inertia about $Y^{\mathcal{B}}$-axis

I_{ZZ}	Vehicle moment of inertia about $Z^{\mathcal{B}}$-axis
l_f	Distance from front wheels to mass center
l_r	Distance from rear wheels to mass center
w	Half track width
w_f	Distance from front wheels to $X^{\mathcal{V}}$-axis
w_r	Distance from rear wheels to $X^{\mathcal{V}}$-axis
w_1	Distance from front left wheel to $X^{\mathcal{V}}$-axis
w_2	Distance from front right wheel to $X^{\mathcal{V}}$-axis
w_3	Distance from rear left wheel to $X^{\mathcal{V}}$-axis
w_4	Distance from rear right wheel to $X^{\mathcal{V}}$-axis

Notation	**Description**
Wheel Parameters	
R_w	Wheel radius
I_w	Wheel moment of inertia
R_i	Wheel radius for wheel i

Notation	**Description**
Optimization	
$\boldsymbol{W_x}$	Weight matrix for states
$\boldsymbol{W_u}$	Weight matrix for control inputs
H_p	Prediction horizon
H_c	Control horizon
t_0	Initial time
t_f	Final time
$\boldsymbol{w}$	Algebraic variables
$\boldsymbol{x}$	Differential (state) variables
$T_{s,h}$	High-level sampling period in Chapter 11
$T_{s,l}$	Sampling period of MPC in Chapter 11
H_l	Prediction horizon of MPC in Chapter 11

Abbreviation	**Description**
ABS	Anti-lock Braking System
ASR	Anti-Slip Regulation
CAN	Controller Area Network
CasADi	Computer algebra system with Automatic Differentiation
CPU	Central Processing Unit

DAE	Differential-Algebraic Equation
EKF	Extended Kalman filter
ESC	Electronic Stability Control system
ESP	Electronic Stability Program
FE	Friction-Ellipse based tire model
GPS	Global Positioning System
IPOPT	Interior Point OPTimizer
LMPC	Linear Model Predictive Control(ler)
MGSS	Mixed-Gaussian State Space
MPC	Model Predictive Control(ler)
MRT	Most Recent Time
NHTSA	National Highway Traffic Safety Administration
NMPC	Nonlinear Model Predictive Control(ler)
ODE	Ordinary Differential Equation
OOSM	Out-of-Sequence Measurement
PID	Proportional Integral Derivative
RBPF	Rao-Blackwellized Particle Filter
RBPS	Rao-Blackwellized Particle Smoother
ROS	Robot Operating System
RTS	Rauch-Tung-Striebel
WF	Weighting-Functions based tire model

Algorithms

The algorithms that are used in the thesis are summarized in the table below.

Abbreviation	Description
A-PF	OOSM particle filter based on Bayesian solution
OOSM-GARP	Gaussian approximation rerun particle filter
PFDISC	Particle filter that discards OOSMs
PFIDEAL	Offline, idealized particle filter
PF-CISI	Particle filter with Complete In-Sequence Information
PF-CISIMI	PF-CISI with selective processing
RBOOSMBS	Rao-Blackwellized OOSM with Backward Simulation
RBPFDISC	RBPF that discards OOSMs
SERBPF	Storage-Efficient RBPF
SEPF	Storage-Efficient Particle Filter
SEPF-GARP	*SEPF* with selective processing

1

Introduction

This thesis addresses topics within nonlinear estimation and optimal control of ground vehicles. The ever-continuing advancements in computing power, sensors, and control theory, have led to an increased interest in autonomous vehicles, illustrated by, for example, the Google car and the DARPA grand challenge [Thrun et al., 2006b]. The inclusion of more sensors gives potential for better estimation and understanding of the vehicle motion, which makes it possible to formulate control principles for improved autonomy. On the other hand, more sensor measurements, arriving with different delays and accuracy, increase demands on the system that is responsible for combining the sensor signals.

1.1 Background

Autonomous, or at least semiautonomous, vehicles have been the subject of much research during the past decades [Thrun et al., 2006a; Shiller and Gwo, 1991]. Examples from the automotive industry are predictive steering control [Falcone et al., 2007] and platooning [Alam et al., 2010]. Moreover, in a production scenario with small batch sizes, the combination of an autonomous mobile robot platform with a conventional robot manipulator mounted on the base offers flexible and cost-efficient assembly solutions. Hence, mobile robot platforms have the potential of reducing the costs for production and improving productivity. Figure 1.1 provides an example of a possible combination.

Ideally, autonomous vehicles should be able to work in unstructured environments, where moving obstacles, such as humans, are present. Autonomous vehicles are increasingly being employed in outdoor environments, where examples are planetary exploration, site inspection, mining, and search and rescue operations [Iagnemma and Dubowsky, 2004]. Outdoor applications include traveling in unknown environments, typically with highly varying wheel-surface interaction and uneven terrain.

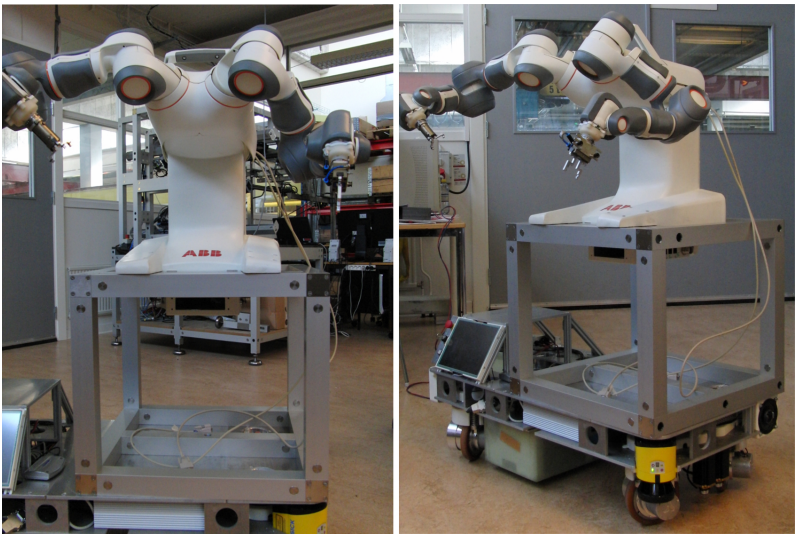

Figure 1.1 Mounting an industrial manipulator on a mobile robot platform offers flexible and cost-efficient assembly solutions. Because the industrial manipulator's work space is now dynamic, the setup also demands more knowledge about the environment.

In both production scenarios and outdoor applications, the vehicles have to navigate autonomously, with automated real-time decision making. It is therefore imperative that the navigation strategies are fast and reliable, and that the robot's environmental perception is of high quality.

The introduction of anti-lock braking systems (ABS) in 1978 [Burton et al., 2004] marked the introduction of control systems for active safety in production cars. In 1995, the Electronic Stability Program (ESP) was introduced as a means to avoid excessive understeering and oversteering [Liebemann et al., 2005; Reif and Dietsche, 2011]. The characteristics of the ESP is to prevent the vehicle from skidding. This is essentially done by controlling yaw rate and body slip toward reference values, which are computed from driver steering input and estimated vehicle velocity. Since the introduction of the ESP, several similar systems have been introduced. These safety systems are commonly referred to as electronic stability control systems (ESCs). There are many other active safety systems available—for example, rollover-prevention controllers and collision-avoidance systems, which both typically use the brakes as main actuators. The brake actuation can also be combined with active suspension and/or active steering. Most active safety systems in production have in common that they control specific variables that have tight interaction

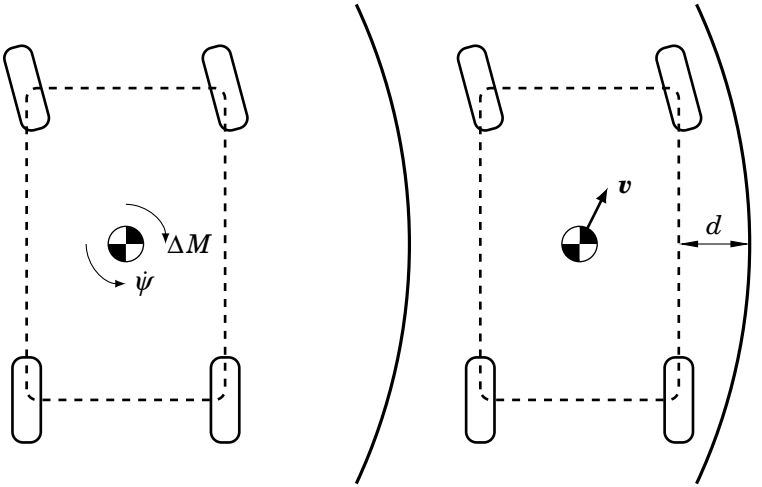

Figure 1.2 An illustration of the difference between traditional yaw-rate controllers (left) and safety systems that take advantage of improved technologies (right). Principally, ESCs maintain vehicle stability by controlling the yaw rate $\dot{\psi}$ through wheel braking. This wheel braking creates a moment ΔM_z, which stabilizes the vehicle. With improved sensing and environmental perception, it is possible to develop high-level controllers, such as lane-keeping controllers, where, for example, combinations of the distance d to the road vicinity and velocity $\boldsymbol{v}$ are controlled.

with the desired vehicle behavior, being the yaw rate for ESCs, roll angle for rollover-prevention systems, and target distance in case of collision-avoidance systems.

Because of the inclusion and combination of sensors such as cameras, radar systems, satellite positioning systems, and inertial sensors, there exist new possibilities for improved vehicle perception. In combination with the availability of braking and steering individual wheels [Jonasson et al., 2011], a spectrum of more advanced safety systems that are not limited to controlling specific signals alone are possible. More high-level control architectures that focus on controlling the overall vehicle behavior—instead of a few characteristics in isolation from each other—are viable. An example of more advanced safety systems is situation-aware lane-following systems based on optimal control, where situation aware essentially means that the vehicle has sufficient knowledge about the surroundings [Lundquist, 2011]. Figure 1.2 contains an illustration of a safety system that takes advantage of improved technologies.

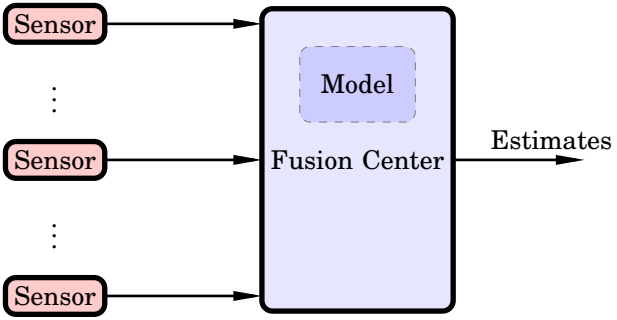

Figure 1.3 An illustration of a sensor-fusion system. Several sensors measure various quantities. The measurements are combined to form state estimates by using a model of the considered system. The estimates can then be used in various applications—for example, in control, surveillance, or fault detection. It is also possible to use the estimates in other estimation algorithms.

1.2 Motivation

Vehicle Estimation

Sensor fusion is the process of extracting information from different sensors, to acquire more knowledge about a system than what would be possible if using the sensors individually, see Figure 1.3. In automotive systems, each sensor has traditionally belonged to a specific active safety system, where one or two sensors have been used in a particular application. With the emergence of sensor fusion in automotive systems, it is possible to improve overall understanding of the vehicle motion, in addition to improving existing applications. ESCs, as an example, have traditionally used measurements of the lateral acceleration and the yaw rate to compute desired control inputs [Isermann, 2006; Savaresi and Tanelli, 2010]. Furthermore, ESCs assume knowledge of the wheel slip, which often is estimated separately from other vehicle states using longitudinal acceleration and wheel-rotation measurements. Sensor fusion enables simultaneous use of many sensors, which can improve tracking accuracy while using fewer estimation algorithms.

In multisensor target-tracking systems, local sensor measurements are typically sent to a common fusion center, where the measurements are fused to form state estimates. Because of different data processing and transmission times, some measurements can arrive when more recent measurements have already been processed. Figure 1.4 illustrates a possible scenario. These delayed measurements are denoted out-of-sequence measurements (OOSMs). They arise in a number of applica-

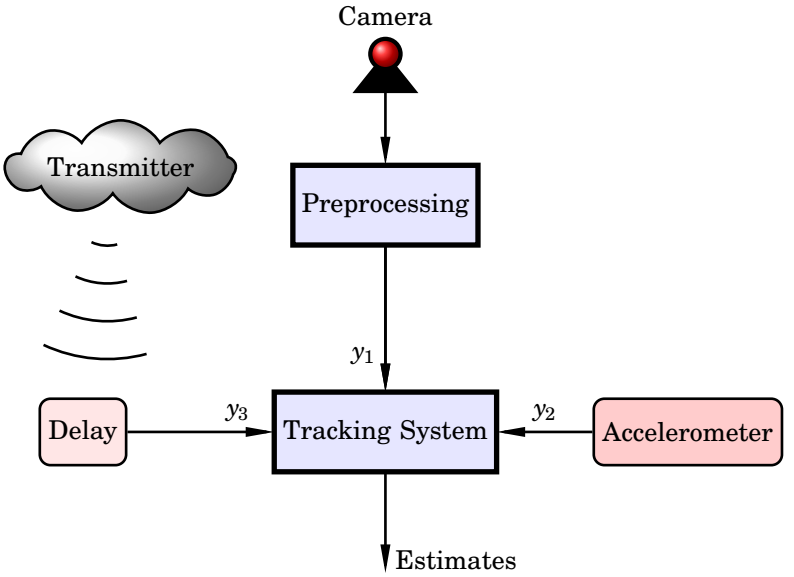

Figure 1.4 An example of a multisensor target-tracking system. The different measurements arise from different types of sensors. Thus, the measurements arrive at the tracking system with different delays.

tions, and with the technological development and improvement of sensors, the OOSM problem is gaining more interest in different applications, see [Bar-Shalom et al., 2001] for a coverage of estimation techniques and applications. More recent examples are data-traffic applications [Jia et al., 2008], visual target tracking for autonomous vehicles [Agnoli et al., 2008], and automotive precrash systems [Muntzinger et al., 2010]. Most research on the inclusion of OOSMs in estimation algorithms have focused on linear systems. However, many systems in the real world have nonlinear characteristics; hence, it is important to also include OOSMs in tracking systems that are aimed at nonlinear systems.

Optimal Control of Ground Vehicles

Optimal control of ground vehicles is interesting for several reasons. One objective is to develop improved active safety and driver-assistance systems for production cars. Numerous studies have shown the importance of ESCs. Investigations in Sweden during the years 1998–2004 showed that ESCs had an effectiveness on fatal crashes of approximately 50%. It was estimated that of the 500 vehicle-related deaths annually, up to 100 could have been avoided had the vehicles been equipped with an ESC.

Figure 1.5 An example of a maneuver that expert drivers can handle while maintaining vehicle stability. Author: Christopher Batt. Made available via Wikimedia Commons.

Also, a National Highway for Traffic Safety Administration (NHTSA) report showed a reduction of single vehicle crashes with 35% [Lie et al., 2005] in the USA. With that said, there are still about 30 000 fatal vehicle crashes per year in the USA alone, see the NHTSA report [NHTSA, 2011]. Moreover, a recent paper points out that the current generation of safety systems is still inferior to the maneuvering performance achievable by expert drivers in critical situations [Funke et al., 2012]. Figure 1.5 gives an example of an expert driver's ability to maintain stability while performing extreme maneuvering.

The long-term goal is obviously to achieve an autonomous vehicle fleet. Before this is possible, several issues have to be solved, technological as well as legislative. Hence, improved driver-assistance systems that are situation aware can be seen as an intermediate step toward autonomy. Optimal control is an enabler for autonomy because it provides a systematic, united approach to vehicle control. Sometimes the optimal-control problem is too complex to use directly for devising new control systems [Sharp and Peng, 2011]. In those cases optimal control at least increases understanding of vehicle dynamics, in the sense that it helps finding model limitations—for example, approximations in the modeling of chassis suspension, see Figure 1.6.

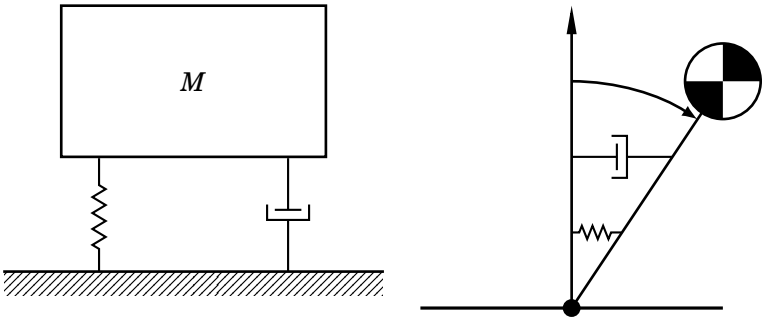

Figure 1.6 An example of different techniques for modeling chassis suspension. One approach (left) is to model each wheel as a spring-damper system, whereas another possibility (right) is to model the whole vehicle as a torsional spring-damper system. Of course, there is also the possibility to neglect suspension altogether. How this modeling is performed may have large impact on the model behavior.

Another motivation for optimal control is task-execution effectiveness. To improve productivity in production scenarios there is a huge potential for replacing industrial robots with, or mount them on, mobile robots, the main reason being increased flexibility and work space. Examples of applications that could benefit from this approach include painting, medical surgery assistance, logistics, and assembly applications. An integral part of the programming and task execution of mobile robots is the path and trajectory generation. A common task is to move the robot from point A to point B, without constraints on the path between the endpoints. However, in certain applications the path between the points is of explicit interest, and thus reliable path tracking is desired. Another scenario is that a high-level path planner determines the geometric path, and a subsequent trajectory generation is to be made [LaValle, 2006]. Naturally, in a path-tracking application where the robot actuators are the limiting factors, a near time-optimal solution robust to model uncertainties is desired in order to maximize productivity.

1.3 Contributions

This thesis presents work on how nonlinear estimation techniques and optimal control can improve vehicle behavior. The main contributions can be summarized as:

- Integration of components for enabling mobile manipulation in a realistic industrial scenario. Position, velocity, and force control is

used for controlling a mobile-manipulator setup in a pick-and-place scenario. The use of an algorithm that incorporates OOSMs ensures reliable estimation performance.

- A particle-filter algorithm that incorporates a method for taking into account that several OOSMs can arrive at the same time step.

- Two particle-filter algorithms that account for that measurements from different sensors often arrive delayed and out of sequence. The algorithms utilize model structure. This enables improved performance compared with previous approaches, when applicable.

- A method for performing combined wheel-slip and motion estimation of ground vehicles. The problem is formulated such that it can be solved using particle-filter methods that utilize linear substructure. Unlike other approaches that deal with slip estimation, the method explicitly models the nonlinear slip dynamics in the state and measurement equations, and combines this with estimating the pose and velocities. In addition, the method only relies on kinematic relations. This significantly reduces parameter uncertainty.

- A thorough investigation of the influence of vehicle- and tire-model configurations for use in automotive safety. The contribution compares vehicle models in aggressive maneuvers.

- An investigation of the influence of road surfaces in optimal road-vehicle maneuvers. The contribution lies in that the comparison is based on experimental data, using an experimentally verified tire model.

- A two-level, hierarchical approach to online trajectory generation for improved vehicle safety and/or autonomy. Both nonconvex and convex optimization techniques are used in a two-level structure with feedback. The first novelty is the use of a nonlinear vehicle model with tire modeling in the optimization problem at the high level. The second novelty is a combined nonconvex/convex control structure at the low level.

- A design flow for real-time, time-optimal trajectory generation and collision avoidance for four-wheeled vehicles. The approach includes both high-level and low-level convex optimization, with feedback from both global and local information. The contribution outlines the whole chain, from modeling to implementation.

1.4 Included Publications

This section states all papers that the author has been involved in, and give the author's contributions to all publications that the thesis is based on.

Berntorp, K. (2013). *Derivation of a Six Degrees-of-Freedom Ground-Vehicle Model for Automotive Applications*. Technical Report ISRN LUTFD2/TFRT--7627--SE. Department of Automatic Control, Lund University, Sweden.

This publication presents a derivation of a two-track vehicle model that incorporates rotations in space as well as load transfer. It also discusses aspects of tire modeling. The vehicle modeling is done using a Newton-Euler approach. The model is derived with control applications in mind.

Berntorp, K., K.-E. Årzén, and A. Robertsson (2011). "Sensor fusion for motion estimation of mobile robots with compensation for out-of-sequence measurements". In: *11th International Conference on Control, Automation and Systems*. Gyeonggi-do, Korea.

The tracking problem for mobile robots is approached by fusing measurements from inertial sensors, wheel encoders, and a camera. An implementation that executes online is done on a four-wheeled pseudo-omnidirectional mobile robot, using a dynamic model with 11 states. The algorithm is analyzed and validated with simulations and experiments.

K. Berntorp was the main contributor to this publication. A. Robertsson assisted in setting up the experimental equipment, and he and K.-E. Årzén gave suggestions for improvements and valuable input on the manuscript.

Berntorp, K., K.-E. Årzén, and A. Robertsson (2012). "Mobile manipulation with a kinematically redundant manipulator for a pick-and-place scenario". In: *2012 IEEE Multi-Conference on Systems and Control*. Dubrovnik, Croatia.

This paper combines a pseudo-omnidirectional mobile robot and a kinematically redundant manipulator for enabling mobile manipulation. The scenario is that of distributing groceries on refilling shelves, and we use a constraint-based task specification methodology to incorporate sensors and geometric uncertainties into the task. Sensor fusion is used for estimating the pose of the mobile base online. Force sensors are utilized to resolve remaining uncertainties. The approach is verified with experiments.

K. Berntorp was the main contributor. K.-E. Årzén outlined the basic idea, and K. Berntorp worked out all details regarding estimation, control, and implementation. A. Robertsson assisted with the experimental setup, and both he and K.-E. Årzén provided valuable comments on the manuscript.

Berntorp, K., K.-E. Årzén, and A. Robertsson (2012). "Storage efficient particle filters with multiple out-of-sequence measurements". In: *15th International Conference on Information Fusion*. Singapore.

OOSMs typically arise when the sensing information in some way uses preprocessing. Here, we treat the multiple OOSM problem for nonlinear models. The proposed method exploits the complete in-sequence information approach and extends it to nonlinear systems. Simulations indicate improved tracking performance for the considered scenario.

K. Berntorp was the main contributor. A. Robertsson and K.-E. Årzén provided valuable comments on the manuscript.

Berntorp, K., A. Robertsson, and K.-E. Årzén (2013). "Rao-Blackwellized out-of-sequence processing for mixed linear/nonlinear state-space models". In: *16th International Conference on Information Fusion*. Istanbul, Turkey.

Here, we investigate the OOSM particle-filtering problem for mixed-Gaussian models, which is a class of dynamic systems that often arise in navigation and tracking applications. The paper includes a simulation study on two benchmark examples.

The idea is due to K. Berntorp, who also derived the models and implemented the examples used in the paper. A. Robertsson and K.-E. Årzén provided valuable comments on the manuscript.

Olofsson, B., K. Lundahl, K. Berntorp, and L. Nielsen (2013). "An investigation of optimal vehicle maneuvers for different road conditions". In: *7th IFAC Symposium on Advances in Automotive Control*. Tokyo, Japan.

The subject of this publication is optimal maneuvers for vehicles on different road surfaces such as asphalt, snow, and ice. This study is motivated by the desire to find control strategies for improved future vehicle safety and driver assistance technologies. We develop vehicle and tire models corresponding to different road conditions and determine the time-optimal maneuver in a hairpin turn for each of these. Our main finding is that there are fundamental differences between how the vehicle should perform the maneuver on different surfaces.

K. Berntorp developed the optimization methodology together with B. Olofsson. K. Berntorp, B. Olofsson, and K. Lundahl performed the optimizations, and K. Lundahl was main responsible for tire-model calibrations. L. Nielsen provided comments and assisted in structuring the manuscript.

Berntorp, K., B. Olofsson, and A. Robertsson (2014). "Path tracking with obstacle avoidance for pseudo-omnidirectional mobile robots using convex optimization". In: *2014 American Control Conference*. Portland, Oregon. Accepted.

This paper considers time-optimal path tracking for the class of pseudo-omnidirectional mobile robots. Using sensor data, objects along the desired path are detected. Subsequently, a new path is planned and the corresponding time-optimal trajectory is found. The robustness of the method and its sensitivity to model errors are analyzed and discussed with extensive simulation results. Moreover, we verify the approach by successful execution on a physical setup.

This publication was developed as a cooperation between K. Berntorp and B. Olofsson, and equal contribution is asserted. A. Robertsson provided comments for the method and gave valuable ideas for improving the manuscript.

Submitted Publications

Berntorp, K., A. Robertsson, and K.-E. Årzén (2014). "Rao-Blackwellized particle filters with out-of-sequence measurement processing". *IEEE Transactions on Signal Processing*. Submitted.

The OOSM particle-filtering problem for mixed-Gaussian models is revisited. We develop and further improve two algorithms that utilize the linear substructure, and provide an extensive simulation study using three different systems. Both algorithms yield estimation improvements when compared with recent particle-filter algorithms for OOSM processing. In some cases the proposed algorithms even deliver accuracy that is similar to the lower performance bounds.

K. Berntorp was the main contributor. He derived the models and implemented the examples used in the paper. A. Robertsson and K.-E. Årzén provided valuable comments on the manuscript.

Berntorp, K. (2014). "Particle filter for combined wheel-slip and vehicle-motion estimation". *IEEE Transactions on Control Systems Technology*. Submitted.

Traditionally, estimation algorithms in automotive systems have been designed to aid a specific application, implying that correlation with other variables is neglected. This paper uses a Bayesian approach to estimation. We derive a model for combined wheel-slip and vehicle-motion estimation. By modeling the coupling between the vehicle states, improved tracking accuracy is achieved. A Rao-Blackwellized particle filter estimates 14 states in total, including key variables in active safety systems, such as longitudinal velocity, roll angle, and wheel slip for all four wheels. One key feature is that the method is robust to vehicle parameter uncertainties, because it only relies on kinematic relationships. The results show that the estimation algorithm provides high-precision tracking in the vast majority of the executions, when evaluated on a demanding scenario. Moreover, a comparison with a slip-estimation algorithm from the literature indicates clear improvements in terms of slip estimation.

Berntorp, K., B. Olofsson, K. Lundahl, and L. Nielsen (2014). "Models and methodology for optimal trajectory generation in safety-critical road-vehicle maneuvers". *Vehicle System Dynamics*. Submitted.

There has not been much research devoted to comparing vehicle models for at-the-limit maneuvers. This paper aims to fill this void by thoroughly investigating six vehicle models in three different maneuvers. The results are extensively analyzed and discussed. We also outline a methodology for how to solve these types of problems.

The original problem formulation is due to L. Nielsen. K. Berntorp, B. Olofsson, and K. Lundahl developed the methodology, sorted out all details, and performed the optimizations. K. Berntorp and B. Olofsson were main responsible for analysis of the results and the manuscript.

Berntorp, K. and F. Magnusson (2014). "Closed-loop optimal control for vehicle autonomy". In: *53rd IEEE Conference on Decision and Control*. Los Angeles, California. Submitted.

This paper presents a hierarchical approach to feedback-based trajectory generation for improved vehicle autonomy. Hierarchical control structures have been used in safety systems before—for example, in ESCs, where a simplified model generates high-level references for a low-level control loop to handle. This paper contains two novelties: First, we include a nonlinear vehicle model already at the high level to generate optimization-based references. Second, we use two model predictive control formulations at the low level for increased robustness, together with a vehicle model that incorporates load transfer and rotations in space. With this structure the two control layers have a physical coupling, which makes it easier for the low-level loop to track the references.

K. Berntorp was the main contributor. He developed the control structure, and implemented and tuned the controllers. F. Magnusson assisted in the implementation, developed the low-level framework needed for the application, and provided valuable comments on the manuscript.

Olofsson, B., K. Berntorp, and A. Robertsson (2014). "A convex approach to path tracking with obstacle avoidance". *IEEE Transactions on Control Systems Technology*. Submitted.

To improve autonomy for wheeled vehicles, we consider the problem of combined trajectory generation and online collision avoidance for four-wheeled vehicles with independent steer and drive actuation on each wheel. An Euler-Lagrange model of the dynamics is derived, and by making appropriate approximations a convex reformulation of the path-tracking problem is developed. This enables the use of time-optimal trajectories during runtime, which combined with model predictive control that achieves feedback from the estimated global position and orientation provides robustness to model uncertainty and disturbances. The proposed approach also incorporates avoidance of moving obstacles, which are not encoded in the map information and thus unknown a priori. We verify the proposed approach by successful execution on a pseudo-omnidirectional mobile robot platform, and compare the method to an algorithm that is currently used on the considered mobile robot platform.

This publication was developed as a cooperation between K. Berntorp and B. Olofsson, and equal contribution is asserted. A. Robertsson provided comments for the method and gave valuable ideas for improving the manuscript.

1.5 Other Publications

The following publications were chosen not to be included in the thesis.

Berntorp, K. (2008). *ESP for Suppression of Jackknifing in an Articulated Bus*. Master's Thesis ISRN LUTFD2/TFRT--5831--SE. Department of Automatic Control, Lund University, Sweden.

Berntorp, K. and J. Nordh (2014). "Rao-Blackwellized particle smoothing for occupancy-grid based SLAM using low-cost sensors". In: *19th IFAC World Congress*. Cape Town, South Africa. Accepted.

Berntorp, K., B. Olofsson, K. Lundahl, B. Bernhardsson, and L. Nielsen (2013). "Models and methodology for optimal vehicle maneuvers applied to a hairpin turn". In: *2013 American Control Conference*. Washington, DC.

Lundahl, K., K. Berntorp, B. Olofsson, J. Åslund, and L. Nielsen (2013). "Studying the influence of roll and pitch dynamics in optimal road-vehicle maneuvers". In: *23rd International Symposium on Dynamics of Vehicles on Roads and Tracks*. Qingdao, China.

Lundahl, K., B. Olofsson, K. Berntorp, J. Åslund, and L. Nielsen (2014). "Towards lane-keeping electronic stability control for road-vehicles". In: *19th IFAC World Congress*. Cape Town, South Africa. Accepted.

Magnusson, F., K. Berntorp, B. Olofsson, and J. Åkesson (2014). "Symbolic transformations of dynamic optimization problems". In: *10th International Modelica Conference*. Lund, Sweden.

Nordh, J. and K. Berntorp (2012). "Extending the occupancy grid concept for low-cost sensor based SLAM". In: *10th International IFAC Symposium on Robot Control*. Dubrovnik, Croatia.

Nordh, J. and K. Berntorp (2013). *pyParticleEst — A Python Framework for Particle Based Estimation*. Technical Report ISRN LUTFD2/TFRT--7628--SE. Department of Automatic Control, Lund University, Sweden.

1.6 Preliminaries

Throughout, vectors and column matrices will be used interchangeably.

- Vector variables are written in boldface letters as in $\boldsymbol{x}$, with its scalar components written in lightface letters as in x.
- Matrices are denoted with capital boldface letters as in $\boldsymbol{A}$.
- $\mathbb{R}^n$ means the real-valued vector space of dimension n. Matrices with dimension $m \times n$ are written as $\mathbb{R}^{m\times n}$.
- The Euclidean norm of a vector $\boldsymbol{x}$ is $\|\boldsymbol{x}\|$ and $\|\boldsymbol{x}\|_{\boldsymbol{Q}} = \sqrt{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}}$.
- The zero column matrix of dimension $n \times 1$ is $\mathbf{0}_n$. We will write out the dimension when it is deemed helpful. The zero and identity matrix of dimension $n \times n$ are written as $\mathbf{0}_{n\times n}$ and $\boldsymbol{I}_{n\times n}$, respectively.
- The time derivative of a variable x is written as $\dot{x}$ or

$$\frac{\mathrm{d}}{\mathrm{d}t}x,$$

  depending on the context.

- The notation
$$\left.\frac{\partial f}{\partial x}\right|_{\bar{x}}$$
means the partial derivative of f evaluated at $\bar{x}$.

1.7 Outline

Chapter 2 gives an overview of control methods that relate to the work in this thesis. A small subset of the available particle-filtering techniques for nonlinear state estimation are briefly reviewed in Chapter 3. Different tire and vehicle models that are common in literature and used throughout the thesis are introduced in Chapter 4. In addition, we give a derivation of a nonlinear vehicle model that incorporates rotations in space as well as chassis suspension. In Chapter 5 we present the notion of out-of-sequence measurements. In addition, previous work and a general problem formulation are stated.

The contributions of this thesis are presented in Chapters 6–12, and can roughly be divided into two areas:

- Nonlinear estimation
- Optimal control of ground vehicles

Nonlinear Estimation

Chapter 6 presents an application study in which OOSMs are compensated for. The application is a mobile robotic setup where a mobile manipulator picks up and places groceries. It is a rather challenging application, which includes estimation using vision and inertial sensing, as well as control of positions, velocities, and forces.

An approach to deal with OOSMs in nonlinear tracking systems when several of them arrive at the same time instant is investigated in Chapter 7. We evaluate the performance in a target-tracking example and compare against related algorithms.

Chapter 8 presents two algorithms for handling OOSMs when a linear substructure is present. The algorithms are derived with computational considerations in mind. We perform an evaluation on three tracking examples and compare the algorithms against related approaches.

Wheel-slip and motion estimation are often considered as two problems in the vehicle community. In Chapter 9 we present a method for estimating 14 states related to vehicle motion, some of which are key variables in automotive safety systems. The chapter includes experimental results, which are extracted from a race-track scenario with aggressive maneuvering.

Optimal Control of Ground Vehicles

Chapter 10 investigates dynamic optimization with applications to road vehicles. Different tire and chassis models are combined, and optimal solutions are found for two maneuvers using dynamic optimization. We thoroughly analyze the optimization results and discuss their implications. This chapter also investigates optimal road-vehicle maneuvers on different surfaces, where the tire-road parameters are based on experimental data.

The conclusions made in Chapter 10 are utilized in Chapter 11, where we outline a hierarchical control structure for improved vehicle autonomy. The chapter presents simulation results and discusses implementation.

Chapter 12 presents a convex approach to real-time, time-optimal trajectory generation and collision avoidance for pseudo-omnidirectional mobile robots. Given a geometric path from a high-level path planner, the approach finds a time-optimal trajectory over the defined path. Unforeseen obstacles, such as humans entering the vicinity of the planned path, are avoided by a model predictive control approach. The chapter gives the whole chain from idea to implementation, as well as verifying the approach using both simulated and experimental results on a mobile robot.

The thesis and its conclusions are summarized in Chapter 13, and Chapter 14 presents directions for future work. Appendix A contains information on the parametrization of rotations in space that is used extensively throughout the thesis. It also contains different parameters that have been used in the automotive applications. Finally, Appendix B contains MATLAB code for an example in Chapter 3.

Figure 1.7 shows a flowchart of the contents and how the chapters connect to each other.

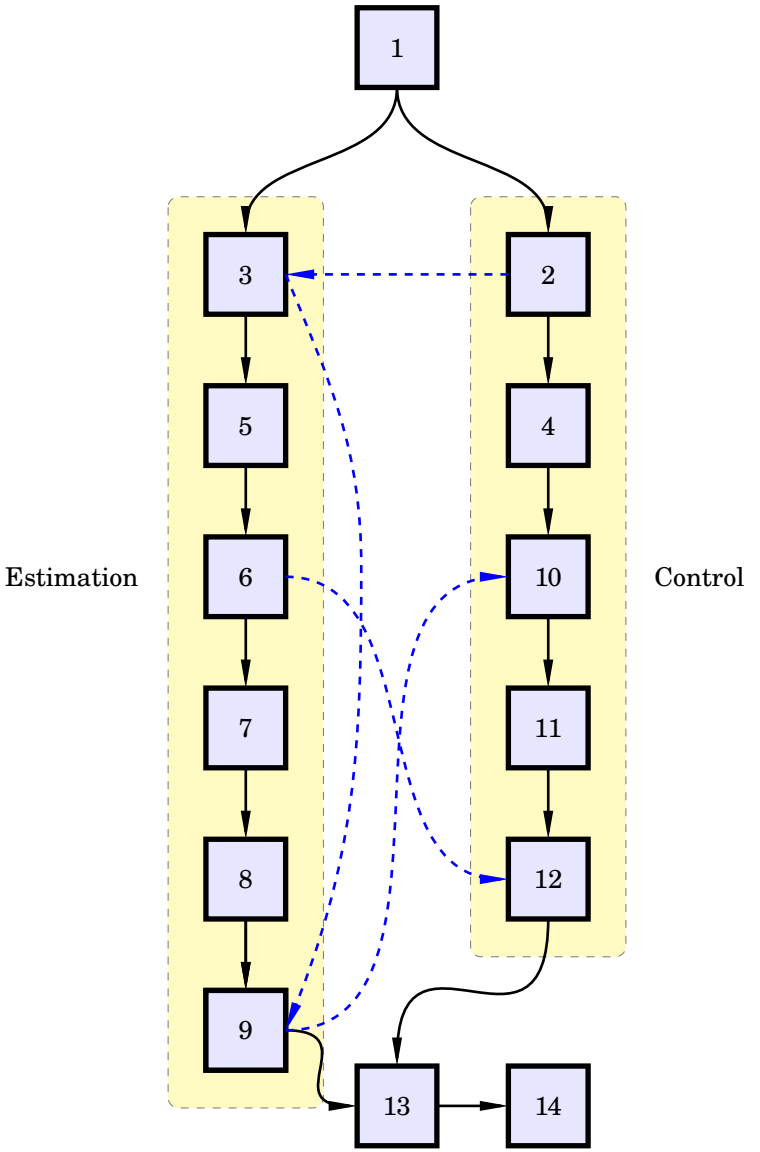

Figure 1.7 A flowchart of the contents. The thesis consists of two tracks. One track treats vehicle estimation and the other covers vehicle control. The blue dashed arrows indicate interconnections between the two tracks. Chapters 3 and 9 should be read before Chapter 10 to get additional understanding. Similarly, Chapters 6 and 12 are tightly connected.

2

Control Concepts

This chapter presents background material on control concepts that are used in the thesis. The aim is to provide so much details that the remaining chapters can be understood without having to consult other material, in addition to introduce the notation that is used throughout the thesis. Each section points out references to appropriate literature where detailed derivations and in-depth discussions can be found.

2.1 PID Control

PID (Proportional Integral Derivative) control is widely used in academia and industry, and it is by far the most common feedback mechanism used in the process industry [Åström and Wittenmark, 1997; Åström and Murray, 2008]. The standard parallel version is in continuous time given by

$$u(t) = K_p\left(e(t) + \frac{1}{K_i}\int_{-\infty}^{t} e(\tau)\,\mathrm{d}\tau + K_d\frac{\mathrm{d}e(t)}{\mathrm{d}t}\right) = P(t) + I(t) + D(t), \quad (2.1)$$

where t is the time, $u(t)$ is the control input, $e(t) = r(t) - y(t)$ is the difference between desired (reference) value $r(t)$ and measured value $y(t)$, K_p is the proportional gain, K_i is the integral gain, and K_d is the derivative gain. The controller consists of three parts. The proportional part concerns the current control error, whereas the integral part and derivative part consider the history and predicted future, respectively. To avoid excessive magnifications of measurement noise, it is common to filter the derivative part with a low-pass filter. By using a first-order filter, the derivative is modified as

$$D(t) = -\frac{K_d}{N_d}\frac{\mathrm{d}D(t)}{\mathrm{d}t} + K_pK_d\frac{\mathrm{d}e(t)}{\mathrm{d}t},$$

where N_d is a parameter that governs the low-pass characteristics. Both model-based and experimental-based tuning methods exist, see [Åström and Hägglund, 2005].

When implemented in a computer, a discretized version is needed. Denote the sampling period with T_s. The standard way of discretizing (2.1) at the sampling instant t_k is to approximate the integral part $I(t)$ with a forward difference—that is,

$$I(t_t + T_s) \approx I(t_k) + \frac{K_p T_s}{K_i} e(t_k).$$

The derivative part is often approximated with a backward approximation, which gives

$$D(t_k) \approx \frac{K_d}{K_d + T_s N_d} D(t_k - T_s) + \frac{K_p K_d N_d}{K_d + T_s N_d} \left(e(t_k) - e(t_k - T_s) \right).$$

In some applications, for example in the process industry, the reference typically changes in steps. This means that the reference is constant for most of the time, with occasional, large changes at specific time instants. In those cases, excessive control signals are avoided by setting the reference to zero in the derivative part.

We will use PID control in Chapter 6, but it is also implicitly used in other parts of the thesis.

2.2 Force Control

Force control [Siciliano and Villani, 1999] is frequently used in robotics. Some examples are deburring [Hsu and Fu, 2000], drilling [Olsson, 2007], and assembly [Linderoth, 2013], where force sensors measure the contact forces. Force feedback can be performed with (2.1). A more common approach is to employ impedance (admittance) control [Hogan, 1984; Spong and Hutchinson, 2006]. The idea is to control the apparent inertia of the system, and a common controller form in Cartesian space is

$$u(t) = \frac{1}{M} \Big(F(t) - F_r(t) - D(\dot{r}(t) - \dot{p}(t)) - K(r(t) - p(t)) \Big), \tag{2.2}$$

where M, D, and K are tuning parameters that describe the desired mass, damper constant, and spring constant of the system, respectively. Moreover, $F(t)$ is the measured force, $F_r(t)$ is the force reference, $p(t)$ is the position, and $\dot{p}(t)$ is the time derivative of p at time t. Figure 2.1 provides the intuition. Impedance controllers are popular in force-control applications because they provide an intuitive control structure for when in contact with stiff environments. Note that the control structure (2.2) allows for performing position and velocity control as well.

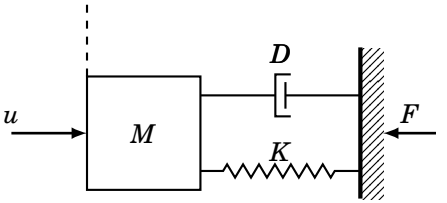

Figure 2.1 An illustration of impedance-based force control. The idea is to change the apparent inertia of the system. The desired behavior is chosen by selecting appropriate mass M, damper constant D, and spring constant K.

2.3 Dynamic Optimization

Dynamic optimization has gained increased attention over the last decade, both in academia and industry [Albanesi et al., 2006; Larsson, 2011; Grover and Andersson, 2012; Sällberg et al., 2012]. Examples of applications are parameter estimation, offline trajectory generation, and online optimization. In one general formulation, it aims to solve optimization problems while allowing for model descriptions written as differential-algebraic equations (DAEs); that is,

$$\boldsymbol{f}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{w}(t), \boldsymbol{u}(t), \boldsymbol{p}) = \boldsymbol{0}, \tag{2.3}$$

where $\boldsymbol{f}$ in general is a vector-valued function and $\boldsymbol{x}(t) \in \mathbb{R}^{n_x}$, $\boldsymbol{w}(t) \in \mathbb{R}^{n_w}$, $\boldsymbol{u}(t) \in \mathbb{R}^{n_u}$, and $\boldsymbol{p} \in \mathbb{R}^{n_p}$ contain the states, algebraic variables, control variables, and system parameters, respectively. Figure 2.2 shows the different classes of optimization problems we will encounter in this thesis, and we will briefly go through the main ideas next. Given the model description (2.3), the optimization problem is formulated on the time interval $t \in [0, t_f]$ as

$$\begin{aligned}
\underset{\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u}, \boldsymbol{p}, t_f}{\text{minimize}} \quad & J(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{w}(t), \boldsymbol{u}(t), \boldsymbol{p}) && \text{(2.4a)} \\
\text{subject to} \quad & \boldsymbol{f}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{w}(t), \boldsymbol{u}(t), \boldsymbol{p}) = \boldsymbol{0} && \text{(2.4b)} \\
& \boldsymbol{f}_0(\dot{\boldsymbol{x}}(0), \boldsymbol{x}(0), \boldsymbol{w}(0), \boldsymbol{u}(0), \boldsymbol{p}) = \boldsymbol{0} && \text{(2.4c)} \\
& \bar{\boldsymbol{f}}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{w}(t), \boldsymbol{u}(t), \boldsymbol{p}) \leq \boldsymbol{0} && \text{(2.4d)} \\
& \bar{\boldsymbol{h}}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{w}(t), \boldsymbol{u}(t), \boldsymbol{p}) = \boldsymbol{0} && \text{(2.4e)} \\
& \bar{\boldsymbol{f}}_{\text{end}}(\dot{\boldsymbol{x}}(t_f), \boldsymbol{x}(t_f), \boldsymbol{w}(t_f), \boldsymbol{u}(t_f), \boldsymbol{p}) \leq \boldsymbol{0} && \text{(2.4f)} \\
& \bar{\boldsymbol{h}}_{\text{end}}(\dot{\boldsymbol{x}}(t_f), \boldsymbol{x}(t_f), \boldsymbol{w}(t_f), \boldsymbol{u}(t_f), \boldsymbol{p}) = \boldsymbol{0}, && \text{(2.4g)}
\end{aligned}$$

where (2.4a) is the scalar-valued objective function, (2.4c) contains the initial conditions, (2.4d) contains the path inequality constraints, and

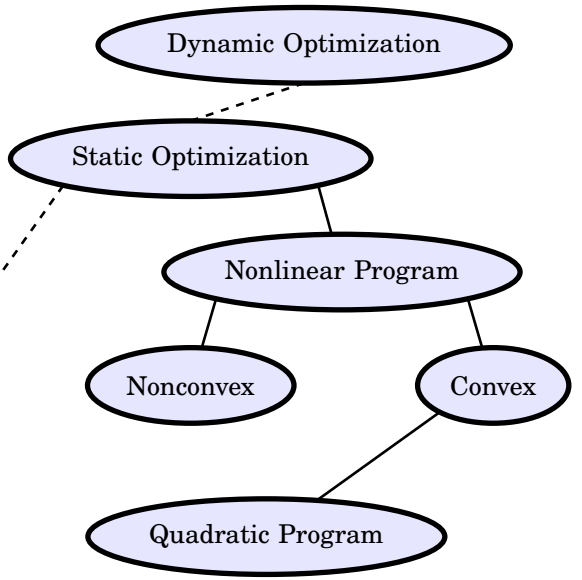

Figure 2.2 The different classes of optimization problems that we will encounter in the thesis and briefly go through in this chapter. The dashed line to the left indicates other classes, such as mixed-integer optimization problems. In this thesis, dynamic optimization problems are (in an approximate manner) formulated as static optimization problems, which are solved by nonlinear programs.

(2.4e) contains the path equality constraints. Moreover, (2.4f) and (2.4g) describe the terminal constraints. These are similar to the path constraints, but are only enforced at the final time t_f. To exemplify, the initial conditions in (2.4c) are often on the form $\boldsymbol{x}(0) = \boldsymbol{x}_0$ and the inequality constraints are typically bounds on the optimization variables (e.g., $-\boldsymbol{u}_{\max} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{\max}$). The time dependency is most often suppressed throughout the thesis.

Formulation of Dynamic Optimization Problems

The optimization problem (2.4) can be solved using two different approaches, which are direct and indirect approaches, see [Biegler, 2010] for a thorough investigation. The indirect approach is based on first-order necessary conditions for optimality [Pontryagin et al., 1964]. The optimality conditions are formulated as a set of DAEs, which results in a two-point boundary-value problem [Cervantes and Biegler, 2009].

The direct approach involves transforming the infinite-dimensional problem (2.4) to a finite-dimensional *nonlinear program* (NLP) [Biegler,

2010] by discretization. The resulting NLP is on the form

$$\begin{aligned} \text{minimize} \quad & \mathcal{J}(\bar{\boldsymbol{x}}) && \text{(2.5a)} \\ \text{subject to} \quad & f_i(\bar{\boldsymbol{x}}) \leq 0, \quad i = 1, \ldots, m && \text{(2.5b)} \\ & h_i(\bar{\boldsymbol{x}}) = 0, \quad i = 1, \ldots, n, && \text{(2.5c)} \end{aligned}$$

where $\bar{\boldsymbol{x}} \in \mathbb{R}^{n_x}$ contains the discretized variables. Further, (2.5a) is a discretized version of (2.4a), where $\mathcal{J} : \mathbb{R}^{n_x} \to \mathbb{R}$, and (2.5b)–(2.5c) are the corresponding discretized inequality and equality constraints, respectively, with $f_i : \mathbb{R}^{n_x} \to \mathbb{R},\ i = 1, \ldots, m$, $h_i : \mathbb{R}^{n_x} \to \mathbb{R}, i = 1, \ldots, n$. The transformation is either done by sequential or simultaneous methods.

Sequential Methods In sequential methods, the control variables are discretized and parametrized by piecewise polynomials. Given initial conditions and control parameters, the DAE system is integrated (simulated) forward in time and used for evaluation of (2.4a). Using the DAE integration an NLP solver finds improved control parameters, whereby the procedure is repeated [Binder et al., 2001]. Sequential methods are relatively easy to construct, but require repeated numerical integration. Further, path constraints are typically handled by introducing extra terms in the penalty function.

Simultaneous Methods Simultaneous methods mainly consist of multiple shooting and direct transcription [Biegler, 2010]. Multiple shooting splits the time domain into smaller elements, and in each element the DAEs are integrated separately from the other segments. Continuity is enforced by including equality constraints at the initial and terminal point, respectively, of two neighboring elements. Control variables are treated in a similar way as for sequential methods, but a difference is that path constraints can be enforced at the grid points. However, they might be violated between the grid points.

Direct transcription methods discretize all algebraic and control variables, as well as the states and the corresponding equations. Typically the discretization makes use of polynomial approximations. The result is a large, sparse NLP of the form (2.5), which is a difference compared with sequential and multiple shooting methods. Thus, direct transcription methods do not need any integrators as the solution is found simultaneously for all time instants. A drawback with this approach is that adaptive discretization schemes are in general not straightforward to include.

Solution of Nonlinear Programs

Two methods to solve NLPs are active-set methods and interior-point methods. Active-set methods essentially aim to repeatedly remove inactive

constraints and solve smaller equality-constrained optimization problems. They assume that a feasible solution is available.

Interior-point methods reduce the full optimization problem by introducing the inequality constraints in the objective function and sequentially solving equality-constrained minimization problems with increasing precision. Feasibility is not achieved until the last iterations. Interior-point methods are typically solved using gradient-based methods, such as Newton's method.

A possible advantage of active-set methods is that the iterations remain feasible as soon as an initial feasible solution has been found. This can be advantageous for online implementations, where it because of computation time may be necessary to terminate before an optimum has been found. However, the need for a feasible initial solution is a disadvantage of active-set methods. Interior-point methods have been shown to be very efficient for large NLPs and for NLPs where certain structure is present. For more detailed presentations of both solver types, see [Biegler, 2010; Maciejowski, 2002].

Dynamic optimization will be used in Chapters 10 and 11, and the optimization problems will be solved by using direct transcription and an interior-point method.

2.4 Convex Optimization

Convex optimization problems are special cases of NLPs (see Figure 2.2 for the hierarchy), in which the optimization problem has certain beneficial properties. The reader is referred to [Boyd and Vandenberghe, 2008] for in-depth discussions of how to formulate and solve convex optimization problems.

Definition 2.1—Convex Sets
A set $\mathbb{S}$ is convex if and only if

$$\theta \boldsymbol{x}_1 + (1-\theta)\boldsymbol{x}_2 \in \mathbb{S} \tag{2.6}$$

for any point $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{S}$ and any $\theta \in [0,1]$. □

The definition implies that the straight line that connects $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ belongs to $\mathbb{S}$. Figure 2.3 gives an illustration of convex sets.

Definition 2.2—Convex Functions
A function $f : \mathbb{R}^{n_x} \to \mathbb{R}$ is convex if and only if it fulfills

$$f(\theta \boldsymbol{x}_1 + (1-\theta)\boldsymbol{x}_2) \leq \theta f(\boldsymbol{x}_1) + (1-\theta) f(\boldsymbol{x}_2) \tag{2.7}$$

for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^{n_x}$ and any $\theta \in [0,1]$. □

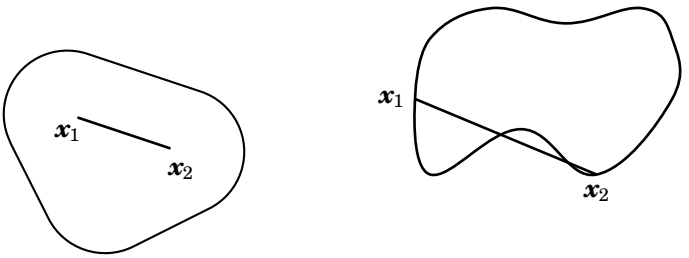

Figure 2.3 A convex set (left) and a nonconvex set (right). Because every point on a line that connects $(\boldsymbol{x}_1,\boldsymbol{x}_2)$ belong to the same set as $(\boldsymbol{x}_1,\boldsymbol{x}_2)$ if and only if the set is convex, the set to the left is convex while the set to the right is not.

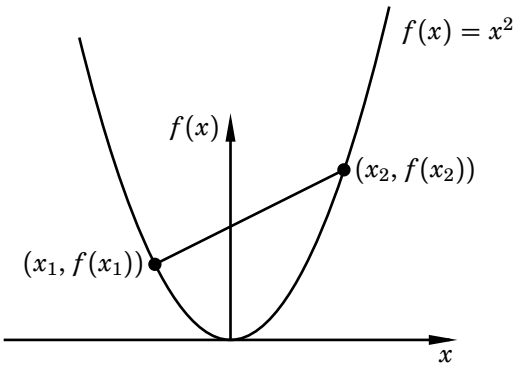

Figure 2.4 An illustration of the convexity concept in the one-dimensional case. For a convex function, in this case $f(x) = x^2$, the line segment between $(x_1, f(x_1))$ and $(x_2, f(x_2))$ lies above the graph of f.

For differentiable functions, Definition 2.2 is equivalent to

$$f(\boldsymbol{x}_1) \geq f(\boldsymbol{x}_2) + \nabla f(\boldsymbol{x}_2)^{\mathrm{T}}(\boldsymbol{x}_1 - \boldsymbol{x}_2), \tag{2.8}$$

where

$$\nabla f(\boldsymbol{x}_2) = \frac{\partial f(\boldsymbol{x}_2)}{\partial \boldsymbol{x}_2} \in \mathbb{R}^{n_x}$$

is the gradient to f at $\boldsymbol{x}_2$. This definition implies that the line segment between $(\boldsymbol{x}_1, f(\boldsymbol{x}_1))$ and $(\boldsymbol{x}_2, f(\boldsymbol{x}_2))$ lies above the graph of f, see Figure 2.4 for an illustration in the one-dimensional case.

Convex Optimization Problems

Again consider the NLP (2.5); that is, consider

$$\begin{aligned} \text{minimize} \quad & \mathcal{J}(\boldsymbol{x}) \\ \text{subject to} \quad & f_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m \\ & h_i(\boldsymbol{x}) = 0, \quad i = 1, \ldots, n, \end{aligned} \tag{2.9}$$

where $\boldsymbol{x} \in \mathbb{R}^{n_x}$ contains the optimization variables. The function $\mathcal{J} : \mathbb{R}^{n_x} \to \mathbb{R}$ is the objective function, $f_i : \mathbb{R}^{n_x} \to \mathbb{R}$, $i = 1, \ldots, m$ are the inequality constraint functions, and $h_i : \mathbb{R}^{n_x} \to \mathbb{R}, i = 1, \ldots, n$ are the equality constraint functions. The optimization problem (2.9) is convex if $\mathcal{J}$ and $\{f_i\}_{i=1}^m$ are convex functions and if $\{h_i\}_{i=1}^n$ are affine—that is, $h_i = \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} - b_i$. Moreover, the feasible set of (2.9) is convex. This class of optimization problems has several attractive properties. First, every locally optimal point $\boldsymbol{x}^*$ to (2.9) is also globally optimal. Second, there exist fast solvers for a large variety of different problem formulations. Third, many problems can be formulated as, or approximated by, convex optimization problems. Convex optimization is therefore used in a wide range of applications.

A quadratic program (QP) is a common special case of (2.9). In QPs, the objective function is quadratic and the equality constraint functions are linear. This yields

$$\text{minimize} \quad \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^{\mathrm{T}}\boldsymbol{x} + d \tag{2.10a}$$

$$\text{subject to} \quad \boldsymbol{G}\boldsymbol{x} - \boldsymbol{r} \leq \boldsymbol{0} \tag{2.10b}$$

$$\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{0}, \tag{2.10c}$$

where $\boldsymbol{G} \in \mathbb{R}^{m \times n_x}$ and $\boldsymbol{A} \in \mathbb{R}^{n \times n_x}$, with a symmetric, positive-definite matrix $\boldsymbol{P} \in \mathbb{R}^{n_x \times n_x}$.

For a detailed presentation of an interior-point method for solving convex optimization problems, see [Boyd and Vandenberghe, 2008]. By utilizing structure, QPs can be solved very efficiently with custom-made solvers, often within a few milliseconds and sometimes even faster. See [Mattingley and Boyd, 2012] for an example of a custom-made solver.

2.5 Model Predictive Control

Model Predictive Control (MPC) is a control technique that has gained attention in several contexts—for example, process control [Maciejowski, 2002; Wang et al., 2007], automotive applications [Del Re et al., 2010;

Hermans et al., 2013], and combustion-engine control [Widd, 2012; Henningsson, 2012]. For thorough descriptions of MPC, see [Rawlings and Mayne, 2009] and [Maciejowski, 2002]. It is an optimization-based control methodology that optimizes system behavior given predictions from a system model. The considered models are in continuous time typically (DAE formulations are also possible) described by the set of differential equations

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad \boldsymbol{x}(0) = \bar{\boldsymbol{x}}, \tag{2.11}$$

where $\boldsymbol{x} \in \mathbb{R}^{n_x}$, $\boldsymbol{u} \in \mathbb{R}^{n_u}$, $\boldsymbol{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$, and $\bar{\boldsymbol{x}}$ is the initial value. It is common to apply piecewise constant control inputs. Hence, (2.11) is often written as the difference equation

$$\boldsymbol{x}_{k+1} = \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k), \quad \boldsymbol{x}_0 = \bar{\boldsymbol{x}}, \tag{2.12}$$

where $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ is the corresponding state vector in discrete time and $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ is the input vector. The function $\boldsymbol{g} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is the discrete-time counterpart to $\boldsymbol{f}$ in (2.11). In the following we outline the MPC formulation in the discrete-time case; however, there exists a continuous-time equivalent [Mayne et al., 2000].

MPC solves a finite horizon optimal-control problem given the state estimate at time index k as initial condition, yielding the optimal future control sequence $\mathcal{U}_k^*$ as

$$\mathcal{U}_k^* = \{\hat{\boldsymbol{u}}_{k|k}^*, \ldots, \hat{\boldsymbol{u}}_{k+H_c-1|k}^*\},$$

where $\hat{\boldsymbol{u}}_{j|k}^*$ is the predicted optimal control input at time index j given information about the system up to time index k and H_c is the control horizon. Subsequently, the first control action $\hat{\boldsymbol{u}}_{k|k}^*$ is applied to the system. The procedure is repeated as soon as a new state estimate is available. Because of this, another name for MPC is receding horizon control. Note that feedback is introduced in the algorithm by repeating the optimization at each time instant. Throughout the thesis, we neglect that the control sequence is based on predictions, and simply write $\boldsymbol{u}_k$ for $\hat{\boldsymbol{u}}_{k|k}$. Further, depending on the actual computation time and sampling periods involved, the control sequence sometimes stretches from time index $k+1$ instead of time index k.

A feature with MPC is that it naturally handles state and control constraints; that is, the state and input vector can be forced to satisfy $\boldsymbol{x}_k \in \mathbb{X}$ and $\boldsymbol{u}_k \in \mathbb{U}$ for some sets $\mathbb{X}$ and $\mathbb{U}$, respectively. The cost function, $J_k(\boldsymbol{x}_k, \mathcal{U}_k)$, reflects the system behavior over a finite prediction horizon H_p, which typically is longer than the control horizon H_c:

$$J_k(\boldsymbol{x}_k, \mathcal{U}_k) = J_f(\boldsymbol{x}_{k+H_p}) + \sum_{i=1}^{H_p} J_{\boldsymbol{x}}(\boldsymbol{x}_{k+i}) + \sum_{i=0}^{H_c-1} J_{\boldsymbol{u}}(\boldsymbol{u}_{k+i}), \tag{2.13}$$

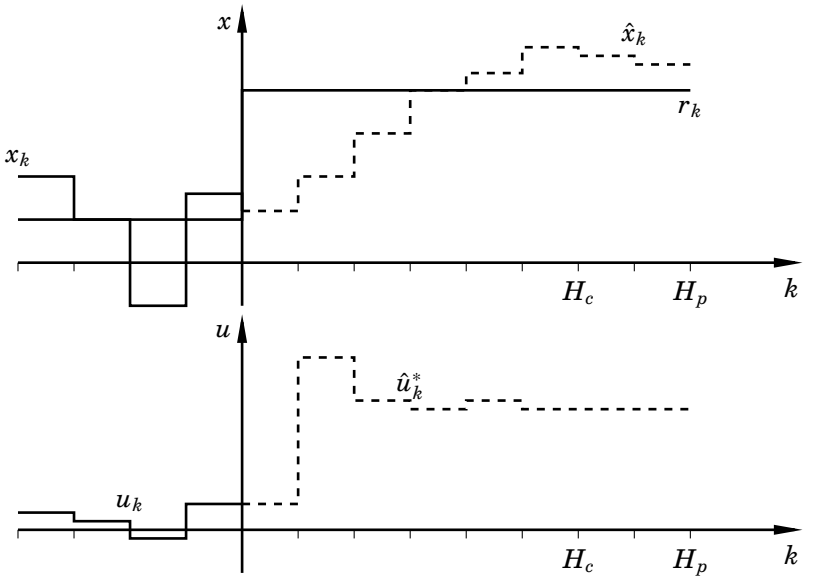

Figure 2.5 The basic idea of model predictive control in the one-dimensional case. The predicted optimal control input sequence $\mathcal{U}_k^*$ gives the predicted state sequence, and the first of these control inputs is applied to the system. The control sequence is found through minimization of a cost function that is specified in terms of initial state x_k and state-reference sequence r_i, for all $i \in \{k+1, \ldots, k+H_p\}$, as well as including a cost on the control inputs.

where $J_{\boldsymbol{x}}(\boldsymbol{x}_{k+i})$ and $J_{\boldsymbol{u}}(\boldsymbol{u}_{k+1})$ are the costs associated with the states and controls, respectively, and $J_f(\boldsymbol{x}_{k+H_p})$ is the terminal cost. Figure 2.5 illustrates the concept. When using $H_p > H_c$ in (2.13), the control input $\boldsymbol{u}_{k+i}$ is held constant for $i \geq H_c$. Moreover, the states are often restricted to reside in a set $\mathbb{X}_f$ at the final time (i.e., $\boldsymbol{x}_{k+H_p} \in \mathbb{X}_f$), where $\mathbb{X}_f \subseteq \mathbb{X}$. This yields the optimization problem

$$
\begin{aligned}
\underset{\mathcal{U}_k}{\text{minimize}} \quad & J_k(\boldsymbol{x}_k, \mathcal{U}_{H_c}) \\
\text{subject to} \quad & \boldsymbol{x}_{i+1} = \boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{u}_i), \quad \forall i \in \{k, \ldots, k+H_p\} \\
& \boldsymbol{x}_k = \bar{\boldsymbol{x}} \\
& \boldsymbol{x}_i \in \mathbb{X}, \quad \forall i \in \{k+1, \ldots, k+H_p\} \\
& \boldsymbol{u}_i \in \mathbb{U}, \quad \forall i \in \{k, \ldots, k+H_c-1\} \\
& \boldsymbol{x}_{k+H_p} \in \mathbb{X}_f.
\end{aligned}
\tag{2.14}
$$

There are numerous papers and books treating stability and feasibility of MPC. The survey paper [Mayne et al., 2000] states four sufficient conditions for stability. Two of the assumptions are that $\mathbb{X}_f$ and $\mathbb{X}$ are closed and $\mathbb{U}$ is compact. Further, the conditions in [Mayne et al., 2000] rely on a terminal cost, terminal constraint, and terminal controller to prove stability. Without imposing terminal constraints, a common approach to show stability is to assume that the horizons are sufficiently long [Maciejowski, 2002; Grimm et al., 2005; Grüne and Rantzer, 2008].

Linear Systems

An important special case in MPC is when the system dynamics are linear, that is,[1]

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{F}\boldsymbol{u}_k, \tag{2.15}$$

where $\boldsymbol{A} \in \mathbb{R}^{n_x \times n_x}$ and $\boldsymbol{F} \in \mathbb{R}^{n_x \times n_u}$ are constant matrices that define the system dynamics. The linear model (2.15) makes it possible to formulate the MPC problem as a QP (i.e., as in (2.10)). The QP formulation has the features:

1. A cost $J_k(\boldsymbol{x}_k, \mathcal{U}_k)$ that is quadratic in the state and input variables
2. Maximum and minimum constraints on the state and input variables
3. Rate constraints on the control variables

The assumption on the cost function means that $J_{\boldsymbol{x}}(\boldsymbol{x}_{k+1})$, $J_{\boldsymbol{u}}(\boldsymbol{u}_{k+1})$, and $J_f(\boldsymbol{x}_{k+H_p})$ in (2.13) are written as

$$\begin{aligned} J_{\boldsymbol{x}}(\boldsymbol{x}_{k+i}) &= (\boldsymbol{x}_{k+i} - \boldsymbol{r}_{k+i})^{\mathrm{T}} \boldsymbol{W}_{\boldsymbol{x}} (\boldsymbol{x}_{k+i} - \boldsymbol{r}_{k+i}) \\ J_{\boldsymbol{u}}(\boldsymbol{u}_{k+i}) &= \boldsymbol{u}_{k+i}^{\mathrm{T}} \boldsymbol{W}_{\boldsymbol{u}} \boldsymbol{u}_{k+i} + \Delta \boldsymbol{u}_{k+i}^{\mathrm{T}} \boldsymbol{W}_{\Delta \boldsymbol{u}} \Delta \boldsymbol{u}_{k+i} \\ J_f(\boldsymbol{x}_{k+H_p}) &= (\boldsymbol{x}_{k+H_p} - \boldsymbol{r}_{k+H_p})^{\mathrm{T}} \boldsymbol{W}_f (\boldsymbol{x}_{k+H_p} - \boldsymbol{r}_{k+H_p}), \end{aligned} \tag{2.16}$$

where the weight matrices $\boldsymbol{W}_{\boldsymbol{x}}$, $\boldsymbol{W}_{\boldsymbol{u}}$, $\boldsymbol{W}_{\Delta \boldsymbol{u}}$, and $\boldsymbol{W}_f$ are symmetric and positive semidefinite matrices, and where $\Delta \boldsymbol{u}_{k+1} = \boldsymbol{u}_{k+1} - \boldsymbol{u}_k$. The cost functions (2.16) correspond to the quadratic objective (2.10a), features 2 and 3 correspond to the linear inequality constraints (2.10b), and the dynamics (2.15) correspond to the equality constraints (2.10c).

We will make use of both linear and nonlinear MPC in an automotive application (Chapter 11), and linear MPC will be used in mobile robotics (Chapter 12).

[1] In the control literature, $\boldsymbol{B}$ usually denotes the input matrix. For consistence with notation in the rest of the thesis, we use $\boldsymbol{F}$ instead.

2.6 Summary

This chapter provided a short summary of the different control methods that will be used throughout the thesis. In addition, references to appropriate reading material were given in each section. The PID controller, which was described in Section 2.1, is a simple, yet powerful method to control different kinds of systems, and is the most wide-spread control technique in industry.

Force control can be implemented in various ways, but is frequently being done using impedance-based controllers, because it provides a way of describing the desired behavior in terms of physical entities. Section 2.2 gave a formulation of one common impedance controller.

We examined different optimization techniques in Sections 2.3–2.5. Dynamic optimization is becoming more and more widespread, both in academia and industry. An important special case of optimization is convex optimization, in which a locally optimal solution is also globally optimal. There are many different convex problem formulations. This chapter discussed quadratic programs in general and model predictive control in particular.

3

Estimation Using Particle Methods

The aim in state estimation is to estimate the state vector $\boldsymbol{x}$ given the measurement vector $\boldsymbol{y}$. Estimation as a field started to evolve during the second half of the 18th century. Gauss, Legendre, Bayes, and Laplace were some of the early contributors. Gauss and Legendre contributed to estimation with, for example, the least-squares solution [Gauss and Davis, 1857; Stigler, 1981]. Bayes and Laplace made contributions within Bayesian probability by, for example, formulating and developing what is now known as Bayes' theorem [Laplace, 1986]. A breakthrough on recursive estimation was given in [Kalman et al., 1960], where the Kalman filter was introduced as a means to employ recursive state estimation of linear, Gaussian state-space systems. The Kalman filter has been successfully applied to numerous types of systems and applications. It provides the minimum mean-square error estimate when the system is linear with Gaussian distributed noise sources.

The particle filter was introduced in [Gordon et al., 1993], and is considered a major contribution within nonlinear recursive estimation. Several ideas had been presented on how to estimate the states of nonlinear systems using extensions of the Kalman-filter framework—for example, the extended Kalman filter (EKF) and the unscented Kalman filter [Julier and Uhlmann, 2004]. However, the particle filter readily solves the estimation problem for nonlinear, non-Gaussian systems, at least in theory. One of the drawbacks with particle filters is increased computational complexity.

In the Bayesian view, the state estimation problem is solved by finding the probability density function $p(\boldsymbol{x}_{s:k}|\boldsymbol{y}_{0:T})$, where $\boldsymbol{x}_{s:k} := \{\boldsymbol{x}_s, \ldots, \boldsymbol{x}_k\}$ and $\boldsymbol{y}_{0:T} := \{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_T\}$. Depending on the values of s and T, three different estimation problems are obtained (Figure 3.1):

- If $s \leq k < T$ it is called the smoothing problem.

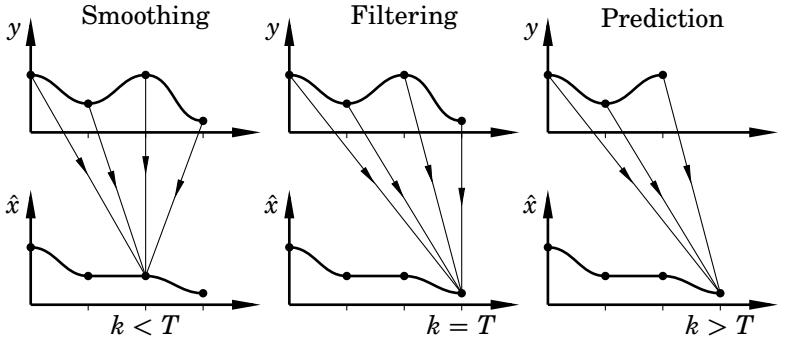

Figure 3.1 An illustration of smoothing, filtering, and prediction for $s = k$. The symbol $\hat{x}$ means the estimate of x.

- If $s = k = T$ it is called the filtering problem.
- If $k \geq s > T$ it is called the prediction problem.

This chapter treats the filtering and smoothing problems.

Consider a state-space model with the dynamics and measurement equation as

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k) \\ \boldsymbol{y}_k &= \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{e}_k, \end{aligned} \tag{3.1}$$

where $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$, $\boldsymbol{y}_k \in \mathbb{R}^{n_y}$, and $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$. Further, the noise processes $\boldsymbol{w}_k \in \mathbb{R}^{n_w}$ and $\boldsymbol{e}_k$ have known densities.[1] The time evolution of $\boldsymbol{x}_k$ is described by the possibly nonlinear function $\boldsymbol{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$ and $\boldsymbol{h} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ denotes the possibly nonlinear measurement function. If (3.1) is a linear model with white, additive, and Gaussian noise; that is, if (3.1) is equal to

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{F}\boldsymbol{u}_k + \boldsymbol{w}_k, \quad \boldsymbol{w}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k) \\ \boldsymbol{y}_k &= \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{e}_k, \quad \boldsymbol{e}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_k), \end{aligned} \tag{3.2}$$

then the Kalman filter is the optimal state estimator [Kalman et al., 1960; Anderson and Moore, 1979]. In (3.2), $\boldsymbol{Q}_k$ is the process-noise covariance matrix, $\boldsymbol{R}_k$ is the measurement-noise covariance matrix, and $\boldsymbol{e}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_k)$ implies that the measurement noise is a sample from the Gaussian distribution, which has zero mean and covariance $\boldsymbol{R}_k$. In this context, optimal means that the state estimates minimize the mean-square estimation error; that is, the state estimates solve

$$\text{minimize} \quad \mathcal{E}(\|\hat{\boldsymbol{x}}_{k|k} - \boldsymbol{x}_k\|^2),$$

[1] In Chapter 2 $\boldsymbol{w}$ denoted algebraic variables, which is not the case here.

where $\mathcal{E}(\boldsymbol{x})$ is the expected value of $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}_{k|k}$ is the state estimate at time index k given measurements up to time index k. The covariance matrix $\boldsymbol{P}_{k|k} \in \mathbb{R}^{n_x \times n_x}$ is associated with the estimate. The Kalman filter solves the estimation problem in a recursive manner, where the filtered estimate is found by correcting the one-step prediction with the measurement at each time instant. The algorithm is summarized in Algorithm 3.1.

Algorithm 3.1—The Kalman Filter

1: **Initialize:** Set $\hat{\boldsymbol{x}}_{0|-1} = \boldsymbol{x}_0$, $\boldsymbol{P}_{0|-1} = \boldsymbol{P}_0$.
2: **for** $k = 0$ **to** T **do**
3: Measurement update:

$$\begin{aligned} \hat{\boldsymbol{x}}_{k|k} &= \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{C}_k\boldsymbol{x}_{k|k-1}) \\ \boldsymbol{P}_{k|k} &= \boldsymbol{P}_{k|k-1} - \boldsymbol{K}_k\boldsymbol{S}_k\boldsymbol{K}_k^{\mathrm{T}} \\ \boldsymbol{S}_k &= \boldsymbol{C}_k\boldsymbol{P}_{k|k}(\boldsymbol{C}_k)^{\mathrm{T}} + \boldsymbol{R}_k \\ \boldsymbol{K}_k &= \boldsymbol{P}_{k|k-1}\boldsymbol{C}_k^{\mathrm{T}}(\boldsymbol{S}_k)^{-1}. \end{aligned} \tag{3.3}$$

4: Time update:

$$\begin{aligned} \hat{\boldsymbol{x}}_{k+1|k} &= \boldsymbol{A}\boldsymbol{x}_{k|k} + \boldsymbol{F}\boldsymbol{u}_k \\ \boldsymbol{P}_{k+1|k} &= \boldsymbol{A}_k\boldsymbol{P}_{k|k}\boldsymbol{A}_k^{\mathrm{T}} + \boldsymbol{Q}_k. \end{aligned} \tag{3.4}$$

5: **end for**

There are several solutions to the linear smoothing problem. One approach is to extend the Kalman filter with a state $\boldsymbol{x}_s$, $s < k$, and then execute the Kalman filter from time t_s to time t_k. This yields a fixed-point estimate, that is, the smoothed estimate at a given point in time. Another possibility is to use the Rauch-Tung-Striebel fixed-interval smoother [Rauch et al., 1965], which computes smoothed estimates over an interval.

Put in a Bayesian framework, (3.1) can be reformulated as

$$\boldsymbol{x}_{k+1} \sim p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{3.5a}$$

$$\boldsymbol{y}_k \sim p(\boldsymbol{y}_k|\boldsymbol{x}_k), \tag{3.5b}$$

where $\boldsymbol{x}_{k+1}$ and $\boldsymbol{y}_k$ are regarded as samples from the respective distribution. The Markov property holds for (3.5), which implies that

$$\begin{aligned} p(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{y}_{0:k-1}) &= p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}), \\ p(\boldsymbol{y}_k|\boldsymbol{x}_{0:k}, \boldsymbol{y}_{0:k-1}) &= p(\boldsymbol{y}_k|\boldsymbol{x}_k). \end{aligned}$$

In the following, we briefly discuss particle filters and particle smoothers. For notational brevity, $\boldsymbol{u}_k$ is omitted. Since $\boldsymbol{u}_k$ can be seen as the mean of the process noise, this omission does not cause any loss of generality. For more details, see, for example, [Arulampalam et al., 2002; Schön et al., 2005; Gustafsson, 2010b].

3.1 Particle Filtering

Particle filters approximate the posterior density $p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k})$,[2] and belong to the class of sequential Monte-Carlo methods, which represent the posterior density with a set of weighted particles [Doucet et al., 2000; Arulampalam et al., 2002]. The first papers on sequential Monte-Carlo methods came in the 1950's, see [Hammersley and Morton, 1954] for one example, but it was in 1993 that [Gordon et al., 1993] provided an implementation of what is now known as the particle filter. One of the novelties in that paper was the inclusion of the crucial resampling step, see a discussion on page 54. Since then it has found its use in numerous applications, see [Gustafsson, 2010a] for a plethora of them. Positioning and tracking applications have been motivators to drive the technological developments for particle filters, and still are [Gustafsson, 2010b].

One key ingredient in the derivation of recursive state estimators using probability density functions is Bayes' theorem (or Bayes' rule): for two variables x and y, it holds that

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}. \tag{3.6}$$

Consider the one-step prediction (prior) of the posterior density expressed in terms of the Chapman-Kolmogorov equation:

$$\begin{aligned} p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k-1}) &= \int p(\boldsymbol{x}_k, \boldsymbol{x}_{k-1}|\boldsymbol{y}_{0:k-1})\,\mathrm{d}\boldsymbol{x}_{k-1} \\ &= \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{y}_{0:k-1})p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{0:k-1})\,\mathrm{d}\boldsymbol{x}_{k-1} \\ &= \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{0:k-1})\,\mathrm{d}\boldsymbol{x}_{k-1}, \end{aligned} \tag{3.7}$$

where the Markov property was used for the third equality. The posterior filtering density

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k})$$

can be rewritten using Bayes' rule (3.6), which gives

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k}) = p(\boldsymbol{x}_k|\boldsymbol{y}_k, \boldsymbol{y}_{0:k-1}) = \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k-1})}{p(\boldsymbol{y}_k|\boldsymbol{y}_{0:k-1})}, \tag{3.8}$$

where

$$p(\boldsymbol{y}_k|\boldsymbol{y}_{0:k-1}) = \int p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k-1})\,\mathrm{d}\boldsymbol{x}_k$$

[2] More correct is to say that particle filters approximate $p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{0:k})$, but here only a simplified, marginalized case is considered. The marginalized case is the most common of the two in the application-oriented literature.

is a normalization constant. Thus, in (3.8), the measurement $\boldsymbol{y}_k$ is used for updating the prior density to obtain the posterior probability density function.

When (3.1) is linear with Gaussian noise sources—that is, when (3.1) is equal to (3.2)—the solutions to (3.7) and (3.8) are analytic and given by the Kalman filter. However, the integrals in (3.7) and (3.8) do not in general admit analytic solutions; therefore approximations, such as the particle filter, must be used. Assume that N particles (hypotheses) $\{\boldsymbol{x}_k^i\}_{i=1}^N$ are sampled from the one-step prior as

$$\boldsymbol{x}_k^i \sim p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k-1}). \tag{3.9}$$

The particle filter then approximates this density with

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k-1}) \approx \hat{p}(\boldsymbol{x}_k|\boldsymbol{y}_{0:k-1}) = \sum_{i=1}^{N} w_{k|k-1}^i \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^i). \tag{3.10}$$

In this context $\delta(\cdot)$ is the Dirac delta function, which fulfills

$$\int_{-\infty}^{\infty} \delta(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = 1, \quad \int_{-\infty}^{\infty} f(\boldsymbol{x})\delta(\boldsymbol{x} - \boldsymbol{x}_0)\,\mathrm{d}\boldsymbol{x} = f(\boldsymbol{x}_0)$$

for a function f. The importance weights $\{w_{k|k-1}^i\}_{i=1}^N$ indicate how likely each particle is, and fulfill

$$\sum_{i=1}^{N} w_{k|k-1}^i = 1, \quad w_{k|k-1}^i \geq 0, \quad \forall i \in \{1,\ldots,N\}.$$

Figure 3.2 shows an illustration of the particle-filter idea.

The weights in (3.10) can be chosen using importance sampling [Robert and Casella, 2004; Marshall, 1956] as follows: Often it is difficult to sample from (3.9) directly. Instead, let $q(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{y}_{0:k})$ be a *proposal density* (the importance density) from which it is straightforward to sample. Using the importance density, (3.7) is rewritten as

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k-1}) = \int q(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{y}_{0:k}) \frac{p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})}{q(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{y}_{0:k})} p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{0:k-1})\,\mathrm{d}\boldsymbol{x}_{k-1}. \tag{3.11}$$

By drawing N samples from the proposal density, that is,

$$\boldsymbol{x}_k^i \sim q(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{y}_{0:k}),$$

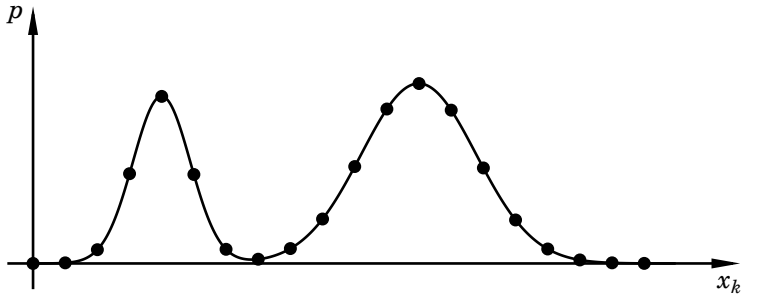

Figure 3.2 A probability density function p and its particle-filter approximation. The weights are indicated by the dots. Analogous to a continuous distribution, the weights sum to one.

the integral in (3.11) approximates to a sum according to

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k-1}) = \sum_{i=1}^{N} \underbrace{\frac{p(\boldsymbol{x}_k^i|\boldsymbol{x}_{k-1})}{q(\boldsymbol{x}_k^i|\boldsymbol{x}_{k-1},\boldsymbol{y}_{0:k})} p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{0:k-1})}_{w_{k|k-1}^i} \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^i). \tag{3.12}$$

The weights $w_{k|k-1}^i$ are defined as

$$w_{k|k-1}^i \propto \frac{p(\boldsymbol{x}_k^i|\boldsymbol{x}_{k-1})}{q(\boldsymbol{x}_k^i|\boldsymbol{x}_{k-1},\boldsymbol{y}_{0:k})} p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{0:k-1}), \tag{3.13}$$

where $\propto$ means proportional to. By inserting (3.12) into (3.8), the following is obtained:

$$\begin{aligned} p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k}) &= \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k-1})}{p(\boldsymbol{y}_k|\boldsymbol{y}_{0:k-1})} \\ &\approx \sum_{i=1}^{N} \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k^i)}{p(\boldsymbol{y}_k|\boldsymbol{y}_{0:k-1})} w_{k|k-1}^i \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^i) \\ &\propto \sum_{i=1}^{N} \underbrace{p(\boldsymbol{y}_k|\boldsymbol{x}_k^i) w_{k|k-1}^i}_{w_k^i} \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^i). \end{aligned}$$

The filtering posterior $p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k})$ is thus approximated as

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k}) \approx \hat{p}(\boldsymbol{x}_k|\boldsymbol{y}_{0:k}) = \sum_{i=1}^{N} w_k^i \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^i), \tag{3.14}$$

where the weights are given by

$$w_k^i \propto p(\boldsymbol{y}_k|\boldsymbol{x}_k^i)w_{k|k-1}^i. \tag{3.15}$$

One obvious choice of importance density is

$$q(\boldsymbol{x}_k^i|\boldsymbol{x}_{k-1}, \boldsymbol{y}_{0:k}) = p(\boldsymbol{x}_k^i|\boldsymbol{x}_{k-1}), \tag{3.16}$$

that is, the state dynamics (3.5a). With this choice of importance density, the weight update equation (3.15) reads

$$w_k^i \propto p(\boldsymbol{y}_k|\boldsymbol{x}_k^i)w_{k-1}^i, \tag{3.17}$$

since $w_{k|k-1} = w_{k-1}^i$ in (3.13). The importance density (3.16) was introduced in [Gordon et al., 1993]. There are, however, other alternatives; see [Arulampalam et al., 2002; Gustafsson, 2010b] for a few examples.

Resampling

A problem with the particle filter is the degeneracy (also referred to as depletion) phenomenon, which means that after a few iterations all but very few particles will have negligible weights. Thus, particles that do not contribute to the posterior approximation are updated in each step, which is inefficient. To remedy this, the particles are resampled. The idea of resampling is to replace particles that have negligible weights with particles that have larger weights. More specifically, resampling involves generating a new set of particles $\{\boldsymbol{x}_k^i\}_{i=1}^N$ from the posterior approximation in such a way that the probability P of generating $\boldsymbol{x}_k^i$ is w_k^i:

$$\mathrm{P}(\boldsymbol{x}_k^i = \boldsymbol{x}_k^j) = w_k^j, \quad \forall i \in \{1, \ldots, N\}. \tag{3.18}$$

The resampling step (3.18) was introduced in [Gordon et al., 1993], and makes the particle filter practically useful. The resampling step greatly reduces the effects of degeneracy, but introduces other negative effects, such as sample impoverishment; thus, particle diversity is lost, which implies that the same particle can be selected many times in the resampling step. The degeneracy effect is often more significant for small process noise. Consequently, resampling is often only performed when the degeneracy is large enough. A measure of degeneracy is the effective sample size, which commonly is approximated by

$$N_{\text{eff}} = \left(\sum_{i=1}^{N} (w_k^i)^2 \right)^{-1}. \tag{3.19}$$

When (3.19) drops below a given threshold N_{thr}, the particles are resampled and the weights are equalized as

$$w_k^i = \frac{1}{N}, \quad \forall i \in \{1, \ldots, N\}.$$

For a discussion on resampling algorithms, see [Arulampalam et al., 2002; Douc, 2005; Hol et al., 2006; Gustafsson, 2010b]

The Algorithm

Algorithm 3.2 provides a version of the particle filter that uses resampling in each time step. There are many refinements of it, but the basic steps remain the same.

Algorithm 3.2—The Particle Filter

1: **Initialize:** Set $\{\boldsymbol{x}_0^i\}_{i=1}^N \sim p_0(x_0)$ and weights $\{w_0^i\}_{i=1}^N$.
2: **for** $k = 0$ **to** T **do**
3: Time update: Generate new particles from (3.5a):

$$\boldsymbol{x}_{k+1}^i \sim p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k^i), \quad \forall i \in \{1, \ldots, N\}.$$

4: Measurement update: Compute weights as in (3.17) and normalize them.
5: Resample according to (3.18).
6: **end for**

3.2 Rao-Blackwellized Particle Filtering

To decrease the required number of particles and the variance of the estimates, and thus to reduce computational complexity, it is advantageous to exploit model structure. This is the idea behind *Rao-Blackwellization*, where the subset of the state space that allows for analytic expressions is marginalized out. Thus, the sampled state space is smaller and it should therefore be possible to use fewer particles [Doucet et al., 2000; Schön et al., 2005]. The term Rao-Blackwellization stems from the Rao-Blackwell theorem [Blackwell, 1947]. The theorem essentially states that if $\hat{\boldsymbol{x}}$ is an unbiased estimate of $\boldsymbol{x}$ and if g is the sufficient statistics,[3] then the expected value of $\hat{\boldsymbol{x}}$ given g, $\mathcal{E}(\hat{\boldsymbol{x}}|g)$, is an unbiased estimate with smaller variance than $\hat{\boldsymbol{x}}$. This transformed estimator is called a Rao-Blackwellized estimator.

[3] As an example, the sufficient statistics for the Gaussian distribution is the mean and covariance.

Similar to particle filters, Rao-Blackwellized particle filters (RBPFs)[4] have found their use in many applications—for example, in aircraft tracking [Schön et al., 2005]; aerial vehicles [Törnqvist et al., 2009]; terrain-aided positioning, radar-based target tracking, bearings-only target tracking, automotive target tracking [Schön et al., 2006]; and simultaneous localization and mapping (SLAM) [Hähnel et al., 2003; Grisetti et al., 2005; Grisetti et al., 2007] in robotics. The RBPFs have become useful because for a lot of applications it is possible to model a major part of the states as linear. Sometimes the RBPF is the only feasible alternative; the number of particles needed to cover the state space grows exponentially with the number of states in many particle-filter implementations [Beskos et al., 2011; Rebeschini and van Handel, 2013].[5]

The details and derivations leading up to the RBPF as we will use it are found in [Schön et al., 2005] and a complexity analysis is given in [Karlsson et al., 2005]. The remainder of this section gives the main ideas behind the RBPF. Assume that the state vector $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ can be partitioned into a linear part $\boldsymbol{z}_k \in \mathbb{R}^{n_z}$ and a nonlinear part $\boldsymbol{\eta}_k \in \mathbb{R}^{n_\eta}$ as $\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{z}_k^{\mathrm{T}} & \boldsymbol{\eta}_k^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$. Further assume that the dynamics (3.1) can be written as

$$\boldsymbol{z}_{k+1} = \boldsymbol{f}(\boldsymbol{\eta}_k) + \boldsymbol{A}(\boldsymbol{\eta}_k)\boldsymbol{z}_k + \boldsymbol{F}(\boldsymbol{\eta}_k)\boldsymbol{w}_k^{\boldsymbol{z}}, \tag{3.20a}$$

$$\boldsymbol{\eta}_{k+1} = \boldsymbol{g}(\boldsymbol{\eta}_k) + \boldsymbol{B}(\boldsymbol{\eta}_k)\boldsymbol{z}_k + \boldsymbol{G}(\boldsymbol{\eta}_k)\boldsymbol{w}_k^{\boldsymbol{\eta}}, \tag{3.20b}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{\eta}_k) + \boldsymbol{C}(\boldsymbol{\eta}_k)\boldsymbol{z}_k + \boldsymbol{e}_k. \tag{3.20c}$$

This model class is denoted mixed-Gaussian state space (MGSS) and is a model class that covers a set of other model classes where marginalization can be used. The process noise $\boldsymbol{w}_k = \begin{bmatrix} (\boldsymbol{w}_k^{\boldsymbol{z}})^{\mathrm{T}} & (\boldsymbol{w}_k^{\boldsymbol{\eta}})^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ is assumed white and Gaussian distributed according to

$$\boldsymbol{w}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k), \quad \boldsymbol{Q}_k = \begin{bmatrix} \boldsymbol{Q}_k^{\boldsymbol{z}} & \boldsymbol{Q}_k^{\boldsymbol{z}\boldsymbol{\eta}} \\ (\boldsymbol{Q}_k^{\boldsymbol{z}\boldsymbol{\eta}})^{\mathrm{T}} & \boldsymbol{Q}_k^{\boldsymbol{\eta}} \end{bmatrix}. \tag{3.21}$$

The process noise and the measurement noise are assumed mutually independent. Furthermore, $\boldsymbol{f}(\boldsymbol{\eta}_k)$, $\boldsymbol{g}(\boldsymbol{\eta}_k)$, $\boldsymbol{h}(\boldsymbol{\eta}_k)$; the system matrices $\boldsymbol{A}(\boldsymbol{\eta}_k)$, $\boldsymbol{B}(\boldsymbol{\eta}_k)$, $\boldsymbol{C}(\boldsymbol{\eta}_k)$; and $\boldsymbol{F}(\boldsymbol{\eta}_k)$, $\boldsymbol{G}(\boldsymbol{\eta}_k)$ all have a possibly nonlinear dependence on $\boldsymbol{\eta}_k$. This dependence is implicit in the following. The measurement noise is assumed white and Gaussian distributed as

$$\boldsymbol{e}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_k).$$

[4] Sometimes also referred to as marginalized particle filters.

[5] This is known as the curse of dimensionality.

The enabler for the RBPF is the key factorization

$$p(\boldsymbol{z}_k, \boldsymbol{\eta}_{0:k}|\boldsymbol{y}_{0:k}) = p(\boldsymbol{z}_k|\boldsymbol{\eta}_{0:k}, \boldsymbol{y}_{0:k})p(\boldsymbol{\eta}_{0:k}|\boldsymbol{y}_{0:k}). \tag{3.22}$$

The factorization in (3.22) allows for first estimating the trajectory $\boldsymbol{\eta}_{0:k}$ with a particle filter (3.14) and then computing $\boldsymbol{z}_k$ analytically with a Kalman filter [Kalman et al., 1960; Anderson and Moore, 1979] for each trajectory $\{\boldsymbol{\eta}_{0:k}\}_{i=1}^N$. It is possible to use a Kalman filter because (3.20a) and (3.20c) combined describe a linear Gaussian system given $\boldsymbol{\eta}_{0:k}$. This implies that

$$p(\boldsymbol{z}_k|\boldsymbol{\eta}_{0:k}, \boldsymbol{y}_{0:k}) = \mathcal{N}(\boldsymbol{z}_k|\hat{\boldsymbol{z}}_{k|k}, \boldsymbol{P}_{k|k}), \tag{3.23}$$

where $\mathcal{N}(\boldsymbol{z}_k|\hat{\boldsymbol{z}}_{k|k}, \boldsymbol{P}_{k|k})$ means the Gaussian density given mean $\hat{\boldsymbol{z}}_{k|k}$ and covariance matrix $\boldsymbol{P}_{k|k}$, $\hat{\boldsymbol{z}}_{k|k} := \hat{\boldsymbol{z}}_{k|k}(\boldsymbol{\eta}_{0:k})$ is the estimate of $\boldsymbol{z}_k$ given the measurements $\boldsymbol{y}_{0:k}$ and the nonlinear trajectory up to time index k, and $\boldsymbol{P}_{k|k} := \boldsymbol{P}_{k|k}(\boldsymbol{\eta}_{0:k})$ is its associated covariance. By combining (3.14) and (3.23), an approximation $\hat{p}(\boldsymbol{z}_k, \boldsymbol{\eta}_{0:k}|\boldsymbol{y}_{0:k})$ to (3.22) is given by the Gaussian mixture

$$\hat{p}(\boldsymbol{z}_k, \boldsymbol{\eta}_{0:k}|\boldsymbol{y}_{0:k}) = \sum_{i=1}^{N} w_k^i \mathcal{N}(\boldsymbol{z}_k|\hat{\boldsymbol{z}}_{k|k}^i, \boldsymbol{P}_{k|k}^i)\delta(\boldsymbol{\eta}_{0:k} - \boldsymbol{\eta}_{0:k}^i),$$

where the weight update resembles that of the standard particle filter. The RBPF is summarized in Algorithm 3.3 in the case when $\boldsymbol{F}_k$ and $\boldsymbol{G}_k$ both are identity matrices. On line 1 in Algorithm 3.3, p_0 and $\boldsymbol{P}_0$ mean the initial distribution and initial covariance, respectively.

Algorithm 3.3—Rao-Blackwellized Particle Filter

1: **Initialize:** Generate $\{\boldsymbol{\eta}_0^i\}_{i=1}^N \sim p_0(\boldsymbol{\eta}_0)$, $\{\boldsymbol{z}_{0|-1}^i\}_{i=1}^N \sim p_0(\boldsymbol{z}_0)$ and for all $i \in \{1, \ldots, N\}$, set $w_{-1}^i = 1/N$, $\boldsymbol{P}_{0|-1}^i = \boldsymbol{P}_0$.

2: **for** $k = 0$ **to** T **do**

3: Particle-filter measurement update: Evaluate the importance weights as

$$\bar{w}_k^i \propto p(\boldsymbol{y}_k|\boldsymbol{\eta}_{0:k}^i, \boldsymbol{y}_{0:k-1})w_{k-1}^i, \quad \forall i \in \{1, \ldots, N\}, \tag{3.24}$$

and normalize according to

$$w_k^i = \frac{\bar{w}_k^i}{\sum_{i=1}^N \bar{w}_k^i}.$$

4: **if** $\left(\sum_{i=1}^{N}(w_k^i)^2\right)^{-1} < N_{\text{thr}}$ **then**

5: Resampling: Resample (with replacement) N particles $\{\boldsymbol{\eta}_k\}_{i=1}^N$ with probabilities $\{w_k^i\}$, reinitialize weights as $w_k^i = 1/N$.

6: **end if**

7: Kalman-filter measurement update:

$$\begin{aligned}
\hat{\boldsymbol{z}}_{k|k}^i &= \hat{\boldsymbol{z}}_{k|k-1}^i + \boldsymbol{K}_k^i(\boldsymbol{y}_k - \boldsymbol{h}_k^i - \boldsymbol{C}_k^i \boldsymbol{z}_{k|k-1}^i) \\
\boldsymbol{P}_{k|k}^i &= \boldsymbol{P}_{k|k-1}^i - \boldsymbol{K}_k^i \boldsymbol{S}_k^i (\boldsymbol{K}_k^i)^{\mathrm{T}} \\
\boldsymbol{S}_k^i &= \boldsymbol{C}_k^i \boldsymbol{P}_{k|k}^i (\boldsymbol{C}_k^i)^{\mathrm{T}} + \boldsymbol{R}_k \\
\boldsymbol{K}_k^i &= \boldsymbol{P}_{k|k-1}^i (\boldsymbol{C}_k^i)^{\mathrm{T}} (\boldsymbol{S}_k^i)^{-1}.
\end{aligned} \tag{3.25}$$

8: Particle-filter time update: Predict new particles,

$$\boldsymbol{\eta}_{k+1}^i \sim p(\boldsymbol{\eta}_{k+1} | \boldsymbol{\eta}_{0:k}^i, \boldsymbol{y}_{0:k}), \quad \forall i \in \{1, \ldots, N\}. \tag{3.26}$$

9: Kalman-filter time update:

$$\begin{aligned}
\hat{\boldsymbol{z}}_{k+1|k}^i &= \boldsymbol{f}_k^i + (\boldsymbol{Q}_k^{\boldsymbol{z}\boldsymbol{\eta}})^{\mathrm{T}} (\boldsymbol{Q}_k^{\boldsymbol{\eta}})^{-1} (\boldsymbol{\eta}_{k+1}^i - \boldsymbol{g}_k^i) + \bar{\boldsymbol{A}}_k^i \bar{\boldsymbol{z}}_{k|k}^i \\
\boldsymbol{P}_{k+1|k}^i &= \bar{\boldsymbol{A}}_k^i \bar{\boldsymbol{P}}_{k|k}^i (\bar{\boldsymbol{A}}_k^i)^{\mathrm{T}} + \boldsymbol{Q}_k^{\boldsymbol{z}} - (\boldsymbol{Q}_k^{\boldsymbol{z}\boldsymbol{\eta}})^{\mathrm{T}} (\boldsymbol{Q}_k^{\boldsymbol{\eta}})^{-1} \boldsymbol{Q}^{\boldsymbol{z}\boldsymbol{\eta}} \\
\bar{\boldsymbol{A}}_k^i &= \boldsymbol{A}_k^i - (\boldsymbol{Q}_k^{\boldsymbol{z}\boldsymbol{\eta}})^{\mathrm{T}} (\boldsymbol{Q}_k^{\boldsymbol{\eta}})^{-1} \boldsymbol{B}_k^i \\
\bar{\boldsymbol{z}}_{k|k}^i &= \hat{\boldsymbol{z}}_{k|k}^i + \boldsymbol{L}_k^i (\boldsymbol{\eta}_{k+1}^i - \boldsymbol{g}_k^i - \boldsymbol{B}_k^i \hat{\boldsymbol{z}}_{k|k}^i) \\
\bar{\boldsymbol{P}}_{k|k}^i &= \boldsymbol{P}_{k|k}^i - \boldsymbol{L}_k^i \boldsymbol{M}_k^i (\boldsymbol{L}_k^i)^{\mathrm{T}} \\
\boldsymbol{L}_k^i &= \boldsymbol{P}_{k|k}^i (\boldsymbol{B}_k^i)^{\mathrm{T}} (\boldsymbol{M}_k^i)^{-1} \\
\boldsymbol{M}_k^i &= \boldsymbol{B}_k^i \boldsymbol{P}_{k|k}^i (\boldsymbol{B}_k^i)^{\mathrm{T}} + \boldsymbol{Q}_k^{\boldsymbol{\eta}}.
\end{aligned} \tag{3.27}$$

10: **end for**

The densities in (3.24) and (3.26) are given by

$$\begin{aligned}
p(\boldsymbol{y}_k | \boldsymbol{\eta}_{0:k}, \boldsymbol{y}_{0:k-1}) &= \mathcal{N}(\boldsymbol{y}_k | \boldsymbol{h}_k + \boldsymbol{C}_k \hat{\boldsymbol{z}}_{k|k-1}, \boldsymbol{C}_k \boldsymbol{P}_{k|k-1} \boldsymbol{C}_k^{\mathrm{T}} + \boldsymbol{R}_k), \\
p(\boldsymbol{\eta}_{k+1} | \boldsymbol{\eta}_{0:k}, \boldsymbol{y}_{0:k}) &= \mathcal{N}(\boldsymbol{\eta}_{k+1} | \boldsymbol{g}_k + \boldsymbol{B}_k \hat{\boldsymbol{z}}_{k|k}, \boldsymbol{B}_k \boldsymbol{P}_{k|k} \boldsymbol{B}_k^{\mathrm{T}} + \boldsymbol{Q}_k^{\boldsymbol{\eta}}).
\end{aligned}$$

For easier access to the theory, Listing B.1 on page 327 provides a MATLAB implementation of Algorithm 3.3. The code is used for a four-dimensional problem in Example 3.1.

EXAMPLE 3.1—RAO-BLACKWELLIZED PARTICLE FILTER IMPLEMENTATION
The system is described by the fourth-order system

$$\begin{aligned}
\boldsymbol{z}_{k+1} &= \begin{bmatrix} 1 & 0.3 & 0 \\ 0 & 0.92 & -0.3 \\ 0 & 0.3 & 0.92 \end{bmatrix} \boldsymbol{z}_k + \boldsymbol{w}_k^{\boldsymbol{z}}, \\
\eta_{k+1} &= \arctan(\eta_k) + \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \boldsymbol{z}_k + w_k^{\eta}, \\
\boldsymbol{y}_k &= \begin{bmatrix} 0.1\eta_k^2 \operatorname{sign}(\eta_k) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 1 & -1 & 1 \end{bmatrix} \boldsymbol{z}_k + \boldsymbol{e}_k,
\end{aligned}$$

where sign is the signum function. The noise sources are mutually independent, white, and Gaussian distributed according to $\boldsymbol{w}_k \sim \mathcal{N}(\boldsymbol{0}, 0.01\boldsymbol{I}_{4\times4})$ and $\boldsymbol{e}_k \sim \mathcal{N}(\boldsymbol{0}, 0.1\boldsymbol{I}_{2\times2})$. This model has previously been used in [Lindsten and Schön, 2010].

First, we define the system parameters. Then simulated data is generated, which is followed by executing the RBPF. The MATLAB code for the example is given next, and the implementation of the filter (line 27 in the code) is found in Listing B.1 on page 327.

```
1  A = [1 0.3 0;...
2      0 0.92 -0.3;...
3      0 0.3 0.92];      %System matrix for linear states
4  B = [1 0 0];          %System matrix for nonlinear state
5  C = [0 0 0; 1 -1 1]; %Measurement matrix
6  R = 0.1*eye(2);       %Measurement noise covariance
7  Q = 0.01*eye(4);      %Process covariance
8  P0 = 1e-6*eye(4);     %initial covariance
9  Tfinal = 50;          %Number of time steps
10 x = zeros(4,Tfinal+1);
11 y = zeros(2,Tfinal);
12 x(:,1) = sqrtm(P0)*randn(4,1);
13
14 for t=1:Tfinal        %Generate data
15     x(:,t+1) = [zeros(3,1);atan(x(4,t))]...
16         + [A;B]*x(1:3,t)+sqrtm(Q)*randn(4,1);
17
18     y(:,t)   = [0.1*x(4,t)^2*sign(x(4,t));0]...
19         +C*x(1:3,t)+sqrtm(R)*randn(2,1);
20 end
21
22 N = 500;              %Number of particles
23 xp = sqrtm(P0)*randn(4,N);
24 P = repmat(P0(1:3,1:3),[1 1 N]);
25 w = ones(1,N)/N;
26 %% Execute RBPF
27 [xmean,xparts] = RBPF(N,y,xp,w,P,A,B,C,R,Q,Tfinal);
```

□

Figure 3.3 shows results that are generated from executing Example 3.1 once. The corresponding estimates of the distributions for the nonlinear state are displayed in Figure 3.4. The estimated distributions have large resemblence with a Gaussian distribution, but differ in their support (the length of the tails differ).

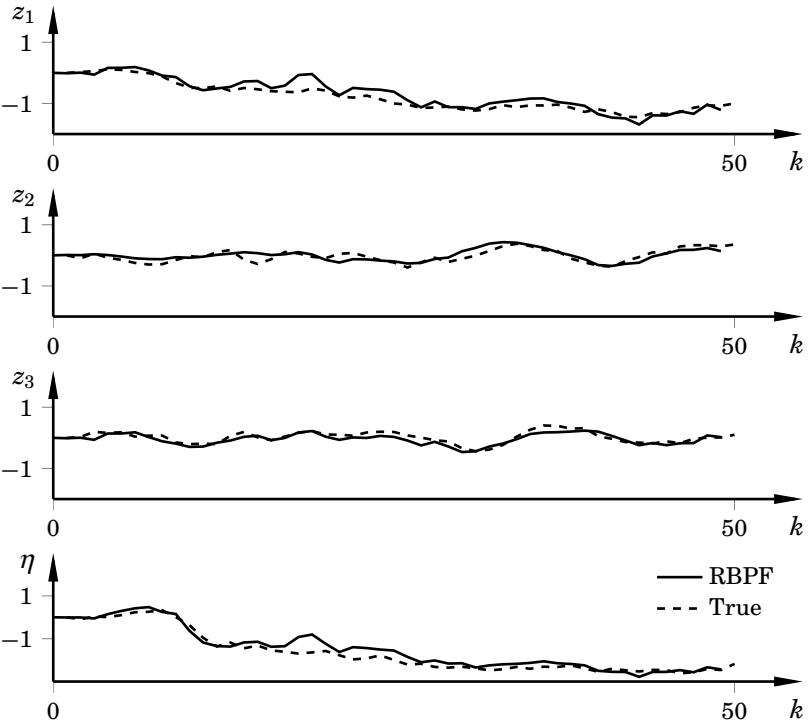

Figure 3.3 True states and estimated states for Example 3.1. The estimates were generated by the code in Listing B.1, corresponding to Algorithm 3.3.

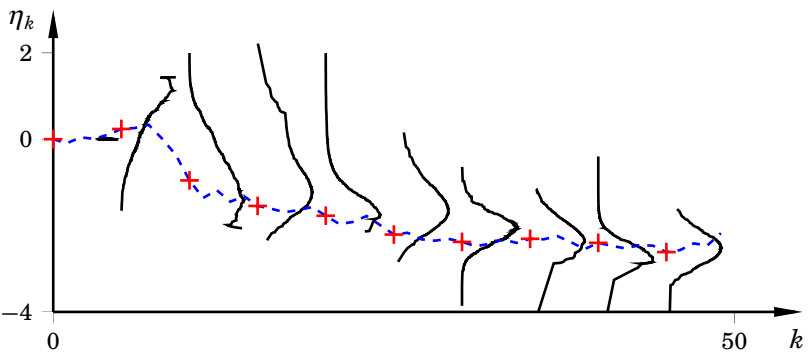

Figure 3.4 Estimated distribution of the nonlinear state (black) at every fifth time step, corresponding to the results in Figure 3.3. The red + correspond to the true states and the dashed line is the resulting trajectory.

3.3 Particle Smoothing

Particle smoothers form approximate solutions to the smoothing density $p(\boldsymbol{x}_{0:T}|\boldsymbol{y}_{0:T})$. Note that the fixed-point, $p(\boldsymbol{x}_k|\boldsymbol{y}_{0:T})$, and fixed-interval, $p(\boldsymbol{x}_{s:k}|\boldsymbol{y}_{0:T})$, smoothing densities can be found by marginalization. If whole trajectories are stored and resampled, the particle filter in (3.14) can also be used for approximating the smoothing density. However, because of the inherent depletion problem in particle filters, this estimate is often degenerate for large time lags. Hence particle smoothing is frequently approached with other algorithms, such as the forward filter/backward simulator (FFBS) [Godsill et al., 2004; Schön et al., 2005] or the two-filter formula [Kitagawa, 1996]. The FFBS has obtained its name from that it uses a particle filter in the forward direction and then iterates backward in time using the filtered estimates. For more details about backward-simulation methods, see [Lindsten and Schön, 2013]. The FFBS utilizes the sequential factorization

$$p(\boldsymbol{x}_{0:T}|\boldsymbol{y}_{0:T}) = p(\boldsymbol{x}_T|\boldsymbol{y}_{0:T}) \prod_{k=0}^{T-1} p(\boldsymbol{x}_k|\boldsymbol{x}_{k+1:T}, \boldsymbol{y}_{0:T}). \tag{3.28}$$

It starts by sampling a state $\boldsymbol{x}'_T = \boldsymbol{x}^i_T$ with probability w^i_T from the filtering approximation $p(\boldsymbol{x}_T|\boldsymbol{y}_{0:T})$ at time index T. Then the use of Bayes' rule and the Markov property inherent in the process model (3.5a) gives that

$$p(\boldsymbol{x}_k|\boldsymbol{x}_{k+1:T}, \boldsymbol{y}_{0:T}) \propto p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k}), \tag{3.29}$$

which results in

$$p(\boldsymbol{x}_{T-1}|\boldsymbol{x}'_T, \boldsymbol{y}_{0:T}) \approx \sum_{i=1}^{N} w^i_{T-1|T} \delta\left(\boldsymbol{x}_{0:T-1} - \boldsymbol{x}^i_{0:T-1}\right),$$

where

$$w^i_{T-1|T} \propto w^i_{T-1} p(\boldsymbol{x}'_T|\boldsymbol{x}^i_{T-1}). \tag{3.30}$$

At the second time step, $\boldsymbol{x}'_{T-1} = \boldsymbol{x}^i_{T-1}$ is sampled from the filtering distribution with probability $w^i_{T-1|T}$. Thus, the particle $\boldsymbol{x}'_T$, which was chosen in the previous time step, is concatenated with $\boldsymbol{x}'_{T-1}$ to form the trajectory $\{\boldsymbol{x}'_{T-1}, \boldsymbol{x}'_T\}$. This recursion is performed until $k = 0$ is reached, whereby an approximation to (3.28) can be formed using the trajectory $\boldsymbol{x}'_{0:T}$. For a more diverse approximation, the algorithm is repeated M times to yield the approximation $\hat{p}(\boldsymbol{x}_{0:T}|\boldsymbol{y}_{0:T})$ to the smoothing density as

$$\hat{p}(\boldsymbol{x}_{0:T}|\boldsymbol{y}_{0:T}) = \frac{1}{M} \sum_{j=1}^{M} \delta\left(\boldsymbol{x}_{0:T} - \boldsymbol{x}^j_{0:T}\right). \tag{3.31}$$

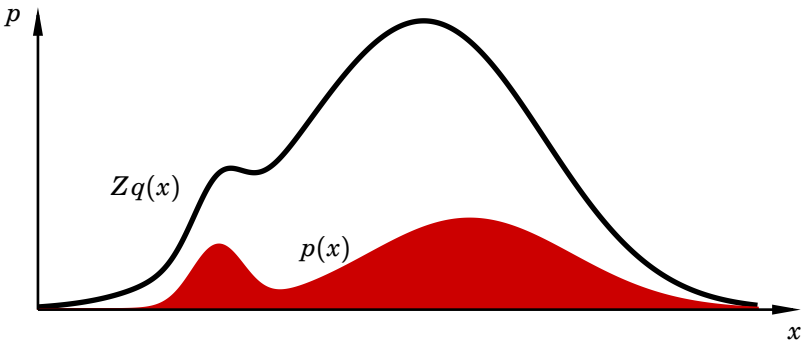

Figure 3.5 An illustration of rejection sampling. Samples are drawn from the area below the black curve. If they are deemed to be samples from the red area, they are accepted. Otherwise, they are rejected.

Alternative Sampling Methods

As presented here, the FFBS assumes that all smoothing weights (3.30) are computed in each time step. This is computationally heavy. There are sampling methods that can be used for decreasing the computational complexity, such as *acceptance-rejection* and *Metropolis-Hastings* sampling methods [Neumann, 1951; Chib and Greenberg, 1995; Gustafsson, 2010b]. Instead of sampling directly from the smoothing weights, these methods are based on sampling from alternative densities. Similarly to importance sampling, which is used in the particle-filter derivation in this chapter, the alternative densities are denoted by proposal distributions.

Rejection sampling assumes that the original density $p(\boldsymbol{x})$ can be bounded by a user-chosen proposal distribution $q(\boldsymbol{x})$ as $p(\boldsymbol{x}) < Zq(\boldsymbol{x})$, $Z > 1$, for all $\boldsymbol{x}$, where Z is a constant. The idea is to sample from the area below $Zq(\boldsymbol{x})$, $\boldsymbol{x}' \sim Zq(\boldsymbol{x})$, and accept $\boldsymbol{x}'$ as a sample from the distribution $p(\boldsymbol{x})$ if $Zq(\boldsymbol{x}')$ is deemed to be below the graph of $p(\boldsymbol{x})$. Otherwise the sample is rejected. Figure 3.5 sketches the idea. The sample is accepted if $Zq(\boldsymbol{x}')n \leq p(\boldsymbol{x}')$, where n is a number between zero and one, drawn from a uniform distribution. If q is poorly chosen, there will be many rejections, thus resulting in large computation time.

3.4 Rao-Blackwellized Particle Smoothing

It is beneficial for smoothing performance to utilize model structure, similar to the filtering case. Compared with filtering, smoothing is not as straightforward. The reason is that the Markovian property (3.29) is lost; that is, the linear part of the state vector (thus also the measurement

likelihoods) depends on the entire nonlinear trajectory. Whole trajectories must therefore be generated to preserve Gaussianity. Some early work on Rao-Blackwellized particle smoothers (RBPS) is found in [Fong et al., 2002]. In [Särkkä et al., 2012], an RBPS was derived for a class of state-space models where the nonlinear part does not depend on the linear part. Another smoother is found in [Lindsten and Schön, 2011], where both the linear and nonlinear part are sampled backward in time.

An alternative, improved RBPS was derived in [Lindsten et al., 2013]. This RBPS seeks the solution to the density $p(\boldsymbol{z}_{0:T}, \boldsymbol{\eta}_{0:T}|\boldsymbol{y}_{0:T})$ for the model class (3.20). The density at time step $k < T$ is approximated by

$$\hat{p}(\boldsymbol{z}_k, \boldsymbol{\eta}_k|\boldsymbol{y}_{0:T}) = \frac{1}{M}\sum_{j=1}^{M} \mathcal{N}(\boldsymbol{z}_k|\hat{\boldsymbol{z}}^j_{k|T}, \boldsymbol{P}^j_{k|T})\delta(\boldsymbol{\eta}_k - \boldsymbol{\eta}^j_k), \tag{3.32}$$

where $\boldsymbol{\eta}^j_k$ is the component of the nonlinear trajectory at time index k and where $\boldsymbol{z}^j_{k|T}$ is conditioned on the nonlinear backward trajectory.

Assume that the trajectory $\boldsymbol{x}'_{k+1:T}$ is given. To compute (3.32), the RBPS in [Lindsten et al., 2013] draws one of the RBPF particles $\{\boldsymbol{\eta}^i_{0:k}\}_{i=1}^N$ with probability $w^i_{k|T}$. By discarding the forward trajectory $\boldsymbol{\eta}^i_{0:k-1}$ the backward trajectory is extended with $\boldsymbol{\eta}^i_k$, yielding the trajectory $\{\boldsymbol{\eta}^i_k, \boldsymbol{\eta}'_{k+1:T}\}$. This procedure is repeated for each time step $k = T-1, \ldots, 0$, resulting in a backward trajectory that can be used for approximating $p(\boldsymbol{\eta}_{0:T}|\boldsymbol{y}_{0:T})$.

The smoothing weights are found by using Bayes rule on the left-hand side of (3.29), which gives that

$$p(\boldsymbol{\eta}_{0:k}|\boldsymbol{\eta}'_{k+1:T}, \boldsymbol{y}_{0:T}) \propto p(\boldsymbol{y}_{k+1:T}, \boldsymbol{\eta}'_{k+1:T}|\boldsymbol{\eta}_{0:k}, \boldsymbol{y}_{0:k})p(\boldsymbol{\eta}_{0:k}|\boldsymbol{y}_{0:k}). \tag{3.33}$$

The second factor on the right-hand side in (3.33) is approximated by the particle-filter part of the RBPF. This results in that (3.33) transforms to

$$\begin{aligned} p(\boldsymbol{\eta}_{0:k}|\boldsymbol{\eta}'_{k+1:T}, \boldsymbol{y}_{0:T}) &\propto p(\boldsymbol{y}_{k+1:T}, \boldsymbol{\eta}'_{k+1:T}|\boldsymbol{\eta}_{0:k}, \boldsymbol{y}_{0:k})p(\boldsymbol{\eta}_{0:k}|\boldsymbol{y}_{0:k}) \\ &\approx \sum_{i=1}^{N} \underbrace{p(\boldsymbol{y}_{k+1:T}, \boldsymbol{\eta}'_{k+1:T}|\boldsymbol{\eta}^i_{0:k}, \boldsymbol{y}_{0:k})w^i_k}_{w^i_{k|T}} \delta\left(\boldsymbol{\eta}_{0:k} - \boldsymbol{\eta}^i_{0:k}\right) \\ &= \sum_{i=1}^{N} w^i_{k|T}\delta\left(\boldsymbol{\eta}_{0:k} - \boldsymbol{\eta}^i_{0:k}\right), \end{aligned}$$

where the smoothing weights were introduced as

$$w^i_{k|T} \propto w^i_k p(\boldsymbol{y}_{k+1:T}, \boldsymbol{\eta}'_{k+1:T}|\boldsymbol{\eta}^i_{0:k}, \boldsymbol{y}_{0:k}). \tag{3.34}$$

The predictive density in (3.34) is expressed as

$$p(\boldsymbol{y}_{k+1:T}, \boldsymbol{\eta}'_{k+1:T} | \boldsymbol{\eta}^i_{0:k}, \boldsymbol{y}_{0:k}) = \int p(\boldsymbol{y}_{k+1:T}, \boldsymbol{\eta}'_{k+1:T} | \boldsymbol{z}^i_k, \boldsymbol{\eta}^i_k) p(\boldsymbol{z}^i_k | \boldsymbol{\eta}^i_{0:k}, \boldsymbol{y}_{0:k}) \, \mathrm{d}\boldsymbol{z}^i_k, \quad (3.35)$$

where the second factor on the right-hand side is given by the Kalman-filter part of the RBPF—that is, (3.23). The density $p(\boldsymbol{y}_{k+1:T}, \boldsymbol{\eta}'_{k+1:T} | \boldsymbol{z}^i_k, \boldsymbol{\eta}^i_k)$ in (3.35) is found by propagating zero, first, and second order moments $\{Z_k, \boldsymbol{\lambda}_k, \boldsymbol{\Omega}_k\}$, dependent on $\boldsymbol{\eta}^i_k$ but independent of $\boldsymbol{z}^i_k$, backward in time as the nonlinear trajectory is drawn. Given $\{Z_k, \boldsymbol{\lambda}_k, \boldsymbol{\Omega}_k\}$, the predictive density in (3.34) is given by

$$p(\boldsymbol{y}_{k+1:T}, \boldsymbol{\eta}'_{k+1:T} | \boldsymbol{z}^i_k, \boldsymbol{\eta}^i_k) \propto Z_k \exp\left(-\frac{1}{2} \left((\boldsymbol{z}^i_k)^{\mathrm{T}} \Omega_k \boldsymbol{z}^i_k - 2\boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{z}^i_k \right) \right).$$

Marginalizing out $\boldsymbol{z}^i_k$ gives the sought density (3.35) as

$$p(\boldsymbol{y}_{k+1:T}, \boldsymbol{\eta}'_{k+1:T} | \boldsymbol{\eta}^i_{0:k}, \boldsymbol{y}_{0:k}) \propto Z_k \det(\boldsymbol{\Lambda}_k)^{-1/2} \exp\left(-\frac{1}{2} \zeta_k \right), \quad (3.36)$$

In (3.36), $\det(\boldsymbol{\Lambda}_k)$ is the determinant of $\boldsymbol{\Lambda}_k$ and

$$\begin{aligned} \zeta_k &= \|\hat{\boldsymbol{z}}_{k|k}\|^2_{\boldsymbol{\Omega}_k} - 2\boldsymbol{\lambda}_k^{\mathrm{T}} \hat{\boldsymbol{z}}_{k|k} - \|\boldsymbol{\Gamma}_k^{\mathrm{T}} (\boldsymbol{\lambda}_k - \boldsymbol{\Omega}_k \hat{\boldsymbol{z}}_{k|k})\|^2_{\boldsymbol{\Lambda}^{-1}}, \\ \boldsymbol{\Lambda}_k &= \boldsymbol{\Gamma}_k^{\mathrm{T}} \boldsymbol{\Omega}_k \boldsymbol{\Gamma}_k + \boldsymbol{I}, \\ \boldsymbol{\Gamma}_k \boldsymbol{\Gamma}_k^{\mathrm{T}} &= \boldsymbol{P}_{k|k}, \text{ where } \|\boldsymbol{\mu}\|^2_{\boldsymbol{\Omega}} = \boldsymbol{\mu}^{\mathrm{T}} \boldsymbol{\Omega} \boldsymbol{\mu}. \end{aligned}$$

When the full backward trajectory $\boldsymbol{\eta}'_{0:T}$ has been computed, the algorithm is typically repeated M times to give a set of backward trajectories $\{\boldsymbol{\eta}^j_{0:T}\}_{j=1}^M$ analogous to (3.31). Note that (3.36) is computed for all N particles, yielding the complexity $O(TMN)$. To find smoothed estimates of the linear states for each nonlinear trajectory, different constrained linear smoothers can be employed. Alternatively, it is possible to run Kalman filters conditioned on the backward trajectories and fuse the estimates with the backward statistics $\{Z_k, \boldsymbol{\lambda}_k, \boldsymbol{\Omega}_k\}$. This finally gives the approximated smoothing density as in (3.32). For further details, see [Lindsten et al., 2013; Lindsten, 2013].

3.5 Summary

This chapter treated estimation, beginning with giving a historical perspective on estimation. We examined the powerful estimation technique particle filtering in Section 3.1. Starting from the desired posterior

$p(\boldsymbol{x}_k, \boldsymbol{y}_{0:k})$ and Bayes' rule, we derived a common form of the particle filter that was summarized in Algorithm 3.2. The particle filter was introduced in the seminal paper by [Gordon et al., 1993], but the roots trace back to the 1950's. Since then it has been widely used in many applications, and it excels in applications where the nonlinearities are severe and/or the noise is non-Gaussian.

Rao-Blackwellized, or marginalized, particle filters were introduced in Section 3.2. In many applications the state vector can be partitioned into a part with nonlinear dynamics and a part with linear dynamics. In these cases it is possible to marginalize out the linear part and find an analytic expression of the corresponding density. This provides improved estimation accuracy and in many cases also decreased execution time. We presented the main idea and the key factorization behind the filter, and one form of it was given in Algorithm 3.3. In addition, to provide for easier access to the theory, Example 3.1 used the MATLAB implementation in Listing B.1 for a test example.

Section 3.3 discussed particle smoothing. In its general formulation, particle smoothers aim to find the density

$$p(\boldsymbol{x}_{0:T}|\boldsymbol{y}_{0:T}).$$

This can be achieved using various approaches. One approach is to store the particles in the particle filter and resample whole trajectories, instead of only the particles at the current time step. More feasible approaches include the two-filter formula and the forward filter/backward simulator. We sketched a derivation of the forward filter/backward simulator smoother, which perhaps is the most widely used approach to particle smoothing.

Finally, Section 3.4 discussed Rao-Blackwellization in particle smoothing. The solution is not as straightforward as in the filtering case, since the process is non-Markovian. This implies that the measurement likelihood depends on the whole backward trajectory. We outlined a recent particle smoother that utilizes Rao-Blackwellization for mixed-Gaussian state-space models.

4

Ground-Vehicle Modeling

This chapter introduces and explains the tire and chassis models used in the thesis. Vehicle dynamics is a field that has been given considerable attention during the last 50 years, and this chapter only treats a small subset of the available models. Different tire models that are common in the literature are introduced and compared in terms of their qualitative behavior. We also discuss aspects of chassis modeling. We conclude the chapter with a derivation of the dynamic equations of a five degrees-of-freedom chassis model. The derivation is based on the Newton-Euler formulation. The coordinate frames that are used in the derivation are described in Appendix A.

For more information about vehicle modeling in general, the reader will benefit from the books [Ellis, 1994; Kiencke and Nielsen, 2005; Reimpell and Betzler, 2005; Isermann, 2006; Rajamani, 2006; Schindler, 2007; Wong, 2008] and the theses [Gäfvert, 2003; Schofield, 2008], whereas tire modeling is given particular attention in [Pacejka, 2006; Svendenius, 2007].

4.1 Ground-Tire Interaction

The dominant interaction between a ground vehicle and its surroundings is via the tires; it is the friction forces between the tires and ground that generate most of the vehicle motion. Figure 4.1 shows a common way to model ground-tire interaction for the longitudinal dynamics [Petersen, 2003; Solyom, 2004], where the tilt about the wheel x-axis has been ignored. The wheel has longitudinal velocity v^x with respect to an inertial system, resolved in the wheel's coordinate system, and the normal load F^z acts on the wheel. A torque balance around the center of the wheel gives

$$\tau = I_w \dot{\omega} - R_w F^x, \tag{4.1}$$

where

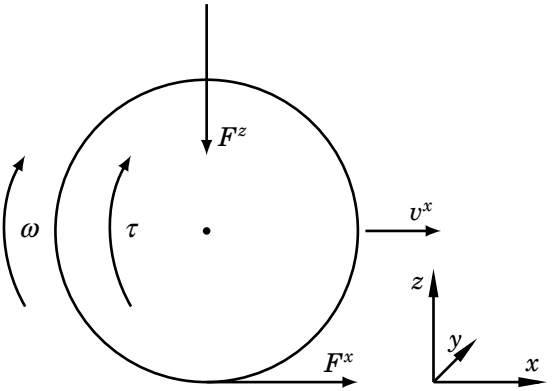

Figure 4.1 A wheel model for longitudinal dynamics. The input torque τ generates a longitudinal friction force F^x, which gives rise to angular velocity ω and longitudinal wheel velocity v^x.

- τ is the applied torque.
- I_w is the wheel moment of inertia.
- ω is the wheel angular velocity.
- R_w is the effective wheel radius—that is, the distance from the wheel center to the road.
- F^x is the longitudinal friction force, which depends on the normal force F^z.

The longitudinal force F^x is generated by applying a wheel torque τ, whereas the lateral force F^y is generated when cornering. Both these quantities are dependent on *slip*, something that will be described next.

Tire Slip

When applying wheel torque, longitudinal slip λ develops. Several definitions exist, all slightly different. In [Pacejka, 2006], it is defined as

$$\lambda := \frac{R_w\omega - v^x}{v^x} = \frac{R_w\omega}{v^x} - 1. \tag{4.2}$$

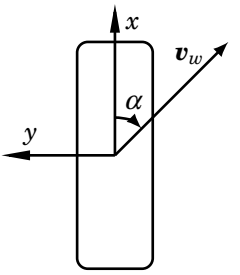

Figure 4.2 The wheel together with its coordinate system seen from above. The wheel velocity vector consists of v^x and v^y, assuming planar motion.

With this definition, $\lambda \in [-1, \infty)$.[1] Another definition, adopted from [Schindler, 2007], is given by

$$\begin{aligned} \lambda &:= \frac{v^x - R_w\omega}{v^x} \quad \text{if } v^x \geq R_w\omega \\ \lambda &:= \frac{v^x - R_w\omega}{R_w\omega} \quad \text{if } R_w\omega > v^x. \end{aligned} \tag{4.3}$$

With the slip definition in (4.3), $\lambda \in [-1, 1]$. The lateral slip angle α is the ratio of the wheel's velocities in the lateral and longitudinal directions, resolved in the wheel's coordinate system. It is often defined as

$$\tan\alpha := -\frac{v^y}{v^x}, \tag{4.4}$$

because a positive α then corresponds to a positive lateral force, see Figure 4.2 for a visualization.

Tire Forces

The interaction between tire and ground is highly complex and nonlinear, and it is hard to capture all aspects of it in low-order models. Still, in many situations it is enough to consider the longitudinal and lateral tire forces as static functions of tire slip, friction coefficient between tire and road, and normal load—that is, as the relations

$$F^x := F^x(\lambda, \alpha, \mu, F^z), \tag{4.5a}$$
$$F^y := F^y(\alpha, \lambda, \mu, F^z). \tag{4.5b}$$

The reason for the different ordering of λ and α between (4.5a) and (4.5b), is to emphasize the different dependence between the slip quantities and

[1] Assuming that both ω and v^x are positive.

forces. Linear approximations may be suitable for small slip. The approximations are

$$\begin{aligned} F^x &\approx C_\lambda \lambda \\ F^y &\approx C_\alpha \alpha, \end{aligned} \tag{4.6}$$

where the longitudinal stiffness C_λ and lateral stiffness C_α are found through linearizations of the respective force-slip relations at zero slip [Wong, 2008]:

$$C_\lambda = \left.\frac{\partial F^x}{\partial \lambda}\right|_{\lambda=0,\alpha=0} \tag{4.7a}$$

$$C_\alpha = \left.\frac{\partial F^y}{\partial \alpha}\right|_{\alpha=0,\lambda=0}. \tag{4.7b}$$

The linear approximations are widely employed in the vehicle community; they are, for example, sometimes used in reference-generation models for safety systems [Rajamani, 2006] and in algorithms that estimate tire-stiffness parameters [Carlson and Gerdes, 2005].

For pure slip—that is, when only one of the slip quantities are nonzero—an experimentally verified slip-force model is the Magic Formula model [Pacejka, 2006], given by

$$F_0(m) = D \sin\Big(C \arctan\big(Bm - E(Bm - \arctan(Bm))\big)\Big), \tag{4.8}$$

where

- B is the stiffness factor.
- C is the shape factor.
- $D = \mu F^z$ is the peak factor.
- E is the curvature factor.
- F_0 is either the longitudinal (F_0^x) or lateral (F_0^y) force and m is either λ or α.

The typical shape of (4.8) in the longitudinal direction for three different surfaces are shown in Figure 4.3. The difference compared with the forces in the lateral direction is that the peaks are often more pronounced in the longitudinal direction. When braking on snow, a locked wheel builds up snow in front of it. Thus, the force curve for snow (and loose gravel, for that matter) often increases monotonically with increased λ. The tire behavior for different surfaces and the implications on vehicle control will be discussed more in Chapter 10.

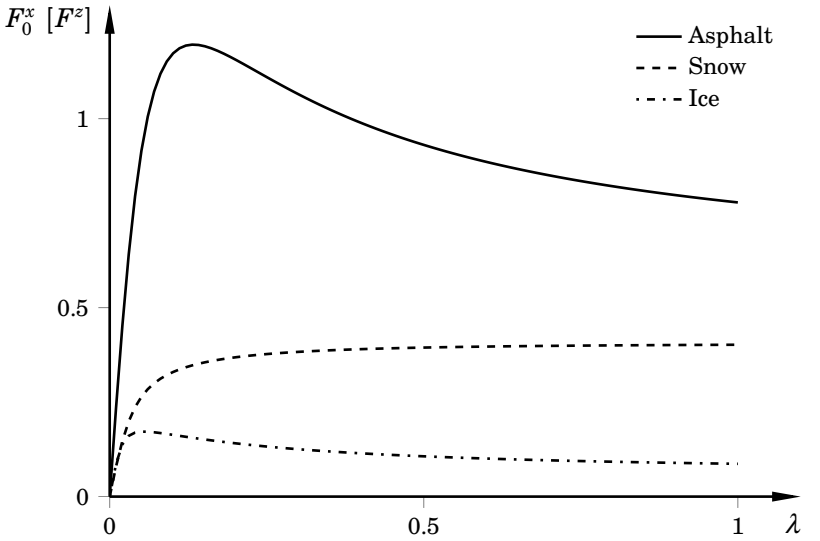

Figure 4.3 Typical shape of longitudinal force as a function of longitudinal slip for surfaces corresponding to asphalt, loose snow, and ice. In this plot, the force increases monotonically with increasing slip on snow. This is often, but not always, the case. The reason for the monotonicity is that the wheel typically builds up snow in front of it as the wheel locks.

Forces for Combined Slip Both (4.6) and (4.8) describe the slip-force interaction for pure slip. A straightforward approach to model combined slip is based on the *friction ellipse*. The friction ellipse is the envelope of the maximum achievable forces. Several models assume the same relationship for intermediate forces. In [Wong, 2008], the friction-ellipse assumption is used for computing the lateral force F^y given the longitudinal force F^x as

$$F^y = F_0^y \sqrt{1 - \left(\frac{F^x}{\mu_x F^z}\right)^2}, \tag{4.9}$$

where F_0^y is computed using (4.8). The main limitation with this model is that the longitudinal force does not explicitly depend on the lateral slip, which is not realistic. Nevertheless, owing to its simplicity it is sometimes adopted in the control literature [Andreasson, 2009; Sundström et al., 2010]. A related model is the Kamm-circle model [Kiencke and Nielsen, 2005; Isermann, 2006]. The resulting slip is defined as $\sigma_{\text{res}} := \sqrt{\lambda^2 + \alpha^2}$,

and the forces for combined slip are

$$\begin{aligned} F^x &= \frac{\lambda}{\sigma_{\text{res}}} F_r(\sigma_{\text{res}}) \\ F^y &= \frac{\alpha}{\sigma_{\text{res}}} F_r(\sigma_{\text{res}}), \end{aligned} \tag{4.10}$$

where $F_r(\sigma_{\text{res}})$ is the resulting force. Variants of (4.10) are used in the automotive literature [Velenis and Tsiotras, 2005; Chakraborty et al., 2013].

Another approach to tire modeling, inspired by the Magic Formula, is to scale the nominal force (4.8) with weighting functions G_α and G_λ, which depend on α and λ [Pacejka, 2006]. The relations for the longitudinal force are

$$F^x = F_0^x G_\alpha, \tag{4.11a}$$
$$G_\alpha = \cos(C_\alpha \arctan(H_\alpha \alpha)), \tag{4.11b}$$
$$H_\alpha = B_{\alpha 1} \cos(\arctan(B_{\alpha 2} \lambda)), \tag{4.11c}$$

and the corresponding relations in the lateral direction are given by

$$F^y = F_0^y G_\lambda, \tag{4.11d}$$
$$G_\lambda = \cos(C_\lambda \arctan(H_\lambda \lambda)), \tag{4.11e}$$
$$H_\lambda = B_{\lambda 1} \cos(\arctan(B_{\lambda 2} \alpha)), \tag{4.11f}$$

where B and C are model parameters. Figure 4.4 contains a comparison between (4.9) and (4.11), where the lateral force is plotted against longitudinal force for $\alpha = 10$ deg. As seen, the longitudinal force increases monotonically with decreasing lateral force for the model based on the friction ellipse, which is not the case for the model based on the weighting functions. Typically, experimental results tend to support the behavior of (4.11) [Pacejka, 2006]. Figure 4.5 shows how the lateral force changes with λ for fixed values of α when using (4.11).

Compared with (4.11), both (4.9) and (4.10) have the advantage that they depend on few parameters. This makes them easy to use.

4.2 Chassis Models

Also chassis models have been given a fair amount of attention in literature, see [Kiencke and Nielsen, 2005; Pacejka, 2006; Wong, 2008; Schofield, 2008] for various models of cars and [Gäfvert, 2003] for derivation of a nine degrees-of-freedom truck-trailer combination.

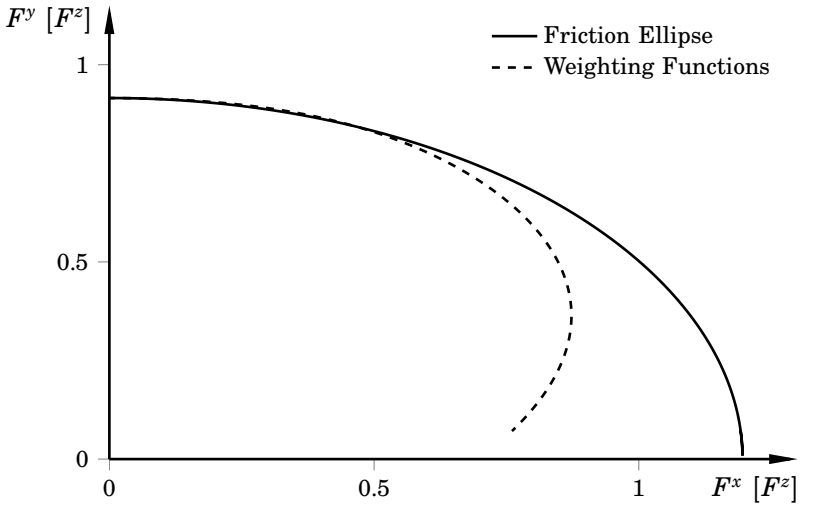

Figure 4.4 Lateral force versus longitudinal force for the models based on the weighting functions and the friction ellipse, respectively, for $\alpha = 10$ deg. The longitudinal force increases monotonically with decreasing lateral force for the friction ellipse. Experimental results tend to support the behavior of the weighting functions (4.11). The parameters correspond to the plot for asphalt in Figure 4.3.

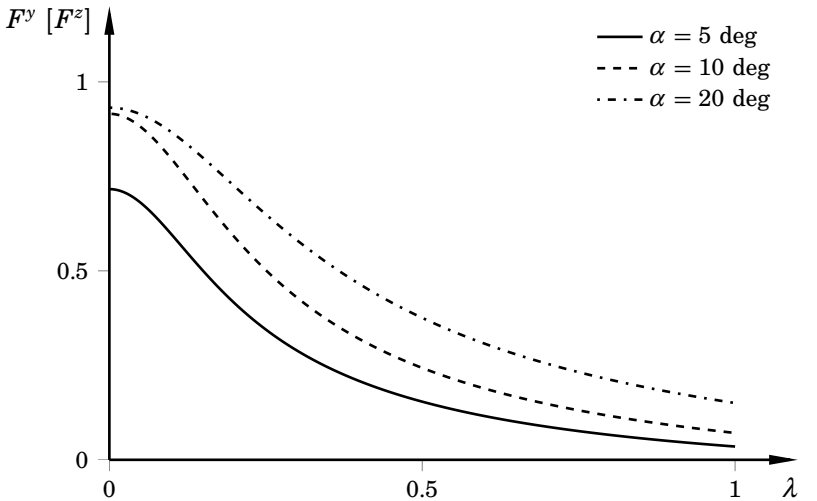

Figure 4.5 Longitudinal-slip impact on the lateral force for different α. The lateral force is computed using (4.11d)–(4.11f) with parameters corresponding to the plot for asphalt in Figure 4.3.

One-Track Model

Figure 4.6 shows a ground-vehicle model that is common in literature. The model is often referred to as the one-track model, the single-track model, or the bicycle model. It only treats planar motion. Hence, it has three degrees of freedom, two translational and one rotational. Further, it lumps together the two wheels on each axle. The model dynamics are straightforward to derive [Rajamani, 2006; Schindler, 2007; Wong, 2008] and are given by

$$\begin{aligned} \dot{v}^X - v^Y\dot{\psi} &= \frac{1}{m}(F_f^x\cos(\delta) + F_r^x - F_f^y\sin(\delta)) \\ \dot{v}^Y + v^X\dot{\psi} &= \frac{1}{m}(F_f^y\cos(\delta) + F_r^y + F_f^x\sin(\delta)) \\ I_{ZZ}\ddot{\psi} &= l_f F_f^y\cos(\delta) - l_r F_r^y + l_f F_f^x\sin(\delta), \end{aligned} \tag{4.12}$$

where v^X, v^Y are the longitudinal and lateral velocities at the mass center; $\dot{\psi}$ is the yaw rate; F_f^x, F_f^y and F_r^x, F_r^y are the longitudinal and lateral tire forces acting at the front and rear wheels, respectively; δ is the steer angle; and I_{ZZ} is the vehicle inertia about the Z-axis. The nominal normal force F_0^z acting on the respective wheel in steady state is given by

$$F_{0,f}^z = mg\frac{l_r}{l}, \quad F_{0,r}^z = mg\frac{l_f}{l},$$

where the wheel base is defined as $l := l_f + l_r$. A variable related to safety systems is the vehicle body-slip (or sideslip) angle β, and is defined as

$$\beta := \arctan\left(\frac{v^Y}{v^X}\right). \tag{4.13}$$

The one-track model has proven valuable in numerous applications, such as optimal motion planning for high-speed driving [Jeon et al., 2013],

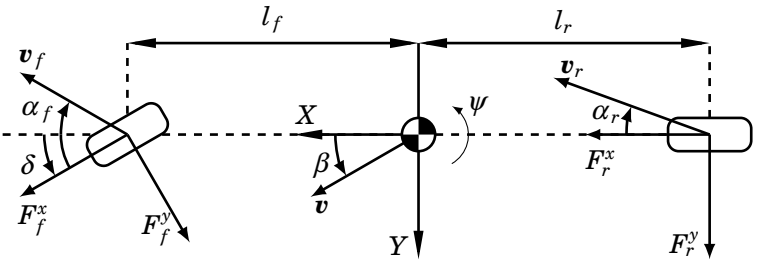

Figure 4.6 A sketch of the one-track model. The slip angle in (4.4) as well as the body-slip angle β in (4.13) are shown.

yaw-rate reference generation [Berntorp, 2008], and road-geometry estimation [Lundquist and Schön, 2011]. It is possible to extend the model to capture pitch dynamics with load transfer (i.e., rotation about the Y-axis) or roll dynamics, but to fully seize the effects of load transfer, it is necessary to consider two-track (four-wheel) models. More or less complicated variants of two-track models can be found in literature; see [Ellis, 1994; Gäfvert, 2003; Kiencke and Nielsen, 2005; Pacejka, 2006; Rajamani, 2006; Schofield, 2008] for a few of them. However, the models are often derived for a specific purpose, resulting in approximations appropriate for the considered application. For completeness we derive a two-track model that incorporates rotations in space. Together with most of the models described in this chapter, it will be used for optimal control in Chapters 10 and 11.

Derivation of a Two-Track Model

We now derive a two-track ground-vehicle model including roll and pitch dynamics. Figure 4.7 shows a schematic of the vehicle model. The derivation is done using a Newton-Euler modeling approach. The resulting chassis model has five degrees of freedom, two translational and three rotational, but the load transfer can be interpreted as a sixth degree of freedom. The derivation is based on [Berntorp, 2013], which also discusses load transfer and possible simplifications of the model.

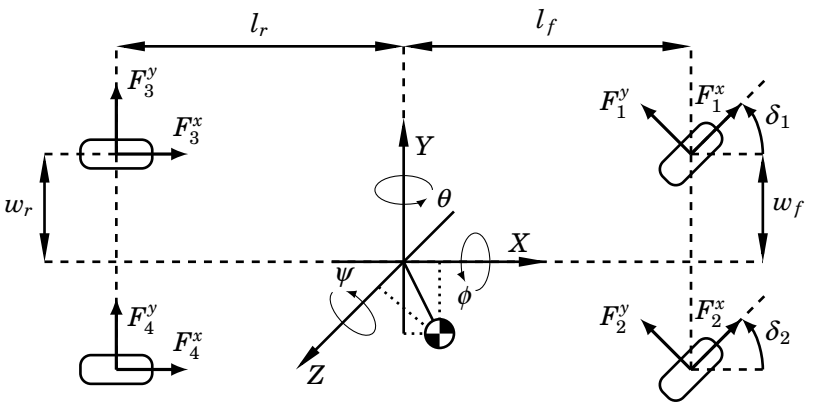

Figure 4.7 A schematic of the two-track vehicle model and its degrees of freedom, with pitch dynamics as well as roll dynamics. The X- and Y-axes reside in the ground plane. The wheels are numbered from the front left wheel to the rear right wheel.

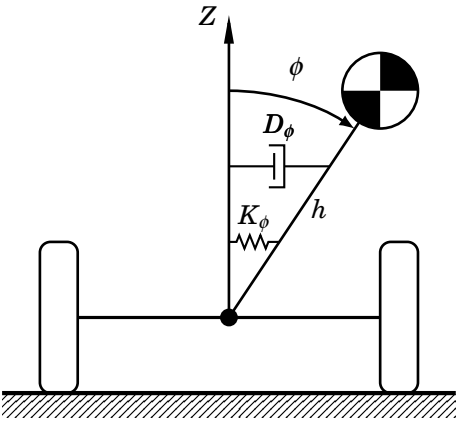

Figure 4.8 An illustration of the suspension system in the roll direction, modeled as a torsional spring-damper system with spring constant K_ϕ and damper constant D_ϕ. This suspension model is used in the derivations of the two-track model, see Figure 4.7. The mass center is located at a distance h from the origin of the vehicle frame. The suspension system is modeled in an equivalent way for the pitch dynamics.

Preliminaries The model incorporates suspension. We model the suspension system as a torsional spring and damper system, where the spring and damper constants for each wheel have been lumped to two constants, one for each degree of freedom. We assume small roll and pitch angles and model the suspension in the roll and pitch directions as two decoupled systems. Figure 4.8 provides an illustration of the roll dynamics. We assume that the wheels at all times are in contact with the ground and that the road is flat. Matrices with dimension 3×1 serve as placeholders for the vector components. The coordinate systems are described in Appendix A. I means an inertial, earth-fixed, frame and $\mathcal{V}$ denotes the vehicle-fixed frame rotated an angle ψ (the yaw) about the Z-axis of I. Similarly, $\mathcal{C}$ indicates the chassis frame, rotated with an angle θ (the pitch) about the Y-axis of $\mathcal{V}$ and with $\mathcal{B}$ we mean the body frame, rotated an angle ϕ (the roll) about the X-axis of $\mathcal{C}$. The rotation matrix from $\mathcal{C}$ to $\mathcal{V}$ is $\boldsymbol{R}_{\mathcal{C}}^{\mathcal{V}}$. The time derivative of a vector $\boldsymbol{v}$ with respect to a specific frame $\mathcal{S}$ is indicated with a subscript as in $\frac{\mathrm{d}}{\mathrm{d}t}\big|_{\mathcal{S}} \boldsymbol{v}$.

Kinematics Assume that $\mathcal{V}$ rotates with the angular velocity vector $\boldsymbol{\xi}$ with respect to the inertial frame I. Then, given a vector $\boldsymbol{v}$,

$$\left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_I \boldsymbol{v} = \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_{\mathcal{V}} \boldsymbol{v} + \boldsymbol{\xi} \times \boldsymbol{v}. \tag{4.14}$$

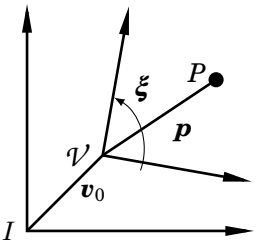

Figure 4.9 The inertial coordinate system I and the noninertial system, here denoted $\mathcal{V}$. The noninertial frame is translating with velocity $\boldsymbol{v}_0$ and rotating with angular velocity $\boldsymbol{\xi}$ relative to I.

Consider a point P with coordinates $\boldsymbol{p}$ with respect to $\mathcal{V}$. The frame $\mathcal{V}$ is moving with translational velocity $\boldsymbol{v}_0$ with respect to I. Then the velocity of P is

$$\boldsymbol{v}_P = \boldsymbol{v}_0 + \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_I \boldsymbol{p} = \boldsymbol{v}_0 + \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_{\mathcal{V}} \boldsymbol{p} + \boldsymbol{\xi} \times \boldsymbol{p}. \tag{4.15}$$

See Figure 4.9 for an illustration. By applying (4.14) to (4.15), an expression for the acceleration is

$$\boldsymbol{a}_P = \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_{\mathcal{V}} (\boldsymbol{v}_0 + \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_{\mathcal{V}} \boldsymbol{p} + \boldsymbol{\xi} \times \boldsymbol{p}) + \boldsymbol{\xi} \times (\boldsymbol{v}_0 + \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_{\mathcal{V}} \boldsymbol{p} + \boldsymbol{\xi} \times \boldsymbol{p}), \tag{4.16}$$

which can be expanded to

$$\boldsymbol{a}_P = \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_{\mathcal{V}} (\boldsymbol{v}_0 + \boldsymbol{\xi} \times \boldsymbol{p}) + \left.\frac{\mathrm{d}^2}{\mathrm{d}t^2}\right|_{\mathcal{V}} \boldsymbol{p} + \boldsymbol{\xi} \times \boldsymbol{v}_0 + \boldsymbol{\xi} \times (\boldsymbol{\xi} \times \boldsymbol{p}) + 2\boldsymbol{\xi} \times \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_{\mathcal{V}} \boldsymbol{p}. \tag{4.17}$$

Kinetics In a Newton-Euler setting the total external forces $\mathbf{F}$ acting on a rigid body B are given by

$$\mathbf{F} = m\boldsymbol{a}_{\mathrm{CoG}} = \int_B \boldsymbol{a}_P \, \mathrm{d}m_P, \tag{4.18}$$

where the integration is performed over all mass elements $\mathrm{d}m_P$. Further, $\boldsymbol{a}_{\mathrm{CoG}}$ denotes the acceleration of the mass center. Likewise, the total external moments acting on the body are equal to

$$\mathbf{M} = \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_I \boldsymbol{I}_{\mathcal{V}} \boldsymbol{\xi}_I = \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_I \int_B \boldsymbol{p} \times \boldsymbol{v}_P \, \mathrm{d}m_P, \tag{4.19}$$

where $\boldsymbol{\xi}_I = \begin{bmatrix}\dot{\phi} & \dot{\theta} & \dot{\psi}\end{bmatrix}^{\mathrm{T}}$ and $\boldsymbol{I}_{\mathcal{V}}$ is the vehicle's moment-of-inertia matrix with respect to the vehicle-fixed frame. By applying (4.14) to (4.19) we get

$$\mathbf{M} = \boldsymbol{I}_{\mathcal{V}}\dot{\boldsymbol{\xi}}_I + \boldsymbol{\xi}_{\mathcal{V}} \times \boldsymbol{I}_{\mathcal{V}}\boldsymbol{\xi}_I, \tag{4.20}$$

where $\boldsymbol{\xi}_{\mathcal{V}} = \begin{bmatrix}0 & 0 & \dot{\psi}\end{bmatrix}^{\mathrm{T}}$ is the angular velocity of the frame in which the formulas are to be derived, in this case $\mathcal{V}$. In (4.20), we have used that $\boldsymbol{I}_{\mathcal{V}}$ is constant in $\mathcal{V}$. The moment of inertia is typically measured in the body frame $\mathcal{B}$. Conveniently, $\boldsymbol{I}_{\mathcal{V}}$ is found by using the formula $\boldsymbol{I}_{\mathcal{V}} = \boldsymbol{R}_C^{\mathcal{V}}\boldsymbol{R}_{\mathcal{B}}^{C}\boldsymbol{I}_{\mathcal{B}}(\boldsymbol{R}_{\mathcal{B}}^{C})^{\mathrm{T}}(\boldsymbol{R}_C^{\mathcal{V}})^{\mathrm{T}}$ [Spong and Hutchinson, 2006], where $\boldsymbol{I}_{\mathcal{B}}$ is the moment of inertia in the body frame, and $\boldsymbol{R}_C^{\mathcal{V}}$ and $\boldsymbol{R}_{\mathcal{B}}^{C}$ are given by (A.1) and (A.2). For simplicity we have assumed that

$$\boldsymbol{I}_{\mathcal{B}} = \begin{bmatrix} I_{XX} & 0 & 0 \\ 0 & I_{YY} & 0 \\ 0 & 0 & I_{ZZ} \end{bmatrix},$$

that is, cross terms are neglected. This gives that

$$\boldsymbol{I}_{\mathcal{V}} = \begin{bmatrix} I_1 & I_2 & I_3 \\ I_2 & I_4 & I_5 \\ I_3 & I_5 & I_6 \end{bmatrix}, \tag{4.21}$$

where

$$\begin{aligned}
I_1 &= \cos^2(\theta)I_{XX} + \sin^2(\theta)\sin^2(\phi)I_{YY} + \sin^2(\theta)\cos^2(\phi)I_{ZZ},\\
I_2 &= \sin(\theta)\sin(\phi)\cos(\phi)(I_{YY} - I_{ZZ}),\\
I_3 &= -\sin(\theta)\cos(\theta)\Big(I_{XX} - I_{YY} + \cos^2(\phi)(I_{YY} - I_{ZZ})\Big),\\
I_4 &= \cos^2(\phi)I_{YY} + \sin^2(\phi)I_{ZZ},\\
I_5 &= \sin(\phi)\cos(\phi)\cos(\theta)(I_{YY} - I_{ZZ}),\\
I_6 &= \sin^2(\theta)I_{XX} + \cos^2(\theta)\Big(\sin^2(\phi)I_{YY} + \cos^2(\phi)I_{ZZ}\Big).
\end{aligned}$$

Chassis Modeling The total forces acting on the vehicle are found from force equilibria in the X- and Y-directions, see Figure 4.7:

$$\begin{aligned}
F^X &= F_1^x\cos(\delta_1) - F_1^y\sin(\delta_1)\\
&\quad + F_2^x\cos(\delta_2) - F_2^y\sin(\delta_2) + F_3^x + F_4^x\\
F^Y &= F_1^x\sin(\delta_1) + F_1^y\cos(\delta_1)\\
&\quad + F_2^x\sin(\delta_2) + F_2^y\cos(\delta_2) + F_3^y + F_4^y.
\end{aligned} \tag{4.22}$$

By performing a torque equilibrium around the vehicle Z-axis, we obtain

$$\begin{aligned} M^Z = l_f\Big(F_1^x \sin(\delta_1) + F_2^x \sin(\delta_2) + F_1^y \cos(\delta_1) + F_2^y \cos(\delta_2)\Big) \\ + w_f\Big(- F_1^x \cos(\delta_1) + F_2^x \cos(\delta_2) + F_1^y \sin(\delta_1) - F_2^y \sin(\delta_2)\Big) \\ - l_r(F_3^y + F_4^y) - w_r(F_3^x + F_4^x). \end{aligned} \quad (4.23)$$

To derive the model we first assume that the noninertial frame $\mathcal{V}$ is translating with velocity vector $\boldsymbol{v}$ relative to the inertial frame. By attaching $\mathcal{V}$ at the mass-center coordinates in the XY-plane, we get that $\boldsymbol{p} = 0$ in (4.15). Thus, the velocity in the X- and Y-directions are

$$\boldsymbol{v} = \begin{bmatrix} v^X & v^Y \end{bmatrix}^{\mathrm{T}}.$$

The translational force equations are found by combining (4.17) and (4.18). Note that we have assumed that $\boldsymbol{p}$ and all its derivatives are zero with respect to $\mathcal{V}$. By rearranging the equations, we get

$$\begin{aligned} \dot{v}^X &= v^Y\dot{\psi} + h\Big(\sin(\theta)\cos(\phi)(\dot{\psi}^2 + \dot{\phi}^2 + \dot{\theta}^2) - \sin(\phi)\ddot{\psi} - 2\cos(\phi)\dot{\phi}\dot{\psi} \\ &\quad - \cos(\theta)\cos(\phi)\ddot{\theta} + 2\cos(\theta)\sin(\phi)\dot{\theta}\dot{\phi} + \sin(\theta)\sin(\phi)\ddot{\phi}\Big) + \frac{F^X}{m} \\ \dot{v}^Y &= -v^X\dot{\psi} + h\Big(- \sin(\theta)\cos(\phi)\ddot{\psi} - \sin(\phi)\dot{\psi}^2 - 2\cos(\theta)\cos(\phi)\dot{\theta}\dot{\psi} \\ &\quad + \sin(\theta)\sin(\phi)\dot{\phi}\dot{\psi} - \sin(\phi)\dot{\phi}^2 + \cos(\phi)\ddot{\phi}\Big) + \frac{F^Y}{m}. \end{aligned} \quad (4.24)$$

The motion equation in the ψ-direction (about the Z-axis) is found by combining (4.20) and (4.21). Because of deflection of the mass center, the external forces in the X- and Y-directions give rise to additional external torques τ^Z, in this case $\tau^Z = -h(F^X \sin(\phi) + F^Y \sin(\theta)\cos(\phi))$:

$$\begin{aligned} \ddot{\psi}(I_{XX}\sin(\theta)^2 + \cos(\theta)^2(I_{YY}\sin(\phi)^2 + I_{ZZ}\cos(\phi)^2)) = M^Z \\ - h\Big(F^X \sin(\phi) + F^Y \sin(\theta)\cos(\phi)\Big). \end{aligned} \quad (4.25)$$

The suspension system gives a contribution $\tau^Y = K_\theta\theta + D_\theta\dot{\theta}$ in the pitch dynamics, where K_θ and D_θ are the rotational spring and damper constants in the θ-direction. Thus, proceeding in the same manner as for the

yaw dynamics, we get

$$\begin{aligned}\ddot{\theta}(I_{YY}\cos(\phi)^2 + I_{ZZ}\sin(\phi)^2) &= -K_\theta\theta - D_\theta\dot{\theta} \\ &+ h\Big(mg\sin(\theta)\cos(\phi) - F^X\cos(\theta)\cos(\phi)\Big) \\ &+ \dot{\psi}\Big(\dot{\psi}\sin(\theta)\cos(\theta)(\Delta I_{XY} + \cos(\phi)^2\Delta I_{YZ}) \\ &- \dot{\phi}\cos(\theta)^2 I_{XX} + \sin(\phi)^2\sin(\theta)^2 I_{YY} \\ &+ \sin(\theta)^2\cos(\phi)^2 I_{ZZ}) - \dot{\theta}\big(\sin(\theta)\sin(\phi)\cos(\phi)\Delta I_{YZ}\big)\Big), \quad (4.26)\end{aligned}$$

where $\Delta I_{YZ} = I_{YY} - I_{ZZ}$ and $\Delta I_{XY} = I_{XX} - I_{YY}$. The third equation of angular motion is in the same manner found to be

$$\begin{aligned}\ddot{\phi}(I_{XX}\cos(\theta)^2 + I_{YY}\sin(\theta)^2\sin(\phi)^2 + I_{ZZ}\sin(\theta)^2\cos(\phi)^2) \\ = -K_\phi\phi - D_\phi\dot{\phi} + h(F^Y\cos(\phi)\cos(\theta) + mg\sin(\phi)) \\ + \dot{\psi}\Delta I_{YZ}\Big(\dot{\psi}\sin(\phi)\cos(\phi)\cos(\theta) + \dot{\phi}\sin(\theta)\sin(\phi)\cos(\phi)\Big) \\ + \dot{\psi}\dot{\theta}(\cos(\phi)^2 I_{YY} + \sin(\phi)^2 I_{ZZ}). \quad (4.27)\end{aligned}$$

Equations (4.22)–(4.27) constitute the chassis two-track model with five degrees of freedom.

The two-track model derived here incorporates suspension modeling. The derivation assumes that the suspension system is modeled as a torsional spring-damper system. This assumption can be relaxed, and a derivation based on more intricate suspension dynamics is possible. The model also assumes that the roll and pitch angles are sufficiently small, which implies that the roll and pitch angles are in the linear region. The relative errors for this approximation exceed 1% at approximately 10 deg.

4.3 Summary

In this chapter we treated two key ingredients in vehicle modeling, namely tire modeling and chassis modeling. We defined the notion of tire slip, which is the main variable when it comes to modeling the static behavior of tire-ground interaction.

The Magic formula was defined. It is the most well known formula for describing tire forces as function of pure longitudinal or lateral slip. There are, however, many more models available, and we mentioned a few of them. There are also many approaches for modeling the tire forces when both longitudinal and lateral slip are nonzero. We described three of them

and compared their characteristics against each other. The weighting-functions model has been experimentally verified for quite aggressive maneuvering. Nonetheless, the friction-ellipse based models are very popular in the control community because of their simplicity.

Section 4.2 discussed chassis models. We described one of the simplest and most commonly used chassis models—that is, the one-track model (single-track model, bicycle model). It has been widely adopted in many applications, perhaps most notably as a reference generator for electronic stability control systems. Still, it has the drawbacks that it only treats planar motion and that it lumps the two wheels on each axle. To provide a more advanced model, we derived a two-track model that incorporates suspension dynamics as well as rotations in space.

5

The Out-of-Sequence Measurement Problem

This chapter introduces the out-of-sequence measurement (OOSM) problem, which will be studied in different contexts over the next three chapters. This chapter also contains a section on related work and a general problem formulation.

5.1 Motivation

Tracking systems frequently employ decentralized sensor platforms that gather measurements. These measurements are then sent to a centralized fusion center, which, given a model of the system and the sensor platforms, correlates the data and estimates the system behavior. More computing power in combination with improved sensor technologies—for example, sophisticated computer-vision algorithms that efficiently extract information from cameras—has led to that the amount of sensors used in tracking systems has increased during the last decades. Obviously, the aim with using more sensors is improved perception of the environment and in the end enhanced system performance.

OOSMs are measurements that arrive after more recent measurements have already been processed. Because the amount of sensors used in tracking is increasing, and because tracking is frequently performed using distributed sensor platforms, tracking systems increasingly often encounter OOSMs [Bar-Shalom et al., 2001]. Delayed measurements occur for several reasons—for example, data preprocessing and communication delays. Nondeterministic transmission times is inherent in many communication protocols, such as in TCP/IP or wireless sensor networks, and is a potential creator of OOSMs. However, OOSMs can also occur when using deterministic communication protocols—for example, because of buffering. That measurements arrive out of sequence implies that they

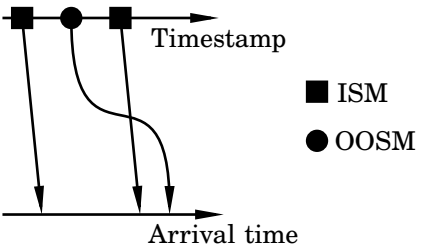

Figure 5.1 An illustration of OOSMs. Note that the ISMs also can be delayed. The difference compared with OOSMs is that they still arrive at the fusion center in the correct order.

are delayed. However, the delay itself is not enough for a measurement to be classified as an OOSM. Rather, only when the delayed measurement is combined with a measurement that arrives in the same order as it occurred in (an in-sequence measurement), is the delayed measurement an OOSM. Figure 5.1 illustrates the situation in the case of one OOSM. It could, for example, be the case that all measurements have the same delay, and are thus in-sequence measurements (ISMs). In this case the solution to the estimation problem would be to process the measurements that have arrived and then predict up to the current time. A common case, however, is that some measurements are ISMs and some are OOSMs. Hence, the ability to account for OOSMs is crucial in modern tracking systems if high accuracy is wanted. Neglecting the time delays of the measurements is a rather common choice [Maskell et al., 2006], but this means discarding information and will lead to inferior tracking performance. On the other hand, to make efficient use of OOSMs in nonlinear tracking systems can be challenging, and is probably why many choose to neglect that they in fact are OOSMs.

OOSMs arise in many applications. Because of the rapid technological development in the automotive industry, where sensors such as lidar, radar, optical cameras, and inertial measurement units are used in production cars, OOSMs have received a lot of attention in automotive applications. How to account for OOSMs in automotive applications is by some considered a key challenge for enabling autonomy [Mauthner et al., 2006]. Practical examples of OOSMs are from automotive collision-avoidance systems, where network links cause transmission delays of radar sensors [Muntzinger et al., 2010; Westenberger et al., 2013]. Another application is tracking of autonomous vehicles, where cameras have become increasingly important for giving spatial information. The processing times of the vision algorithms, in combination with inertial measurement units, often cause OOSMs, see [Ranganathan et al., 2007; Jia et al., 2008].

5.2 Problem Formulation

When studying the OOSM problem, it is customary to assume that some measurements experience negligible delay, implying that they can be processed without taking the delay into account. It is only the measurements that have delays so large that they become OOSMs that actually are compensated for. In the following, t_k means the timestamp of a measurement and t_k^a is the arrival time of the same measurement. Definition 5.1 gives a formal definition of OOSMs and ISMs [Zhang and Bar-Shalom, 2012a].

Definition 5.1—[Zhang and Bar-Shalom, 2012a]
Given a measurement $\boldsymbol{y}_{k_1}$, if there exists another measurement $\boldsymbol{y}_{k_2}$ with $t_{k_2}^a < t_{k_1}^a$ and $t_{k_2} > t_{k_1}$, then $\boldsymbol{y}_{k_1}$ is an OOSM. Otherwise, $\boldsymbol{y}_{k_1}$ is an ISM. □

We use the notation $\boldsymbol{y}_\tau$ to indicate that the measurement is an OOSM. In general, when discussing the OOSM problem we consider the scenario of possibly nonlinear state dynamics, possibly nonlinear measurement relations, and additive process and measurement noise. In other words, the systems are on the same form

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{w}_k, \\ \boldsymbol{y}_k &= \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{e}_k. \end{aligned} \tag{5.1}$$

Without loss of generality, we will assume that $\boldsymbol{u}_k = \boldsymbol{0}$ and thus that the system is driven by noise. Another assumption is that the timestamps of the ISMs are the same as their arrival times (i.e., $t_k = t_k^a$).

The problem formulation can be stated as follows: Suppose that measurements up to time index k (i.e., time t_k) have been processed. Thus, an expression for the posterior probability density function $p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k})$ is available. Then one or several delayed measurements $\boldsymbol{y}_\tau$ arrive, and the problem is to update with the OOSM—that is, to form

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k}, \boldsymbol{y}_\tau). \tag{5.2}$$

Note that when the system is linear with white, Gaussian noise, to find (5.2) is equivalent to update the state estimate and associated covariance in the Kalman-filter framework. Moreover, when the timestamp t_τ is bounded as $t_\tau \in [t_{k-1}, t_k)$, the problem of finding (5.2) is referred to as the 1-step lag problem. More generally, for a positive integer l, when the timestamp is bounded as $t_\tau \in [t_{k-l}, t_{k-l+1})$ the problem is referred to as the l-step lag problem, see Figure 5.2. Note that the OOSM index τ need not be an integer; that is, the timestamp of the OOSM can have any value between two sampling instants.

Many target-tracking systems estimate the states of multiple targets. While tracking these targets, both measurements from the target and

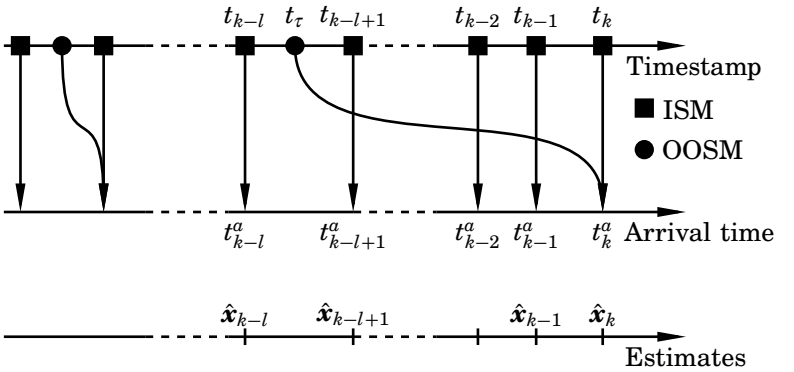

Figure 5.2 An illustration of the l-step lag OOSM problem, where the circles denote OOSMs and squares refer to ISMs. OOSMs $\boldsymbol{y}_\tau$ arising from time t_τ arrive at time t_k, and are subsequently utilized to compute an updated $\hat{\boldsymbol{x}}_k$.

false alarms (clutter), for example caused by measurement noise, can reach the tracking systems. Data association deals with the problem of selecting the measurement that most probably is originated from the object to be tracked. Throughout, we discard measurements delayed more than $l_{\max}$ time steps, with $l_{\max}$ predefined, and assume that all sensors give detection of the correct targets each time. Hence, data association and clutter are beyond the scope of this thesis [Karlsson and Gustafsson, 2001; Särkkä et al., 2004; Maskell et al., 2006].

5.3 Related Work

Over the last decades there has been substantial research considering OOSMs for tracking. An overview of initial work spanning to the late 1990's is found in [Blackman and Popoli, 1999]. In [Bar-Shalom, 2002], the optimal solution (i.e., the minimum mean-square error solution) to the 1-step lag problem was derived. An efficient, suboptimal approach for the l-step lag problem is given in [Bar-Shalom et al., 2004], and [Zhang et al., 2010] contains an algorithm that handles sensor bias. Optimal solutions to the l-step problem for different available information and using different approaches have been derived in, for example, [Zhang et al., 2005; Koch, 2009; Shen et al., 2009].

One of the drawbacks with the presented approaches is that only the most recent estimate is updated with the OOSM. In real-life scenarios there are sometimes multiple OOSMs arriving, either in succession or interleaved with ISMs. What one then is faced with is to update with

multiple OOSMs. In this case the preceding approaches will in general not be optimal. The first optimal solution to the multi-OOSM problem was derived in [Shen et al., 2009]. The solution assumes that the OOSMs are not interleaved with the ISMs. The first general optimal solution with multiple OOSMs, denoted the complete in-sequence information (CISI) approach, was presented in [Zhang and Bar-Shalom, 2012a]. A number of approaches yielding the optimal solution were compared in terms of complexity, among them a fixed-point smoother, a fixed-interval smoother, an *fading-information* approach, and an algorithm using the *equivalent-measurement method*. The conclusion was that the fixed-point smoother approach is superior to the fixed-interval smoothing approach, the fading information approach, and the equivalent-measurement method in terms of computational demands. The storage requirements in [Zhang and Bar-Shalom, 2012a] are only the mean and covariances for the $l_{\max}$ last time steps. For the scenario with several OOSMs arriving simultaneously, the CISI approach is applied sequentially, still giving optimality. Another, similar solution, is found in [Govaers and Koch, 2012], which is based on a generalized, computationally efficient Rauch-Tung-Striebel algorithm [Rauch et al., 1965].

Nonlinear Systems

All work presented so far is for linear systems. The implementation-wise easiest extension for nonlinear systems is to use extended Kalman-filter (EKF) type approximations. For systems with significant nonlinearities and/or non-Gaussian noise, the use of EKFs can lead to poor performance. Several methods for exploring OOSMs in the more general particle-filter framework have been proposed. An approach where the measurement equation is allowed to be nonlinear was outlined in [Orton and Marrs, 2001; Orton and Marrs, 2005]. The particle weights are first updated without the OOSM. They are then modified utilizing the OOSM in a Markov chain Monte Carlo smoothing step to overcome the problem of degeneracy in the particle filter. This approach stores the N particles for the last $l_{\max}$ time steps, where $l_{\max}$ is the predetermined maximum delay. Unfortunately, it also needs a linear state-transition model. Another drawback with this approach is that the storage requirements are large, since all particles have to be stored for the last $l_{\max}$ steps. A workaround for this was described in [Mallick et al., 2002], where an invertible state-transition matrix is assumed. This matrix is then used for retrodiction of the states back to the time of the OOSM. The only storage requirements in this algorithm are the mean and covariances for the last $l_{\max}$ steps. A comparison between particle filters and Kalman filters for OOSM filtering is found in [Mallick and Marrs, 2003]. For the considered system,

which is linear, the two types of estimators perform similarly.

An extension of the work in [Mallick et al., 2002] was presented in [Orguner and Gustafsson, 2008], where the assumption of a linear, invertible state-transition matrix was removed. In addition, [Orguner and Gustafsson, 2008] derived a particle filter for OOSM updates using an orthogonal approach, denoted storage-efficient particle filter (*SEPF*). *SEPF* handles nonlinear state-space models with white, Gaussian noise. It is computationally fast and memory efficient, since it only stores and processes means, covariances, and measurements for the last $l_{\max}$ time steps. Because there is only one measurement vector $\boldsymbol{y}_k$ at each time instant, and because the dimension of the measurements usually is less than that of the states, the storage requirements for the measurements are often minor compared with storing the particles. Different fixed-point smoothers determine the likelihood of the measurement given each particle at the current time. The likelihood is then utilized to update the weight of the particle. When [Orguner and Gustafsson, 2008] compared an extended fixed-point smoother, an unscented Kalman smoother, and a particle smoother on a highly nonlinear example, the fixed-point smoother outperformed the unscented Kalman smoother and the particle smoother despite demanding less computational power. As mentioned previously, *SEPF* usually performs very well. However, as pointed out in [Orguner and Gustafsson, 2008], the performance of *SEPF* sometimes suffers when the OOSMs introduce a large change in the estimated filtering distribution. This problem is partially overcome in [Liu et al., 2010], where an algorithm for detecting these OOSMs was derived, denoted SEPF with selective processing (*SEPF-GARP*). In [Oreshkin et al., 2011], the approach in [Liu et al., 2010] was extended with an optimization-based algorithm for separating between informative and uninformative OOSMs. Moreover, an exact Bayesian solution and its corresponding particle filter implementation, denoted *A-PF*, were derived in [Zhang and Bar-Shalom, 2012b] for general nonlinear models with white noise. One drawback with this algorithm is that it is computationally expensive. For OOSMs that have larger delays than one sample, its complexity is $O((l-1)N^3 + N^2)$, where N is the number of particles in the forward filter. This complexity usually prohibits real-time implementations for anything but the smallest systems.

6

Out-of-Sequence Measurements in Robotics

Chapter 5 gave an introduction to the OOSM problem as well as mentioned a few applications where they occur. This chapter presents an application of an OOSM solution to a physical setup. It is based on the work in [Berntorp et al., 2011; Berntorp et al., 2012], and treats a combined indoor navigation and mobile-manipulation example. The scenario is that of distributing groceries. This chapter does not intend to derive any new models or estimation techniques. Rather, it shows how different sensor and process-model combinations, in combination with classic control theory, can be combined to perform complex tasks.

We first give an introduction to the application and why it is studied, and then present the experimental equipment. The chapter proceeds with a section on modeling, which is followed by a summary of the estimation algorithm. The second part of the chapter is devoted to mobile manipulation. Finally, a summary and some remarks are given.

6.1 Motivation and Problem Description

The last two decades, attention has been drawn to combine mobile robots with manipulators. The ambition has not only been to improve existing applications, but also to enable new applications. Several potential applications were mentioned in Chapter 1—for example, manufacturing, assembly, and medical surgery assistance. Another area is service robotics for domestic applications. Examples of approaches in this direction are ROMAN [Hanebeck et al., 1997], the PR2 [Wyrobek et al., 2008], and the Care-O-Bot [Reiser et al., 2009a]. There are several challenges when it comes to mobile manipulation; for example, the question of how to coordinate between the mobile base and the manipulator is a nontrivial issue. Another issue is how to define and specify the control task. In this

chapter we use a light-weight, two-armed industrial robot combined with a pseudo-omnidirectional mobile base within a pick-and-place scenario, where the stationary robot manipulator places groceries on the mobile base while both the mobile base and manipulator arm are moving. To incorporate sensor measurements, introduce uncertainties such as the uncertainty of the pose estimates into the control problem, and coordinate movements, the constraint-based task-specification methodology (iTaSC), described in [De Schutter et al., 2007], is used.

We use the OOSM solution in [Bar-Shalom et al., 2004] to provide real-time estimates of the position and orientation of the mobile base. The method in [Bar-Shalom et al., 2004] assumes a linear model with white, Gaussian noise. It is a computationally efficient, approximate method, and is based on the equivalent-measurement concept. Here, equivalent measurement means that all measurements from the current time to the OOSM time are replaced with a function that depends on the predicted state estimates from the OOSM time and a modified process noise. The OOSM timestamp must be bounded by a positive integer $l_{\max}$ as $t_\tau \in [t_{k-l_{\max}}, t_k)$, but the delays do not have to be known prior to the OOSM arrival. Thus we assume the l-step lag problem defined in Section 5.2.

The estimator fuses inertial sensor, wheel-encoder, and vision measurements using an extended Kalman-filter (EKF) approach. A timestamp is attached to every camera measurement. The extension from [Bar-Shalom et al., 2004] is that the algorithm executes on a physical setup in real time.

6.2 Related Work

How to combine internal and external sensors for position and velocity estimation in real time is crucial for autonomous mobile robots. Methods based on wheel encoders, inertial measurement units (IMUs), and global positioning systems (GPS) are common to localize mobile robots, see [Yi et al., 2007; Yi et al., 2009] and references therein. The major drawback with using GPS indoors is well known; the GPS signal strength is often heavily attenuated, as well as scattered by walls and roofs. Therefore, using GPS is in general not an option when navigating indoors. Dead-reckoning approaches based on wheel encoders and IMUs all suffer from integration and accumulation of erroneous signals—for example, caused by noise or bias in the sensors. Vision can be used for removing the drift caused by dead reckoning. Vision for mobile-robot navigation is by no means a new concept, see the survey [DeSouza and Kak, 2002], which covers the developments from the 1980's to the late 1990's. Vision in combination with inertial sensors has been investigated before, see [Corke et al., 2007] for

an introduction and [Armesto et al., 2004] for multi-rate fusion of inertial and visual sensors, to mention a few examples.

Fusion of IMU and vision for state estimation of mobile robots has been studied before in slightly varying setups. In [Nuetzi et al., 2011], IMU data is fused with vision in a simultaneous localization and mapping framework, by employing an EKF. In [Choi et al., 2007], OOSM state estimation considering wheel encoders and a GPS with time delay is performed and verified by simulations. Another work is [Jia et al., 2008], which uses the method in [Bar-Shalom, 2002] by assuming that the delay is within a fraction of a sample. An autonomous vehicle's target is estimated using a constant-velocity model with the help of vehicle-mounted cameras, and inertial sensors are used for estimating the stereo-cameras' motion.

An example of motion coordination was given in [Carriker et al., 1989], where the coordination problem is formulated as a nonlinear optimization problem. The resulting solution is evaluated on a two degrees-of-freedom mobile base combined with a three degrees-of-freedom manipulator. Other examples of motion coordination are found in [Khatib et al., 1996], [Khatib, 1999], where the coordination is solved by modeling the mobile base as the mechanism with coarse and slow dynamics, with the manipulator viewed as the fast and accurate device. The coordination between the two devices is then done by considering internal forces. In [Meeussen et al., 2010] the problem of door opening was considered. The mobile base motion is independent of the arm motion, whereas the arm motion is coupled to the mobile base through sensing of the reaction forces of the environment.

6.3 Experimental Setup

The mobile base, see Figure 6.1, which was built and designed at Fraunhofer IPA in Stuttgart, Germany, has previously been used in the DESIRE project [Reiser et al., 2009b]. It is a four-wheeled pseudo-omnidirectional mobile robot equipped with eight motors, two for each wheel.[1] Figure 6.2 shows a sketch of the robot and its degrees of freedom. For the experiments in this chapter, the mobile robot is equipped with a six degrees-of-freedom IMU from Xsens [Xsens Technologies B.V., 2010], which is aligned with the robot's coordinate frame. The IMU provides measurements with a rate of 50 Hz. An internal calibration procedure accounts for imperfections in the physical alignment of each component, gains, offsets, and temperature relations. Using the calibration, the accelerometer and gyro vectors, expressed in the IMU's local coordinate frame, are computed using an onboard processor.

[1] The robot needs to turn its wheels, thus prohibiting true omnidirectionality.

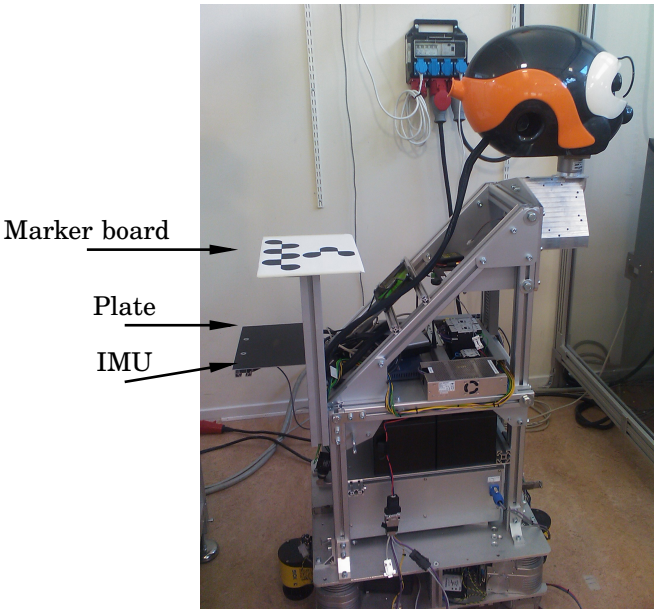

Figure 6.1 The mobile robot used for the experiments. The IMU is placed below the black plate to the left in the picture, and the feature-detection pattern used for the vision algorithm is located above the black plate.

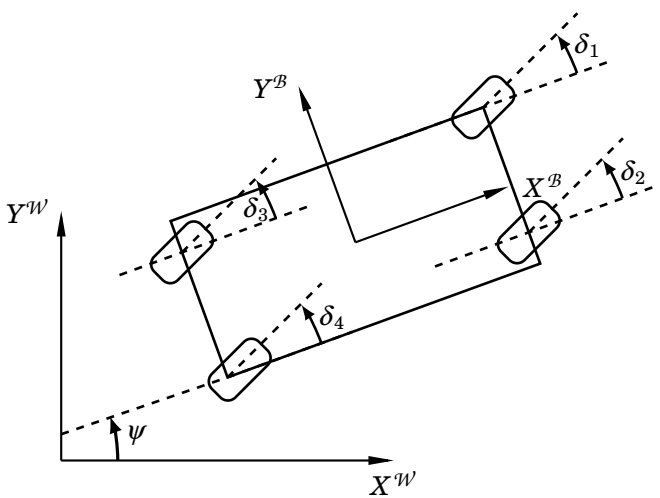

Figure 6.2 A sketch of the robot and its degrees of freedom.

The wheel-encoder position and velocity measurements are extracted with a rate of 20 Hz. The robot is controlled and sensor data are acquired using the ROS software package [ROS, 2014]. A roof mounted camera is situated above the robot's workspace. Using the camera the absolute position of the robot, as well as the orientation are calculated. The algorithms for object tracking and feature detection are not included here, but are surveyed in [Yilmaz et al., 2006]. In this experimental setup, the camera vision algorithm provides position measurements with an update rate between 1.5–2 Hz. The update rate differs because of jitter and non-deterministic computational demands. Moreover, the camera provides a timestamp associated with each frame, thus enabling OOSM compensation.

Mobile Manipulator

FRIDA [Kock et al., 2011], see Figure 6.3, is a dual-arm lightweight manipulator that is used in the pick-and-place scenario and for ground-truth evaluations. Both arms have seven degrees of freedom, which means that they have one redundant degree of freedom each. The robot is designed to be intrinsically safe, which is accomplished by having low payload and robot inertia, a mechanical design free from sharp edges, as well as covering exposed regions with soft padding. Also, power and speed limitations together with collision detection are implemented. The robot is controlled with the ABB IRC5 robot control system, which has been extended with an external control system (see [Blomdell et al., 2005] and [Blomdell et al., 2010]) that makes it possible to alter the references for the low-level joint velocity and position loops. The external controllers are designed in MATLAB/Simulink. The Real-Time workshop toolbox is used for code generation. The resulting program is executed on a Linux Xenomai PC with communication between the IRC5 control system and the external control system via a dedicated Ethernet connection. For communication between the two robots, Java with PalCom, see [Åkesson et al., 2012; Fors, 2009], is used. The communication executes with a rate of 50 Hz.

6.4 Modeling

When working with moving objects, different coordinate frames have to be used. For the combined vehicle and camera system, three coordinate systems are of particular interest:

1. World ($\mathcal{W}$): This frame is considered an inertial frame and is fixed to the base of FRIDA. The robot pose is estimated with respect to this frame.

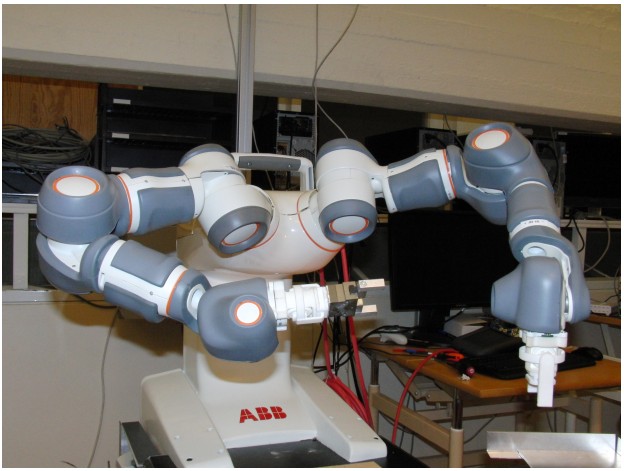

Figure 6.3 The ABB light-weight robot FRIDA used both in the pick-and-place scenario and for verification of the estimation accuracy. To evaluate estimation performance, FRIDA's endpoint (flange) of the left arm tracks the robot, and the position differences between FRIDA and the mobile robot are used as an error estimate.

2. Body ($\mathcal{B}$): This frame is fixed to the mobile robot, with its origin placed at the position of the IMU. Both the IMU and wheel-encoder measurements are resolved in this frame.

3. Camera ($\mathcal{C}$): The camera measurements are given in this frame, which is fixed relative to $\mathcal{W}$.

Assume that the position of $\mathcal{B}$ expressed in $\mathcal{W}$ is $\boldsymbol{p}^{\mathcal{W}}$, and that the rotation is given by a yaw ψ, a pitch θ, and a roll ϕ (see Appendix A). Then the transformation between the frames is performed by an addition of $\boldsymbol{p}^{\mathcal{W}}$ followed by a rotation given by the rotation matrix

$$\boldsymbol{R}_{\mathcal{B}}^{\mathcal{W}} := \begin{bmatrix} c_\theta c_\psi & -c_\phi s_\psi + s_\phi s_\theta c_\psi & s_\phi s_\psi + c_\phi s_\theta c_\psi \\ c_\theta s_\psi & c_\phi c_\psi + s_\phi s_\theta s_\psi & -s_\phi c_\psi + c_\phi s_\theta s_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix},$$

where c_θ and s_θ are short for $\cos(\theta)$ and $\sin(\theta)$, respectively, and similarly for the other angles.

The mobile base is modeled by a state-space model. In congruence with the standard notation used throughout the thesis, at time t_k the robot state vector is denoted $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ is the input vector. It is safe to assume that the mobile base moves in a plane, because it navigates indoors on

Table 6.1 Mobile base states with description and dimension.

Notation	Description
$\boldsymbol{p} \in \mathbb{R}^2$	Position in world coordinates
$\boldsymbol{v} \in \mathbb{R}^2$	Velocity in world coordinates
$\boldsymbol{a} \in \mathbb{R}^2$	Acceleration in world coordinates
$\boldsymbol{b_a} \in \mathbb{R}^2$	Bias states for acceleration measurement
$\psi \in \mathbb{R}$	Yaw angle relative to the world frame
$\dot{\psi} \in \mathbb{R}$	Yaw velocity in world coordinates
$b_{\dot{\psi}} \in \mathbb{R}$	Bias state for yaw velocity measurement

flat surfaces. The state vector consists of

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{p}_k \\ \boldsymbol{v}_k \\ \boldsymbol{a}_k \\ \boldsymbol{b}_{\boldsymbol{a},k} \\ \psi_k \\ \dot{\psi}_k \\ b_{\dot{\psi},k} \end{bmatrix} \in \mathbb{R}^{11}, \tag{6.1}$$

where $\boldsymbol{p}_k \in \mathbb{R}^2$ is the position, $\boldsymbol{v}_k \in \mathbb{R}^2$ is the velocity, $\boldsymbol{a}_k \in \mathbb{R}^2$ is the acceleration, which are all both given and expressed in $\mathcal{W}$, $\psi_k \in \mathbb{R}$ is the yaw angle, and $\dot{\psi}_k \in \mathbb{R}$ is the yaw rate. Moreover, the state vector is extended with bias states $\boldsymbol{b}_{k,\boldsymbol{a}} \in \mathbb{R}^2$ and $b_{k,\dot{\psi}} \in \mathbb{R}$ to account for sensor imperfections. Note that both the acceleration and yaw rate are introduced as states in (6.1), but they could be used as inputs instead. Including them as states gives a larger state space, but increases flexibility since the model can then be used for prediction of accelerations and yaw rates as well. Table 6.1 summarizes the state vector.

Process Model

The process model describes the time evolution (i.e., the dynamics) of the state vector $\boldsymbol{x}_k$. There are of course many ways, with different complexity, to model this. For rigid bodies, it is common to employ constant-acceleration or constant-velocity models [Gustafsson, 2010b]. By modeling the process as a constant-acceleration process, the model that describes the time evolution of $\boldsymbol{x}_k$ becomes

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{w}_k, \tag{6.2}$$

with

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_{2\times2} & T_s\boldsymbol{I}_{2\times2} & \frac{T_s^2}{2}\boldsymbol{I}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_2 & \boldsymbol{0}_2 & \boldsymbol{0}_2 \\ \boldsymbol{0}_{2\times2} & \boldsymbol{I}_{2\times2} & T_s\boldsymbol{I}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_2 & \boldsymbol{0}_2 & \boldsymbol{0}_2 \\ \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{I}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_2 & \boldsymbol{0}_2 & \boldsymbol{0}_2 \\ \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{I}_{2\times2} & \boldsymbol{0}_2 & \boldsymbol{0}_2 & \boldsymbol{0}_2 \\ \boldsymbol{0}_2^{\mathrm{T}} & \boldsymbol{0}_2^{\mathrm{T}} & \boldsymbol{0}_2^{\mathrm{T}} & \boldsymbol{0}_2^{\mathrm{T}} & 1 & T_s & 0 \\ \boldsymbol{0}_2^{\mathrm{T}} & \boldsymbol{0}_2^{\mathrm{T}} & \boldsymbol{0}_2^{\mathrm{T}} & \boldsymbol{0}_2^{\mathrm{T}} & 0 & 1 & 0 \\ \boldsymbol{0}_2^{\mathrm{T}} & \boldsymbol{0}_2^{\mathrm{T}} & \boldsymbol{0}_2^{\mathrm{T}} & \boldsymbol{0}_2^{\mathrm{T}} & 0 & 0 & 1 \end{bmatrix}. \tag{6.3}$$

In (6.3), T_s is the sampling period, $\boldsymbol{I}_{2\times2}$ is the 2×2 identity matrix, $\boldsymbol{0}_{2\times2}$ is the 2×2 zero matrix, and $\boldsymbol{0}_2$ is the 2×1 zero vector. The process noise $\boldsymbol{w}_k$ is assumed independent, white, and Gaussian distributed with zero mean according to

$$\boldsymbol{w}_k \sim \mathcal{N}(\boldsymbol{0}_{11}, \boldsymbol{Q}), \quad \boldsymbol{Q} = \boldsymbol{F}\boldsymbol{Q}^{\boldsymbol{w}}\boldsymbol{F}^{\mathrm{T}}, \tag{6.4}$$

where

$$\boldsymbol{F} = \begin{bmatrix} \frac{T_s^3}{6}\boldsymbol{I}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_2 & \boldsymbol{0}_2 \\ \frac{T_s^2}{2}\boldsymbol{I}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_2 & \boldsymbol{0}_2 \\ T_s\boldsymbol{I}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{0}_2 & \boldsymbol{0}_2 \\ \boldsymbol{0}_{2\times2} & T_s\boldsymbol{I}_{2\times2} & \boldsymbol{0}_2 & \boldsymbol{0}_2 \\ \boldsymbol{0}_2^{\mathrm{T}} & \boldsymbol{0}_2^{\mathrm{T}} & \frac{T_s^2}{2} & 0 \\ \boldsymbol{0}_2^{\mathrm{T}} & \boldsymbol{0}_2^{\mathrm{T}} & T_s & 0 \\ \boldsymbol{0}_2^{\mathrm{T}} & \boldsymbol{0}_2^{\mathrm{T}} & 0 & T_s \end{bmatrix}$$

and $\boldsymbol{Q}^{\boldsymbol{w}} \in \mathbb{R}^{6\times6}$ is a diagonal matrix with noise parameters for the acceleration states, acceleration bias, yaw rate state, and yaw rate bias. Note that (6.2) is linear and thus fits directly into the framework that is used for OOSM processing.

Measurement Model

The vision system provides position and yaw-angle measurements ($\boldsymbol{p}_{m,k}$ and $\psi_{m,k}$, respectively), and the velocities $\boldsymbol{v}_{m,k}^{\mathcal{B}}$ are extracted from the wheel-encoder measurements. Moreover, the IMU measures accelerations and angular velocities in the IMU frame $\mathcal{B}$. Since we use acceleration and yaw rate as states, the IMU measurements enter in the measurement

equation. Thus, the measurements are

$$\boldsymbol{y}_k = \begin{bmatrix} \boldsymbol{p}_{m,k} \\ \boldsymbol{v}^{\mathcal{B}}_{m,k} \\ \boldsymbol{a}^{\mathcal{B}}_{m,k} \\ \psi_{m,k} \\ \dot{\psi}_{m,k} \end{bmatrix} \in \mathbb{R}^8. \tag{6.5}$$

Note that $\boldsymbol{v}^{\mathcal{B}}_{m,k}$ in (6.5) is not given directly by the wheel encoders, but needs to be transformed using the forward kinematics. If longitudinal and lateral slip are neglected, investigation of Figures 6.2 and 6.4 straightforwardly gives the relations for the velocity vector of the robot's geometric center, $\boldsymbol{v}^{\mathcal{B}}_{\mathrm{CoG},k}$, as

$$\boldsymbol{v}^{\mathcal{B}}_{\mathrm{CoG},k} = \begin{bmatrix} \sum_{i=1}^{4} \cos(\delta_{i,k}) R_i \omega_{i,k} \\ \sum_{i=1}^{4} \sin(\delta_{i,k}) R_i \omega_{i,k} \end{bmatrix}.$$

The velocity measurement vector $\boldsymbol{v}^{\mathcal{B}}_{\mathrm{CoG},k}$ gives the velocity at the geometric center. Therefore, $\boldsymbol{v}^{\mathcal{B}}_{\mathrm{CoG},k}$ has to be translated to the IMU's location prior to being used in (6.5). This is done as

$$\boldsymbol{v}^{\mathcal{B}}_{m,k} = \boldsymbol{v}^{\mathcal{B}}_{\mathrm{CoG},k} + \dot{\psi}_k \times \boldsymbol{p}^{\mathcal{B}}_{\mathrm{CoG},k}, \tag{6.6}$$

where $\boldsymbol{p}^{\mathcal{B}}_{\mathrm{CoG},k}$ is the relative position between the IMU and the geometric center.

The measurement model that relates the measurement vector (6.5) to

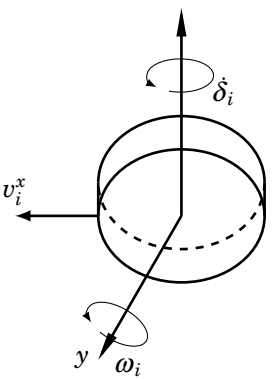

Figure 6.4 Wheel i and its two rotational degrees of freedom.

the states is given by

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{e}_k = \begin{bmatrix} \boldsymbol{p}_k \\ \boldsymbol{R}_{\mathcal{B}}^{\mathcal{W}} \boldsymbol{v}_k \\ \boldsymbol{R}_{\mathcal{B}}^{\mathcal{W}} \boldsymbol{a}_k + \boldsymbol{b}_{\boldsymbol{a},k} \\ \psi_k \\ \dot{\psi}_k + b_{\dot{\psi},k} \end{bmatrix} + \boldsymbol{e}_k. \tag{6.7}$$

The measurement noise $\boldsymbol{e}_k$ is assumed to be independent, white, and Gaussian distributed, which leads to

$$\boldsymbol{e}_k \sim \mathcal{N}(\boldsymbol{0}_8, \boldsymbol{R}),$$

where the covariance matrix $\boldsymbol{R} \in \mathbb{R}^{8\times 8}$ is determined from experiments. The rotation matrix introduces nonlinearities in (6.7), but these are mild.

6.5 Filter Design

To solve the state estimation problem, we use an EKF. The forward-filter algorithm that is used is Algorithm 3.1 on page 50, with linearized dynamics. The algorithm consists of two steps; the time update step and the measurement update step. In the time update step, (6.2) is used for updating the states and covariances.

When a measurement arrives, (6.5) is used for updating the mean of the state estimate. To update the covariance estimate, the measurement model is linearized at $\hat{\boldsymbol{x}}_{k|k-1}$ (i.e., the one-step prediction of the estimated mean), which yields

$$\boldsymbol{C}_k = \left. \frac{\mathrm{d}\boldsymbol{h}_k(\boldsymbol{x}_k)}{\mathrm{d}\boldsymbol{x}_k} \right|_{\hat{\boldsymbol{x}}_{k|k-1}}.$$

With this calculation of the measurement matrix, Algorithm 3.1 is used for updating with the in-sequence measurements.

Remark 6.1
From (6.6) it is clear that the velocity at the geometric center depends on both wheel-encoder and yaw-rate measurements. However, we neglect this and only treat the wheel-encoder measurements as disturbed by noise. □

Handling Out-of-Sequence Measurements

The camera delivers pictures with a rate of approximately 30 Hz, but the vision algorithm only produces about two position and angle measurements per second on average. The wheel encoders produce measurements at 20 Hz and the IMU delivers measurements at 50 Hz. Thus, the

measurements from the vision algorithms are OOSMs, and the delay is therefore accounted for using an OOSM approach. As pointed out in Section 5.3, there are many alternatives to account for measurements that arrive delayed in time for linear systems. The approach used here, which was derived in [Bar-Shalom et al., 2004], assumes the retrodicted[2] noise to be zero. This assumption implies that the algorithm is only suboptimal. When an OOSM arrives, the estimate at time index k given measurements up to time index k, $\hat{\boldsymbol{x}}_{k|k}$, is retrodicted to the OOSM time index τ using equivalent measurements. The algorithm is outlined below, but for more details the reader is referred to [Bar-Shalom et al., 2004].

The algorithm assumes a maximum OOSM delay $l_{\max}$ and that the OOSM time t_τ for each OOSM is bounded by a positive integer l as $\tau \in [k-l, k-l+1)$. Starting with the estimate $\hat{\boldsymbol{x}}_{k|k}$, the retrodiction of the state from k to τ is

$$\hat{\boldsymbol{x}}_{\tau|k} = \boldsymbol{A}_{\tau,k}\hat{\boldsymbol{x}}_{k|k} = \boldsymbol{A}_\tau \boldsymbol{A}^{-l}\hat{\boldsymbol{x}}_{k|k}, \tag{6.8}$$

where $\boldsymbol{A}_\tau$ is $\boldsymbol{A}$ in (6.3) with T_s replaced with $T_s(t_\tau - t_{k-l})$. The covariance for the state retrodiction is

$$\boldsymbol{P}_{\tau|k} = \boldsymbol{A}_{\tau,k}\left(\boldsymbol{P}_{k|k} + \boldsymbol{Q}_{k,\tau} - \boldsymbol{P}^{\boldsymbol{xw}}_{k,\tau|k} - (\boldsymbol{P}^{\boldsymbol{xw}}_{k,\tau|k})^{\mathrm{T}}\right)\boldsymbol{A}^{\mathrm{T}}_{\tau,k},$$

where $\boldsymbol{Q}_{k,\tau}$ is the accumulated process noise covariance matrix, which is equal to $\boldsymbol{Q}$ in (6.4) with T_s replaced with $T_s(t_\tau - t_{k-l})$. Moreover,

$$\boldsymbol{P}^{\boldsymbol{xw}}_{k,\tau|k} = \boldsymbol{P}_{k|k}\boldsymbol{P}_{k|k-l}\boldsymbol{P}^{\boldsymbol{ww}}_{k,\tau|k}, \tag{6.9}$$

where $\boldsymbol{P}^{\boldsymbol{xw}}_{k,\tau|k}$ is the cross-covariance between the accumulated process noise and the current state. The covariance between the state at time index k and the OOSM is computed as

$$\boldsymbol{P}^{\boldsymbol{xy}}_{k,\tau|k} = \left(\boldsymbol{P}_{k|k} - \boldsymbol{P}^{\boldsymbol{xw}}_{k,\tau|k}\right)\boldsymbol{A}^{\mathrm{T}}_{\tau,k}\boldsymbol{C}^{\mathrm{T}}_\tau. \tag{6.10}$$

Using the cross-covariance (6.10), the gain used for the EKF update is

$$\boldsymbol{K}_{k|\tau} = \boldsymbol{P}^{\boldsymbol{xy}}_{k,\tau|k}\boldsymbol{S}^{-1}_\tau,$$

where $\boldsymbol{S}_\tau$ is given by the expression

$$\boldsymbol{S}_\tau = \boldsymbol{C}_\tau \boldsymbol{P}_{\tau|k}\boldsymbol{C}^{\mathrm{T}}_\tau + \boldsymbol{R}.$$

The new, updated estimate $\hat{\boldsymbol{x}}_{k|k,\tau}$ is

$$\hat{\boldsymbol{x}}_{k|k,\tau} = \hat{\boldsymbol{x}}_{k|k} + \boldsymbol{K}_{k|\tau}\left(\boldsymbol{y}_\tau - \hat{\boldsymbol{y}}_{\tau|k}\right), \tag{6.11}$$

[2] Backward predicted.

where the retrodicted OOSM is equal to

$$\hat{\boldsymbol{y}}_{\tau|k} = \boldsymbol{C}_\tau \hat{\boldsymbol{x}}_{\tau|k}, \tag{6.12}$$

and the retrodicted state is given by (6.8). Finally, the covariance for the updated state estimate is

$$\boldsymbol{P}_{k|\tau} = \boldsymbol{P}_{k|k} - \boldsymbol{P}^{\boldsymbol{xy}}_{k,\tau|k} \boldsymbol{S}_\tau^{-1} (\boldsymbol{P}^{\boldsymbol{xy}}_{k,\tau|k})^{\mathrm{T}}. \tag{6.13}$$

Note that no storage of the old state estimates is needed. The only stored covariance matrix that is explicitly needed for the algorithm is $\boldsymbol{P}_{k-l|k-l}$, which is used in (6.9). The storage requirements for the algorithm in addition to the standard EKF is the storage of $\{\boldsymbol{P}_{k-m|k-m}\}_{m=1}^{l_{\max}}$, corresponding to $l_{\max} n_x (n_x + 1)/2$ scalars, where n_x is the number of states. The algorithm is summarized in Algorithm 6.1

Algorithm 6.1—OOSM algorithm

1: **Initialize:** Set $\hat{\boldsymbol{x}}_{-1|-1} = \boldsymbol{x}_{-1}$, $\boldsymbol{P}_{-1|-1} = \boldsymbol{P}_{-1}$.
2: **for** $k = 0$ **to** T **do**
3: Update the state estimate and covariance with (3.4).
4: When velocity and/or IMU measurements arrive, update the state estimate and covariance with (3.3).
5: When a delayed camera measurement arrives, use (6.11)–(6.13).
6: **end for**

6.6 Tracking-Performance Evaluation

This section will first give experimental results for two different scenarios. These scenarios are intended to showcase how the position estimates differ from the camera measurements. In the first scenario, the robot drives around randomly, with longitudinal, translational, as well as rotational movements. The second scenario consists of straight-line driving, and we show a 35 s excerpt from the whole experiment. Then, we assess the performance in an evaluation where we use FRIDA as ground truth.

Evaluation against Camera Sensor

Figure 6.5 shows the measured and estimated pose (position and orientation) for one of the two scenarios. The estimates cohere well with the measured quantities, although some discrepancies exist. One reason for the discrepancies is that dead-reckoning errors will cause the Kalman-filter estimates to become less reliable as time progresses without any new position information. Another cause for discrepancies is that the IMU is

not perfectly aligned with the robot, which implies that it senses nonplanar motion; for example, the contribution from gravity to the longitudinal and lateral acceleration measurements are nonzero. If investigating Figure 6.5 closely, it is possible to see small discontinuities (jumps) in the position estimates when compensating for the OOSMs. This shows that even though approximately 25 IMU measurements arrive between every camera measurement, the position measurements contain significant information. Further, the discontinuities give an indication of how fast the estimation algorithm would diverge (which it would) without absolute position information.

The second scenario consists of a straight-line path with the yaw angle held constant. Figure 6.6 shows the longitudinal position. The estimates are very close to the camera measurements, and a comparison of Figure 6.6 with Figure 6.5 gives that the estimated robot motion is smoother for the second scenario (i.e., the less complex maneuver), as expected.

Evaluation against Ground Truth

To verify the tracking accuracy we use FRIDA, see Figure 6.3, as ground truth. FRIDA has an absolute accuracy of approximately 0.25 mm mean error and 1 mm maximum error in steady state. The state estimates from the estimation algorithm are sent to FRIDA at a rate of 50 Hz, and FRIDA is controlled to keep the position of the flange of its left arm constant relative to the mobile base. With an ideal control system, the flange only moves away from its reference position if the pose and velocity estimates are incorrect. The control system in FRIDA is, of course, not ideal, but still the procedure gives some information about the performance.

Figure 6.7 shows the results from an experiment where the mobile base drives back and forth as well as rotating back and forth. The upper plot displays the difference between the mobile base position estimates and the X- and Y-coordinates of the robots' left flange. The lower plot shows the estimated mobile base velocities. The minor deviations that occur are when the base accelerates. This can be explained by the relatively low sampling rates of the wheel encoders and vision algorithm, but also because the system is not perfectly controlled. The maximum error is about 15 mm, but is often well below 10 mm. The results indicate that the estimation is smooth and correct enough to perform high-precision coordinated control, at least when the acceleration is not too large.

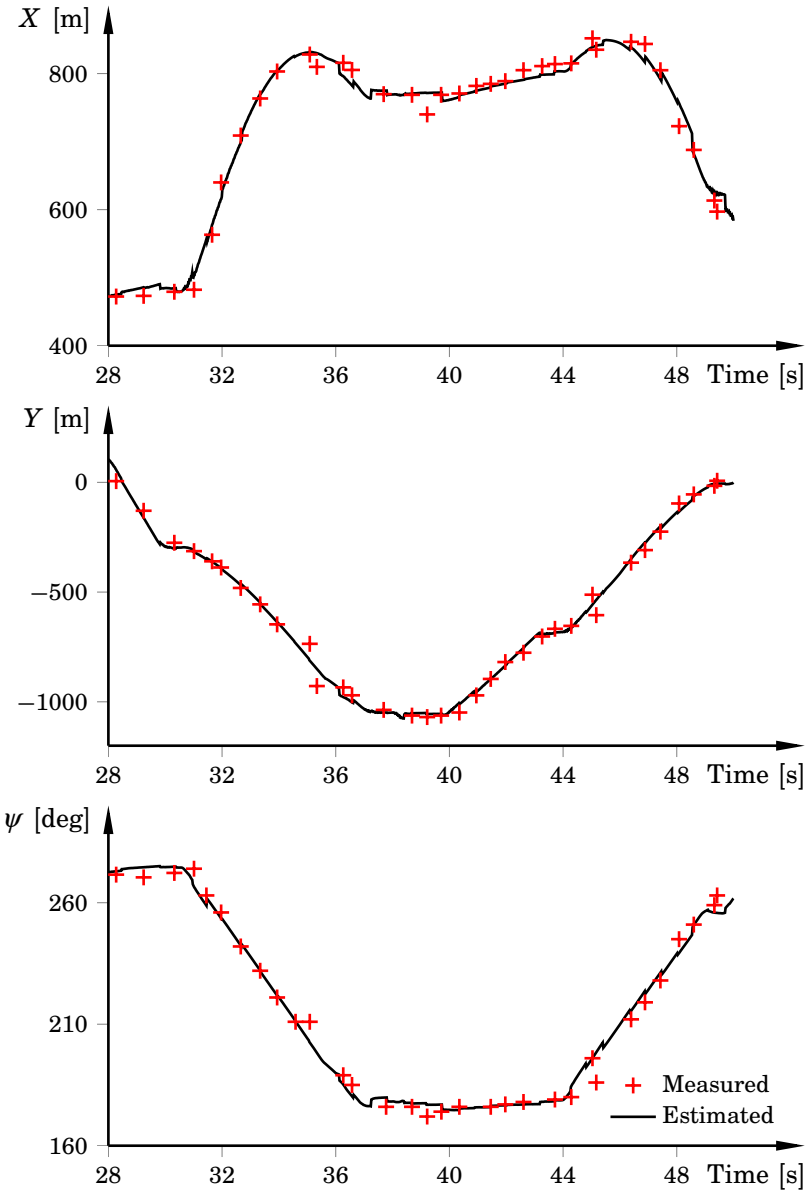

Figure 6.5 Estimated trajectories compared with the camera measurements for the Kalman-filter approach with OOSM compensation. The red + indicate the camera measurements. The camera measurements have been moved to their correct place in time.

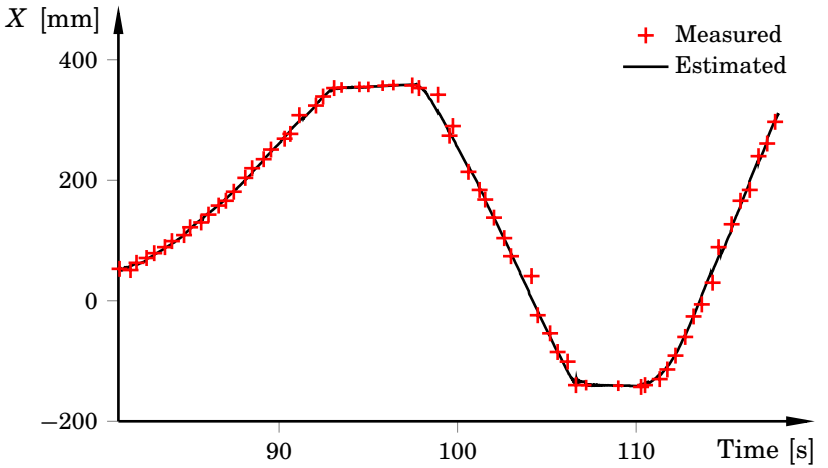

Figure 6.6 Estimated longitudinal trajectory compared with the camera measurements for the Kalman-filter approach with OOSM compensation. The red + indicate the camera measurements. The camera measurements have been moved to their correct place in time. Compared with Figure 6.5 the estimates are now smoother throughout, which owes to the less complex maneuver.

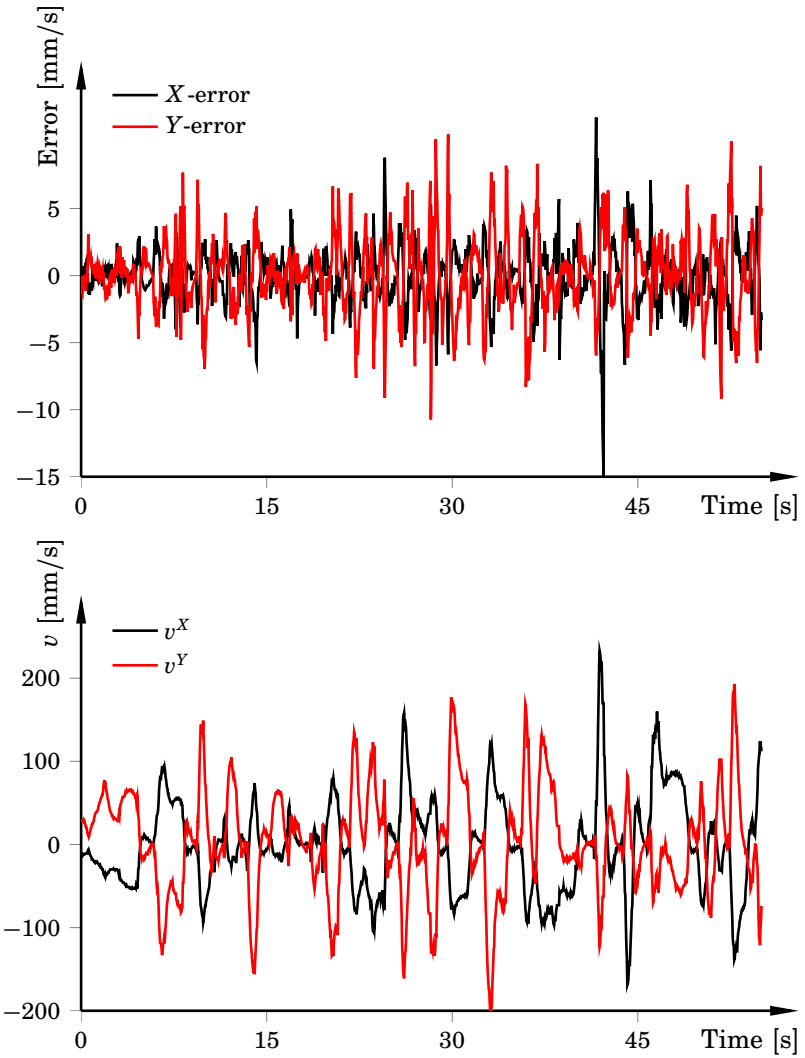

Figure 6.7 Verification of the uncertainty-estimation algorithm. The upper plot shows the longitudinal (X) and lateral global errors between the flange of FRIDA's left arm and the position of the mobile base. The lower plot shows the estimated global velocities of the mobile robot. The position error is at most approximately 15 mm (at 42 seconds), but most often it is well below 10 mm.

6.7 Application: A Pick-and-Place Scenario

The considered application is a pick-and-place scenario. The aim is to pick up and place cans using a mobile manipulator, but the framework and possible use cases of the robots are not restricted to this scenario. Examples of applications that can be approached, estimated, and controlled using a similar procedure and experimental setup are assembly and milling.

The pick-and-place scenario is as follows: The manipulator first picks up a can, which is positioned at a fixed, known position. The size of the can is unknown but, of course, has to fit into the gripper's fingers. The shape of the can is assumed cylindrical. The manipulator then places the can at the corner of the plate while the mobile base drives around. Force control ensures that the item is placed in the correct position. Figure 6.8 shows the setup. Figure 6.9 contains a flowchart of the scenario.

- In state 1, FRIDA picks up the item.
- When entering state 2, the mobile base starts moving with varying velocity. FRIDA moves to a position above the plate, with the orientation chosen such that the coordinate frames of both robots are aligned in the Z-direction.
- In state 3, FRIDA initiates a search motion using velocity control in the Z-direction. The position and orientation are kept constant in the other directions.
- When contact is made, the state machine enters state 4, where the control in the Z-direction now switches to force control.
- In states 4 and 5, FRIDA searches for contact using velocity control in the Y- and X-directions, respectively. As soon as contact is made, the controller switches back to position control. Finally, the item is released and the procedure can either be restarted or terminated.

Task Specification

We use a constraint-based task specification framework (iTaSC) for specifying the task, see [De Schutter et al., 2007] for a thorough description. The main ideas behind the framework and how it is used in the pick-and-place scenario is explained next.

The constraint-based task-specification framework specifies the relative motion of objects by introducing constraints. Constraints can express geometric relationships, force relationships, velocity relationships, or some other relationships. These constraints are specified using kinematic chains. Normally a kinematic chain contains two *object frames*, o_1

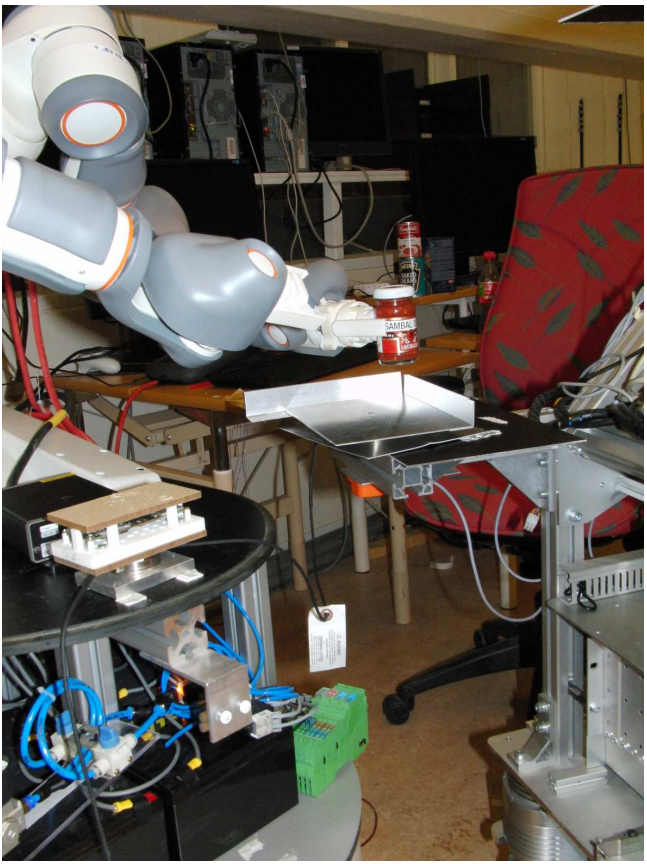

Figure 6.8 The setup used in the pick-and-place scenario. FRIDA's left gripper is used for picking up and placing groceries on the mobile base.

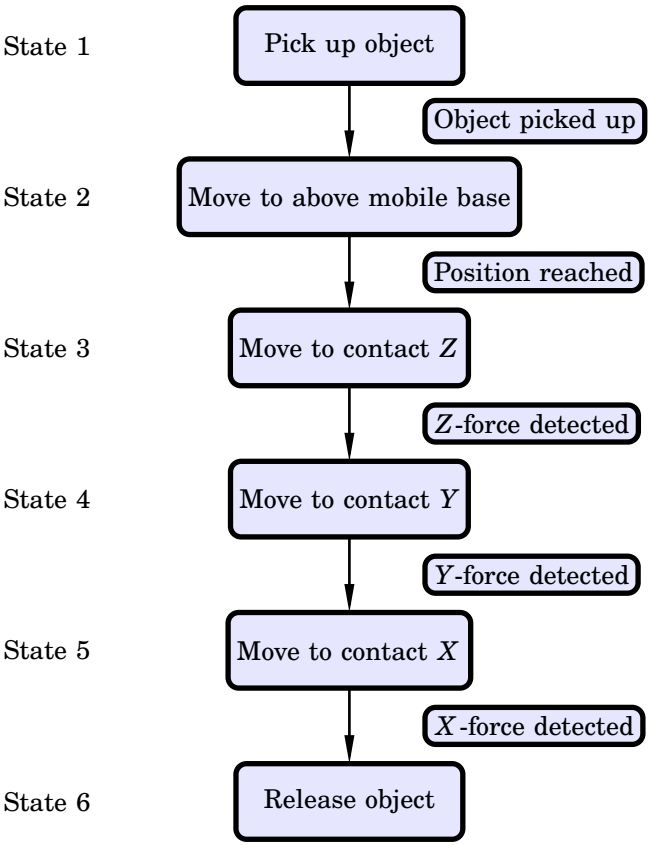

Figure 6.9 A flowchart describing the pick-and-place strategy used in the experiments in Section 6.7. The intermediate boxes between two states indicate the rules that govern the respective switches.

and o_2, and two *feature frames*, f_1 and f_2. The object frames are rigidly attached to the manipulated object and the object that manipulates, respectively. The feature frames should be attached in such a way that they simplify the problem of specifying the constraints that define the task. Furthermore, they should be linked to o_1 and o_2, respectively. The different transformations between the frames are either constant or time varying. In total there are six degrees of freedom distributed over the transformations, and these are represented by the feature coordinates $\boldsymbol{\chi}_f$. The feature coordinates are usually partitioned as

$$\boldsymbol{\chi}_f = \begin{bmatrix} \boldsymbol{\chi}_{f_1}^{\mathrm{T}} & \boldsymbol{\chi}_{f_2}^{\mathrm{T}} & \boldsymbol{\chi}_{f_3}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}},$$

where $\boldsymbol{\chi}_{f_1}$ represents the relative motion of f_1 with respect to o_1, $\boldsymbol{\chi}_{f_2}$ represents the relative motion of f_2 with respect to f_1, and $\boldsymbol{\chi}_{f_3}$ represents the relative motion of o_2 with respect to f_2. To obtain the feature coordinates, the inverse kinematics of the kinematic chains have to be solved for.

Normally not all transformations are exactly known, which implies that some uncertainties exist. These uncertainties are modeled by introducing auxiliary transformations, placed between the frames where the uncertainties occur. The degrees of freedom of the uncertainties are given by $\boldsymbol{\chi}_u$, the uncertainty coordinates; for example, frame o_1' could model the uncertainty of o_1, with the degrees of freedom between the two frames represented by $\boldsymbol{\chi}_u$.

The variables that are of interest to constrain (i.e., to control) are chosen by specifying desired outputs $\boldsymbol{y}_d$ as functions of the robot joint coordinates $\boldsymbol{q}$ and feature coordinates $\boldsymbol{\chi}_f$ as

$$\boldsymbol{y}_d = \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{\chi}_f), \tag{6.14}$$

If the kinematic chain is chosen appropriately, $\boldsymbol{y}_d$ directly corresponds to some or all of the feature coordinates $\boldsymbol{\chi}_f$. If there are fewer outputs than the degrees of freedom, the robot system will be underconstrained. The redundancy can then be utilized to perform an additional task—for example, minimizing the norm of the joint velocities.

Task Specification for the Pick-and-Place Scenario We use one kinematic chain to model the pick-and-place scenario. The feature and uncertainty coordinates connect the different transformations according to:

- The world frame $\mathcal{W}$ is rigidly connected to FRIDA.
- Frame o_1 is fixed to $\mathcal{W}$. The unknown position of the mobile base is modeled by o_1', and is estimated by Algorithm 6.1.

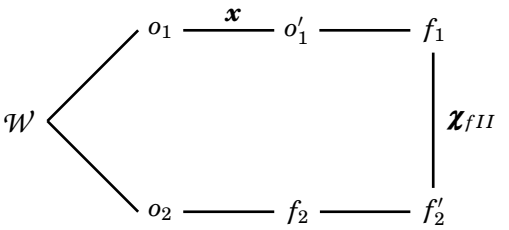

Figure 6.10 Feature and uncertainty coordinates and how the frames connect to each other. The primed frame represents the uncertain frame while the other represent the true frames. The transformation between $\mathcal{W}$ and o_2 is given by FRIDA's forward kinematics, which depends on the feature coordinates $\boldsymbol{\chi}_{fII}$ and the position of the mobile base.

- Frame f_1 is located at one of the corners of the plate, see Figure 6.1. It is related to o_1' by a constant translation.

- Frame o_2 is fixed to FRIDA's left flange. It is related to $\mathcal{W}$ by the kinematics of FRIDA.

- Frame f_2 is connected to the tool center point, related to o_2 by a constant translation. The unknown geometry of the gripped item is modeled by f_2'.

For a clarification of how the the different transformations connect to each other, see Figure 6.10. The feature coordinates are collected into the transformation between f_1 and f_2—that is, the relative position between the mobile base and FRIDA. Hence, $\boldsymbol{\chi}_{f_2} \in \mathbb{R}^6$ contains the three translational and three rotational degrees of freedom. Since all degrees of freedom are collected between f_1 and f_2, both $\boldsymbol{\chi}_{f_1}$ and $\boldsymbol{\chi}_{f_3}$ are void.

The first three feature coordinates are Cartesian translations along the axes of f_1, and the last three are rotational coordinates parametrized by ZYX (yaw-pitch-roll) Euler angles, see Section 6.4 and Appendix A. The length, width, and height of the gripped object are modeled by the transformation between f_2 and f_2'. No explicit uncertainty coordinates are used for modeling this uncertainty, because the object dimensions are accounted for using force control; that is, the motion is guarded by a force sensor when searching for contact. When reaching contact in the Z-direction the motion is force controlled to maintain contact throughout, while the X- and Y-directions are both position controlled. Furthermore, all feature coordinates are chosen as outputs.

Computing the Feature Coordinates

The transformations in Figure 6.10 form a kinematic loop

$$L(\boldsymbol{q}, \boldsymbol{\chi}_f) = \mathbf{0} \tag{6.15}$$

for each kinematic chain. The kinematic loop $L(\boldsymbol{q}, \boldsymbol{\chi}_f)$ in (6.15) is constituted by the product of all n involved transformation matrices. To exemplify, let $\boldsymbol{T}_i$ be a transformation matrix. Then

$$L(\boldsymbol{q}, \boldsymbol{\chi}_f) = \boldsymbol{T}_1 \cdots \boldsymbol{T}_n = \mathbf{0}.$$

Note that because we use Euler angles to model rotation and we know, or have estimates of, the feature coordinates at the previous time step, the solution to the inverse kinematics is analytic and given by inversion of the corresponding transformation matrix.

Manipulator Control

The constraints in the task specification are either position, velocity, or force based. Because the control system for FRIDA only allows to alter the joint velocity and position references, the constraint equation has to include feedback to not violate the constraints. How to provide feedback at joint level with the references set at Cartesian level is straightforward and described in [De Schutter et al., 2007]: The idea is to find a relationship between the desired output velocities $\dot{\boldsymbol{y}}_d$ and the robot joint coordinates $\boldsymbol{q}$. This relationship is found by differentiation of (6.14), which gives

$$\dot{\boldsymbol{y}}_d = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\chi}_f} \dot{\boldsymbol{\chi}}_f. \tag{6.16}$$

Differentiation of the kinematic loop (6.15) and insertion into (6.16) yields the relation

$$\boldsymbol{B}\dot{\boldsymbol{q}} = \dot{\boldsymbol{y}}_d. \tag{6.17}$$

Here $\boldsymbol{B}$ is a matrix that is composed of the different jacobians involved. Because the task specification only has six constraints (three translational and three rotational) and the mobile base and FRIDA have more than six degrees of freedom, the matrix $\boldsymbol{B}$ is not square. Hence, the solution to (6.17) is in general not unique. A workaround for this is to use the pseudoinverse $\boldsymbol{B}^{\#}$ [Golub and Van Loan, 1996], which is equal to

$$\boldsymbol{B}^{\#} = \boldsymbol{M}^{-1}\boldsymbol{B}^{\mathrm{T}}(\boldsymbol{B}\boldsymbol{M}^{-1}\boldsymbol{B}^{\mathrm{T}})^{-1}$$

for a suitable weighting matrix $\boldsymbol{M}$. The feedback is now introduced by modifying $\dot{\boldsymbol{y}}_d$ with a feedforward term $\dot{\boldsymbol{y}}_{ff}$ and a controller C as

$$\dot{\boldsymbol{y}}_d = \dot{\boldsymbol{y}}_{ff} + C. \tag{6.18}$$

For the pick-and-place scenario, velocity control is performed with PID control, see (2.1) on page 36, and both position and force control are done with impedance controllers ((2.2) on page 37). Note that the control references are sent to FRIDA; that is, FRIDA is main responsible for controlling the behavior.

Experimental Results

Figure 6.11 shows the state sequence (see Figure 6.9 for the flowchart) from an experiment and Figure 6.12 displays the corresponding force data. The threshold set to trigger transitions to state 4 is -4 N for the force in the Z-direction, which is exceeded after approximately 5 s. To trigger transitions to states 5 and 6 we set the threshold to 2 N for the Y- and X-directions, respectively. The force in the Z-direction is controlled to -5 N as soon as contact is made. The rotational coordinates are controlled to maintain zero velocity in all states from state 3 and onward.

Figure 6.13 contains the estimated velocities for the mobile base together with the desired velocities for FRIDA (i.e., the output from (6.18)) from the same experiment. The desired velocities are expressed in $\mathcal{B}$ (i.e., the frame of the mobile base). Approximately between 1.8–5 s the velocity in the Z-direction is controlled to keep constant relative velocity, whereas the other positions are position controlled using impedance controllers. Note that the velocity references in both X and Y vary. This has to do with that the impedance controllers are tuned to avoid too large contact forces, rather than keeping exactly the same position. Between approximately 5–8.5 s the Y-velocity is controlled, and between roughly 9–10.8 s the X-velocity is controlled. The release of the can occurs where the control signal in Z goes to nonzero velocity at 10.8 s. The path that the mobile base traversed and the state transitions are found in Figure 6.14.

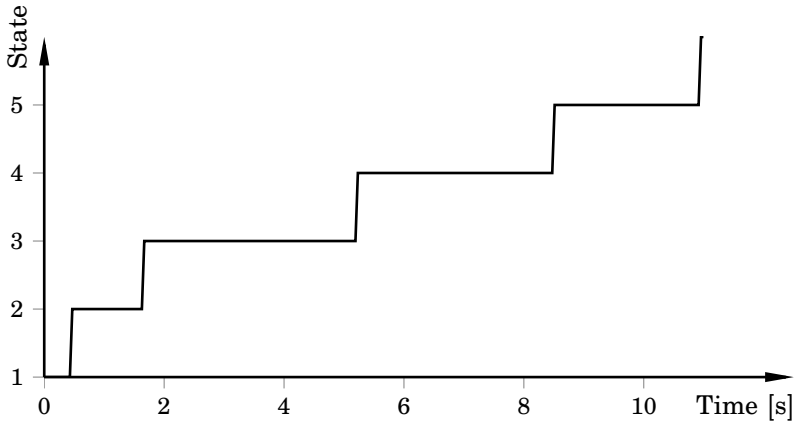

Figure 6.11 State sequence from the pick-and-place scenario (Section 6.7). The state sequence is from the same experiment as Figures 6.12–6.14.

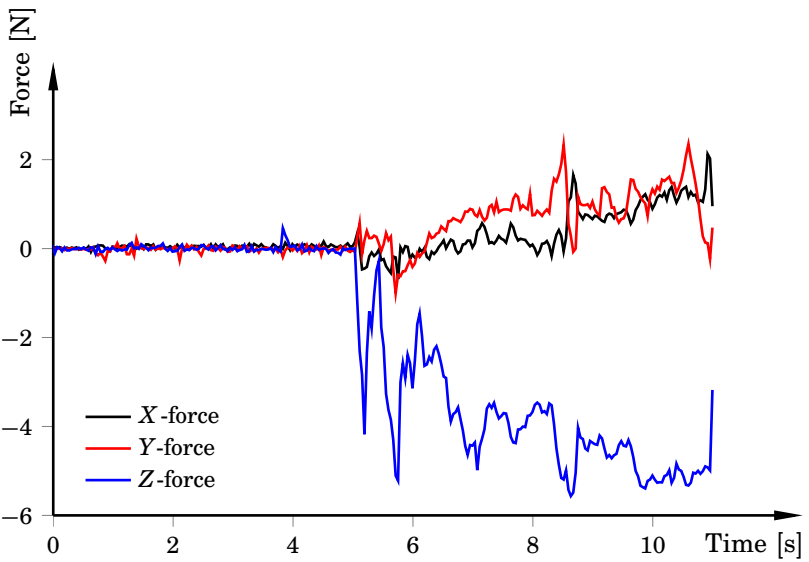

Figure 6.12 Force data from the pick-and-place scenario (Section 6.7) for the same experiment as in Figure 6.11. The Z-force is controlled to -5 N from state 4 and onward. The threshold for activating the force control in the Z-direction is set to -4 N. The outputs in the Y- and X-directions are position controlled except in states 4 and 5, respectively, where they are velocity controlled.

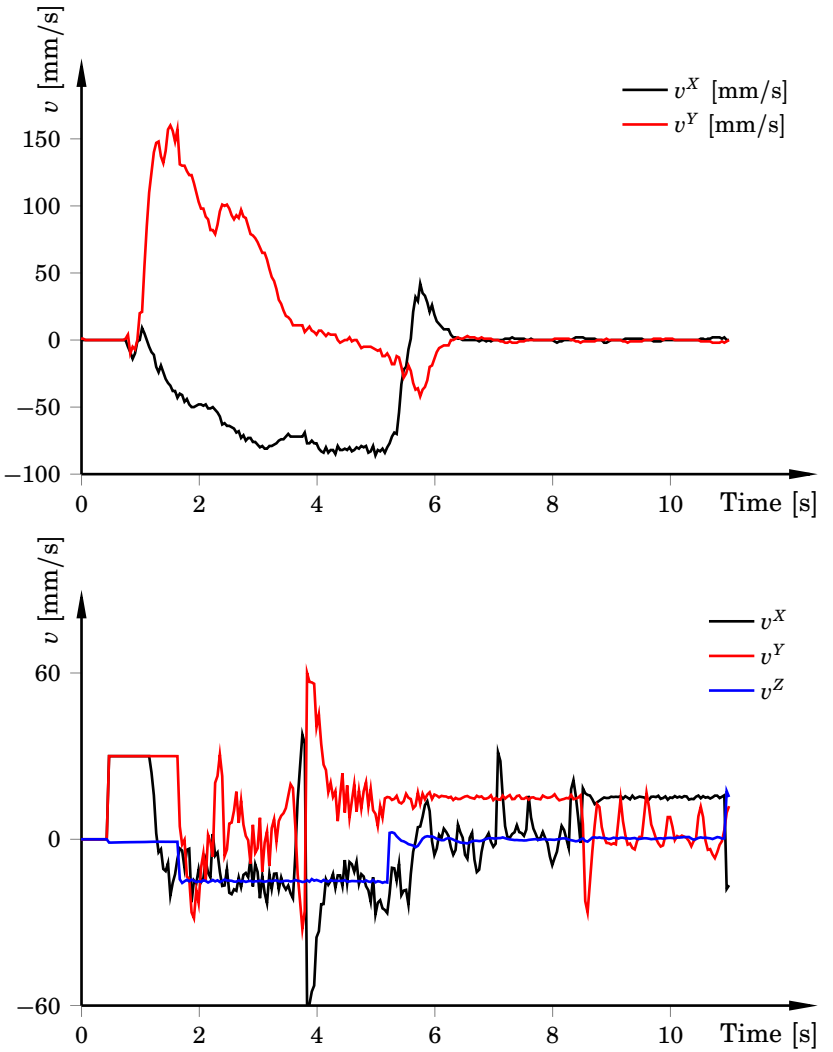

Figure 6.13 Velocity data for the mobile base and desired output velocities from an experiment. The upper plot shows velocities of the mobile base. The lower plot displays the control signals—that is, the modified desired output velocities in (6.18)—resolved in the mobile platform's coordinate system. The control signals for the ZYX-Euler angles are not showed, since they are practically zero throughout the experiment.

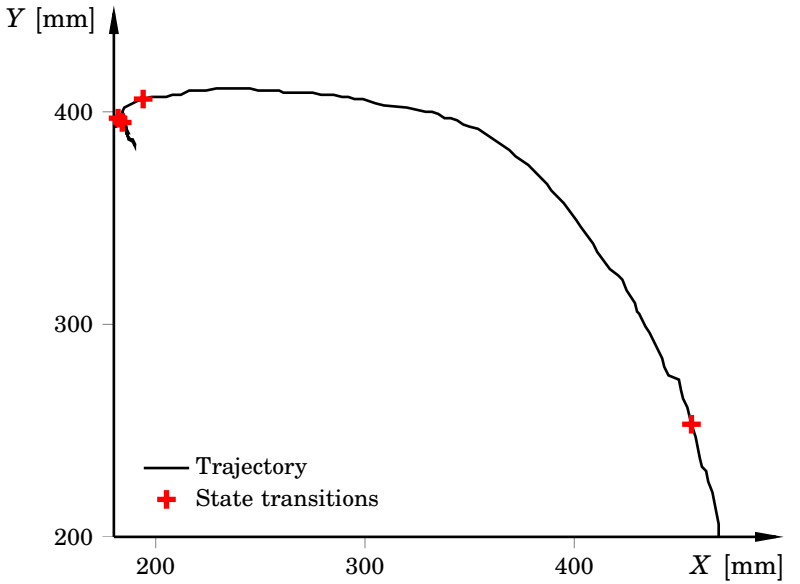

Figure 6.14 Traversed path of the mobile base together with the state transitions, shown as red +. The path is for the origin of frame o'_1 (i.e., the plate on the mobile base). The transitions start from state 1 up to state 6, all in all five transitions. The transitions to states 2 and 3 occur in the lower right part of the figure, whereas the other three transitions occur in the upper left part.

6.8 Concluding Remarks

The iTaSC framework proved to be suitable for the application at hand. A benefit is that it is well suited for specifying the task at a rather high level. The framework is more powerful than what was needed for our application, but it is easy to envision scenarios where both robot arms, possibly with more mobile robots involved, are used. We used Euler angles for specifying the task. They are intuitive to use, but a problem with Euler angles is that they have representation singularities. There are several solutions to remedy this. One solution is to use different parametrizations of the angle representation depending on the robot-system configuration. Another approach is to instead use quaternions [Hamilton, 1844], which can be incorporated into the framework, see [Stolt et al., 2012].

The robot coordination was done using a very crude approach; that is, the stationary manipulator was responsible for all high-precision motion. This is a common-sense method, which has been used before [Meeussen et al., 2010]. However, it is likely that involving both robots in the feedback process can improve performance.

The results indicate that a rather simple, linear constant-acceleration process model can be used for achieving estimation performance that is good enough for online scenarios. The only nonlinearity in the estimation model enters in the measurement equations, and it is a rather mild one considering that the rotation is assumed to occur in one dimension only. Still, to perform more integrated coordinated movements, the estimation performance must probably be improved. The estimation algorithm executed at 50 Hz, whereas the vision algorithm executed with approximately 2 Hz. Increasing the update rates will most probably improve performance. Another way to achieve better estimation performance is to use models that better capture the dynamic effects in the robots. As an example, if the robot system is modeled as one robot instead of two separate robots, it is possible to improve performance. The reason is that the force sensor contains information about the position of the robot's grippers. By incorporating the force-sensor measurements together with the forward kinematics of the stationary manipulator into the estimation framework, it is possible to extract more information about the position of the mobile base. In this chapter, the model used for estimation is based on rigid-body kinematics. This assumption is typically violated when performing manipulation, since the contact forces give rise to deflections. Thus, by taking this into consideration, improved estimation and control accuracy will be achieved. However, it is possible that incorporating more complex dynamics, and possibly more intricate noise distributions that more correctly model the eigenfrequencies in the system, gives a model that is not suitable for linear estimation techniques.

6.9 Summary

This chapter provided an example of the type of applications OOSMs occur in and how they can be dealt with. The method that accounted for the OOSMs assumes a linear model with Gaussian noise. This was achieved by using a planar, constant-acceleration model and linearizing the dynamics at each time step. Results showed that the estimated pose is smooth, with minor jumps in the estimates when the OOSM arrives.

We discussed a pick-and-place scenario where a mobile manipulator system manipulated objects. The setup was successfully verified in experiments. The scenario involved several aspects of vehicle modeling and control, being robot localization and position, velocity, and force control.

7

Particle Filters for Out-of-Sequence Processing

In many scenarios, the OOSM filters that assume linear dynamics perform very well. Chapter 6 provided an application where the estimation accuracy is good enough to successfully execute a relatively complex task. However, the vast majority of systems are not linear, and in many cases the nonlinearities cannot be well approximated by linearizations. As pointed out in Section 3.1, the particle filter is especially popular in positioning and navigation/tracking applications. In mobile robotics, the particle filter is used for localization [Fox et al., 2000] and in many of the state-of-the-art simultaneous localization and mapping (SLAM) algorithms [Thrun et al., 2006a], see [Montemerlo and Thrun, 2003; Hähnel et al., 2003] for different particle-filter approaches.

This chapter discusses how to account for OOSMs in particle filters. As mentioned in Chapter 5, it is not obvious how to update with the OOSM in a sensible way. Sometimes the OOSM update can even lead to worse performance than what would be achieved by simply discarding it. For some types of sensor configurations, it is particularly important how the update is done. Many OOSM filters only update the most recent estimate. This was what was done in Chapter 6. Only updating the most recent estimate is, however, not always the right thing to do when several OOSMs arrive interleaved with each other.[1]

Here, we adapt the optimal[2] complete in-sequence information (CISI) fixed-point smoother for linear systems, described in [Zhang and Bar-Shalom, 2012a], to nonlinear systems and use it in the context of particle filters. The CISI approach updates all states between the OOSM timestamp and the current time. This implies better performance than

[1] The problem for when several OOSMs arrive at the same time step is called the multi-OOSM problem. It was briefly discussed in Section 5.3.

[2] Optimal in the mean-square sense.

if only updating the current estimate. The rationale for our approach is that if the CISI method is optimal for linear systems, it should perform better for nonlinear systems than other linearized approaches. The proposed approach involves a seemingly minor, yet effective, extension to the storage-efficient particle filter (*SEPF*) reported in [Orguner and Gustafsson, 2008] and the storage-efficient particle filter with selective processing (*SEPF-GARP*) reported in [Liu et al., 2010]. We evaluate the proposed algorithm on a target-tracking example and compare the performance against related OOSM particle filters, in terms of average estimation accuracy, robustness, and execution time. The chapter ends with a discussion and by drawing some conclusions.

7.1 Related Work

Many of the research papers within OOSM processing were mentioned in Section 5.3. Several of them derive optimal OOSM solutions for linear, Gaussian systems. For the multi-OOSM problem; that is, when more than one of the OOSMs are interleaved, there are a handful of different solutions. There are not many optimal algorithms when it comes to particle-filter algorithms. In fact, there is only one, namely the approach described in [Zhang and Bar-Shalom, 2012b]. That algorithm is denoted by *A-PF*. For general nonlinear systems with white noise, and for a sufficiently large number of particles N, the distribution estimated by *A-PF* converges to the distribution that is estimated by a filter that has access to all measurements. The problem with *A-PF* is that it is computationally expensive; the computational complexity is $O(N^3)$, thus prohibiting online execution for anything but the smallest problems.

Chapter 8 will cover *A-PF* in more detail. For now, focus will be on particle-filter algorithms that are aimed at online execution. Approaches toward this direction are taken in, for example, [Orguner and Gustafsson, 2008] and [Liu et al., 2010], and we will go through them after a formal problem formulation.

7.2 Problem Formulation

The considered systems are on a similar form as (5.1):

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}_{k+1,k}(\boldsymbol{x}_k) + \boldsymbol{w}_{k+1,k}, \tag{7.1a}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{e}_k. \tag{7.1b}$$

The state-transition function $\boldsymbol{f}_{k+1,k} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ and the process noise $\boldsymbol{w}_{k+1,k}$ have been appended with explicit indexing for later use, but when

the indexing is not used they have the standard interpretation. In (7.1b), $\boldsymbol{h} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ is the measurement function, $\boldsymbol{e}_k$ is the measurement noise, and $\boldsymbol{y}_k$ is the measurement vector. Moreover, $\boldsymbol{w}_{k+1,k}$ and $\boldsymbol{e}_k$ are mutually independent, white, and Gaussian distributed according to

$$\begin{aligned} \boldsymbol{w}_{k+1,k} &\sim \mathcal{N}(\boldsymbol{0}_{n_x}, \boldsymbol{Q}_k), \\ \boldsymbol{e}_k &\sim \mathcal{N}(\boldsymbol{0}_{n_y}, \boldsymbol{R}_k). \end{aligned}$$

As in Section 5.2, the timestamp is referred to as t_k and t_k^a means the corresponding measurement arrival time. Denote the set of in-sequence measurements (ISMs) generated in the interval $[0, k]$ as $\mathcal{Y}_k$. Let $\mathcal{Z}_k$ denote the set of OOSMs available at time t_k. Then $\boldsymbol{y}_{0:k}$ means $\mathcal{Y}_k \cup \mathcal{Z}_{k-1}$, that is, the ISMs available at time t_k and the OOSMs available at time t_{k-1}.

A definition of OOSMs and ISMs was given in Definition 5.1. With this definition in mind, we define the notion of *most recent time* next:

Definition 7.1—[Zhang and Bar-Shalom, 2012a]
Given an OOSM $\boldsymbol{y}_\tau$, if

$$t_{m(\tau)} = \max\{t_k;\ \forall t_k,\ t_k^a < t_\tau^a\},$$

then $t_{m(\tau)}$ is the most recent time (MRT) corresponding to $\boldsymbol{y}_\tau$. □

What Definition 7.1 says is that the MRT is the largest timestamp of all measurements that arrived before the OOSM. As mentioned in Section 5.3, when discussing linear, Gaussian systems, it is only under special circumstances that it is enough to update the most recent mean and covariance and still have optimality. The case for when this approach is optimal is when the OOSM scenario is of *type I*, see [Zhang and Bar-Shalom, 2012a] for the theorem. The type I scenario is explained in Definition 7.2.

Definition 7.2—[Zhang and Bar-Shalom, 2012a]
If for any two OOSMs $\boldsymbol{y}_{\tau_1}$ and $\boldsymbol{y}_{\tau_2}$, where $\boldsymbol{y}_{\tau_1}$ arrives before $\boldsymbol{y}_{\tau_2}$ (i.e., $t_{\tau_1}^a < t_{\tau_2}^a$), we have that the most recent time corresponding to $\boldsymbol{y}_{\tau_1}$ is before the timestamp of $\boldsymbol{y}_{\tau_2}$ (i.e., $t_{m(\tau_1)} < t_{\tau_2}$), then the OOSM scenario is of type I. □

An interpretation is that an OOSM scenario is *not* of Type I if a measurement occurs after $\boldsymbol{y}_{\tau_2}$ but arrives before $\boldsymbol{y}_{\tau_1}$. An example of a type I OOSM scenario is shown in Figure 7.1. A scenario that does not fulfill the assumptions in Definition 7.2 is shown in Figure 7.2. The scenario in Figure 7.2 is not of type I since $\boldsymbol{y}_{\tau_2}$ arrives before $\boldsymbol{y}_{\tau_1}$, but the timestamp of $\boldsymbol{y}_{\tau_2}$ is after the MRT of $\boldsymbol{y}_{\tau_1}$. Only the CISI approach guarantees optimality when the OOSM scenario is not of type I, which is often the case when having more than one sensor producing OOSMs.

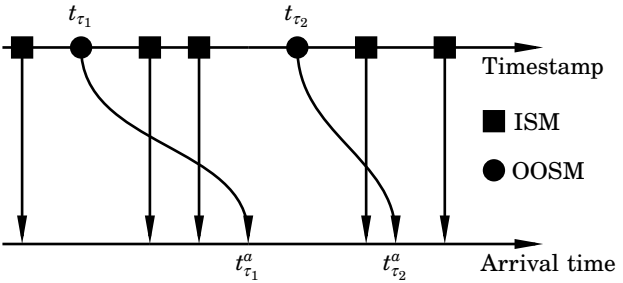

Figure 7.1 An example of the type I OOSM scenario when, for linear systems, it is optimal to use the OOSMs for updating the most recent estimate only. For an explanation of OOSMs and ISMs, see Definition 5.1.

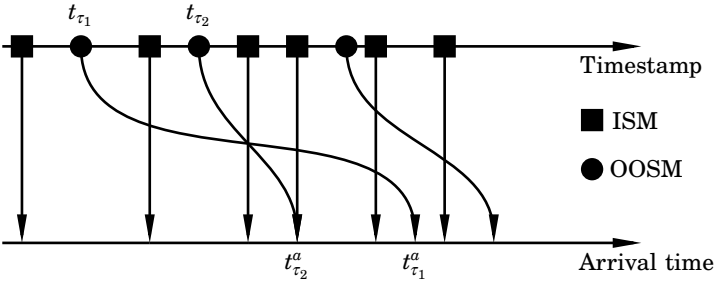

Figure 7.2 An example of an OOSM scenario which is not of type I, when one must update all states between the OOSM time and the current time to guarantee optimality.

Throughout the chapter we assume that the timestamp t_τ of each OOSM is bounded as $t_\tau \in [t_{k-l}, t_{k-l+1})$ for a positive integer l, where $l \in \{1, \ldots, l_{\max}\}$. The constant $l_{\max}$ is chosen a priori but we do not assume prior knowledge of the timestamps.

As stated in Section 5.2, assume that an estimate of the filtering posterior $p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k})$ is available. Then, assume that n OOSMs $\{\boldsymbol{y}_{\tau_1}, \ldots, \boldsymbol{y}_{\tau_n}\}$ arrive. The aim is to update the filtering posterior with the OOSMs; that is, the aim is to find

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k}, \boldsymbol{y}_{\tau_1}, \ldots, \boldsymbol{y}_{\tau_n}).$$

For simplicity in notation, in the remainder of this chapter, $\boldsymbol{y}_\tau$ is also used for denoting several OOSMs.

7.3 Particle Filters with Out-of-Sequence Measurements

The forward filter we use for processing of ISMs is based on Algorithm 3.2, with the addition that we at each time step k form the state and covariance estimates as

$$\hat{\boldsymbol{x}}_{k|k} = \sum_{i=1}^{N} w_k^i \boldsymbol{x}_k^i, \tag{7.2a}$$

$$\hat{\boldsymbol{P}}_{k|k} = \sum_{i=1}^{N} w_k^i (\boldsymbol{x}_k^i - \hat{\boldsymbol{x}}_{k|k})(\boldsymbol{x}_k^i - \hat{\boldsymbol{x}}_{k|k})^{\mathrm{T}}. \tag{7.2b}$$

These estimates are the minimum mean-square error estimates of the mean and associated covariances [Schön, 2006]. Moreover, we do conditional resampling according to (3.18) when the effective sample number in (3.19) fulfills $N_{\text{eff}} < 2N/3$. The algorithm is summarized in Algorithm 7.1.

Algorithm 7.1— Particle Filter with Conditional Resampling

1: **Initialize:** Set $\{\boldsymbol{x}_0^i\}_{i=1}^N \sim p_0(\boldsymbol{x}_0)$ and weights $\{w_0^i\}_{i=1}^N$.
2: **for** $k = 0$ **to** T **do**
3: Time update: Generate new particles from (7.1a):

$$\boldsymbol{x}_{k+1}^i \sim p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k^i), \quad \forall i \in \{1,\dots,N\}.$$

4: Measurement update: Compute weights according to

$$\bar{w}_k^i \propto p(\boldsymbol{y}_k|\boldsymbol{x}_k^i) w_{k-1}^i, \quad \forall i \in \{1,\dots,N\}.$$

and normalize them as $w_k^i = \bar{w}_k^i / \sum_{i=1}^N \bar{w}_k^i$.
5: **if** $\left(\sum_{i=1}^N (w_k^i)^2\right)^{-1} < N_{\text{thr}}$ **then**
6: Resample according to $\mathrm{P}(\boldsymbol{x}_k^i = \boldsymbol{x}_k^j) = w_k^j, \quad \forall i \in \{1,\dots,N\}$.
7: **end if**
8: Compute mean and covariance from (7.2).
9: **end for**

Next, we briefly go through *SEPF* in [Orguner and Gustafsson, 2008] and its extension *SEPF-GARP* in [Liu et al., 2010].

Storage Efficient Particle Filter—SEPF

SEPF is motivated by that there are usually much more particles than measurements in a particle filter. Hence, particle storage should be avoided, if possible. The starting point is that measurements until time

t_k have been processed using a particle filter, such as the one in Algorithm 7.1. The $(k+1)$th measurement arrives delayed, with the timestamp t_τ bounded between two samples as $t_\tau \in [t_{k-l}, t_{k-l+1})$ for a positive integer l. Note that a type I scenario as depicted in Figure 7.1 is assumed.

SEPF utilizes Bayes' rule as

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k,\tau}) = \frac{p(\boldsymbol{y}_\tau|\boldsymbol{x}_k, \boldsymbol{y}_{0:k})}{p(\boldsymbol{y}_\tau|\boldsymbol{y}_{0:k})} p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k}). \tag{7.3}$$

Substitution of (3.14) on page 53 into (7.3) leads to

$$\begin{aligned} p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k,\tau}) &\approx \sum_{i=1}^{N} \frac{p(\boldsymbol{y}_\tau|\boldsymbol{x}_k^i, \boldsymbol{y}_{0:k})}{p(\boldsymbol{y}_\tau|\boldsymbol{y}_{0:k})} w_k^i \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^i) \\ &= \sum_{i=1}^{N} w_{k|k,\tau}^i \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^i), \end{aligned} \tag{7.4}$$

where the weights are generated by

$$w_{k|k,\tau}^i \propto p(\boldsymbol{y}_\tau|\boldsymbol{x}_k^i, \boldsymbol{y}_{0:k}) w_k^i. \tag{7.5}$$

SEPF computes the likelihood

$$p(\boldsymbol{y}_\tau|\boldsymbol{x}_k^i, \boldsymbol{y}_{0:k}) = \int p(\boldsymbol{y}_\tau|\boldsymbol{x}_\tau) p(\boldsymbol{x}_\tau|\boldsymbol{x}_k^i, \boldsymbol{y}_{0:k}) \, \mathrm{d}t$$

in two steps. First, it approximates $p(\boldsymbol{x}_\tau|\boldsymbol{x}_k^i, \boldsymbol{y}_{0:k})$ with an extended Kalman fixed-point smoother [Biswas and Mahalanabis, 1973], where $\boldsymbol{x}_k^i$ is treated as a measurement. Second, it approximates $p(\boldsymbol{y}_\tau|\boldsymbol{x}_\tau)$ using an extended Kalman-filter (EKF) approach, which enables computation of (7.5).

Sometimes *SEPF* suffers from that the OOSM update can lead to severe mismatch between filter weights and updated weights, which drastically reduces the effective sample number N_{eff}. *SEPF* suppresses this problem by computing the approximate effective sample number (3.19) before and after the OOSM update, $N_{\text{eff}}^{\text{pri}}$ and $N_{\text{eff}}^{\text{post}}$, respectively. If

$$\frac{N_{\text{eff}}^{\text{post}}}{N_{\text{eff}}^{\text{pri}}} < \gamma_2, \tag{7.6}$$

where $\gamma_2 \ll 1$, the OOSM is discarded. This leads to that OOSMs that reduce the effective sample number by more than a factor γ_2 are dispatched.

REMARK 7.1
SEPF processes single OOSMs. In the multi-OOSM problem, as we consider, the smoothing and corresponding weight update is applied sequentially when several OOSMs arrive simultaneously. □

Selective Processing—SEPF-GARP

A more systematic approach, compared with (7.6), to decide which OOSMs to neglect and which to process, was derived in [Liu et al., 2010]. The approach uses either the mutual-information metric or the Kullback-Leibler divergence metric [Kullback and Leibler, 1951] to find out how informative the delayed measurement is in relation to the state. The former approach is summarized here. The mutual information between the measurement $\boldsymbol{y}_\tau$ and the state $\boldsymbol{x}_k$ is defined as

$$I(\boldsymbol{y}_\tau, \boldsymbol{x}_k|\boldsymbol{y}_{0:k}) := \int \log\left(\frac{p(\boldsymbol{y}_\tau, \boldsymbol{x}_k|\boldsymbol{y}_{0:k})}{p(\boldsymbol{y}_\tau|\boldsymbol{y}_{0:k})p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k})}\right) p(\boldsymbol{y}_\tau, \boldsymbol{x}_k|\boldsymbol{y}_{0:k})\, \mathrm{d}\boldsymbol{y}_\tau \mathrm{d}\boldsymbol{x}_k. \quad (7.7)$$

Thus, to compute (7.7) it is enough to find the joint distribution $p(\boldsymbol{y}_\tau, \boldsymbol{x}_k|\boldsymbol{y}_{0:k})$, which is done by approximating it as a Gaussian distribution:

$$p(\boldsymbol{y}_\tau, \boldsymbol{x}_k|\boldsymbol{y}_{0:k}) \approx \mathcal{N}\left(\begin{bmatrix}\boldsymbol{y}_\tau \\ \boldsymbol{x}_k\end{bmatrix} \middle| \begin{bmatrix}\hat{\boldsymbol{y}}_{\tau|\tau} \\ \hat{\boldsymbol{x}}_{k|k}\end{bmatrix}, \begin{bmatrix}\boldsymbol{R}_{\boldsymbol{y}_\tau} & \boldsymbol{R}_{\boldsymbol{y}_\tau \boldsymbol{x}_k} \\ \boldsymbol{R}_{\boldsymbol{x}_k \boldsymbol{y}_\tau} & \boldsymbol{R}_{x_k}\end{bmatrix}\right),$$

where $\hat{\boldsymbol{y}}_{\tau|\tau} = \boldsymbol{h}(\hat{\boldsymbol{x}}_{\tau|\tau})$ and $\hat{\boldsymbol{x}}_{k|k}$ is found from (7.2a). The resulting mutual information is approximated with

$$I(\boldsymbol{y}_\tau, \boldsymbol{x}_k|\boldsymbol{y}_{0:k}) \approx \hat{I}(\boldsymbol{y}_\tau, \boldsymbol{x}_k|\boldsymbol{y}_{0:k}) = \frac{1}{2}\log\left(\frac{\|\boldsymbol{R}_{\boldsymbol{x}_k}\|}{\|\boldsymbol{R}_{\boldsymbol{x}_k} - \boldsymbol{R}_{\boldsymbol{x}_k \boldsymbol{y}_\tau}\boldsymbol{R}_{\boldsymbol{y}_\tau}^{-1}\boldsymbol{R}_{\boldsymbol{y}_\tau \boldsymbol{x}_k}\|}\right). \quad (7.8)$$

The resulting filter with selective processing works as follows: When an OOSM arrives, (7.8) is computed. The involved covariance matrices and means are found through repeated EKF recursions from time t_τ to the current time t_k on a system formed by augmenting the state $\boldsymbol{x}_k$ with the measurement $\boldsymbol{y}_\tau$. It is initialized with the estimated mean and covariance at time t_τ given by (7.2); that is, it is initialized with

$$\begin{aligned}
\boldsymbol{R}_{\boldsymbol{x}_\tau} &= \boldsymbol{P}_{\tau|\tau}, \\
\boldsymbol{R}_{\boldsymbol{y}_\tau} &= \boldsymbol{C}_\tau \boldsymbol{P}_{\tau|\tau} \boldsymbol{C}_\tau^{\mathrm{T}} + \boldsymbol{R}_\tau, \\
\boldsymbol{R}_{\boldsymbol{y}_\tau \boldsymbol{x}_\tau} &= \boldsymbol{R}_{\boldsymbol{x}_\tau \boldsymbol{y}_\tau}^{\mathrm{T}} = \boldsymbol{C}_\tau \boldsymbol{R}_{\boldsymbol{x}_\tau}, \\
\boldsymbol{C}_\tau &= \left.\frac{\partial \boldsymbol{h}_\tau(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_k}\right|_{\hat{\boldsymbol{x}}_{\tau|\tau}}.
\end{aligned}$$

If the resulting mutual-information approximation (7.8) is below a threshold γ_1, the OOSM is discarded. If the mutual-information approximation (7.8) is above γ_1, meaning that

$$\hat{I}(\boldsymbol{y}_\tau, \boldsymbol{x}_k|\boldsymbol{y}_{0:k}) > \gamma_1, \quad (7.9)$$

the OOSM is deemed informative and is thus processed. The algorithm proceeds with executing *SEPF* from Section 7.3. If condition (7.6) holds; that is, if

$$\frac{N_{\text{eff}}^{\text{post}}}{N_{\text{eff}}^{\text{pri}}} < \gamma_2,$$

then *SEPF* terminates. Instead, a particle filter that executes from time index $k-l$ to the current time is activated, involving all (ordered) measurements, including $\boldsymbol{y}_\tau$, from time index $k-l$ to time index k. This rerun particle filter, denoted Gaussian-approximation rerun particle filter (*OOSM-GARP*), is initialized with the estimated mean and covariance at time index $k-l$. The thresholds γ_1 and γ_2 should be seen as governing the tradeoff between complexity and accuracy. The rerun particle filter is typically much more computationally demanding than *SEPF*.

REMARK 7.2
One of the assumptions *SEPF-GARP* makes is that the measurements can be easily ordered. The data association (see Section 5.2) itself can be a demanding problem in some applications [Maskell et al., 2006], and thus a rerun filter should be avoided. Also, a rerun filter has relatively high computational complexity. □

SEPF for the Multi-OOSM Problem—PF-CISI(MI)

We now present a version of *SEPF* that not only updates the estimate at the latest time step, but also all estimates between the OOSM time t_τ and the current time t_k. Assume that state estimates $\hat{\boldsymbol{x}}_{j|j}$, for all $j \in \{k-l_{\max}, \ldots, k\}$, and their associated covariances $\boldsymbol{P}_{j|j}$, computed from (7.2), are stored and available. The extension of *SEPF* with the procedure from [Zhang and Bar-Shalom, 2012a] proceeds as follows: The OOSM weight update (7.5) is also here approximated with a Gaussian density:

$$p(\boldsymbol{y}_\tau|\boldsymbol{x}_k^i, \boldsymbol{y}_{0:k}) \approx \mathcal{N}(\boldsymbol{y}_\tau|\hat{\boldsymbol{y}}_{\tau|k}^i, \Sigma_{\tau|k}). \tag{7.10}$$

The mean $\hat{\boldsymbol{y}}_{\tau|k}^i$ is found by inserting a fixed-point smoothing estimate of the state in (7.1b), whereas $\boldsymbol{\Sigma}_{\tau|k}$ is found by the corresponding EKF measurement update as in

$$\boldsymbol{\Sigma}_{\tau|k} = \boldsymbol{C}_\tau \boldsymbol{P}_{\tau|k} \boldsymbol{C}_\tau^{\mathrm{T}} + \boldsymbol{R}_\tau.$$

Since a Gaussian approximation is used, it suffices to propagate means and covariances. Hence, when an OOSM with timestamp t_τ arrives, we predict the estimate $\hat{\boldsymbol{x}}_{k-l|k-l}$ forward to time index τ. This gives $\hat{\boldsymbol{x}}_{\tau|k-l}$ with the associated covariance as

$$\boldsymbol{P}_{\tau|k-l} = \boldsymbol{A}_{\tau,k-l} \boldsymbol{P}_{k-l|k-l} \boldsymbol{A}_{\tau,k-l}^{\mathrm{T}} + \boldsymbol{Q}_{\tau,k-l}, \tag{7.11}$$

where $\boldsymbol{A}_{\tau,k-l}$ is found by linearization:

$$\boldsymbol{A}_{\tau,k-l} = \left.\frac{\partial \boldsymbol{f}_{\tau,k-l}(\boldsymbol{x}_{k-l})}{\partial \boldsymbol{x}_{k-l}}\right|_{\hat{\boldsymbol{x}}_{k-l|k-l}}.$$

For $j = k-l+1,\ldots,k-1$, the OOSM-updated mean and covariance in each time step are recursively found as

$$\begin{aligned}\hat{\boldsymbol{x}}_{j|j,\tau} &= \hat{\boldsymbol{x}}_{j|j} + \boldsymbol{K}_j(\boldsymbol{y}_\tau - \boldsymbol{h}_\tau(\hat{\boldsymbol{x}}_{\tau|j}))\\ \boldsymbol{P}_{j|j,\tau} &= \boldsymbol{P}_{j|j} - \boldsymbol{K}_j \boldsymbol{S}_\tau \boldsymbol{K}_j^{\mathrm{T}},\end{aligned} \tag{7.12}$$

where

$$\begin{aligned}\boldsymbol{C}_\tau &= \left.\frac{\partial \boldsymbol{h}_\tau(\boldsymbol{x}_\tau)}{\partial \boldsymbol{x}_\tau}\right|_{\hat{\boldsymbol{x}}_{\tau|j}},\\ \boldsymbol{S}_\tau &= \boldsymbol{C}_\tau \boldsymbol{P}_{\tau|j} \boldsymbol{C}_\tau^{\mathrm{T}} + \boldsymbol{R}_\tau,\\ \boldsymbol{K}_j &= \boldsymbol{P}_{j,\tau|j} \boldsymbol{C}_\tau^{\mathrm{T}} \boldsymbol{S}_\tau^{-1},\end{aligned}$$

and where $\boldsymbol{P}_{j,\tau|j}$ is the cross-covariance between the states at time index j and τ. To compute (7.12), the smoothed estimates and covariances as well as the cross-covariances are needed. If $j = k-l+1$, the quantities are given by

$$\begin{aligned}\hat{\boldsymbol{x}}_{\tau|j} &= \hat{\boldsymbol{x}}_{\tau|j-1} + \boldsymbol{V}_j \boldsymbol{P}_{j|j-1}^{-1}(\hat{\boldsymbol{x}}_{j|j} - \hat{\boldsymbol{x}}_{j|j-1})\\ \boldsymbol{P}_{\tau|j} &= \boldsymbol{P}_{\tau|j-1} - \boldsymbol{V}_j \boldsymbol{P}_{j|j-1}^{-1}(\boldsymbol{P}_{j|j-1} - \boldsymbol{P}_{j|j})\boldsymbol{P}_{j|j-1}^{-1} V_j^{\mathrm{T}}\\ \boldsymbol{P}_{j,\tau|j} &= \boldsymbol{P}_{j|j} \boldsymbol{P}_{j|j-1}^{-1} \boldsymbol{V}_j^{\mathrm{T}}\\ \boldsymbol{V}_j &= \boldsymbol{P}_{\tau|j-1} \boldsymbol{A}_{j,\tau}^{\mathrm{T}}.\end{aligned} \tag{7.13}$$

For $j = k-l+2,\ldots,k-1$, the quantities are instead given by

$$\begin{aligned}\hat{\boldsymbol{x}}_{\tau|j} &= \hat{\boldsymbol{x}}_{\tau|j-1} + \boldsymbol{V}_j \boldsymbol{P}_{j|j-1}^{-1}(\hat{\boldsymbol{x}}_{j|j} - \hat{\boldsymbol{x}}_{j|j-1})\\ \boldsymbol{P}_{\tau|j} &= \boldsymbol{P}_{\tau|j-1} - \boldsymbol{V}_j \boldsymbol{P}_{j|j-1}^{-1}(\boldsymbol{P}_{j|j-1} - \boldsymbol{P}_{j|j})\boldsymbol{P}_{j|j-1}^{-1} \boldsymbol{V}_j^{\mathrm{T}}\\ \boldsymbol{P}_{j,\tau|j} &= \boldsymbol{P}_{j|j} \boldsymbol{P}_{j|j-1}^{-1} \boldsymbol{V}_j^{\mathrm{T}}\\ \boldsymbol{V}_j &= \boldsymbol{P}_{j-1,\tau|j-1} \boldsymbol{A}_{j,j-1}^{\mathrm{T}}.\end{aligned} \tag{7.14}$$

At the last time step, the approach of using the particles $\{\boldsymbol{x}_k^i\}_{i=1}^N$ as measurements is adopted [Orguner and Gustafsson, 2008]. This means that at the last step, $j = k$, $\hat{\boldsymbol{x}}_{\tau|j}$ is updated using the Kalman smoother formulae,

see [Biswas and Mahalanabis, 1973], according to

$$\begin{aligned}
\hat{\boldsymbol{x}}^i_{\tau|k,i} &= \hat{\boldsymbol{x}}_{\tau|k-1} + \boldsymbol{K}_k \left(\hat{\boldsymbol{x}}^i_k - \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1|k-1}) \right) \\
\boldsymbol{P}_{\tau|k,i} &= \boldsymbol{P}_{\tau|k-1} - \boldsymbol{P}_{k-1,\tau|k-1} \boldsymbol{C}_k^T \boldsymbol{K}_k^T \\
\boldsymbol{K}_k &= \boldsymbol{P}_{k-1,\tau|k-1} \boldsymbol{C}_k^{\mathrm{T}} \boldsymbol{S}_k^{-1} \\
\boldsymbol{S}_k &= \boldsymbol{C}_k \boldsymbol{P}_{k-1|k-1} \boldsymbol{C}_k^{\mathrm{T}} + \boldsymbol{Q}_k \\
\boldsymbol{C}_k &= \left. \frac{\partial \boldsymbol{f}(\boldsymbol{x}_{k-1})}{\partial \boldsymbol{x}_{k-1}} \right|_{\hat{\boldsymbol{x}}_{k-1|k-1}},
\end{aligned} \tag{7.15}$$

for all $i \in \{1, \ldots, N\}$, where $\hat{\boldsymbol{x}}^i_{\tau|k,i}$ means the ith smoothed estimate given measurements up to time t_{k-1} and particle i at time t_k. Note that a single smoother executes to time t_{k-1}. At the last time step, N smoothed estimates are computed using the particles at time t_k.

Computational Considerations Compared with *SEPF* and *SEPF-GARP*, for each $j \in \{k-l, \ldots, k-1\}$, an additional linearization of the measurement equation (7.1b) at the estimate $\hat{\boldsymbol{x}}_{\tau|j}$ for use in the OOSM update (7.12) has to be performed. Furthermore, for $j = k-l+1$ and $j = k-l+2, \ldots, k-1$, (7.13) and (7.14) need the linearization of the system dynamics (7.1a) at $\hat{\boldsymbol{x}}_{j-1|j-1}$ to propagate the covariances. In addition, in each step the algorithm needs the inverses of $\boldsymbol{S}_\tau$ (in (7.12)) and $\boldsymbol{P}_{j|j-1}$ (in (7.13) and (7.14)). Thus, compared with *SEPF* and *SEPF-GARP*, additional matrix operations are performed in each step. The impact of these additional operations of course depends on the system size and the amount of resources available, but for systems of moderate size the difference should be small.

Algorithm Summary Algorithm 7.2 contains a summary of the resulting algorithm, which we denote by *PF-CISIMI*. In this version, the second condition on line 5 is always true. We have also implemented the algorithm without mutual-information computations (i.e., the second condition on line 5 is always false). This algorithm is denoted by *PF-CISI*. When several OOSMs arrive simultaneously, the algorithms are applied sequentially. The storage requirements of the algorithm are

$$\{\hat{\boldsymbol{x}}_{j|j}, \hat{\boldsymbol{P}}_{j|j}\}_{j=k-l_{\max}}^{k-1}.$$

Thus, the smoother does not need the measurements but instead propagates the stored means and covariances.

Algorithm 7.2—*PF-CISIMI* and *PF-CISI*

Input: $\{\{\boldsymbol{x}_k^i, w_k^i\}_{i=1}^N, \{\hat{\boldsymbol{x}}_{j|j}, \hat{\boldsymbol{P}}_{j|j}\}_{j=k-l}^k, \{\boldsymbol{y}_{\tau_m}\}_{m=1}^n\}$

When n OOSMs arrive, sort them with largest delay first.
Set `accept` $= 0$.
for $m = 1$ **to** n **do**
  Set $\boldsymbol{y}_\tau = \boldsymbol{y}_{\tau_m}$.
  if $I(\boldsymbol{y}_\tau, \boldsymbol{x}_k | \boldsymbol{y}_{0:k}) < \gamma_1$ and *PF-CISIMI* **then**
    Discard $\boldsymbol{y}_\tau$.
  else
    Set `accept` $= 1$.
    Predict up to time t_τ, yielding $\hat{\boldsymbol{x}}_{\tau|k-l}$ and $\boldsymbol{P}_{\tau|k-l}$.
    for $j = k-l+1$ **to** $k-1$ **do**
      if $j = k-l+1$ **then**
        Compute $\hat{\boldsymbol{x}}_{\tau|j}$, $\boldsymbol{P}_{\tau|j}$, and $\boldsymbol{P}_{j,\tau|j}$ using (7.13).
      else
        Compute $\hat{\boldsymbol{x}}_{\tau|j}$, $\boldsymbol{P}_{\tau|j}$, and $\boldsymbol{P}_{j,\tau|j}$ using (7.14).
      end if
      Compute $\hat{\boldsymbol{x}}_{j|j,\tau}$ and $\boldsymbol{P}_{j|j,\tau}$ using (7.12).
    end for
    for $i = 1$ **to** N **do**
      Compute $\hat{\boldsymbol{x}}_{\tau|k,i}^i$ and $\boldsymbol{P}_{\tau|k,i}$ by applying (7.15).
      Apply (7.10) and update weights using (7.5).
    end for
    Compute $N_{\text{eff}}^{\text{post}} = \left(\sum_{i=1}^N (w_{k|k,\tau}^i)^2\right)^{-1}$.
    if $N_{\text{eff}}^{\text{post}} < \gamma_2 N_{\text{eff}}^{\text{pri}}$ **then**
      discard $\boldsymbol{y}_\tau$.
    end if
  end if
end for
if `accept` **then**
  Form the new estimates from (7.2).
end if

Output: $\{\{\hat{\boldsymbol{x}}_{j|j,\tau}, \boldsymbol{P}_{j|j,\tau}\}_{j=k-l+1}^{k-1}, \{w_{k|k,\tau}^i\}_{i=1}^N\}$

Note that the differences when comparing with *SEPF* and *SEPF-GARP* are in the implementation of the EKS:

- All estimates between the OOSM time and the current time are updated.
- *OOSM-GARP* (i.e., the rerun filter) is not used as a last resort when both γ_1 (mutual-information threshold) in (7.9) and γ_2 (threshold for effective sample number) in (7.6) are violated.

REMARK 7.3
In lines 23–25 of Algorithm 7.2 we discard the OOSM if the effective sample number changes the distribution much, instead of also executing *OOSM-GARP*. This improves execution speed. □

7.4 Numerical Results

This section validates Algorithm 7.2 using root-mean-square errors (RMSEs) and compares it against a number of different particle filters, among them *SEPF* and *SEPF-GARP*. Let $\boldsymbol{x}_{k,j}$ and $\hat{\boldsymbol{x}}_{k,j}$ denote the true and estimated state vector, respectively, at time index k of the jth of K Monte-Carlo simulations. The RMSE of the state vector at time index k is then computed as

$$\text{RMSE}_k = \sqrt{\frac{1}{K}\sum_{j=1}^{K}\|\boldsymbol{x}_{k,j} - \hat{\boldsymbol{x}}_{k,j}\|^2}. \tag{7.16}$$

The time-averaged RMSE is found by computing the mean of the RMSE.

Simulation Model

The simulation example is similar to the ones used in [Orguner and Gustafsson, 2008] and [Liu et al., 2010]. A target moves in a plane, with the motion modeled as a five-state coordinated-turn model [Bar-Shalom et al., 2001; Gustafsson, 2010b]. The geometric path forms a circle of radius 500 m. The initial position $\boldsymbol{p}_0$ and velocity $\boldsymbol{v}_0$ are set to

$$\boldsymbol{p}_0 = \begin{bmatrix}-500 & 500\end{bmatrix}^{\text{T}}, \quad \boldsymbol{v}_0 = \begin{bmatrix}0 & 55\end{bmatrix}^{\text{T}}, \tag{7.17}$$

expressed in m and m/s, respectively. The motion lasts for 40 seconds. By introducing position, velocity, and turn rate $\dot{\psi}_k$ as states,

$$\boldsymbol{x}_k = \begin{bmatrix}p_k^X & p_k^Y & v_k^X & v_k^Y & \dot{\psi}_k\end{bmatrix}^{\text{T}},$$

the discrete-time model for the coordinated turn is

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} 1 & 0 & \dfrac{\sin(\dot{\psi}_k T_s)}{\dot{\psi}_k} & -\dfrac{1-\cos(\dot{\psi}_k T_s)}{\dot{\psi}_k} & 0 \\ 0 & 1 & \dfrac{1-\cos(\dot{\psi}_k T_s)}{\dot{\psi}_k} & \dfrac{\sin(\dot{\psi}_k T_s)}{\dot{\psi}_k} & 0 \\ 0 & 0 & \cos(\dot{\psi}_k T_s) & -\sin(\dot{\psi}_k T_s) & 0 \\ 0 & 0 & \sin(\dot{\psi}_k T_s) & \cos(\dot{\psi}_k T_s) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_k + \boldsymbol{w}_k.$$

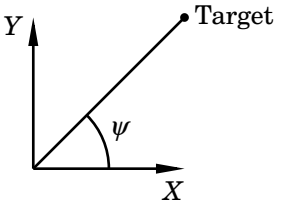

Figure 7.3 Bearing sensors measure the angle ψ relative to the object.

This model is nonlinear in $\dot{\psi}_k$. The process noise $\boldsymbol{w}_k$ is white and Gaussian distributed, with zero mean and covariance matrix

$$\boldsymbol{Q}_k = \text{diag}\left(\begin{bmatrix} 30^2 & 30^2 & 10^2 & 10^2 & 0.1^2 \end{bmatrix}\right), \tag{7.18}$$

where diag(·) is the diagonal matrix. To track the target, three bearings-only sensors (Figure 7.3), common in tracking applications, send measurements to the fusion center. The sampling period is set to $T_s = 1$ s. The sensors are located at

$$S_1 = (-200, 0), \quad S_2 = (200, 0), \quad S_3 = (-750, 750).$$

The geometric path for a simulation is shown in Figure 7.4. In addition, the sensor locations are shown as red +. The measurement model for the bearings-only measurements is

$$\boldsymbol{y}_k = \boldsymbol{h}_k + \boldsymbol{e}_k = \begin{bmatrix} \arctan\left(\dfrac{p_k^Y - S_1^Y}{p_k^X - S_1^X}\right) \\ \arctan\left(\dfrac{p_k^Y - S_2^Y}{p_k^X - S_2^X}\right) \\ \arctan\left(\dfrac{p_k^Y - S_3^Y}{p_k^X - S_3^X}\right) \end{bmatrix} + \boldsymbol{e}_k.$$

The measurement noise $\boldsymbol{e}_k$ is white and Gaussian distributed, with zero mean and covariance matrix

$$\boldsymbol{R}_k = 0.05\boldsymbol{I}_{3\times 3}. \tag{7.19}$$

The process noise and measurement noise are mutually independent. In this setup, the second and third sensors have serious communication issues: For each of the two sensors, the probability P_{oosm} that a measurement arrives from a sensor is $\text{P}_{\text{oosm}} = 0.7$. In addition, if a measurement arrives, it has delay d. The delays are drawn randomly from a discrete uniform distribution in the interval $[0, 5]$ s. Thus, each of the two sensors

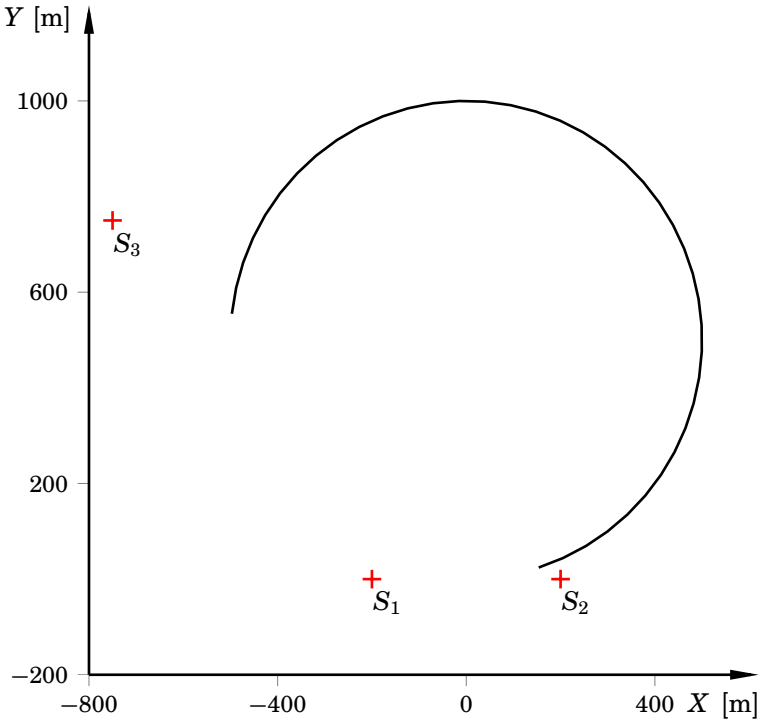

Figure 7.4 The geometric path and the sensor locations (red +) used in the simulation example. The path is taken from a 40 s simulation.

only successfully transmit 30% of the measurements to the communication center, and those measurements that arrive are delayed between zero and five seconds. The initial estimate $\hat{\boldsymbol{x}}$ is set to

$$\hat{\boldsymbol{x}}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$$

for all filters, with initial covariance $\boldsymbol{P}_0$ set to

$$\boldsymbol{P}_0 = \mathrm{diag}\left(\begin{bmatrix} 250^2 & 250^2 & 30^2 & 30^2 & 0.1^2 \end{bmatrix}\right). \tag{7.20}$$

When compared with the true initial state in (7.17) and the process noise distribution in (7.18), it is clear that the initial knowledge about the target's state is modest.

The thresholds for the mutual-information value in (7.9) and when to execute the rerun filter in *SEPF-GARP*, respectively, are set to $\gamma_1 = 0.05$ and $\gamma_2 = 0.025$. We use the same γ_1 and γ_2 in *PF-CISIMI*. Moreover, *SEPF* and *PF-CISI* use $\gamma_2 = 0.025$ for choosing when to discard the OOSMs.

Benchmarked Filters

Seven different particle filters are implemented in MATLAB, all based on Algorithm 7.1. All seven implementations are highly optimized, and take full advantage of vectorization techniques. The filters are:

- *PFIDEAL*: An offline, idealized particle filter that collects measurements from all sensors with zero delay. Better results than with this filter is not achievable, given the considered particle-filter algorithm.
- *PFDISC*: A particle-filter implementation that discards all OOSMs. Thus, it only processes the ISMs: It processes sensor S_1 every sample, but only S_2 and S_3 when they arrive with zero delay.
- *SEPF*: The storage-efficient particle filter derived in [Orguner and Gustafsson, 2008], see Section 7.3, page 119.
- *SEPF-GARP*: The storage-efficient particle filter with selective processing derived in [Liu et al., 2010], see Section 7.3, page 121.
- *OOSM-GARP*: The Gaussian-approximation rerun particle filter described in [Liu et al., 2010] and in Section 7.3. Usually *OOSM-GARP* is part of *SEPF-GARP*, but is here used as a standalone algorithm for OOSM processing. Hence, for every OOSM that arrives, the algorithm reorders and reprocesses all measurements.
- *PF-CISIMI*: The particle filter outlined in Algorithm 7.2, page 125.
- *PF-CISI*: The particle filter outlined in Algorithm 7.2 but without the mutual-information metric computation on line 5.

EKF versions of linear algorithms [Bar-Shalom et al., 2004; Zhang and Bar-Shalom, 2012a] were also implemented; however, those algorithms are omitted since they only sporadically converge for the considered example.

Results

Figure 7.5 displays the RMSE values for the position $\boldsymbol{p}_k$ when executing 2000 Monte-Carlo simulations. The lower plot is a zoomed-in version of the upper plot. In addition, Figure 7.6 shows the corresponding results for the velocity $\boldsymbol{v}_k$. All filters use $N = 2000$ particles. The first observation is that the filter that discards the OOSMs (*PFDISC*) has much worse performance than all other filters. This implies that it is not fruitful to discard all OOSMs in this example. *OOSM-GARP* has better convergence than the OOSM algorithms in the transient phase. Both *PF-CISI* and *PF-CISIMI* seem to have improved convergence rates compared with *SEPF*

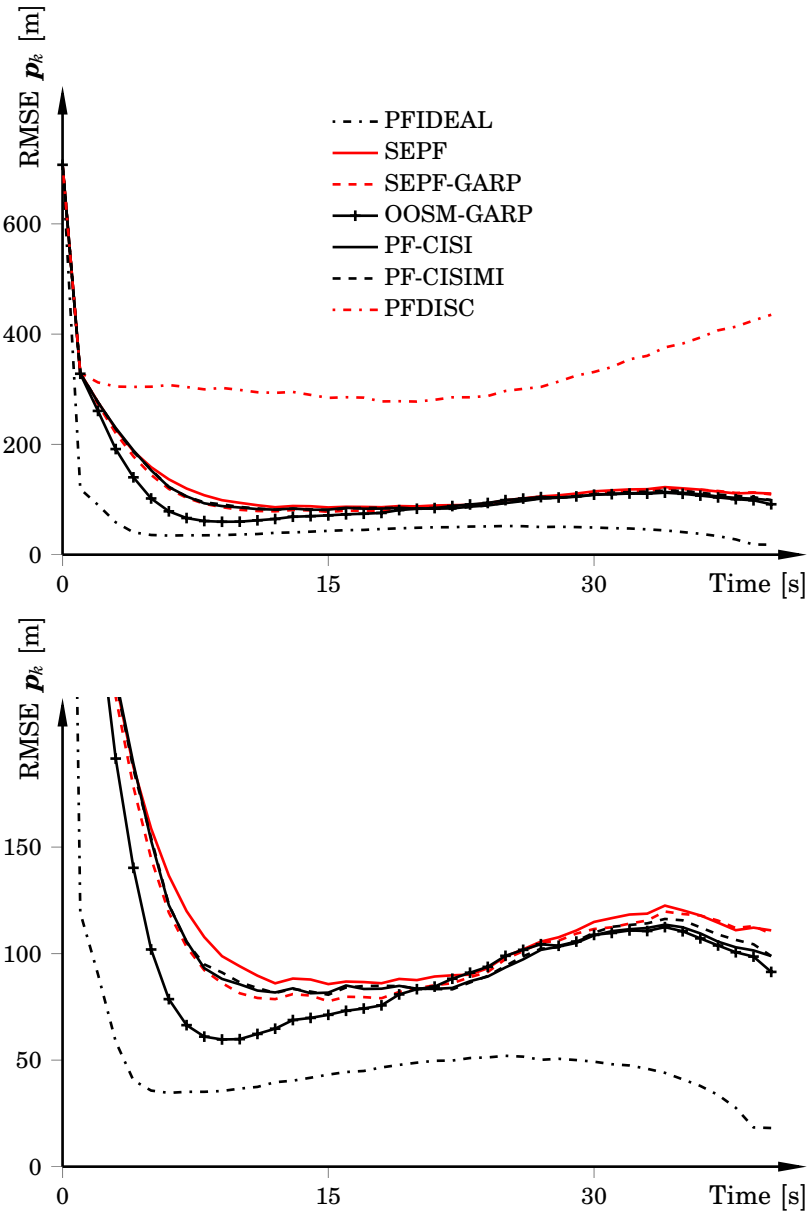

Figure 7.5 Position RMSE values for the different particle filters over 2000 Monte-Carlo simulations for 2000 particles. The lower plot is a zoomed in version of the upper.

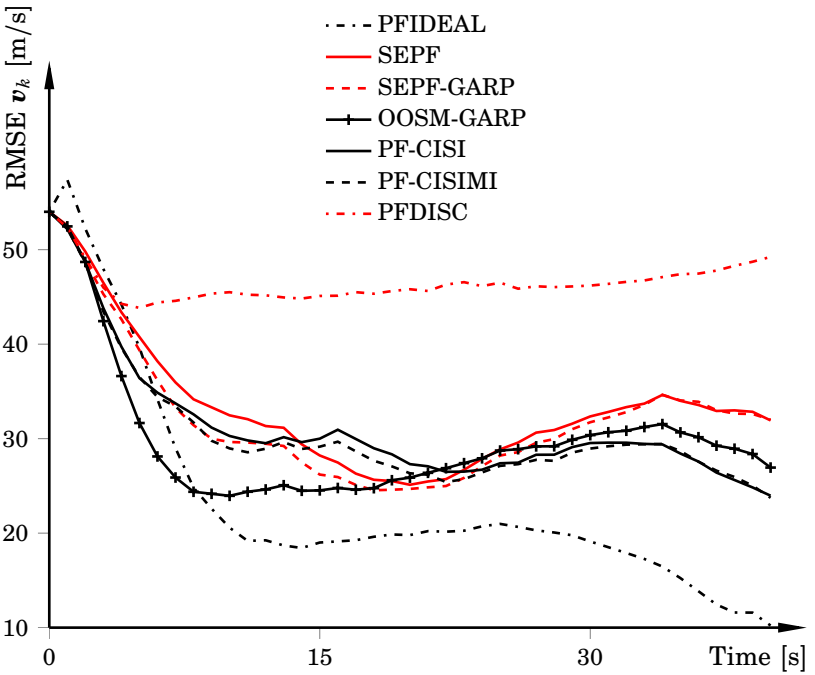

Figure 7.6 Velocity RMSE values for the different particle filters over 2000 Monte-Carlo simulations for 2000 particles. The CISI approaches *PF-CISI* and *PF-CISIMI* seem to have better convergence rate than the other OOSM algorithms.

and *SEPF-GARP*. Moreover, *PF-CISI* is very close in tracking performance to both selective-processing procedures (i.e., *PF-CISIMI* and *SEPF-GARP*). The reason for this can be found in Table 7.1. The mutual-information computation in *PF-CISIMI* discards approximately 20% of the OOSMs, but only 0.06% of the OOSMs that are deemed informative are discarded by the second threshold; that is, almost none of the OOSMs reduce the particle weights significantly when used in *PF-CISIMI*. The reason for this is probably that the CISI-based OOSM update in *PF-CISIMI* and *PF-CISI* is more robust than the update in *SEPF-GARP* and *SEPF*. This is validated by that *SEPF-GARP* discards about 21% of the OOSMs, which is approximately the same as for *PF-CISIMI*, but discards about six times more of the OOSMs (i.e., execute the rerun filter). Hence, the CISI-based OOSM update provides better correlation between the ISMs and the OOSMs.

The time-averaged RMSEs for the OOSM algorithms are shown in Table 7.2. In round numbers, the CISI approaches result in 5–8% performance improvements compared with *SEPF*.

Table 7.1 The middle column shows the percentage of the OOSMs that are discarded by the selective processing logic (see (7.9)). The right column shows the percentage of the OOSMs that are discarded by the effective sample computation (7.6). For *SEPF-GARP*, this also means the percentage of *OOSM-GARP* executions.

Algorithm	γ_1 [%]	γ_2 [%]
SEPF	-	0.9
SEPF-GARP	21.26	0.4
PF-CISI	-	0.07
PF-CISIMI	22.41	0.06

Table 7.2 Time-averaged RMSE values corresponding to Figures 7.5 and 7.6. The CISI approaches perform best of the OOSM algorithms, *OOSM-GARP* excluded. The smallest errors for the OOSM algorithms are indicated in bold font.

Algorithm	$\boldsymbol{p}$ [m]	$\boldsymbol{v}$ [m/s]
PFDISC	330.2	46.4
SEPF	133.7	33.3
SEPF-GARP	128.4	32.2
PF-CISI	127.9	31.3
PF-CISIMI	128.7	30.8
OOSM-GARP	**117.5**	**29.9**
PFIDEAL	61.7	23.6

Computational Complexity As mentioned, the MATLAB code is highly optimized, but for several reasons a comparison between execution times should be interpreted with care. A comparison between maximum computation times can be made by measuring the execution times and compute the maximum.[3] Because of memory management and operating system interruptions, this is not necessarily a good indication of the actual time taken by the algorithm.

Another procedure is to compute the average of the execution times over the simulation period for all 2000 Monte-Carlo simulations. Then an indication of the maximum execution time is the maximum of these 2000 simulations. The result will underestimate the actual maximum execution time—nevertheless, it provides some information. Table 7.3 shows the results after using the explained procedure. MATLAB's `tic` and `toc` functionality measured the execution times.

[3] The maximum time is interesting since it for real-time applications is important that the deadlines are met.

Table 7.3 Execution times, measured in milliseconds, for 2000 Monte-Carlo simulations using 2000 particles. The time is for executing the algorithm once (i.e., one time step), including the forward filter. The smallest execution time for the OOSM algorithms is indicated in bold font. The results are from a standard laptop equipped with an Intel i5 CPU.

Algorithm	Time [ms]
PFDISC	3.4
PFIDEAL	3.8
PF-CISIMI	**7.8**
SEPF	7.9
SEPF-GARP	8.6
PF-CISI	8.8
OOSM-GARP	11.6

We see that the CISI-based approach with selective processing (*PF-CISIMI*) has the shortest maximum time of all OOSM algorithms, and that *SEPF* has similar execution time. Moreover, *SEPF-GARP* has about 10% larger execution time than *PF-CISIMI*. This shows that *PF-CISIMI* is able to deliver tracking performance that is as good as *SEPF-GARP* while reducing computational demands. The rerun filter (*OOSM-GARP*) is by far the slowest. It is worth stressing that *SEPF-GARP* and *OOSM-GARP* in reality have larger maximum execution time (caused by the rerun filter), because the maximum of the *average* execution time for a whole sequence is here used as measure.

Outliers To give an indication of robustness, Figure 7.7 displays error-bar plots for *SEPF*, *SEPF-GARP*, and the proposed *PF-CISI*. Clearly, there are more outliers for *SEPF*. Moreover, *PF-CISI* has fewer outliers than *SEPF-GARP*. Note that *PF-CISIMI* (not shown) has about the same number of outliers as *PF-CISI*. The reason is that the OOSM algorithms are the same for these two filters. The only thing that *PF-CISIMI* benefits from is that some of the OOSMs that now are neglected by the mutual information calculation would otherwise possibly yield outliers.

20 000 Particles If we instead use 20 000 particles, the position RMSEs in Figure 7.8 and the corresponding velocity RMSEs in Figure 7.9 are obtained. The time-averaged RMSE values are shown in Table 7.4. When comparing with the results for 2000 particles (Figures 7.5–7.6 and Table 7.2), the performance of both CISI-based approaches have improved significantly. That *OOSM-GARP* has not improved much is an indication of that any further increase of the number of particles will have small impact on tracking performance.

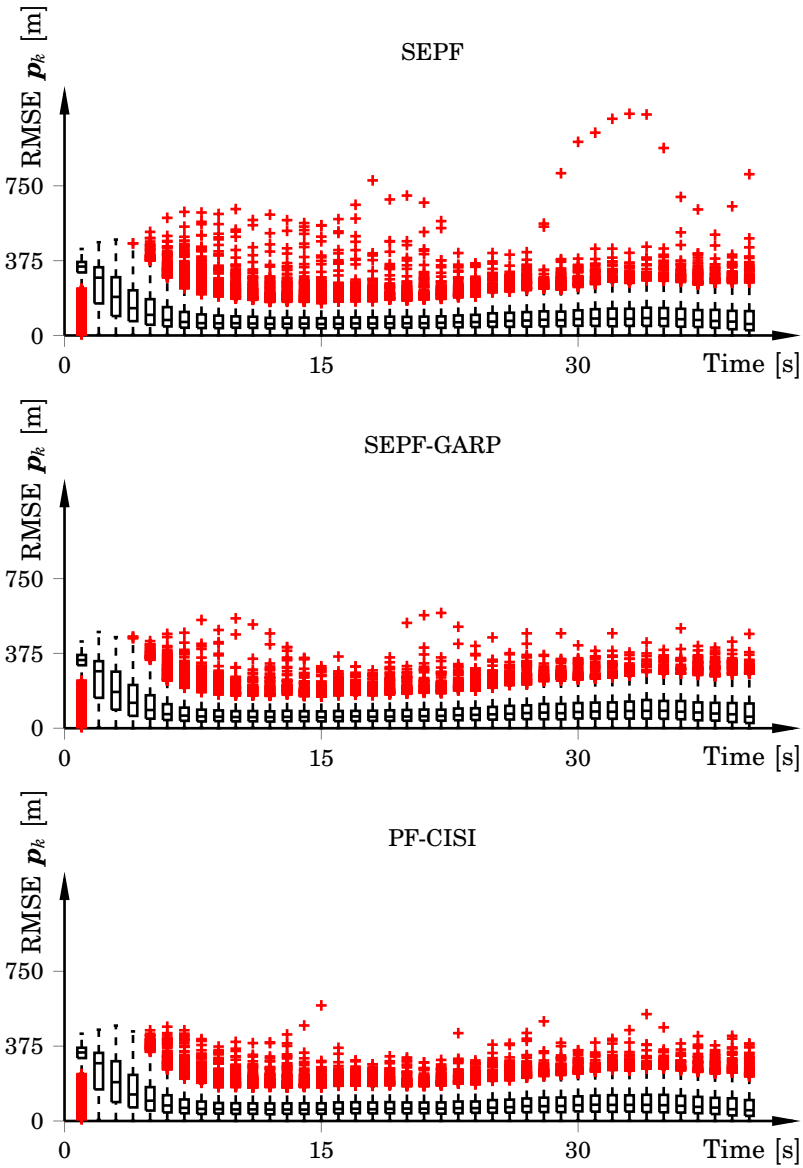

Figure 7.7 Error-bar plots. The boxes indicate the lower and upper quartiles and the medians. Outliers (red +) are position errors that are larger than approximately 2.7 standard deviations, corresponding to 99.3% coverage for Gaussian distributed data. Executing *PF-CISI* results in fewer outliers than for the two other algorithms.

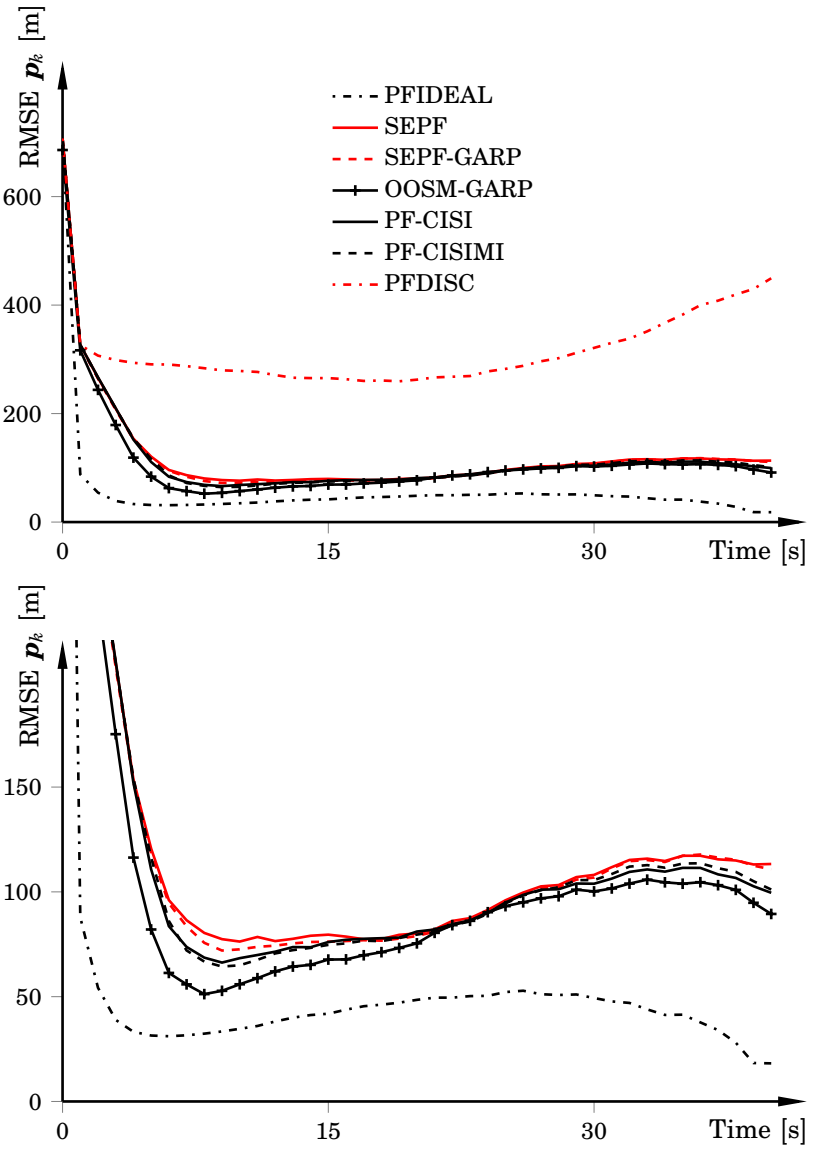

Figure 7.8 Position RMSE values for the different particle filters over 2000 Monte-Carlo simulations for 20 000 particles. Compared with Figure 7.5, where 2000 particles are used, *PF-CISI* and *PF-CISIMI* are now closer to *OOSM-GARP* in performance.

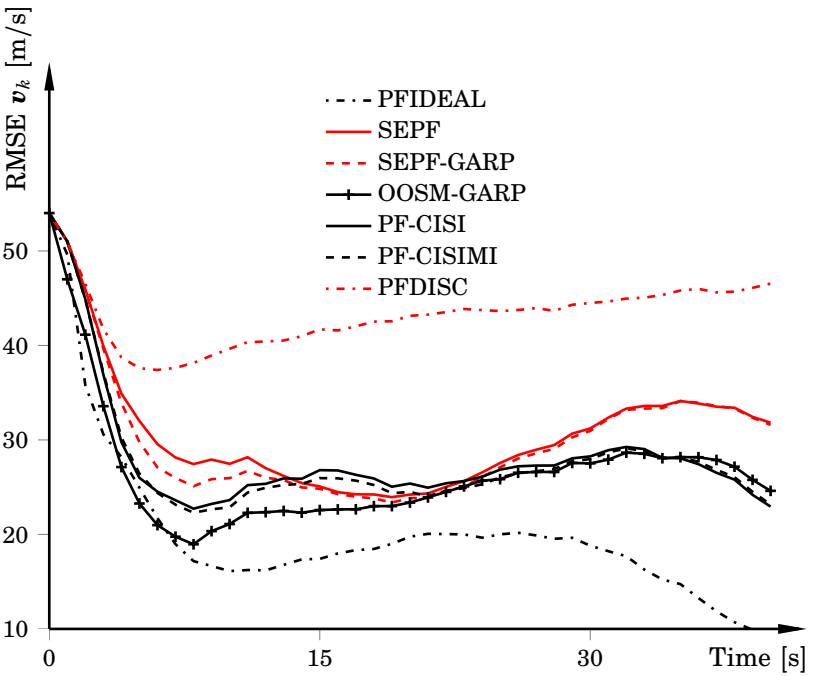

Figure 7.9 Velocity RMSE values for the different particle filters over 2000 Monte-Carlo simulations for 20 000 particles. Compared with Figure 7.6 where 2000 particles are used, *PF-CISI* and *PF-CISIMI* are now closer to *OOSM-GARP* in performance (and sometimes even better). Note that *OOSM-GARP* has not improved much compared with Figure 7.6, indicating that the tracking potential is close to be reached.

Table 7.4 Time-averaged RMSE values for 2000 Monte-Carlo simulations with 20 000 particles, corresponding to Figures 7.8 and 7.9. Compared with the case for 2000 particles, the CISI approaches have improved more than both *SEPF-GARP* and *OOSM-GARP*. The smallest errors for the OOSM algorithms are indicated in bold font.

Algorithm	$\boldsymbol{p}$ [m]	$\boldsymbol{v}$ [m/s]
PFDISC	318.4	43.2
SEPF	124.3	30.8
SEPF-GARP	122.8	30.1
PF-CISI	119.8	28.2
PF-CISIMI	120.1	**27.9**
OOSM-GARP	**113.4**	28.1
PFIDEAL	58.8	20.1

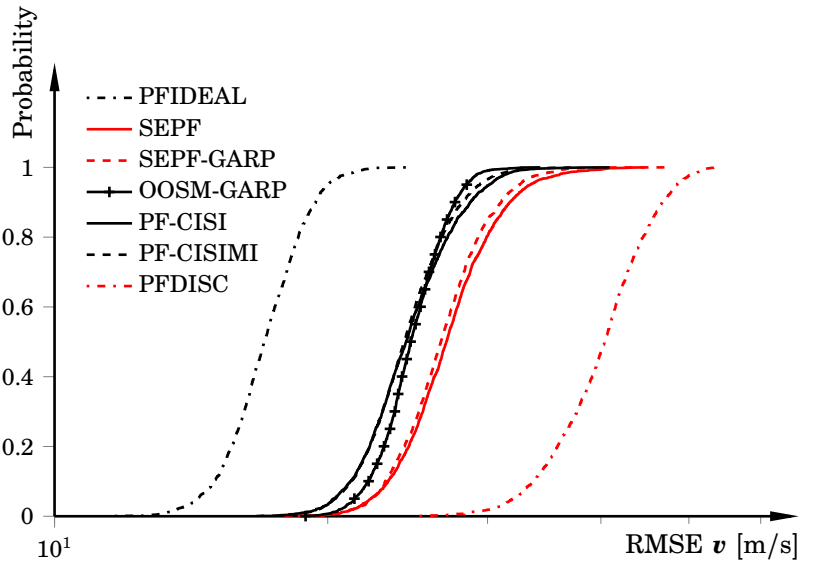

Figure 7.10 The probability of time-averaged RMSE for the velocity vector using the different filters, computed as a cumulative distribution function over the 2000 Monte-Carlo simulations. All filters use 20 000 particles. Both *PF-CISI* and *PF-CISIMI* are very similar to *OOSM-GARP* in performance. To quantify, the probability that the RMSE for *PF-CISIMI* is below 30 m/s is 95%. The corresponding numbers for *OOSM-GARP* and *SEPF-GARP* are 99% and 85%, respectively.

To further illustrate the robustness, Figure 7.10 shows the likelihood of the respective time-averaged RMSE, computed as a cumulative function over the 2000 Monte-Carlo simulations. This gives further measure of how reproducible the results are, in that it tells how large portion of the executions that are below a given error. Both *PF-CISI* and *PF-CISIMI* provide smaller errors for a given probability than the other OOSM algorithms, and are very close (and sometimes even better) to *OOSM-GARP* in performance. The RMSEs are more evenly distributed for *OOSM-GARP* compared with *PF-CISIMI* and *PF-CISI*, but the difference is small.

7.5 Conclusion

The results in Section 7.4 indicate that both *PF-CISI* and *PF-CISIMI* are more robust while maintaining execution-time effectiveness compared with *SEPF*. Similarly, the approaches maintain tracking accuracy (and improve the accuracy when using many particles) while decreasing execution time compared with *SEPF-GARP*. In addition, when the particle filter uses many particles, the CISI approaches seem to perform similarly to *OOSM-GARP*. This implies that the algorithms have higher tracking potential (for the considered multi-OOSM problem) than the other OOSM algorithms as the number of particles grows. Further, the algorithms seem to result in fewer outliers and more predictable results for the considered multi-OOSM scenario.

We validated the algorithms on one system only. The initial state knowledge, see (7.20), was assumed small. The measurement-noise covariance (7.19) was only 0.05 rad^2, whereas the position and velocity covariances (7.18) were 900 m^2 and 100 m^2/s^2, respectively. Hence, the uncertainty in the process dynamics is large. By using an algorithm that makes more use of information in the measurements, the resulting performance improvement will be relatively large. For systems that do not have large process uncertainty, the performance gain in using a CISI approach might be smaller.

7.6 Summary

This chapter presented an alternative implementation to the storage-efficient particle-filter algorithm for OOSM processing in [Liu et al., 2010], which in its turn builds on the results in [Orguner and Gustafsson, 2008]. In essence, the extension adapts the CISI-based fixed-point smoother solution in [Zhang and Bar-Shalom, 2012a] to the nonlinear case. The CISI approaches not only update the particle weights together with the state estimate and associated covariance at the last time step,

$$\hat{\boldsymbol{x}}_{k|k}, \hat{\boldsymbol{P}}_{k|k},$$

but also all estimates and covariances between the timestamp of the OOSM and the current time,

$$\{\hat{\boldsymbol{x}}_{j|j}, \hat{\boldsymbol{P}}_{j|j}\}_{j=k-l+1}^{k}.$$

In the linear, Gaussian setting this guarantees optimality even when several OOSMs arrive at the same time step, irrespective of the arrival order. The multi-OOSM problem typically arises when there are several sensors

that deliver OOSMs. Alternatively, the multi-OOSM problem could arise when one sensor delivers OOSMs with highly varying time delays. Although the optimality results do not extend to the nonlinear OOSM problem, the motivation for updating all estimates and covariances is that it can increase tracking performance also in the nonlinear case.

In the spirit of [Orguner and Gustafsson, 2008; Liu et al., 2010; Oreshkin et al., 2011], we validated the algorithms using a coordinated-turn target-tracking example. In our setup, two of the three sensors delivered OOSMs with different delays. Thus, the considered example yielded a multi-OOSM problem.

8

Rao-Blackwellized Out-of-Sequence Processing

Chapter 7 was concerned with the OOSM problem for general nonlinear models with Gaussian noise. Special focus was on achieving high tracking performance while allowing for both low computational complexity and low storage requirements. As already pointed out in Section 3.2, even better tracking performance and execution time are possible to achieve if the models contain a tractable substructure.

In this chapter we treat the OOSM problem assuming that a linear, Gaussian substructure is present in the otherwise nonlinear model. Next a motivation and outline of the contributions are given. Then we give a formal problem formulation that introduces a few new concepts compared with those in Sections 5.2 and 7.2. This is followed by the derivations of two OOSM algorithms, which are evaluated in an extensive simulation study. The chapter ends with a discussion.

8.1 Motivation and Contributions

The models used in this chapter are the mixed-Gaussian state-space (MGSS) models described by (3.20). For MGSS models, it is possible to use the Rao-Blackwellized particle filter/smoother for improved performance compared with the standard particle filter/smoother, as was discussed in Chapter 3. As also mentioned in Chapter 3, MGSS models are common in target tracking, positioning, and navigation [Gustafsson et al., 2002; Schön et al., 2005], to mention a few examples. Two occurrences are when the system equations are almost linear and/or the measurement equations are highly nonlinear, see [Rong Li and Jilkov, 2001; Rong Li and Jilkov, 2003]. As discussed in Section 3.2, specific applications that RBPFs are used in are simultaneous localization and mapping (SLAM)

[Grisetti et al., 2005; Grisetti et al., 2007; Berntorp and Nordh, 2014], automotive applications [Schön et al., 2006; Lundquist et al., 2014], and aircraft tracking [Schön et al., 2005]. Note that the robotic navigation example in Chapter 6 and the target-tracking example in Chapter 7 classify as MGSS. This also holds for the application we will consider in Chapter 9. Furthermore, because tracking applications often combine several sensors, some of which use sophisticated preprocessing, it is clear that gains can be made if utilizing model structure for the OOSM updates.

Previous work has provided the exact Bayesian inference solution and the corresponding particle-filter implementation to the OOSM problem, see the reference [Zhang and Bar-Shalom, 2012b] and the brief description in Section 5.3. The major contributions in this chapter are

- The Bayesian formulation when utilizing substructure in nonlinear state-space models
- Two algorithms for the OOSM problem that utilize the substructure

This is the first time that marginalization is used in conjunction with OOSMs. In particular, exploitation of the conditionally-linear Gaussian substructure that is present in the model class provides improved tracking performance and competitive computational demands compared with current state-of-the-art OOSM algorithms.

This chapter presents two alternative approaches for solving the OOSM problem. The first algorithm (*SERBPF*) can be regarded as a generalization of the storage-efficient particle filter (*SEPF*), reported in [Orguner and Gustafsson, 2008] and evaluated in Chapter 7, with the derivations adapted to the MGSS setting. *SERBPF* is, however, not an extension of *PF-CISIMI*, which we presented in Chapter 7, because it does not consider the multi-OOSM problem and does not perform selective processing. Except for the measurements, the approach only stores a subset of the particles over a predefined time interval. These particles are then used in an additional forward filter to associate the current estimates with the OOSMs. Because of its computational simplicity, it is well suited for real-time applications. The second algorithm (*RBOOSMBS*) instead adapts a backward-simulation approach for the association task. This implies that when the number of particles and smoothing trajectories are sufficiently large, *RBOOSMBS* achieves close to optimal tracking performance at the OOSM arrival times.

The aim with the algorithms is that they should be possible to use in realistic online applications. Therefore computational simplifications are made when feasible. In addition, one of the three examples that are used for benchmarking is a target-tracking example with characteristics that are common in typical tracking scenarios.

8.2 Related Work

We gave numerous references to different OOSM filters in Section 5.3. Those that are most related to the research in this chapter are described below.

Some of the most related particle-filter algorithms were discussed and evaluated in Chapter 7—for example, *SEPF* [Orguner and Gustafsson, 2008], the storage efficient particle filter with selective processing (*SEPF-GARP*) given in [Liu et al., 2010], and the algorithm *PF-CISI* that we presented in Chapter 7.

Another algorithm is *A-PF*. It is rooted in exact Bayesian inference, and allows for a general nonlinear Markov process model, with white process and measurement noise sources. It assumes that the OOSMs are of type I (defined in Definition 7.2). When this holds, for a sufficiently large number of particles, the estimated distribution in *A-PF* converges to the distribution that is estimated by a filter that has access to all measurements. Hence, a comparison against *A-PF* indicates how the algorithms compete with an algorithm that is optimal when no substructure is present. It is worth mentioning that *A-PF* is computationally expensive. For OOSMs that have larger delays than one sample, its complexity is $O((l-1)N^3 + N^2)$, N being the number of particles and l the OOSM delay.

Other related research is [Oreshkin et al., 2011], in which an optimization-based approach to selective processing was derived. This approach is not considered here. The reason is that the main difference compared with *SEPF-GARP* is computational efficiency and not tracking performance.

8.3 Problem Formulation

Most notation has been used before, but some is used with slightly different meaning. Hence, it is restated for ease of reading. Throughout the chapter, $p(\boldsymbol{x}_k|\boldsymbol{y}_{m:k})$ denotes the conditional probability density function of the state vector $\boldsymbol{x}$ at time $t_k \in \mathbb{R}$ conditioned on the measurement-vector sequence $\boldsymbol{y}_{m:k}$ from time t_m to time t_k. As for the RBPF in Section 3.2, we assume that the state vector $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ can be partitioned into a linear part $\boldsymbol{z}_k$ and a nonlinear part $\boldsymbol{\eta}_k$ as

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{z}_k^{\mathrm{T}} & \boldsymbol{\eta}_k^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}.$$

The considered MGSS models are on the form

$$\boldsymbol{z}_{k+1} = \boldsymbol{f}(\boldsymbol{\eta}_k) + \boldsymbol{A}(\boldsymbol{\eta}_k)\boldsymbol{z}_k + \boldsymbol{w}_k^{\boldsymbol{z}}(\boldsymbol{\eta}_k), \tag{8.1a}$$

$$\boldsymbol{\eta}_{k+1} = \boldsymbol{g}(\boldsymbol{\eta}_k) + \boldsymbol{B}(\boldsymbol{\eta}_k)\boldsymbol{z}_k + \boldsymbol{w}_k^{\boldsymbol{\eta}}(\boldsymbol{\eta}_k), \tag{8.1b}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{\eta}_k) + \boldsymbol{C}(\boldsymbol{\eta}_k)\boldsymbol{z}_k + \boldsymbol{e}_k(\boldsymbol{\eta}_k), \tag{8.1c}$$

where the process noise

$$\boldsymbol{w}_k(\boldsymbol{\eta}_k) = \begin{bmatrix} (\boldsymbol{w}_k^{\boldsymbol{z}}(\boldsymbol{\eta}_k)^{\mathrm{T}} & \boldsymbol{w}_k^{\boldsymbol{\eta}}(\boldsymbol{\eta}_k)^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$$

and the measurement noise $\boldsymbol{e}_k(\boldsymbol{\eta}_k)$ are mutually independent, white, and Gaussian distributed according to

$$\boldsymbol{w}_k(\boldsymbol{\eta}_k) \sim \mathcal{N}(\boldsymbol{0}_{n_x}, \boldsymbol{Q}_k(\boldsymbol{\eta}_k)), \quad \boldsymbol{Q}_k(\boldsymbol{\eta}_k) = \begin{bmatrix} \boldsymbol{Q}_k^{\boldsymbol{z}}(\boldsymbol{\eta}_k) & \boldsymbol{Q}_k^{\boldsymbol{z}\boldsymbol{\eta}}(\boldsymbol{\eta}_k) \\ \boldsymbol{Q}_k^{\boldsymbol{z}\boldsymbol{\eta}}(\boldsymbol{\eta}_k)^{\mathrm{T}} & \boldsymbol{Q}_k^{\boldsymbol{\eta}}(\boldsymbol{\eta}_k) \end{bmatrix}, \tag{8.2a}$$

$$\boldsymbol{e}_k(\boldsymbol{\eta}_k) \sim \mathcal{N}(\boldsymbol{0}_{n_y}, \boldsymbol{R}_k(\boldsymbol{\eta}_k)), \tag{8.2b}$$

with $\boldsymbol{Q}_k^{\boldsymbol{\eta}}(\boldsymbol{\eta}_k)$ and $\boldsymbol{R}_k(\boldsymbol{\eta}_k)$ invertible. For notational brevity, the dependence on $\boldsymbol{\eta}_k$ is implicit in what follows. Note that (8.1) differs slightly from the model formulation (3.20) in that we have transferred $\boldsymbol{F}_k$ and $\boldsymbol{G}_k$ into the process and measurement noise sources in (8.2).

For the OOSM filtering task, $\mathcal{Y}_k$ denotes the set of in-sequence measurements generated in the interval $[0, k]$. The symbol $\mathcal{Z}_k$ indicates the set of OOSMs generated in the interval $[0, k]$ available at time index k. Moreover, $\boldsymbol{y}_{0:k}$ represents the set $\mathcal{Y}_k \cup \mathcal{Z}_{k-1}$. Further, $\hat{\boldsymbol{z}}_{k|k} := \hat{\boldsymbol{z}}_{k|k}(\boldsymbol{\eta}_{0:k})$ is the linear state estimate given the trajectory $\boldsymbol{\eta}_{0:k}$ and measurements $\boldsymbol{y}_{0:k}$, and $\boldsymbol{P}_{k|k} := \boldsymbol{P}_{k|k}(\boldsymbol{\eta}_{0:k})$ is its associated covariance.

Suppose that an estimate of the filtering posterior $p(\boldsymbol{z}_k, \boldsymbol{\eta}_k | \boldsymbol{y}_{0:k})$ exists at time t_k, where $\boldsymbol{z}_k$ is conditioned on $\boldsymbol{\eta}_{0:k}$. Assume that an OOSM $\boldsymbol{y}_\tau \in \mathcal{Z}_k$ with timestamp $t_\tau \in [t_{k-l}, t_{k-l+1})$ arrives, see Figure 8.1. The Rao-Blackwellized OOSM filtering task is to update the particle weights and linear estimates at time t_k with $\boldsymbol{y}_\tau$; that is, the aim is to obtain $p(\boldsymbol{x}_k | \boldsymbol{y}_{0:k}, \boldsymbol{y}_\tau) = p(\boldsymbol{z}_k, \boldsymbol{\eta}_k | \boldsymbol{y}_{0:k}, \boldsymbol{y}_\tau)$.

8.4 Rao-Blackwellized Out-of-Sequence Processing

The aim is to estimate the density

$$p(\boldsymbol{z}_k, \boldsymbol{\eta}_k | \boldsymbol{y}_{0:k}, \boldsymbol{y}_\tau), \tag{8.3}$$

as described in Section 8.3. To utilize the linear structure in the system dynamics (8.1), factorize (8.3) in the same manner as for the standard RBPF. This gives

$$p(\boldsymbol{z}_k, \boldsymbol{\eta}_k | \boldsymbol{y}_{0:k}, \boldsymbol{y}_\tau) = p(\boldsymbol{z}_k | \boldsymbol{\eta}_k, \boldsymbol{y}_{0:k}, \boldsymbol{y}_\tau) p(\boldsymbol{\eta}_k | \boldsymbol{y}_{0:k}, \boldsymbol{y}_\tau). \tag{8.4}$$

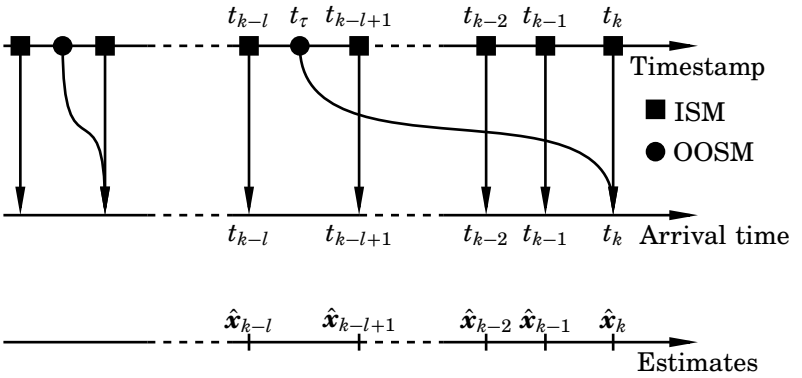

Figure 8.1 An illustration of the l-step lag OOSM problem. Measurements $\boldsymbol{y}_{0:k}$ have arrived and been used in the estimation process to calculate $\hat{\boldsymbol{x}}_k$. Then, an OOSM $\boldsymbol{y}_\tau$ arising from time t_τ arrives at time t_k, and is subsequently utilized to compute an updated $\bar{\boldsymbol{x}}_k$.

Using Bayes' rule on the second factor of (8.4) yields

$$p(\boldsymbol{\eta}_k|\boldsymbol{y}_{0:k},\boldsymbol{y}_\tau) = \frac{p(\boldsymbol{y}_\tau|\boldsymbol{\eta}_k,\boldsymbol{y}_{0:k})p(\boldsymbol{\eta}_k|\boldsymbol{y}_{0:k})}{p(\boldsymbol{y}_\tau|\boldsymbol{y}_{0:k})} \propto p(\boldsymbol{y}_\tau|\boldsymbol{\eta}_k,\boldsymbol{y}_{0:k})p(\boldsymbol{\eta}_k|\boldsymbol{y}_{0:k}). \quad (8.5)$$

By approximating the last term in (8.5) with the particle-filter part of the RBPF, (8.5) transforms to

$$p(\boldsymbol{\eta}_k|\boldsymbol{y}_{0:k},\boldsymbol{y}_\tau) \approx \sum_{i=1}^{N} w^i_{k|k,\tau}\delta\left(\boldsymbol{\eta}_k - \boldsymbol{\eta}^i_k\right), \quad (8.6)$$

where the weights are

$$w^i_{k|k,\tau} \propto w^i_k p(\boldsymbol{y}_\tau|\boldsymbol{\eta}^i_k,\boldsymbol{y}_{0:k}). \quad (8.7)$$

Moreover, it holds that

$$p(\boldsymbol{y}_\tau|\boldsymbol{\eta}^i_k,\boldsymbol{y}_{0:k}) = \int p(\boldsymbol{y}_\tau|\boldsymbol{z}^i_k,\boldsymbol{\eta}^i_k,\boldsymbol{y}_{0:k})p(\boldsymbol{z}^i_k|\boldsymbol{\eta}^i_k,\boldsymbol{y}_{0:k})\,\mathrm{d}\boldsymbol{z}^i_k. \quad (8.8)$$

Thus, to update the posterior $p(\boldsymbol{\eta}_k|\boldsymbol{y}_{0:k})$ with the OOSM $\boldsymbol{y}_\tau$ to form (8.6), the weight update (8.7) needs estimates of the likelihoods

$$\left\{p\left(\boldsymbol{y}_\tau|\boldsymbol{z}^i_k,\boldsymbol{\eta}^i_k,\boldsymbol{y}_{0:k}\right)\right\}_{i=1}^{N}.$$

To incorporate the OOSM $\boldsymbol{y}_\tau$ in the first factor on the right-hand side in (8.4), recast it as

$$p(\boldsymbol{z}_k|\boldsymbol{\eta}_k, \boldsymbol{y}_{0:k}, \boldsymbol{y}_\tau) = \frac{p(\boldsymbol{y}_\tau|\boldsymbol{z}_k, \boldsymbol{\eta}_k, \boldsymbol{y}_{0:k})p(\boldsymbol{z}_k|\boldsymbol{\eta}_k, \boldsymbol{y}_{0:k})}{\int p(\boldsymbol{y}_\tau|\boldsymbol{z}_k, \boldsymbol{\eta}_k, \boldsymbol{y}_{0:k})p(\boldsymbol{z}_k|\boldsymbol{\eta}_k, \boldsymbol{y}_{0:k})\,\mathrm{d}\boldsymbol{z}_k}. \tag{8.9}$$

The RBPF can be used for approximating the second factor in the numerator of (8.9). The first term in the numerator equals the first term on the right-hand side of (8.8). Hence, what remains is to evaluate the densities

$$\left\{p(\boldsymbol{y}_\tau|\boldsymbol{z}_k^i, \boldsymbol{\eta}_k^i, \boldsymbol{y}_{0:k})\right\}_{i=1}^N.$$

In the following, we present two different approaches for computing

$$p(\boldsymbol{y}_\tau|\boldsymbol{x}_k^i, \boldsymbol{y}_{0:k}) = p(\boldsymbol{y}_\tau|\boldsymbol{z}_k^i, \boldsymbol{\eta}_k^i, \boldsymbol{y}_{0:k})$$

and thus performing the particle-filter and Kalman-filter OOSM updates.

OOSM Processing with Supporting RBPF—SERBPF

For the first algorithm (*SERBPF*), the focus is on finding a computationally efficient method that approximately computes $p(\boldsymbol{y}_\tau|\boldsymbol{x}_k^i, \boldsymbol{y}_{0:k})$. The idea is that by explicitly utilizing the linear substructure (8.1a), we can use approximations and still improve performance compared with existing OOSM algorithms.

To derive the algorithm, start with computing $p(\boldsymbol{y}_\tau|\boldsymbol{x}_k^i, \boldsymbol{y}_{0:k})$ as

$$p(\boldsymbol{y}_\tau|\boldsymbol{x}_k^i, \boldsymbol{y}_{0:k}) = \int p(\boldsymbol{y}_\tau|\boldsymbol{z}_\tau, \boldsymbol{\eta}_{0:\tau})p(\boldsymbol{z}_\tau, \boldsymbol{\eta}_{0:\tau}|\boldsymbol{x}_k^i, \boldsymbol{y}_{0:k})\,\mathrm{d}\boldsymbol{z}_\tau \mathrm{d}\boldsymbol{\eta}_{0:\tau}. \tag{8.10}$$

The density $p(\boldsymbol{z}_\tau, \boldsymbol{\eta}_{0:\tau}|\boldsymbol{x}_k^i, \boldsymbol{y}_{0:k})$ in (8.10) is a smoothing density. Rewrite it as

$$p(\boldsymbol{z}_\tau, \boldsymbol{\eta}_{0:\tau}|\boldsymbol{x}_k^i, \boldsymbol{y}_{0:k}) \propto p(\boldsymbol{x}_k^i, \boldsymbol{y}_{k-l+1:k-1}|\boldsymbol{z}_\tau, \boldsymbol{\eta}_{0:\tau})p(\boldsymbol{z}_\tau, \boldsymbol{\eta}_{0:\tau}|\boldsymbol{y}_{0:k-l}), \tag{8.11}$$

with $\boldsymbol{y}_k$ removed because $\boldsymbol{x}_k^i$ is given. Furthermore, the density $p(\boldsymbol{z}_\tau, \boldsymbol{\eta}_{0:\tau}|\boldsymbol{y}_{0:k-l})$ in (8.11) can be approximated by applying the RBPF and using a prediction to time t_τ. This gives

$$p(\boldsymbol{z}_\tau, \boldsymbol{\eta}_{0:\tau}|\boldsymbol{y}_{0:k-l}) = p(\boldsymbol{z}_\tau, \boldsymbol{\eta}_{0:\tau}|\boldsymbol{z}_{k-l}, \boldsymbol{\eta}_{0:k-l})p(\boldsymbol{z}_{k-l}, \boldsymbol{\eta}_{0:k-l}|\boldsymbol{y}_{0:k-l}).$$

The first density on the right-hand side in (8.11) is a measurement update using both $\boldsymbol{y}_{k-l+1:k-1}$ and $\boldsymbol{x}_k^i$ as measurements. Thus, we need to propagate the past, $\{\boldsymbol{z}_\tau, \boldsymbol{\eta}_{0:\tau}\}$, to update with $\{\boldsymbol{x}_k^i, \boldsymbol{y}_{k-l+1:k-1}\}$.

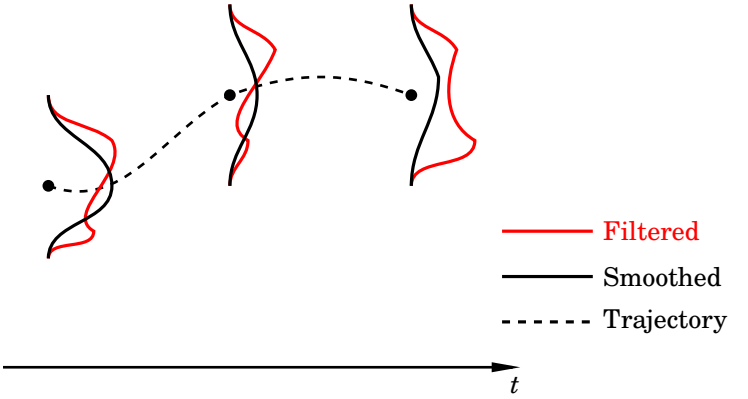

Figure 8.2 The trajectory and the corresponding filtered and smoothed estimated densities. The intuition is that because smoothing utilizes more information than filtering, it typically reduces multimodality and uncertainty. Thus, the smoothing density in general needs fewer particles than the filtering density.

Forward Propagation As discussed in Section 3.3, the RBPF is also an approximate solution to the smoothing problem, but the estimate is often degenerate for large smoothing intervals because of the inherent depletion problem in particle filters. Nevertheless, if the smoothing interval is small, the RBPF should be an adequate approximation. Consequently, we apply a supporting (additional) RBPF to find (8.11). With this approach, both the smoothing weights and the linear smoothing estimates are efficiently found in one forward sweep: Initialize an additional RBPF from time index $k-l$. Because (8.11) is a smoothing density, the supporting RBPF typically needs fewer particles than the original RBPF. The motivation for this is that because more measurements are available when performing smoothing than filtering, the uncertainty, and thus the number of particles needed, is smaller. Figure 8.2 illustrates the idea.

To apply the supporting RBPF, start with sampling $M_{\text{FF}} \ll N$ estimates $\{\hat{\boldsymbol{z}}^j_{k-l|k-l}, \boldsymbol{\eta}^j_{k-l}\}_{j=1}^{M_{\text{FF}}}$ from the filtering density at time index $k-l$.[1] Then predict the estimates up to time index τ and augment the linear state vector to

$$\boldsymbol{\zeta}^j_m = \begin{bmatrix} \boldsymbol{z}^j_m \\ \boldsymbol{z}^j_\tau \end{bmatrix}, \tag{8.12}$$

with initialization according to

[1] Note that performing this sampling already in the original forward RBPF, at each time step, reduces the storage requirements.

$$\boldsymbol{\zeta}_\tau^j = \begin{bmatrix} \hat{\boldsymbol{z}}_{\tau|k-l}^j \\ \hat{\boldsymbol{z}}_{\tau|k-l}^j \end{bmatrix}, \quad \bar{\boldsymbol{P}}_\tau^j = \begin{bmatrix} \boldsymbol{P}_{\tau|k-l}^j & \boldsymbol{P}_{\tau|k-l}^j \\ \boldsymbol{P}_{\tau|k-l}^j & \boldsymbol{P}_{\tau|k-l}^j \end{bmatrix}. \tag{8.13}$$

Then for each $m \in [k-l+1, k-1]$, execute the supporting RBPF with resampling as usual, where the lower part of $\boldsymbol{\zeta}_m^j$ in (8.13) is the linear smoothing estimate, the lower right block of $\bar{\boldsymbol{P}}_m^j$ in (8.13) gives the smoothing covariance, and the upper right block of $\bar{\boldsymbol{P}}_m^j$ gives the cross-covariance $\boldsymbol{P}_{m,\tau|m}^j$ between the states at time index m and τ. At time t_k, use $\{\hat{\boldsymbol{z}}_{k|k}^i, \boldsymbol{\eta}_k^i\}_{i=1}^N$ as measurements. At the end of the recursion, an approximation of the smoothing density (8.11) is therefore given by the Gaussian mixture

$$p(\boldsymbol{z}_\tau, \boldsymbol{\eta}_{0:\tau} | \boldsymbol{x}_k^i, \boldsymbol{y}_{0:k}) \approx \sum_{j=1}^{M_{\text{FF}}} q_{\tau|k,i}^j \mathcal{N}(\boldsymbol{z}_\tau | \hat{\boldsymbol{z}}_{\tau|k,i}^j, \boldsymbol{P}_{\tau|k,i}^j) \delta(\boldsymbol{\eta}_{0:\tau} - \boldsymbol{\eta}_{0:\tau}^j), \tag{8.14}$$

where $q_{\tau|k,i}^j$ are the smoothing weights given both measurements up to time t_{k-1} and state estimate i at time t_k.

Updating with the OOSM Given (8.14), an approximation of (8.10) is

$$p(\boldsymbol{y}_\tau | \boldsymbol{x}_k^i, \boldsymbol{y}_{0:k}) \approx \sum_{j=1}^{M_{\text{FF}}} q_{\tau|k,i}^j p(\boldsymbol{y}_\tau | \hat{\boldsymbol{z}}_{\tau|k,i}^j, \boldsymbol{\eta}_{0:\tau}^j), \tag{8.15}$$

with the measurement likelihood given by

$$p(\boldsymbol{y}_\tau | \boldsymbol{z}_{\tau|k,i}^j, \boldsymbol{\eta}_{0:\tau}^j) = \mathcal{N}(\boldsymbol{y}_\tau | \hat{\boldsymbol{y}}_{\tau|k,i}^j, \boldsymbol{\Sigma}_{\tau|k,i}^j). \tag{8.16}$$

In (8.16), the mean and covariance are computed as

$$\begin{aligned} \hat{\boldsymbol{y}}_{\tau|k,i}^j &= \boldsymbol{h}_\tau^j + \boldsymbol{C}_\tau^j \hat{\boldsymbol{z}}_{\tau|k,i}^j, \\ \boldsymbol{\Sigma}_{\tau|k,i}^j &= \boldsymbol{C}_\tau^j \boldsymbol{P}_{\tau|k,i}^j (\boldsymbol{C}_\tau^j)^{\mathrm{T}} + \boldsymbol{R}_\tau^j. \end{aligned}$$

With (8.15) inserted into (8.8), the particle weights (8.7) become

$$w_{k|k,\tau}^i \propto w_k^i \sum_{j=1}^{M_{\text{FF}}} q_{\tau|k,i}^j p(\boldsymbol{y}_\tau | \hat{\boldsymbol{z}}_{\tau|k,i}^j, \boldsymbol{\eta}_{0:\tau}^j). \tag{8.17}$$

To find the linear estimates and their covariances after the update with the OOSM, and hence to find (8.9), we utilize Proposition 8.1.

PROPOSITION 8.1
Given (8.15), for each $i \in \{1, \ldots, N\}$, (8.9) is given by

$$p(\boldsymbol{z}_k^i | \boldsymbol{\eta}_k^i, \boldsymbol{y}_{0:k}, \boldsymbol{y}_\tau) = \mathcal{N}(\boldsymbol{z}_k | \hat{\boldsymbol{z}}_{k|k,\tau}^i, \boldsymbol{P}_{k|k,\tau}^i),$$

where

$$\begin{aligned}
\hat{\boldsymbol{z}}^i_{k|k,\tau} &= \hat{\boldsymbol{z}}^i_{k|k} + \boldsymbol{E}^i \\
\boldsymbol{P}^i_{k|k,\tau} &= \boldsymbol{P}^i_{k|k} + \sum_{j=1}^{M} q^j_{\tau|k,i} \Big((\boldsymbol{E}^{j,i} - \boldsymbol{E}^i)(\boldsymbol{E}^{j,i} - \boldsymbol{E}^i)^{\mathrm{T}} \\
&\quad - \boldsymbol{W}^{j,i}_{k,\tau} \boldsymbol{\Sigma}^j_{\tau|k,i} (\boldsymbol{W}^{j,i}_{k,\tau})^{\mathrm{T}} \Big) \\
&= \sum_{j=1}^{M} q^j_{\tau|k,i} \boldsymbol{W}^{j,i}_{k,\tau} \boldsymbol{e}^{j,i}_{\tau} \\
\boldsymbol{E}^{j,i} &= \boldsymbol{W}^{j,i}_{k,\tau} \boldsymbol{e}^{j,i}_{\tau} \\
\boldsymbol{e}^{j,i}_{\tau} &= \boldsymbol{y}_{\tau} - \hat{\boldsymbol{y}}^j_{\tau|k,i} \\
\boldsymbol{W}^{j,i}_{k,\tau} &= \boldsymbol{P}^j_{k,\tau|k,i} (\boldsymbol{C}^j_{\tau})^{\mathrm{T}} (\boldsymbol{\Sigma}^j_{\tau|k,i})^{-1}.
\end{aligned} \tag{8.18}$$

□

The algorithm is denoted by *SERBPF* and is summarized in Algorithm 8.1. The algorithm's storage requirements are

$$\left\{ \{\hat{\boldsymbol{z}}^j_{m|m}, \boldsymbol{P}^j_{m|m}, \boldsymbol{\eta}^j_m\}_{m=k-l_{\max}}^{k-1}, \{\boldsymbol{y}_m\}_{m=k-l_{\max+1}}^{k-1} \right\}_{j=1}^{M_{\mathrm{FF}}},$$

where $l_{\max}$ is the predetermined maximum delay in the number of time steps. Because the RBPF has computational complexity $O(N)$ [Karlsson et al., 2005], the algorithm has complexity $O(lM_{\mathrm{FF}} + M_{\mathrm{FF}}N)$.

Algorithm 8.1—SERBPF

Input: $\{\hat{\boldsymbol{x}}^j_{k-l|k-l}, \boldsymbol{P}^j_{k-l|k-l}, \boldsymbol{x}^i_k, \boldsymbol{P}^i_{k|k}, w^i_k, \boldsymbol{y}_{k-l+1:k-1}\}_{j=1}^{M_{\mathrm{FF}}}$

1: Predict up to time t_τ, yielding $p(\boldsymbol{z}^j_\tau, \boldsymbol{\eta}^j_{0:\tau} | \boldsymbol{y}_{0:k-l})$.
2: Initialize smoothing weights as $q^j_{\tau|k-l} = 1/M_{FF}$.
3: Augment the state vector as in (8.12) and initialize with (8.13).
4: **for** $m = k - l + 1$ **to** $k - 1$ **do**
5: Perform RBPF time update to time t_m using (3.26) and (3.27).
6: Perform weight update according to forward-filter measurement likelihood (3.24).
7: **if** $\left(\sum_{j=1}^{M_{\mathrm{FF}}} (q^j_m)^2\right)^{-1} < N_{\mathrm{thr}}$ **then**
8: Resample M_{FF} new particles with replacement. Keep track of $\{\boldsymbol{\eta}^j_{0:\tau}\}_{j=1}^{M_{\mathrm{FF}}}$ and equalize weights: $q^j_{\tau|m} = 1/M_{\mathrm{FF}}, \quad \forall j \in \{1, \ldots, M_{\mathrm{FF}}\}$.
9: **end if**
10: Perform Kalman-filter measurement update using (3.25).
11: **end for**
12: Perform RBPF time update to time t_k using (3.26) and (3.27).

13: Use $\{\hat{\boldsymbol{z}}^i_{k|k}, \boldsymbol{\eta}^i_k\}_{i=1}^N$ as measurements with (8.1a) and (8.1b) as measurement likelihoods.
14: Update weights as in (8.17).
15: Update linear estimates by applying (8.18).
Output: $\{\hat{\boldsymbol{z}}^i_{k|k,\tau}, \boldsymbol{P}^i_{k|k,\tau}, w^i_{k|k,\tau}\}_{i=1}^N$

OOSM Update Using Backward Simulation—RBOOSMBS

Algorithm 8.1 is computationally efficient, but the generated smoothing density may, especially for large delays, suffer from degeneracy. Accordingly, the estimate of the measurement likelihood that is used when associating the current estimate with the OOSM may be inaccurate. To avoid this, and thus to improve estimation accuracy, the next algorithm instead uses a backward-simulation approach to the OOSM update.

As a start, now instead rewrite $p(\boldsymbol{y}_\tau|\boldsymbol{x}^i_k, \boldsymbol{y}_{0:k})$ as

$$p(\boldsymbol{y}_\tau|\boldsymbol{x}^i_k, \boldsymbol{y}_{0:k}) = \int p(\boldsymbol{y}_\tau|\boldsymbol{z}_\tau, \boldsymbol{\eta}_{0:\tau}) \cdot p(\boldsymbol{z}_\tau, \boldsymbol{\eta}_\tau, \boldsymbol{\eta}_{0:k-1}|\boldsymbol{x}^i_k, \boldsymbol{y}_{0:k})\, \mathrm{d}\boldsymbol{z}_\tau \mathrm{d}\boldsymbol{\eta}_\tau \mathrm{d}\boldsymbol{\eta}_{0:k-1} \quad (8.19)$$

for insertion into and subsequent weight update in (8.8) and (8.7), respectively, where

$$p(\boldsymbol{z}_\tau, \boldsymbol{\eta}_\tau, \boldsymbol{\eta}_{0:k-1}|\boldsymbol{x}^i_k, \boldsymbol{y}_{0:k}) = p(\boldsymbol{z}_\tau|\boldsymbol{\eta}_{0:\tau}, \boldsymbol{\eta}_{k-l:k-1}, \boldsymbol{x}^i_k, \boldsymbol{y}_{0:k}) \cdot p(\boldsymbol{\eta}_{0:\tau}|\boldsymbol{\eta}_{k-l:k-1}, \boldsymbol{x}^i_k, \boldsymbol{y}_{0:k}) p(\boldsymbol{\eta}_{k-l:k-1}|x^i_k, \boldsymbol{y}_{0:k}).$$

For later use in the linear state update (8.9), rewrite $p(\boldsymbol{y}_\tau|\boldsymbol{x}^i_k, \boldsymbol{y}_{0:k})$ as

$$p(\boldsymbol{y}_\tau|\boldsymbol{x}^i_k, \boldsymbol{y}_{0:k}) = \int p(\boldsymbol{y}_\tau|\boldsymbol{z}_\tau, \boldsymbol{\eta}_\tau) \cdot p(\boldsymbol{z}_\tau, \boldsymbol{\eta}_\tau, \boldsymbol{\eta}_{k-l:k-1}|\boldsymbol{x}^i_k, \boldsymbol{y}_{0:k})\, \mathrm{d}\boldsymbol{z}_\tau \mathrm{d}\boldsymbol{\eta}_\tau \mathrm{d}\boldsymbol{\eta}_{k-l:k-1}, \quad (8.20)$$

where

$$p(\boldsymbol{z}_\tau, \boldsymbol{\eta}_\tau, \boldsymbol{\eta}_{k-l:k-1}|\boldsymbol{x}^i_k, \boldsymbol{y}_{0:k}) = p(\boldsymbol{z}_\tau|\boldsymbol{\eta}_\tau, \boldsymbol{\eta}_{k-l:k-1}, \boldsymbol{x}^i_k, \boldsymbol{y}_{0:k}) \cdot p(\boldsymbol{\eta}_\tau|\boldsymbol{\eta}_{k-l:k-1}, \boldsymbol{x}^i_k, \boldsymbol{y}_{0:k}) p(\boldsymbol{\eta}_{k-l:k-1}|\boldsymbol{x}^i_k, \boldsymbol{y}_{0:k}). \quad (8.21)$$

In both (8.19) and (8.20), the last term $p(\boldsymbol{\eta}_{k-l:k-1}|\boldsymbol{x}^i_k, \boldsymbol{y}_{0:k})$ factorizes in the same manner as the sequential factorization (3.28) in the FFBS on page 61:

$$p(\boldsymbol{\eta}_{k-l:k-1}|\boldsymbol{x}^i_k, \boldsymbol{y}_{0:k}) = p(\boldsymbol{\eta}_{k-1}|\boldsymbol{x}^i_k, \boldsymbol{y}_{0:k}) \cdot \prod_{m=k-l}^{k-2} p(\boldsymbol{\eta}_m|\boldsymbol{\eta}_{m+1:k-1}, \boldsymbol{x}^i_k, \boldsymbol{y}_{0:k}). \quad (8.22)$$

This smoothing density can be solved for in a similar way to how the Rao-Blackwellized particle smoother, described in Section 3.3, solves it [Lindsten et al., 2013]. However, the smoother derived in [Lindsten et al., 2013] assumes that the covariance matrix in (8.2a) is diagonal (i.e., $\boldsymbol{w}_k^{\boldsymbol{z}}$ and $\boldsymbol{w}_k^{\boldsymbol{\eta}}$ are mutually independent). This restriction is here taken into account by decorrelating the noise in the same manner as in [Schön et al., 2005], which yields a modified system. This modified system can then be used in the derivations in [Lindsten et al., 2013], which amounts to replacing $\boldsymbol{f}_k$, $\boldsymbol{A}_k$, and the covariance matrix for the process noise acting on the linear states, $\boldsymbol{Q}_k^{\boldsymbol{z}}$, with

$$\begin{aligned}
\bar{\boldsymbol{f}}_k &= \boldsymbol{f}_k + \boldsymbol{Q}_k^{\boldsymbol{z}\boldsymbol{\eta}}(\boldsymbol{Q}_k^{\boldsymbol{\eta}})^{-1}(\boldsymbol{\eta}_{k+1} - \boldsymbol{g}_k),\\
\bar{\boldsymbol{A}}_k &= \boldsymbol{A}_k - \boldsymbol{Q}_k^{\boldsymbol{z}\boldsymbol{\eta}}(\boldsymbol{Q}_k^{\boldsymbol{\eta}})^{-1}\boldsymbol{B}_k,\\
\bar{\boldsymbol{Q}}_k^{\boldsymbol{z}} &= \boldsymbol{Q}_k^{\boldsymbol{z}} - \boldsymbol{Q}_k^{\boldsymbol{z}\boldsymbol{\eta}}(\boldsymbol{Q}_k^{\boldsymbol{\eta}})^{-1}(\boldsymbol{Q}_k^{\boldsymbol{z}\boldsymbol{\eta}})^{\mathrm{T}},
\end{aligned}$$

respectively, in the derivations. Also, we adapt the smoother to be applicable to the OOSM scenario and take a computationally efficient rejection-sampling approach for finding the backward trajectories, instead of computing the smoothing weights for each particle in each time step. How to find (8.22) is described next.

Finding the Nonlinear Backward Trajectories First, at time t_k, instead of drawing $\boldsymbol{\eta}_k' = \boldsymbol{\eta}_k^j$ with probability w_k^j for all $j \in \{1, \ldots, M_{\mathrm{BS}}\}$, giving the starting point for M_{BS} backward trajectories, we must choose $\boldsymbol{\eta}_k' = \boldsymbol{\eta}_k^i$ for each of the N forward particles to associate the filtered estimates with the OOSM. Thus, in total M_{BS} backward trajectories are associated with each forward particle. Second, if the smoothing-weight computation (3.34) was to be computed for all particles in each time step for appending the trajectories, the complexity would be $O(N^2 M_{\mathrm{BS}})$ in each time step. However, by using the assumption that smoothing densities are less complex than filtering densities, see Figure 8.2, it is feasible to sample $D < N$ particles instead in each time step to form D smoothing weights. Note that this assumption is not true in general. Rather, it depends on the structure of the measurement equation and the information it contains. This is elaborated on in Sections 8.5 and 8.6.

There exist computationally efficient particle smoothers that create backward trajectories without explicitly computing the smoothing weights, with sustained algorithm behavior [Douc et al., 2012]. As will be clear later, the smoothing weights are only needed at time step $k - l$. Thus, the approach in [Douc et al., 2012] can be adapted to our MGSS model setting for $m = k - 1, \ldots, k - l + 1$: The aim is to sample from the discrete distribution formed by the smoothing weights

$$\{w_{m|k}^{j,d}\}_{d=1}^{D},$$

without actually computing all smoothing weights. Here, we use the forward weights $\{w_m^d\}_{d=1}^D$ as a proposal distribution, for which we already know the distribution. By noting that the transition density in (3.36) is bounded from above as

$$p\left(\boldsymbol{y}_{m+1:k}, \boldsymbol{\eta}_{m+1:k-1}^j, \boldsymbol{\eta}_k^i \middle| \boldsymbol{\eta}_{0:m}^d, \boldsymbol{y}_{0:m}\right) \leq \sigma_m^j$$

where

$$\sigma_m^j = \max_{d=1,\ldots,D} Z_m^{j,d} \det(\boldsymbol{\Lambda}_m^{j,d})^{-1/2},$$

we can perform rejection sampling [Neumann, 1951] to extend each backward trajectory as $\{\boldsymbol{\eta}_m^j, \boldsymbol{\eta}_{m+1:k-1}^j, \boldsymbol{\eta}_k^i\}$. Algorithm 8.2 summarizes the algorithm, and a (very) brief discussion about rejection sampling is found on page 62.

Algorithm 8.2—Rejection Sampling

$L = \{1, \ldots, M_{\text{BS}}\}$.
while L is not empty **do**
  Set $n = \text{size}(L)$.
  Sample $\{I(j)\}_{j=1}^n$ independently with probabilities proportional to $\{w_m^{I(j)}\}_{j=1}^n$.
  Sample $\{U(j)\}_{j=1}^n$ independently and uniformly over $[0, 1]$.
  for $j = 1$ **to** n **do**
    if $U(j)\sigma_m^{L(j)} \leq p(\boldsymbol{y}_{m+1:k}, \boldsymbol{\eta}_{m+1:k-1}^{L(j)}, \boldsymbol{\eta}_k^i | \boldsymbol{\eta}_{0:m}^{I(j)}, \boldsymbol{y}_{0:m})$ **then**
      Set $J(L(j)) = I(j)$.
      Set $L = L \setminus L(j)$.
    end if
  end for
end while
return Backward-trajectory indices $\{J(j)\}_{j=1}^{M_{\text{BS}}}$.

There is no upper bound on the number of executions of the while-loop in Algorithm 8.2. Hence, a threshold $C_{\max}$ for the maximum number of iterations is set. If L is still empty after $C_{\max}$ iterations, an index is sampled from the smoothing weights (3.34) by computing the transition densities for the indices that have not already been selected. Even though Algorithm 8.2 occasionally does not succeed in finding an index, on average it provides a noticeable speedup compared with evaluating all weights at each time step. Note that for $M_{\text{BS}} = D$ and D large, Algorithm 8.2 has close to linear computational complexity in the number of particles.

At (the last) time step $k-l$, for each trajectory, draw D forward particles $\{\boldsymbol{\eta}_{0:k-l}^d\}_{d=1}^D$ with probability w_{k-l}^d and compute the smoothing weights

(3.34), using (3.35) and (3.36), for these D particles. This implies that the smoothing density (8.22) at time t_{k-l} approximates to

$$p(\boldsymbol{\eta}_{k-l:k-1}|\boldsymbol{x}_k^i, \boldsymbol{y}_{0:k}) \approx \frac{1}{M_{\text{BS}}} \sum_{j=1}^{M_{\text{BS}}} \sum_{d=1}^{D} w_{k-l|k,i}^{j,d} \delta(\boldsymbol{\eta}_{0:k-l} - \boldsymbol{\eta}_{0:k-l}^d). \tag{8.23}$$

Finally, by drawing a particle $\boldsymbol{\eta}_{k-l}$ with probability $w_{k-l|k,i}^{j,d}$ for each j, an approximation to (8.22) based on the mean of the full backward trajectories is

$$p(\boldsymbol{\eta}_{k-l:k-1}|\boldsymbol{x}_k^i, \boldsymbol{y}_{0:k}) \approx \frac{1}{M_{\text{BS}}} \sum_{j=1}^{M_{\text{BS}}} \delta(\boldsymbol{\eta}_{k-l:k-1} - \boldsymbol{\eta}_{k-l:k-1}^j). \tag{8.24}$$

The reason for approximating (8.22) in two different ways is that the linear part of the state vector relies on access to the backward density (8.24). However, every extension of the backward trajectory introduces approximation errors. Thus, (8.23) is a better choice than (8.24) for updating the particle weights with the OOSM.

Updating the Weights and Linear Estimates To update the particle weights with the OOSM, insert (8.23) into the expression for the measurement likelihood (8.19). This implies that

$$p(\boldsymbol{y}_\tau|\boldsymbol{x}_k^i, \boldsymbol{y}_{0:k}) \approx \frac{1}{M_{\text{BS}}} \sum_{j=1}^{M_{\text{BS}}} \sum_{d=1}^{D} w_{\tau|k,i}^{j,d} p(\boldsymbol{y}_\tau|\hat{\boldsymbol{z}}_{\tau|k-l}^d, \boldsymbol{\eta}_{0:\tau}^d). \tag{8.25}$$

Here, $w_{\tau|k,i}^{j,d} = w_{k-l|k,i}^{j,d}$ since only a time update differs between t_τ and t_{k-l}. Inserting (8.25) into (8.8) and using (8.7) gives the weights after processing the l-step lag OOSM as

$$w_{k|k,\tau}^i \propto w_k^i \sum_{j=1}^{M_{\text{BS}}} \sum_{d=1}^{D} w_{\tau|k,i}^{j,d} p(\boldsymbol{y}_\tau|\hat{\boldsymbol{z}}_{\tau|k-l}^d, \boldsymbol{\eta}_\tau^d), \tag{8.26}$$

where $p(\boldsymbol{y}_\tau|\hat{\boldsymbol{z}}_{\tau|k-l}^d, \boldsymbol{\eta}_\tau^d)$ is computed in a similar way to (8.16):

$$p(\boldsymbol{y}_\tau|\hat{\boldsymbol{z}}_{\tau|k-l}^d, \boldsymbol{\eta}_\tau^d) = \mathcal{N}(\boldsymbol{y}_\tau|\hat{\boldsymbol{y}}_{\tau|k-l}^d, \boldsymbol{\Sigma}_{\tau|k-l}^d),$$

in which

$$\begin{aligned}
\hat{\boldsymbol{y}}_{\tau|k-l}^d &= \boldsymbol{h}_\tau^d + \boldsymbol{C}_\tau^d \hat{\boldsymbol{z}}_{\tau|k-l}^d, \\
\boldsymbol{\Sigma}_{\tau|k-l}^d &= \boldsymbol{C}_\tau^d \boldsymbol{P}_{\tau|k-l}^d (\boldsymbol{C}_\tau^d)^{\text{T}} + \boldsymbol{R}_\tau^d.
\end{aligned}$$

To update the linear filtering density (8.9) (i.e., $p(\boldsymbol{z}_k|\boldsymbol{\eta}_k, \boldsymbol{y}_{0:k}, \boldsymbol{y}_\tau)$), the measurement update step (8.20) needs an estimate of the linear smoothing density

$$p(\boldsymbol{z}_\tau|\boldsymbol{\eta}_\tau, \boldsymbol{\eta}^j_{k-l:k-1}, \boldsymbol{x}^i_k, \boldsymbol{y}_{0:k}),$$

that is, the first density on the right-hand side of (8.21), for which it is possible to resort to different linear smoothers (conditioned on the generated backward trajectories (8.24) and the measurements). Here we choose a Rauch-Tung-Striebel (RTS) fixed-interval smoother [Rauch et al., 1965], which for the MGSS model class is given by Proposition 8.2:

PROPOSITION 8.2
Given the model (8.1), the filtering density $p(\boldsymbol{z}_k|\boldsymbol{\eta}_{0:k}, \boldsymbol{y}_{0:k})$, and the trajectory $\boldsymbol{\eta}_{m:k}$, the marginal smoothing density for $\boldsymbol{z}_m$ is given by

$$p(\boldsymbol{z}_m|\boldsymbol{\eta}_{m:k}, \boldsymbol{y}_{0:k}) = \mathcal{N}(\boldsymbol{z}_m|\hat{\boldsymbol{z}}_{m|k}, \boldsymbol{P}_{m|k}),$$

where

$$\begin{aligned}
\hat{\boldsymbol{z}}_{m|k} &= \hat{\boldsymbol{z}}^*_{m|m} + \boldsymbol{K}_m(\hat{\boldsymbol{z}}_{m+1|k} - \hat{\boldsymbol{z}}_{m+1|m}) \\
\boldsymbol{P}_{m|k} &= \boldsymbol{P}^*_{m|m} + \boldsymbol{K}_m(\boldsymbol{P}_{m+1|k} - \boldsymbol{P}_{m+1|m}) \\
\boldsymbol{P}_{m,k|m} &= \boldsymbol{P}_{m+1,k|m+1}\boldsymbol{K}^{\mathrm{T}}_m \\
\boldsymbol{K}_m &= \boldsymbol{P}^*_{m|m}\boldsymbol{A}^{\mathrm{T}}_m\boldsymbol{P}^{-1}_{m+1|m} \\
\hat{\boldsymbol{z}}_{m+1|m} &= \bar{\boldsymbol{f}}_m + \bar{\boldsymbol{A}}_m\hat{\boldsymbol{z}}^*_{m|m} \\
\boldsymbol{P}_{m+1|m} &= \bar{\boldsymbol{A}}_m\boldsymbol{P}^*_{m|m}\bar{\boldsymbol{A}}^{\mathrm{T}}_m + \bar{\boldsymbol{Q}}^{\boldsymbol{z}}_m \\
\bar{\boldsymbol{f}}_m &= \boldsymbol{f}_m + (\boldsymbol{Q}^{\boldsymbol{z}\boldsymbol{\eta}}_m)^{\mathrm{T}}(\boldsymbol{Q}^{\boldsymbol{\eta}}_m)^{-1}(\boldsymbol{\eta}_{m+1} - \boldsymbol{g}_m) \\
\bar{\boldsymbol{A}}_m &= \boldsymbol{A}_m - (\boldsymbol{Q}^{\boldsymbol{z}\boldsymbol{\eta}}_m)^{\mathrm{T}}(\boldsymbol{Q}^{\boldsymbol{\eta}}_m)^{-1}\boldsymbol{B}_m \\
\bar{\boldsymbol{Q}}^{\boldsymbol{z}}_m &= \boldsymbol{Q}^{\boldsymbol{z}}_m - (\boldsymbol{Q}^{\boldsymbol{z}\boldsymbol{\eta}}_m)^{\mathrm{T}}(\boldsymbol{Q}^{\boldsymbol{\eta}}_m)^{-1}\boldsymbol{Q}^{\boldsymbol{z}\boldsymbol{\eta}} \\
\hat{\boldsymbol{z}}^*_{m|m} &= \hat{\boldsymbol{z}}_{m|m} + \boldsymbol{L}_m(\boldsymbol{\eta}_{m+1} - \boldsymbol{g}_m - \boldsymbol{B}_m\hat{\boldsymbol{z}}_{m|m}) \\
\boldsymbol{P}^*_{m|m} &= \boldsymbol{P}_{m|m} - \boldsymbol{L}_m\boldsymbol{M}_m\boldsymbol{L}^{\mathrm{T}}_m \\
\boldsymbol{L}_m &= \boldsymbol{P}_{m|m}\boldsymbol{B}_m\boldsymbol{M}^{-1}_m \\
\boldsymbol{M}_m &= \boldsymbol{B}_m\boldsymbol{P}_{m|m}\boldsymbol{B}^{\mathrm{T}}_m + \boldsymbol{Q}^{\boldsymbol{\eta}}_m.
\end{aligned} \tag{8.27}$$

□

By using Proposition 8.2 to iterate the linear states back from t_k to t_{k-l} and then performing a time update to t_τ, it holds that the linear smoothing density is

$$p(\boldsymbol{z}_\tau|\boldsymbol{\eta}_\tau, \boldsymbol{\eta}^j_{k-l:k-1}, \boldsymbol{x}^i_k, \boldsymbol{y}_{0:k}) = \mathcal{N}(\boldsymbol{z}_\tau|\hat{\boldsymbol{z}}^j_{\tau|k,i}, \boldsymbol{P}^j_{\tau|k,i}). \tag{8.28}$$

With (8.24) and (8.28) inserted in (8.20), an approximation of $p(\boldsymbol{y}_\tau|\boldsymbol{x}^i_k, \boldsymbol{y}_{0:k})$ is

$$p(\boldsymbol{y}_\tau|\boldsymbol{x}^i_k, \boldsymbol{y}_{0:k}) \approx \hat{p}(\boldsymbol{y}_\tau|\boldsymbol{x}^i_k, \boldsymbol{y}_{0:k}) = \frac{1}{M_{\mathrm{BS}}}\sum_{j=1}^{M_{\mathrm{BS}}} p(\boldsymbol{y}_\tau|\boldsymbol{\eta}^j_\tau, \hat{\boldsymbol{z}}^j_{\tau|k,i}) \tag{8.29}$$

where, again, the measurement likelihood is computed similarly to (8.16), namely

$$p(\boldsymbol{y}_\tau|\boldsymbol{z}^j_{\tau|k,i},\boldsymbol{\eta}^j_\tau)=\mathcal{N}(\boldsymbol{y}_\tau|\hat{\boldsymbol{y}}^j_{\tau|k,i},\boldsymbol{\Sigma}^j_{\tau|k,i}),$$

where

$$\begin{aligned}\hat{\boldsymbol{y}}^j_{\tau|k,i}&=\boldsymbol{h}^j_\tau+\boldsymbol{C}^j_\tau\hat{\boldsymbol{z}}^j_{\tau|k,i},\\ \boldsymbol{\Sigma}^j_{\tau|k,i}&=\boldsymbol{C}^j_\tau\boldsymbol{P}^j_{\tau|k,i}(\boldsymbol{C}^j_\tau)^{\mathrm{T}}+\boldsymbol{R}^j_\tau.\end{aligned}$$

Finally, we update the linear estimates and the associated covariances, and thereby find (8.9) for each $i\in\{1,\dots,N\}$, using the measurement update (8.29) as

$$\begin{aligned}\hat{\boldsymbol{z}}^i_{k|k,\tau}&=\hat{\boldsymbol{z}}^i_{k|k}+\frac{1}{M_{\mathrm{BS}}}\sum_{j=1}^{M_{\mathrm{BS}}}\boldsymbol{W}^{j,i}_{k,\tau}\boldsymbol{e}^{j,i}_\tau\\ \boldsymbol{P}^i_{k|k,\tau}&=\boldsymbol{P}^i_{k|k}-\frac{1}{M_{\mathrm{BS}}}\sum_{j=1}^{M_{\mathrm{BS}}}\boldsymbol{W}^{j,i}_{k,\tau}\boldsymbol{\Sigma}^j_{\tau|k,i}(\boldsymbol{W}^{j,i}_{k,\tau})^{\mathrm{T}}\\ \boldsymbol{W}^{j,i}_{k,\tau}&=\boldsymbol{P}^j_{k,\tau|k,i}(\boldsymbol{C}^{j,i}_\tau)^{\mathrm{T}}(\boldsymbol{\Sigma}^j_{\tau|k,i})^{-1}\\ \boldsymbol{\Sigma}^j_{\tau|k,i}&=\boldsymbol{C}^{j,i}_\tau\boldsymbol{P}^j_{\tau|k,i}(\boldsymbol{C}^{j,i}_\tau)^{\mathrm{T}}+\boldsymbol{R}_\tau\\ \boldsymbol{e}^{j,i}_\tau&=\boldsymbol{y}_\tau-\boldsymbol{h}^{j,i}_\tau-\boldsymbol{C}^{j,i}_\tau\hat{\boldsymbol{z}}^j_{\tau|k,i}.\end{aligned}\tag{8.30}$$

The update (8.30) is given in [Zhang and Bar-Shalom, 2012a] for the purely linear, Gaussian setting, and the extension is straightforward. The algorithm is denoted by *RBOOSMBS* and is summarized in Algorithm 8.3. The storage requirements of the algorithm are $\{\hat{\boldsymbol{z}}^j_{m|m},\boldsymbol{P}^j_{m|m},\boldsymbol{\eta}^j_m,w^j_m,\boldsymbol{y}_m\}_{j=1}^D$ for $m=k-l_{\max},\dots,k-1$. Without using rejection sampling, Algorithm 8.3 has computational complexity $O(lNM_{\mathrm{BS}}D)$. This complexity is most often reduced when utilizing Algorithm 8.2, and should be compared with $O(lM_{\mathrm{FF}}+M_{\mathrm{FF}}N)$ for *SERBPF* and $O((l-1)N^3+N^2)$ for *A-PF* in [Zhang and Bar-Shalom, 2012b].

Algorithm 8.3—RBOOSMBS

Input: $\underset{m=k-l}{\overset{k-1}{}}\{\hat{\boldsymbol{x}}^d_{m|m},\boldsymbol{P}^d_{m|m},w^d_m,\boldsymbol{x}^i_k,\boldsymbol{P}^i_{k|k},w^i_k,\boldsymbol{y}_m\}_{d=1}^D$

1: **for** $i=1$ **to** N **do**
2: Set $\boldsymbol{\eta}^j_k=\boldsymbol{\eta}^i_k$ for all $j\in\{1,\dots,M_{\mathrm{BS}}\}$.
3: **for** $m=k-1$ **to** $k-l$ **do**
4: **if** $m>k-l$ **then**
5: Execute Algorithm 8.2.
6: **else**
7: **for** $j=1$ **to** M_{BS} **do**

for $d = 1$ **to** D **do**
Sample forward particle $\boldsymbol{\eta}_m^d$ with probability w_m^d.
Compute the unnormalized smoothing weight $\bar{w}_{m|k,i}^{j,d}$ using (3.34) and (3.36).
end for
Normalize: $w_{m|k,i}^{j,d} = \bar{w}_{m|k,i}^{j,d} \left(\sum_{d=1}^{D} \bar{w}_{m|k,i}^{j,d} \right)^{-1}$.
Set $J(j) = d$ with probability $w_{m|k,i}^{j,d}$.
end for
end if
Set $\boldsymbol{\eta}_{m:k}^j = \{\boldsymbol{\eta}_m^{J(j)}, \boldsymbol{\eta}_{m+1:k}^j\}$ for all $j \in \{1, \ldots, M_{\mathrm{BS}}\}$.
Perform a backward RTS step using (8.27).
end for
Update weight w_k^i using (8.26).
Update mean and covariance using (8.30).
end for

Output: $\{\hat{\boldsymbol{x}}_{k|k,\tau}^i, \boldsymbol{P}_{k|k,\tau}^i, w_{k|k,\tau}^i\}_{i=1}^N$

REMARK 8.1
By using $D \leq N$ particles in Algorithm 8.3, it is possible to trade tracking performance against computation requirements. For unimodal densities it may suffice with using $D \ll N$ particles to get adequate performance, thus saving processing time. See Figure 8.3 for an illustration. Moreover, D can also be used as a tradeoff with the number of smoothing iterations M_{BS}, since sometimes it may be advantageous to perform the smoothing iterations many times instead of utilizing all N particles in each smoothing step. □

REMARK 8.2
If the measurements are stored it is theoretically possible to reorder and reprocess the measurements when an OOSM arrives. Reprocessing includes redoing the data association, which in itself can be challenging [Zhang and Bar-Shalom, 2012b]. Furthermore, *SERBPF* has lower computational complexity and demands less storage for $M_{\mathrm{FF}} \ll N$, compared with reordering and reprocessing. □

8.5 Numerical Results

This section presents a simulation study, in which the proposed algorithms are evaluated on three examples by comparing their performance against various particle filters. For a fair comparison of the algorithms'

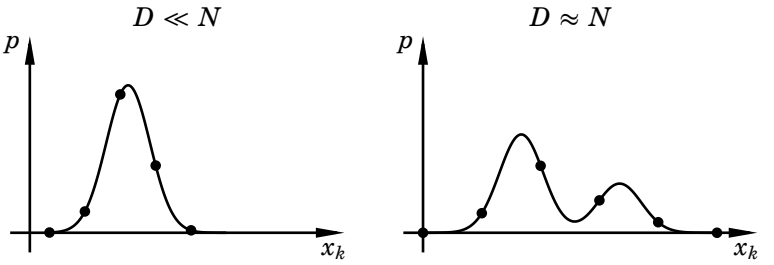

Figure 8.3 The density to the left illustrates an example where it may suffice with only using a subset of the available particles for approximating the density well. To the right is an example where a larger amount of particles typically is needed.

abilities to process the OOSMs, all filters use identical RBPFs in the forward direction, see Algorithm 3.3. The algorithms are implemented in MATLAB and converted to `mex`-files using the Coder toolbox. No measures to code optimization have been taken. The root-mean-square error (RMSE) of the weighted mean at each time step and the time average of it are here used as performance measures, see (7.16) on page 126.

Benchmarked Filters

The methods compared in this section are:

- *RBPFDISC*: An RBPF that discards all OOSMs and thus only processes measurements with zero delay.
- *RBPF*: An offline, idealized RBPF that collects all measurements from both sensors with zero delay. This filter serves as a performance benchmark.
- *A-PF*: The algorithm found in [Zhang and Bar-Shalom, 2012b], derived from exact Bayesian inference.
- *PF-CISI*: The algorithm described in Algorithm 7.2 in Chapter 7, which uses a fixed-point extended Kalman smoother for associating the OOSMs.
- *SERBPF*: The first method proposed in this chapter, described in Section 8.4 and summarized in Algorithm 8.1.
- *RBOOSMBS*: The proposed backward-simulation based method, described in Section 8.4 and summarized in Algorithm 8.3.

Example 1

This example considers the fourth-order MGSS system

$$\begin{aligned} \boldsymbol{z}_{k+1} &= \begin{bmatrix} 1 & 0.3 & 0 \\ 0 & 0.92 & -0.3 \\ 0 & 0.3 & 0.92 \end{bmatrix} \boldsymbol{z}_k + \boldsymbol{w}_k^{\boldsymbol{z}} \\ \eta_{k+1} &= \arctan(\eta_k) + \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \boldsymbol{z}_k + \boldsymbol{w}_k^{\eta} \\ \boldsymbol{y}_k &= \begin{bmatrix} 0.1\eta_k^2 \operatorname{sign}(\eta_k) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 1 & -1 & 1 \end{bmatrix} \boldsymbol{z}_k + \boldsymbol{e}_k. \end{aligned} \tag{8.31}$$

The noise sources are mutually independent, white, and Gaussian distributed according to

$$\begin{aligned} \boldsymbol{w}_k &\sim \mathcal{N}(\boldsymbol{0}_4, 0.01\boldsymbol{I}_{4\times 4}), \\ \boldsymbol{e}_k &\sim \mathcal{N}(\boldsymbol{0}_2, 0.1\boldsymbol{I}_{2\times 2}). \end{aligned}$$

Model (8.31) has been used for benchmarking purposes in research before [Lindsten and Schön, 2010]. Note that it is the same system as in Example 3.1, page 58.

In this chapter, the second sensor (i.e., the second element in $\boldsymbol{y}_k$) has communication issues:[2] A measurement arrives with probability 0.5 and is delayed according to a discrete-valued uniform distribution in the interval $[1, 4]$ samples, with the sampling period $T_s = 1$ s. This should be interpreted as that 50% of the packets of the second sensor are lost on their way to the communication center, and the packets that arrive are delayed between one and four seconds. As a result of that only 50% of the measurements arrive on average, the performance of *RBPF* is impossible to reproduce with any of the other methods. The initial state $\boldsymbol{x}_0$ and covariance matrix $\boldsymbol{P}_0$ for all filters are

$$\begin{aligned} \boldsymbol{x}_0 &= \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}, \\ \boldsymbol{P}_0 &= \operatorname{diag}\left(\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}\right). \end{aligned}$$

The following results are for 20 000 Monte-Carlo simulations, with $T = 50$ time steps in each simulation.

Tracking Performance Figures 8.4 and 8.5 show the RMSE values for all four states, two states in each figure, using $N = 100$ particles in the

[2] Note that simulations where instead the first sensor had communication issues were also performed. This resulted in that all filters had similar performance. The reason is that the second measurement contains more information; consequently, in the following the results focus on when the second sensor delivers OOSMs.

forward filters. Further, *SERBPF* uses $M_{\mathrm{FF}} = 0.1N$ particles in the supporting RBPF and *RBOOSMBS* uses $M_{\mathrm{BS}} = 1, D = N$. Clearly, *RBOOSMBS* performs better than the other methods in terms of RMSE, followed by *SERBPF*. Intuitively, the performance difference between *RBOOSMBS* and *SERBPF* should increase when the delay increases.

At a first glance, the performance differences might seem minor. To give an indication of how close the proposed algorithms are to the lower performance bounds, and to show that the improvements are indeed significant, Figure 8.6 contains a close-up of the RMSE for z_k^2 when also comparing with a particle filter that reorders and reprocesses the measurements when an OOSM arrives (i.e., an optimal filter given the information). This filter is denoted *RERUN* in the figure. As seen, *RBOOSMBS* is very close (approximately 1.5%) to the performance of *RERUN* when compared with the other filters. Moreover, *RBOOSMBS* performs approximately 6% and 10% better than *SERBPF* and *A-PF*, respectively. The other linear states show similar results and for z^1 and η, also the performance of *SERBPF* is close to that of *RERUN*. The relatively low performance of *A-PF* is related to that 100 particles is not enough to reliably associate the estimates with the OOSMs when the model structure in (8.31) is unexploited. Moreover, *PF-CISI* hardly outperforms *RBPFDISC*; that is, in this example there is little gain in using *PF-CISI* compared with discarding the OOSMs. This is caused by the use of the mean of all the particles for initializing the weight update, which does not allow for a good approximation of the smoothing density in this case. This observation again underlines that it is crucial how the OOSM update is done. The estimated distribution for the filtering solution when all measurements are in-sequence measurements (i.e., for *RBPF*), using 500 particles, is found in Figure 3.4 on page 60. Note that the estimated distributions in the OOSM case are different, the reason being less sensor information.

Table 8.1 shows the time-averaged RMSE values for both $N = 100$ and $N = 200$ particles, with the results when using $M_{\mathrm{FF}} = N$ in *SERBPF* added for comparison. The results for *RBOOSMBS* when using $D = 0.25N$ are also added. Using $M_{\mathrm{FF}} = 0.1N$ instead of $M_{\mathrm{FF}} = N$ in *SERBPF* gives almost no decrease in estimation accuracy but much less computational complexity. Likewise, using $D = 0.25N$ in *RBOOSMBS* gives an insignificant change in estimation performance but considerably reduced computation time.

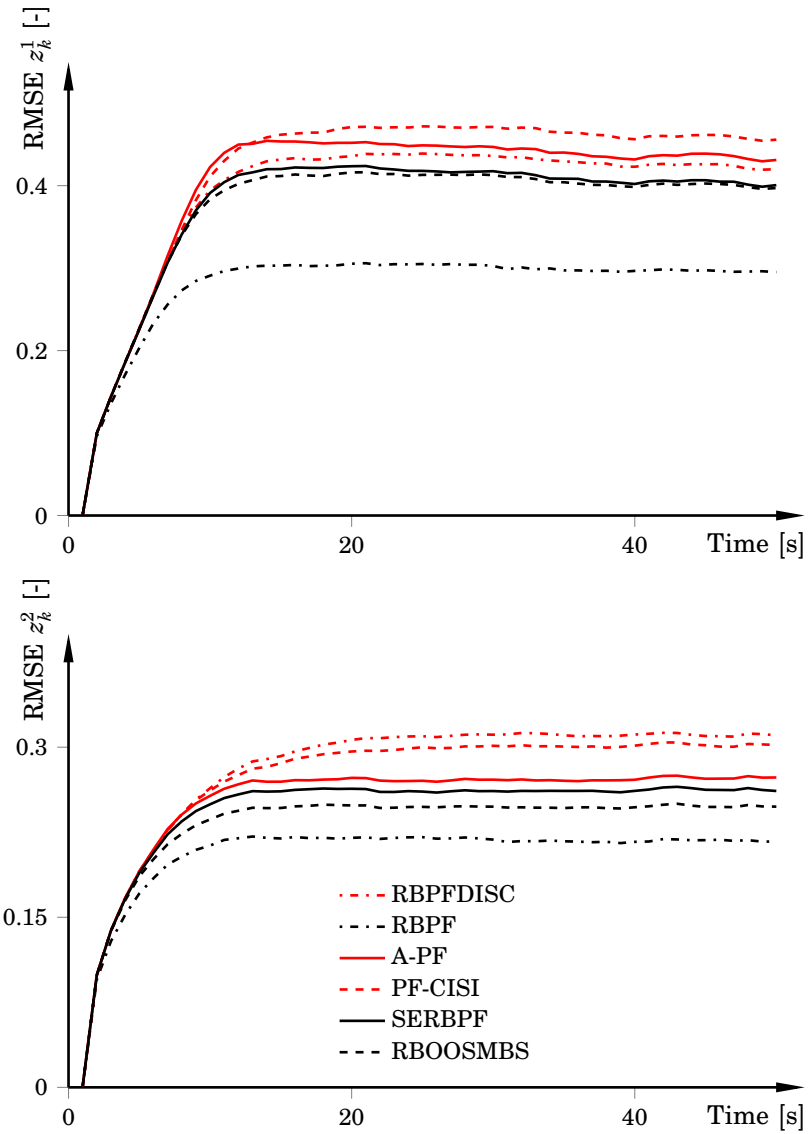

Figure 8.4 RMSEs of the two first linear states in Example 1 for 20 000 Monte-Carlo simulations using 100 particles. *SERBPF* uses $M_{\text{FF}} = 0.1N$ particles in the supporting RBPF and *RBOOSMBS* uses $M_{\text{BS}} = 1$ backward trajectory, something that is enough in most cases for the considered lags. *RBOOSMBS* performs best of the OOSM algorithms, followed by *SERBPF*.

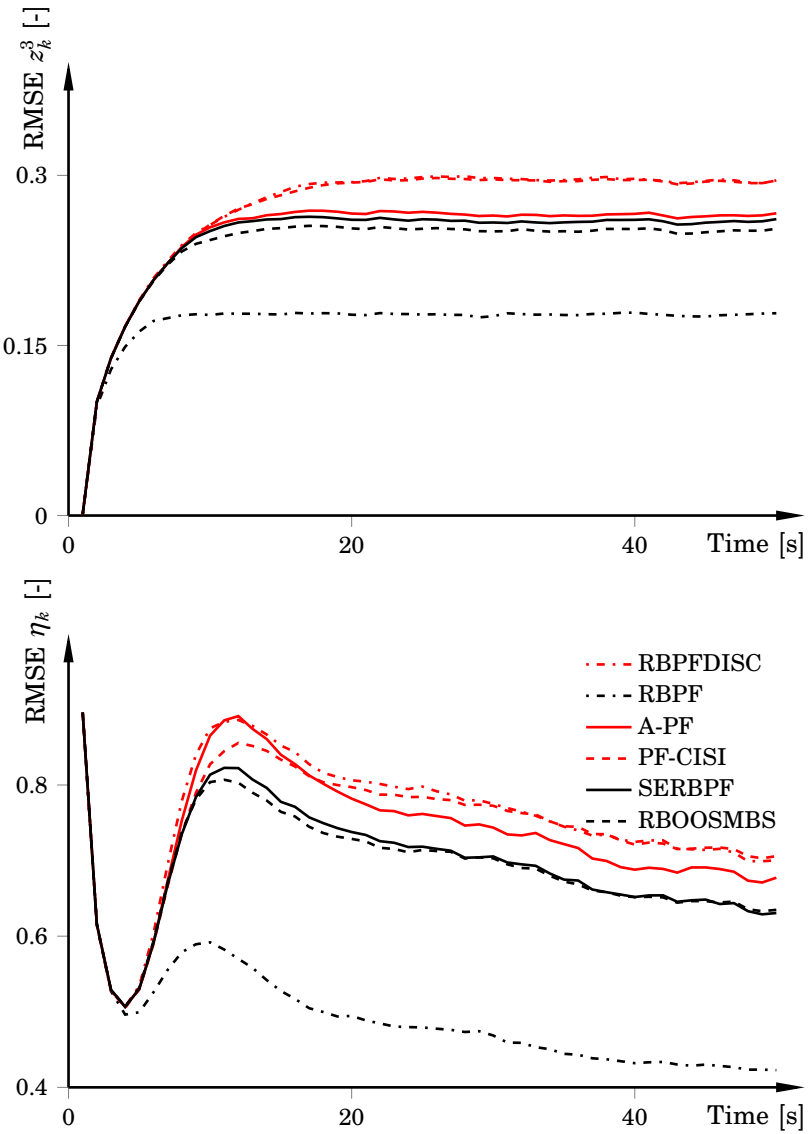

Figure 8.5 RMSEs of the third linear state and the nonlinear state in Example 1 with the same settings as in Figure 8.4. *RBOOSMBS* performs best of the OOSM algorithms followed by *SERBPF*, but the difference is quite small between the two algorithms for the nonlinear state.

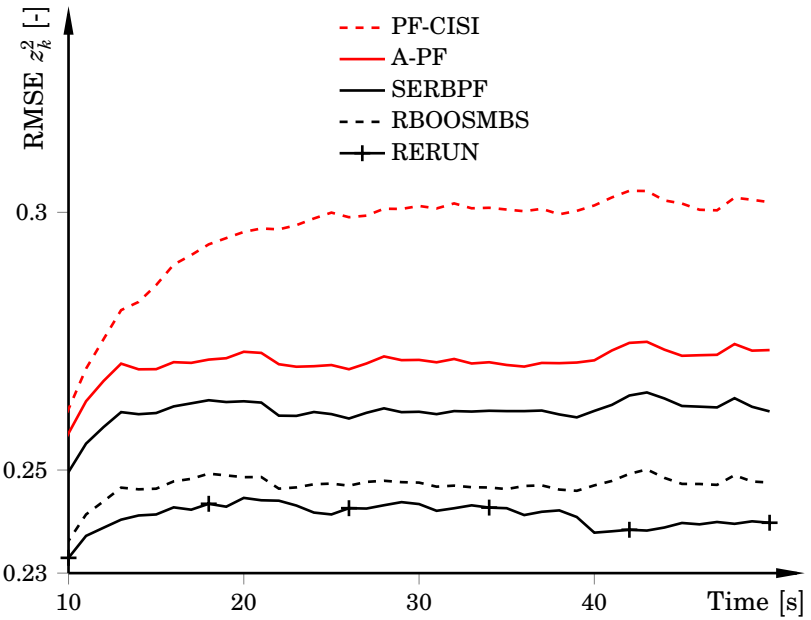

Figure 8.6 A close-up of the RMSEs for z_k^2 in Example 1 obtained by using the OOSM particle filters, with the same settings as in Figure 8.4. *RBOOSMBS*, and to some extent *SERBPF*, have similar performance as *RERUN*.

Table 8.1 Time-averaged RMSE values for Example 1, obtained by executing 20 000 Monte-Carlo simulations with the same settings as in Figure 8.4. For the algorithms that account for OOSMs, the smallest errors are shown in bold font. The number of particles is indicated in the top row. In this example, it is possible to drastically reduce both M_{FF} and D while retaining estimation accuracy.

	$N = 100$			
Algorithm	z^1	z^2	z^3	η
RBPFDISC	0.391	0.281	0.270	0.756
PF-CISI	0.418	0.273	0.269	0.747
A-PF	0.403	0.253	0.248	0.734
SERBPF, $M_{\text{FF}} = 0.1N$	0.378	0.246	0.245	0.707
SERBPF, $M_{\text{FF}} = N$	0.376	0.244	0.243	0.696
RBOOSMBS, $D = 0.25N$	0.371	0.232	**0.236**	0.690
RBOOSMBS, $D = N$	**0.370**	**0.231**	**0.236**	**0.687**
RBPF	0.279	0.206	0.170	0.494
	$N = 200$			
Algorithm	z^1	z^2	z^3	η
RBPFDISC	0.389	0.280	0.269	0.752
PF-CISI	0.414	0.271	0.267	0.740
A-PF	0.400	0.250	0.245	0.728
SERBPF, $M_{\text{FF}} = 0.1N$	0.374	0.244	0.242	0.693
SERBPF, $M_{\text{FF}} = N$	0.373	0.243	0.241	0.689
RBOOSMBS, $D = 0.25N$	0.369	**0.231**	**0.235**	0.686
RBOOSMBS, $D = N$	**0.368**	**0.231**	**0.235**	**0.684**
RBPF	0.278	0.205	0.170	0.490

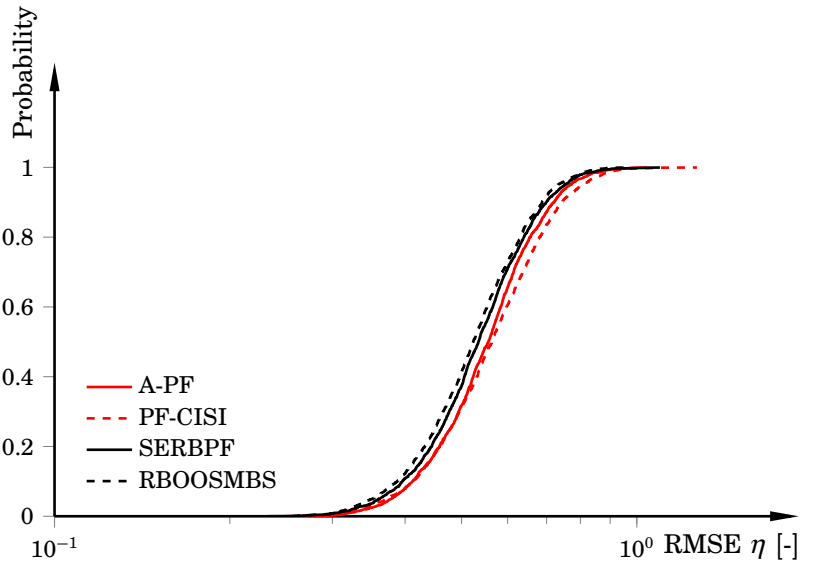

Figure 8.7 The likelihood of time-averaged RMSE for the nonlinear state using the different OOSM filters, with the same settings as in Figures 8.4 and 8.5, computed as a cumulative distribution function. Both *RBOOSMBS* and *SERBPF* have higher probability of achieving smaller errors than the other approaches. As a quantification, *SERBPF* has a probability of 90% to achieve errors smaller than 0.69, whereas the probability for the same error with *A-PF* is 85%.

To indicate the tracking robustness, Figure 8.7 shows the likelihood of the respective time-averaged RMSE over the 20 000 Monte-Carlo simulations. Both *RBOOSMBS* and *SERBPF* have slightly larger likelihood for a smaller error than the other OOSM algorithms. These results are consistent with those presented before. The results for the other states are similar.

Computational Complexity The average execution times of 100 Monte-Carlo simulations as function of forward particles for *A-PF*, *SERBPF*, and *RBOOSMBS* are shown in Figure 8.8 when the delay is fixed to $l = 3$. The implementation is performed in MATLAB and no measures to code optimization have been taken. The execution time for one set of particles is language and implementation dependent. Thus, the absolute execution time is not particularly interesting. Nevertheless, Figure 8.8 indicates the relative complexity increase for increasing N. For $N = 200$, setting $D = 0.25N$ in *RBOOSMBS* gives a reduction in execution time with 75% compared with $D = N$, whereas $M_{\text{FF}} = 0.1N$ gives a 87.5% decrease compared with $M_{\text{FF}} = N$.

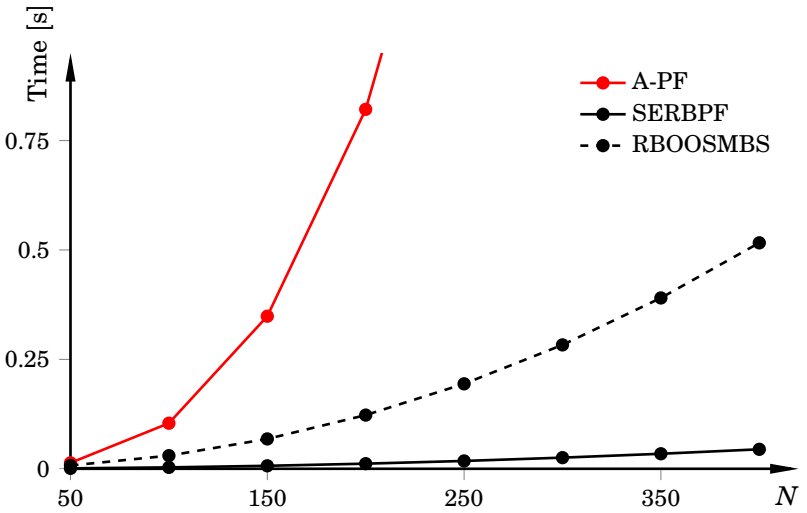

Figure 8.8 Average execution times of 100 Monte-Carlo simulations when implemented in MATLAB for varying number of forward particles. The delay is $l = 3$. We have set $M_{\text{FF}} = N$, $M_{\text{BS}} = 1$, and $D = N$. A decrease to $M_{\text{FF}} = 0.1N$ for *SERBPF* gives a speedup factor of eight for $N = 200$. Similarly, setting $D = 0.25N$ in *RBOOSMBS* gives a speedup factor of four. Note that the execution time for each algorithm includes the computation time of the RBPF used in the forward filtering. The results are for the system in Example 1.

Example 2

In this evaluation we use a fifth-order MGSS model. The nonlinear part is given by

$$\eta_{k+1} = 0.5\eta_k + \theta_k \frac{\eta_k}{1+\eta_k^2} + 8\cos(1.2k) + w_k^\eta, \tag{8.32a}$$

$$y_k = 0.05\eta_k^2 + e_k, \tag{8.32b}$$

with the noise processes distributed according to

$$w_k^\eta \sim \mathcal{N}(0, 0.005),$$
$$e_k \sim \mathcal{N}(0, 0.1).$$

The case when $\theta_k = 25$ in (8.32a) has been used in several papers and books, among them [Gordon et al., 1993; Gustafsson, 2010b; Zhang and Bar-Shalom, 2012b]. Here, θ_k is the output from a linear system with

dynamics given by

$$\boldsymbol{z}_{k+1} = \begin{bmatrix} 3 & -1.691 & 0.849 & -0.3201 \\ 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \end{bmatrix} \boldsymbol{z}_k + \boldsymbol{w}_k^{\boldsymbol{z}} \tag{8.33}$$
$$\theta_k = 25 + \begin{bmatrix} 0 & 0.04 & 0.044 & 0.008 \end{bmatrix} \boldsymbol{z}_k,$$

where $\boldsymbol{w}_k^{\boldsymbol{z}} \sim \mathcal{N}(\boldsymbol{0}_4, 0.01\boldsymbol{I}_{4\times 4})$. Again, the noise sources are mutually independent, white, and Gaussian distributed. The system (8.32)–(8.33) has previously been used in, for example, [Lindsten et al., 2013]. Figure 8.9 shows one realization of the process. Note that the nonlinear state is squared in the measurement equation (8.32b), leading to a bimodal filtering posterior. The initial state and covariance matrix are set to zero for all states, in each filter. The results are based on two data sets, both generated by executing 20 000 Monte-Carlo simulations twice, with 100 time steps in each simulation. The sampling period is $T_s = 1$ s. In both data sets $N = 400$. In the first data set every second measurement is delayed one time step (i.e., $l = 1$), whereas in the second data set every third measurement is delayed two time steps (i.e., $l = 2$). Note that in this example all measurements arrive. Hence, at the OOSM arrival times the performance of *RBPF* is attainable, for a sufficiently large number of particles.

As an illustration of the bimodality, Figure 8.10 displays the estimated posterior for the nonlinear state at every fifth time step, together with the true state. There is a bimodality in the estimated posterior at $t = 31, 65, 86$ s, which shows that it is a demanding filtering problem, even when all measurements arrive without delay.

Tracking Performance Table 8.2 presents the time-averaged RMSE values at the OOSM arrival times (i.e., $k = 1, 3, \ldots, 99$ and $k = 1, 4, \ldots, 100$, respectively) for η_k and θ_k. The different smoothing parameters are indicated in column one. *PF-CISI* is omitted because of inadequate handling of multimodal distributions. The tracking performance of *SERBPF* is superior to both *RBOOSMBS* and *A-PF* for $l = 1$. The reason is that for $l = 1$, the smoothing in *SERBPF* yields a better approximation than only using $M_{\text{BS}} = 1$ in *RBOOSMBS*. Setting $M_{\text{FF}} = 0.4N$ renders performance close to that obtained for $M_{\text{FF}} = N$. However, using fewer than $M_{\text{FF}} = 0.4N$ particles—for example, $M_{\text{FF}} = 0.1N$—does not give better performance than discarding the OOSMs (this is not shown). It is interesting to note that *SERBPF* with $M_{\text{FF}} = 0.4N$ performs better than *RBOOSMBS* with $M_{\text{BS}} = 1$. Furthermore, decreasing D in *RBOOSMBS* gives deficient performance for virtually all $D < N$.

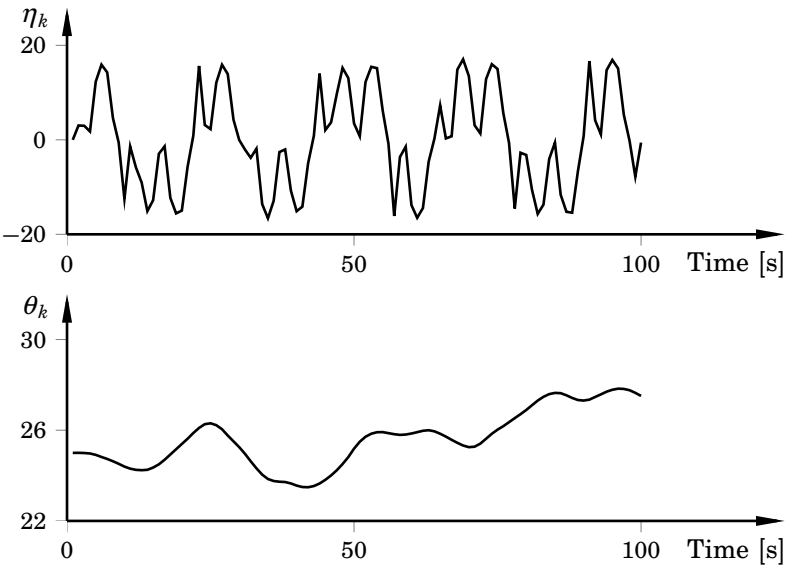

Figure 8.9 One realization of the model that is used in Example 2. The nonlinear state η_k and the output θ_k from the linear subsystem are shown.

Table 8.2 Time-averaged RMSE values at the OOSM arrival times for Example 2, using 20 000 Monte-Carlo simulations. The OOSM delays are indicated in the top row. The errors between $l = 1$ and $l = 2$ are not comparable, since they are measured at different time steps. Setting M_{FF} much smaller than $0.4N$ gives insufficient tracking performance.

	$l = 1$		$l = 2$	
Algorithm	η	θ	η	θ
RBPFDISC	0.453	0.954	0.514	0.895
A-PF	0.448	0.940	0.508	0.892
SERBPF, $M_{\mathrm{FF}} = 0.4N$	0.430	0.856	0.493	0.850
SERBPF, $M_{\mathrm{FF}} = N$	**0.428**	**0.853**	0.484	**0.847**
RBOOSMBS, $D = 0.5N$	0.445	0.950	0.496	0.892
RBOOSMBS, $D = N$	0.437	0.881	**0.476**	0.857
RBPF	0.414	0.844	0.443	0.839

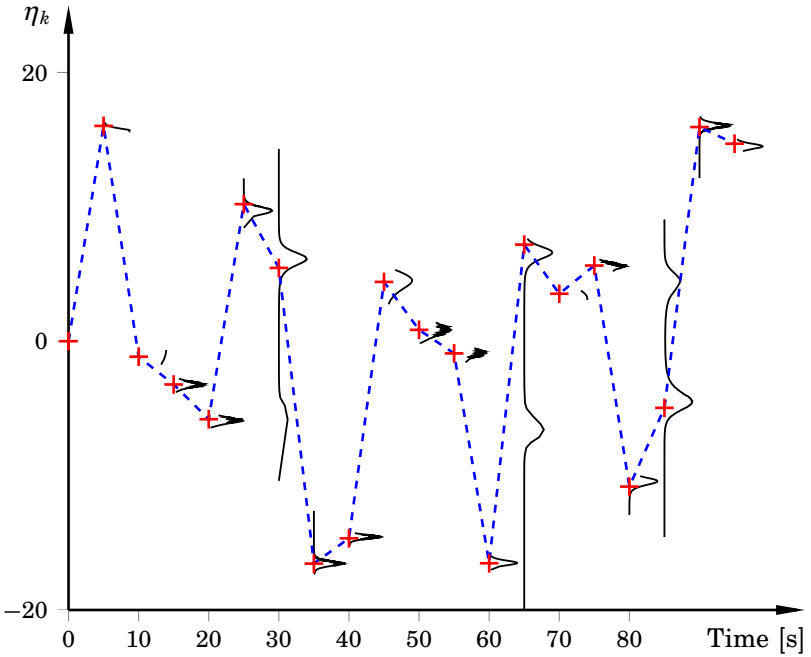

Figure 8.10 Estimated distribution of one realization for the nonlinear state in Example 2 every fifth second. The estimated distribution (black curve), the true state (red +), and resulting estimated trajectory (dashed line) are shown. It is clear that the posterior is bimodal (e.g., at $t = 31, 65, 86$ s).

For $l = 2$, *RBOOSMBS* and *SERBPF* have similar tracking performance, both for η_k and θ_k. The reason for *RBOOSMBS* not consistently performing better than *SERBPF* is that this is a demanding estimation problem where *RBOOSMBS* benefits from using more than one backward trajectory per forward particle. The problem complexity is also indicated by both approaches being more sensitive to the values of D and M_{FF}, respectively. Note that for the fifth-order model given by (8.32)–(8.33), $N = 400$ particles is not enough to consistently approximate the (bimodal) filtering posterior with high accuracy. Still, the estimation accuracy of *SERBPF* for $M_{\text{FF}} = 0.4N$ is close to $M_{\text{FF}} = N$, and indicates that *SERBPF* is quite robust to increased OOSM delay.

An observation is that *A-PF* performs only slightly better than discarding the OOSMs, both for $l = 1$ and $l = 2$. Again, this has probably to do with that $N = 400$ particles is not enough to reliably estimate the filtering posterior. *A-PF* uses a backward recursion to associate the estimates with

the OOSM, which in each step depends on the forward filter particles and weights. Thus, as for the smoothers in *SERBPF* and *RBOOSMBS*, the performance of the OOSM association hinges on a reliable filtering-posterior approximation. The approximation errors accumulate as the OOSM delay increases, which in the end renders an OOSM update that lacks in performance.

Example 3

The third example deals with estimating the states of an aircraft using a simplified two-dimensional constant-acceleration process model [Karlsson et al., 2005]. As opposed to Example 1 and Example 2, this is an example that has a physical interpretation.

The states to be estimated are the position $\boldsymbol{p}_k \in \mathbb{R}^2$, velocity $\boldsymbol{v}_k \in \mathbb{R}^2$, and acceleration $\boldsymbol{a}_k \in \mathbb{R}^2$. The measurements are the range r_k and bearing ψ_k from the radar system to the aircraft. The dynamic equations are linear, with the nonlinearities entering in the measurement equations. Using the sampling period $T_s = 1$ s, the model is

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} p^X_{k+1} \\ p^Y_{k+1} \\ v^X_{k+1} \\ v^Y_{k+1} \\ a^X_{k+1} \\ a^Y_{k+1} \end{bmatrix} = \left[\begin{array}{cc|cccc} 1 & 0 & 1 & 0 & \frac{1}{2} & 0 \\ 0 & 1 & 0 & 1 & 0 & \frac{1}{2} \\ \hline 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \boldsymbol{x}_k + \boldsymbol{w}_k, \tag{8.34a}$$

$$\boldsymbol{y}_k = \begin{bmatrix} r_k \\ \psi_k \end{bmatrix} = \begin{bmatrix} \sqrt{(p^X_k)^2 + (p^Y_k)^2} \\ \arctan\left(\dfrac{p^Y_k}{p^X_k}\right) \end{bmatrix} + \boldsymbol{e}_k, \tag{8.34b}$$

This model can be written on the form (8.1), where the lines in (8.34a) indicate the partition into linear and nonlinear states. Since the position vector $\boldsymbol{p}_k$ shows up nonlinearly in (8.34b), corresponding to (8.1c), it is modeled as the nonlinear part of the state vector. The other four states are linear. Note that the linear and nonlinear states show up in reversed order compared with the standard formulation (8.1). The process noise for $\boldsymbol{p}_k$ is white and Gaussian distributed as

$$\boldsymbol{w}^{\boldsymbol{\eta}}_k \sim \mathcal{N}(\boldsymbol{0}_2, 10\boldsymbol{I}_{2\times 2}),$$

whereas the process noise for the linear states ($\boldsymbol{v}_k$ and $\boldsymbol{a}_k$) are distributed as

$$\boldsymbol{w}^{\boldsymbol{z}}_k \sim \mathcal{N}(\boldsymbol{0}_4, \boldsymbol{Q}^{\boldsymbol{z}}_k), \quad \boldsymbol{Q}^{\boldsymbol{z}}_k = \mathrm{diag}\left(\begin{bmatrix} 1 & 1 & 0.01 & 0.01 \end{bmatrix}\right).$$

The measurement noise $\boldsymbol{e}_k$ is distributed as

$$\mathcal{N}(\mathbf{0}_2, \boldsymbol{R}_k), \quad \boldsymbol{R}_k = \operatorname{diag}\left(\begin{bmatrix} 100 & 10^{-6} \end{bmatrix}\right).$$

In this example the communication with the bearing measurements (i.e., the second element of $\boldsymbol{y}_k$) is corrupted, and on average only 50% of the packets arrive. Moreover, those packages that eventually arrive are delayed between $[0, 3]$ samples according to a discrete uniform distribution.

Tracking Performance Figure 8.11 shows the RMSE for the position p_k^X and velocity v_k^X for 4000 Monte-Carlo simulations using 200 particles, with the initial state and covariance matrix for all filters set to

$$\begin{aligned} \boldsymbol{x}_0 &= \begin{bmatrix} 2000 & 2000 & 10 & 10 & 0 & 0 \end{bmatrix}^{\mathrm{T}}, \\ \boldsymbol{P}_0 &= \operatorname{diag}\left(\begin{bmatrix} 100^2 & 100^2 & 25^2 & 25^2 & 0.2^2 & 0.2^2 \end{bmatrix}\right). \end{aligned}$$

Again the proposed filters perform very well. Compared with the other examples, the filter that associates the OOSMs with the current states using an extended Kalman smoother, *PF-CISI*, now also performs well in terms of RMSE. This is unsurprising since the state dynamics is linear; hence, the Kalman smoother will only be approximate in the measurement update steps. Still, *PF-CISI* has worse tracking performance than the other filters. For this example, especially for the position estimates, *SERBPF* with $M_{\mathrm{FF}} = 0.2N$ performs better than the rest of the OOSM filters. A close-up of the velocity RMSE in Figure 8.12 verifies this, except for some transient behavior in *A-PF*. Various different noise settings in combination with different initial estimates and number of particles have been tried out, and although the backward-simulation based approach, *RBOOSMBS*, most often performs best, *SERBPF* frequently yields very similar performance. A reason that *RBOOSMBS* does not consistently outperform *SERBPF* is probably the linear process dynamics; it is possible to use very few particles for the smoother in *SERBPF* and still obtain accurate estimation accuracy when compared with only using one backward trajectory in *RBOOSMBS*. Note that all OOSM filters are quite close to *RBPF* in performance.

Execution Time The execution time for one time step when the OOSM delay is $l = 3$ s is shown in Table 8.3. To reduce the effects of memory management and operating-system intervention, Table 8.3 displays the minimum execution time. This differs to the approach used for the results in Figure 8.8, where the mean execution time was used as measure. Obviously the execution time is smallest for *PF-CISI*, but the computation time for *SERBPF* is competitive, especially considering that the code is not optimized for performance. In a real-time implementation, the improved tracking performance of *SERBPF* has to be weighed against the lower execution time of *PF-CISI*.

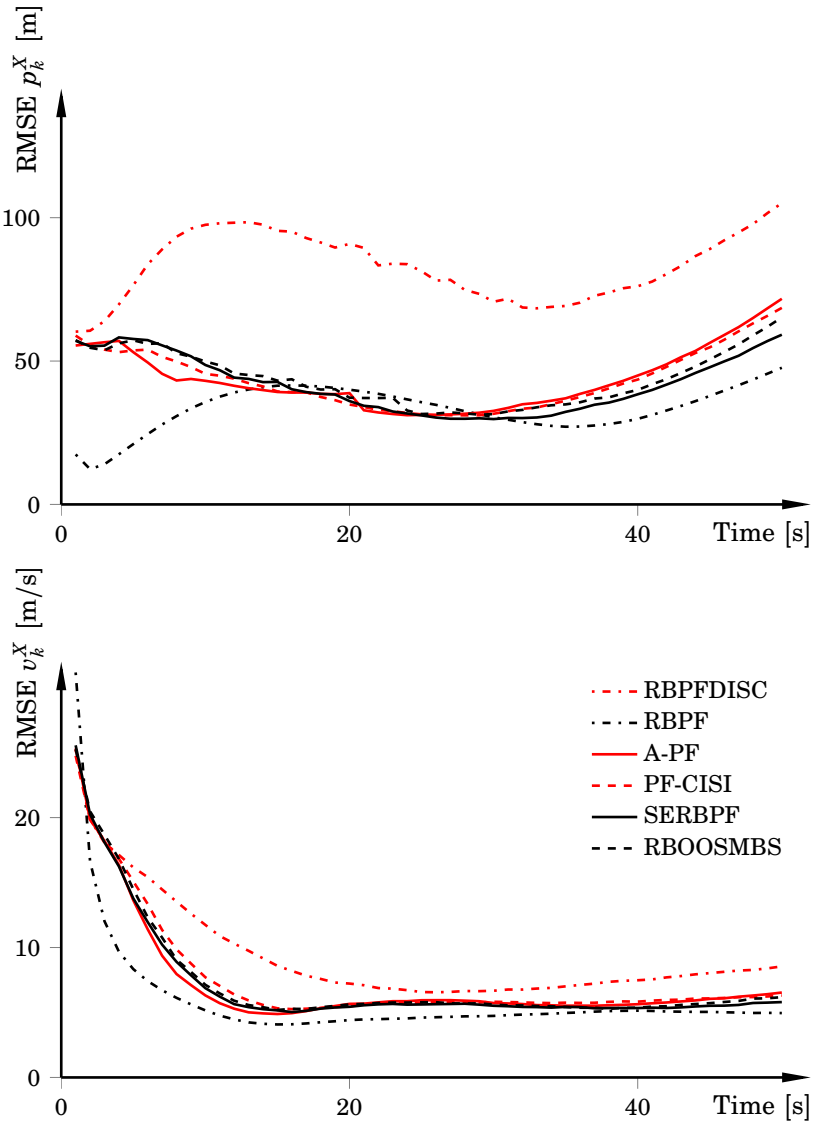

Figure 8.11 RMSEs of the position p_k^X and velocity v_k^X in Example 3 for 4000 Monte-Carlo simulations. All filters use $N = 200$ particles, and $M_{\text{FF}} = 0.2N$. Further, $D = N$ and $M_{\text{BS}} = 1$. The results for p_k^Y and v_k^Y are similar, because of symmetry.

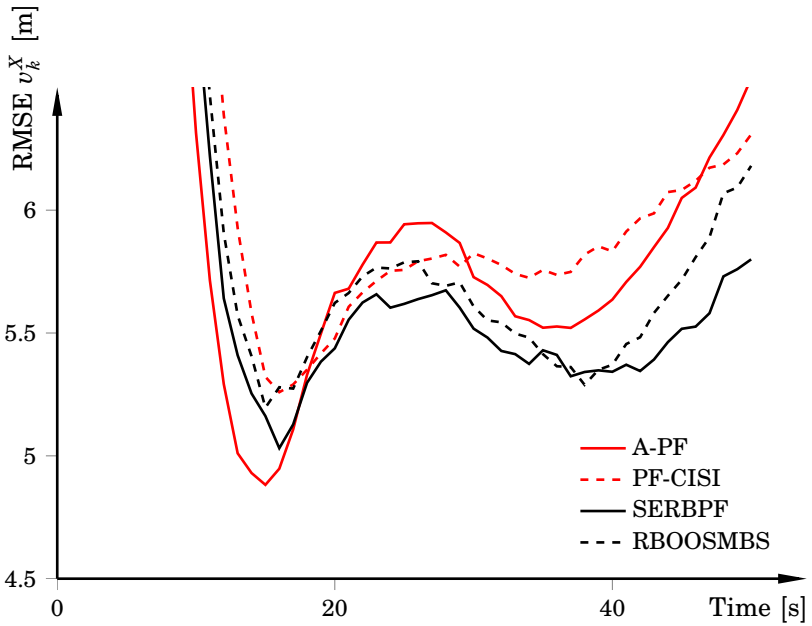

Figure 8.12 A close-up of the RMSE for the velocity v_k^X, using the same settings as in Figure 8.11. *SERBPF* performs best in this example, except for initial transients.

Table 8.3 Minimum execution time for one time step in Example 3, with OOSM delay l set to $l = 3$ s. The implementation is done on a standard desktop PC with an Intel i7 CPU. The implementation is not optimized for performance.

$N = 200$	
Algorithm	Time [s]
A-PF	1.6
RBOOSMBS, $D = N$	0.62
SERBPF, $M_{\text{FF}} = 0.2N$	$3.1 \cdot 10^{-3}$
PF-CISI	$6.2 \cdot 10^{-4}$

Outliers Figures 8.13 and 8.14 display error-bar plots for the four OOSM algorithms *PF-CISI*, *SERBPF* (Figure 8.13); and *A-PF*, *RBOOSMBS* (Figure 8.14). They have been grouped to compare filters that are more focused toward online execution and tracking performance, respectively. There are much more outliers for *PF-CISI* than *SERBPF*. Thus, *SERBPF* does not only give an improvement in tracking performance, but also increased robustness. This is important for online tracking systems.

Figure 8.14 shows that *RBOOSMBS* gives rise to fewer outliers than *A-PF*, which is expected. It is also interesting that *SERBPF* (Figure 8.13) has fewer outliers than *A-PF*. The reason is obviously that *SERBPF* utilizes model structure; still, it is worth stressing that *A-PF* is based on exact Bayesian inference while *SERBPF* uses a rather crude smoothing approximation with $M_{\mathrm{FF}} = 0.2N$ to associate with the OOSMs. These results complement the RMSE errors in Figures 8.11 and 8.12 in that it is important to not only monitor the tracking errors but also the variance of the tracking errors, which is crucial for online tracking systems.

8.6 Discussion

For the considered examples and delays, it is often sufficient to only use one backward trajectory per particle in *RBOOSMBS* and still get excellent estimation accuracy. This is important, especially considering that the computational complexity grows with the number of backward trajectories. The reason for why one backward trajectory per particle often is sufficient is most probably that each trajectory in the smoothing iterations is fixed to a specific forward particle at the endpoint, thus reducing the number of distinct backward trajectories. The performance of *RBOOSMBS* also turned out to be rather insensitive to the choice of D when compared with having $D = N$. However, for some cases it is essential that $D \approx N$. This is, for example, the case when the number of forward particles is small and/or the posterior has a multimodal behavior, as in Example 2. This is also true for the number of backward trajectories. Thus, the choice is problem dependent.

Experience indicates that the number of iterations in the while-loop in Algorithm 8.2 depends heavily on the proposal distribution. If the proposal distribution mimics the target distribution well, an index will often be found after a few iterations only. Hence, in those cases it can be worthwhile to allow for a large $C_{\max}$. However, if the proposal distribution differs a lot from the target distribution, $C_{\max}$ is preferably set to a small value since the maximum number of iterations will probably be reached anyway. In the simulation study the limit was, quite arbitrarily, set to $C_{\max} = D/4$, but the choice is, again, highly problem dependent.

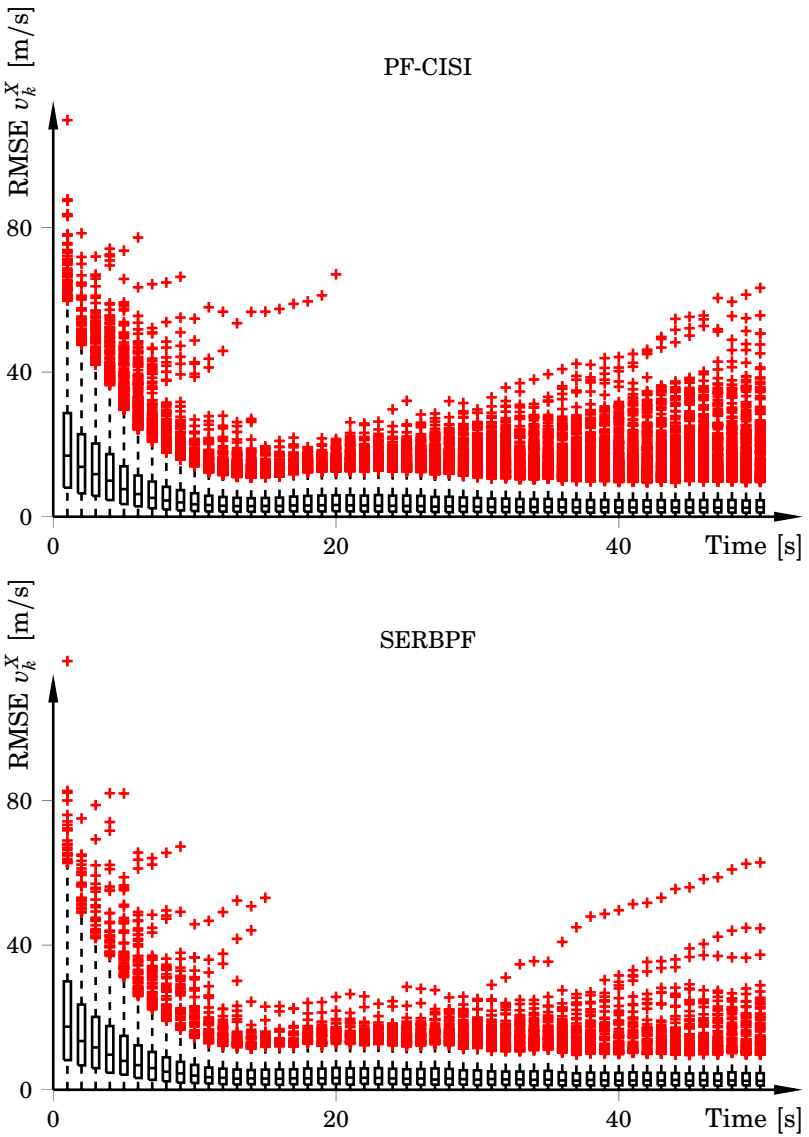

Figure 8.13 Error-bar plots for *PF-CISI* and *SERBPF* corresponding to Example 3 (see Figure 8.11). The boxes indicate the lower and upper quartiles and the medians. Outliers (red +) are position errors that are larger than approximately 2.7 standard deviations, corresponding to 99.3% coverage for Gaussian distributed data. Executing *SERBPF* results in much less outliers than for *PF-CISI*.

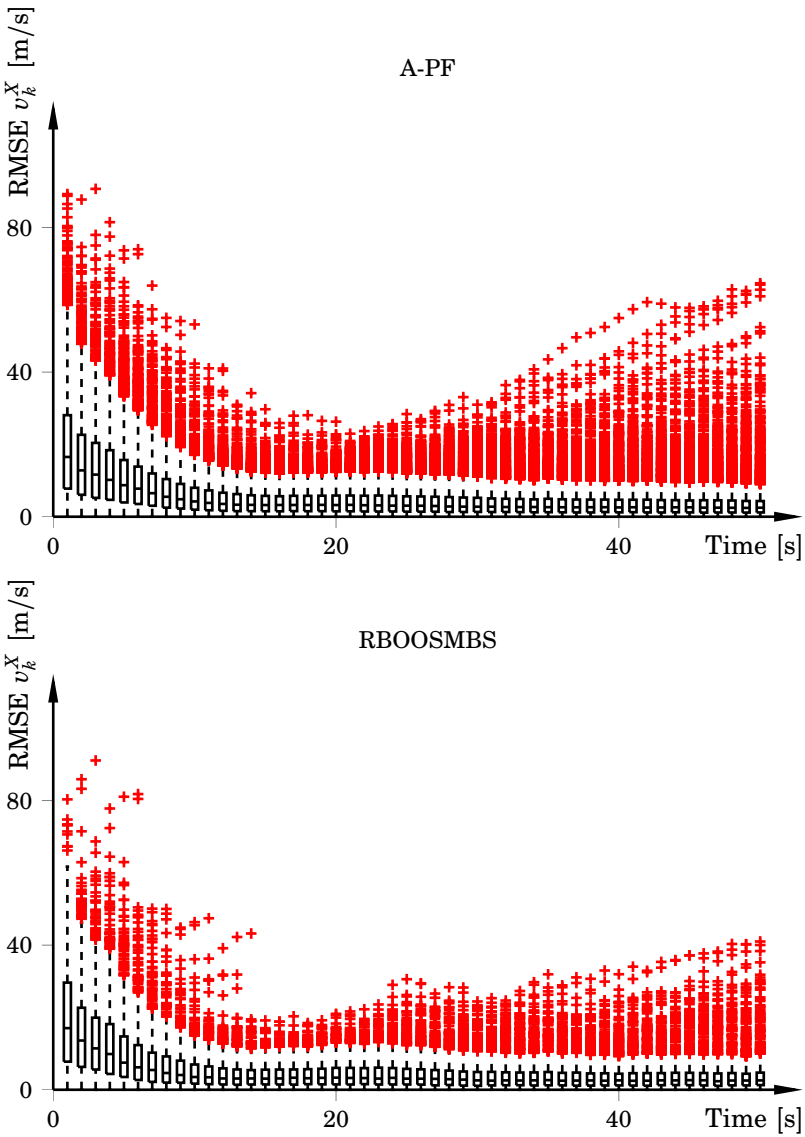

Figure 8.14 Error-bar plots for *A-PF* and *RBOOSMBS* corresponding to Example 3 (see Figure 8.11). The boxes indicate the lower and upper quartiles and the medians. Outliers (red +) are position errors that are larger than approximately 2.7 standard deviations, corresponding to 99.3% coverage for Gaussian distributed data. Executing *RBOOSMBS* results in less outliers than for *A-PF* and also compared with Figure 8.13.

Although $C_{\max}$ was reached in some cases, the rejection sampling on average provided a noticeable speedup compared with explicitly computing all smoothing weights. Note that adaptive schemes can be designed, which possibly can improve performance.

For the considered examples, a suitable number of particles in the supporting RBPF in *SERBPF* considering the tradeoff between performance and computation time, is $0.05N \leq M_{\mathrm{FF}} \leq 0.4N$ depending on noise levels and the number of forward particles used. When N is large the value of M_{FF} can typically be chosen relatively small. The reason is that the filter weights are more accurate; hence, when sampling from the filter weights at time index $k-l$, more consistent samples are drawn than when using few particles. Although the choice of M_{FF} is more insensitive than the choice of D in *RBOOSMBS*, sometimes the tracking performance deteriorates for small M_{FF}, see Example 2 in Section 8.5. Therefore, the choice is also dependent on the degree of multimodality in the posterior.

In many practical applications, it is likely that the number of particles used in the smoothing iterations in both *SERBPF* and *RBOOSMBS* can be drastically reduced without sacrificing estimation accuracy. This is emphasized for *SERBPF* by the results for Example 3 in Section 8.5, where the model structure is similar to those used in many tracking applications, although somewhat simplified. The outlier plots in Figures 8.13 and 8.14 show that reducing M_{FF} to 20% of the number of forward particles N in the RBPF still gives very robust tracking performance. It should be mentioned that setting D smaller than N in Example 3 does not drastically decrease tracking performance, although setting it lower than $D = 0.5N$ seems to render poor performance (depending on noise levels, obviously). Consequently, both algorithms can be feasible alternatives in online implementations. However, with the computing power currently available, real-time tracking of full-scale problems is only possible for *SERBPF*.

8.7 Summary

This chapter presented two new algorithms for OOSM processing considering mixed-Gaussian state-space models, which is a rather general class of conditionally linear state-space models. Both algorithms use Rao-Blackwellization to exploit the conditionally linear Gaussian substructure in the model. One of them focus on fast execution and the other on estimation accuracy. Simulation examples showed that both approaches yield improvements, both in terms of RMSE and repeatability, when compared with recent particle filter algorithms for OOSM processing. In terms of computational complexity, *SERBPF* is the most viable option. The results showed that *RBOOSMBS* is the preferred choice if execution time is not

a concern. Both out-of-sequence measurements and the considered model class are common in target tracking, positioning, and navigation scenarios. Thus, the developed algorithms enable performance improvements in relevant filtering applications.

8.8 Proofs

Proof Proposition 8.1 Follows by application of results in [Schön et al., 2005] on how to calculate expected means and their covariances in RBPFs and in [Zhang and Bar-Shalom, 2012a] on how to update linear states with the OOSM. □

Proof Proposition 8.2 It holds that

$$p(\boldsymbol{z}_m|\boldsymbol{\eta}_{m:k}, \boldsymbol{y}_{0:k}) = \int p(\boldsymbol{z}_m, \boldsymbol{z}_{m+1}|\boldsymbol{\eta}_{m:k}, \boldsymbol{y}_{0:k}) \, \mathrm{d}\boldsymbol{z}_{m+1}.$$

Moreover, using Bayes' rule and the Markov property of the linear states, gives that

$$\begin{aligned} p(\boldsymbol{z}_m|\boldsymbol{\eta}_{m:k}, \boldsymbol{y}_{0:k}) = p(\boldsymbol{z}_m|\boldsymbol{\eta}_{m:m+1}, \boldsymbol{y}_{0:m}) \\ \cdot \int \frac{p(\boldsymbol{z}_{m+1}|\boldsymbol{z}_m, \boldsymbol{\eta}_{m:m+1}) p(\boldsymbol{z}_{m+1}|\boldsymbol{\eta}_{m:k}, \boldsymbol{y}_{0:k})}{p(\boldsymbol{z}_{m+1}|\boldsymbol{\eta}_{m:m+1}, \boldsymbol{y}_{0:m})} \, \mathrm{d}\boldsymbol{z}_{m+1}. \end{aligned}$$

The density $p(\boldsymbol{z}_{m+1}|\boldsymbol{\eta}_{m:k}, \boldsymbol{y}_{0:k})$ is given by the previous smoothing step. The rest is analogous to the constrained Kalman-filter derivations for the RBPF in [Schön et al., 2005]. □

9

Particle Filter for Wheel-Slip and Motion Estimation

The last two chapters have dealt with particle-filter algorithms for handling out-of-sequence measurements. The algorithms are extensions of the standard particle-filter variants with smoothing techniques. In this chapter we address the state-estimation problem in vehicles using a standard Rao-Blackwellized particle filter.

9.1 Motivation and Contributions

Vehicle safety systems have traditionally relied on their own set of sensors to estimate the states needed for the particular control application. With the improved computing and networking capabilities in modern vehicles during the last decade, made available by, for example, the controller area network (CAN) bus [Johansson et al., 2005], opportunities exist for the different systems to exchange information. It is therefore possible to utilize sensor-fusion techniques within automotive systems, something that has gained interest in recent years [Lundquist, 2011].

In the automotive industry, it is common practice to use a high-level safety system for computing the brake forces that are needed to stabilize the vehicle. It is then up to an anti-lock braking system (ABS) to achieve the desired brake forces through wheel-slip control [Solyom, 2004; Johansen et al., 2003; Berntorp, 2008; Savaresi and Tanelli, 2010], see Figure 9.1 for a typical force-slip relation. The ABS uses estimates of the vehicle velocity and longitudinal wheel slip. These quantities are typically estimated using wheel-speed sensor signals and/or measurements of longitudinal acceleration [Gustafsson, 1997; Savaresi and Tanelli, 2010]. Although the estimates are reliable in many situations there is much room for improvements, especially for all-wheel drive vehicles and when cornering [Solyom, 2004; Imsland et al., 2006; Savaresi and Tanelli, 2010;

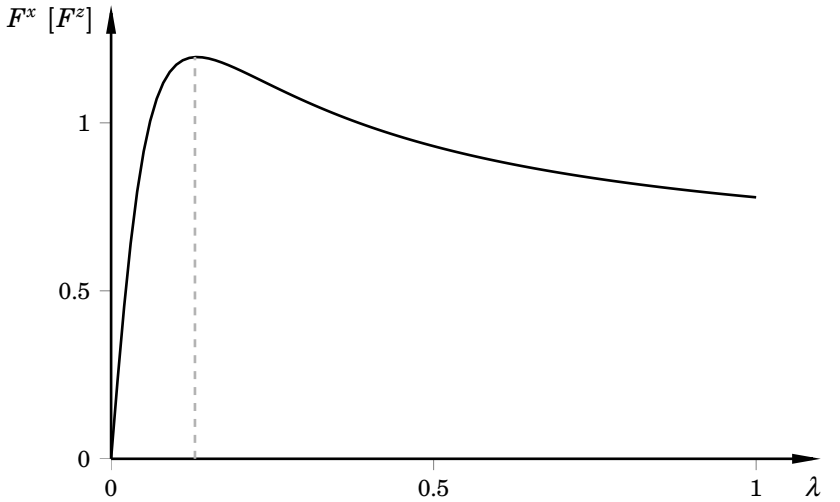

Figure 9.1 Longitudinal force F^x as function of longitudinal wheel slip λ. ABS systems aim to control the wheel slip around, possibly time varying, reference values, determined by a high-level safety system. The behavior is stable to the left of the peak and unstable to the right of the peak. Hence, reliable estimates of the wheel slip is necessary for good performance.

Reif and Dietsche, 2011]. As discussed in [Savaresi and Tanelli, 2010], insufficient knowledge of wheel slip can cause instability in the closed-loop control. Examples of high-level safety systems are electronic stability control systems (ESCs), which rely on lateral-acceleration, steer-angle, and yaw-rate measurements to estimate the lateral acceleration and vehicle sideslip angle [Tseng et al., 1999], and rollover-avoidance systems, which rely on roll-rate and lateral-acceleration measurements to estimate the roll angle. Typically there is a cross dependence between the states; for example, the roll angle affects the longitudinal and lateral accelerations in the ABS and ESC, respectively, and vice versa. Further, [Gustafsson, 2009] points out that accurate state information is more important than advanced algorithms in current vehicles. With improved sensor-fusion techniques in automotive systems, however, advanced model-based control algorithms and principles are possible.

This chapter outlines a novel approach to model-based joint wheel-slip and motion estimation of four-wheeled ground vehicles. The method fuses wheel-encoder, acceleration, gyro, and (optionally) global positioning system (GPS) position measurements to create estimates of the vehicle's pose, translational velocities, and wheel slip. The main contributions are:

- Unlike other approaches that deal with slip estimation, we here ex-

plicitly model the nonlinear slip dynamics in the state and measurement equations. The slip dynamics are then combined with a dynamic model of the translational and rotational entities, which are all estimated simultaneously in an RBPF.

- The method only relies on kinematic relations, which makes it robust to parameter uncertainties; for example, neither parameters describing the ground-wheel interaction, which otherwise need to be estimated for multiple terrain types [Svendenius, 2007], nor mass parameters are required. A drawback with the proposed approach is that some of the states enter the dynamic equations in a highly nonlinear fashion. However, a vast majority of the states are either linear or close to linear, which makes the state estimation problem tractable in terms of execution time.

The approach is validated using a Volkswagen Golf V 2008, equipped with external sensors for measuring wheel slip, vehicle velocities, and rotation angles with high accuracy [Lundahl et al., 2013b]. These measurements are used as ground truth.

We will next go through related work. The assumptions that are introduced for enabling the algorithm are then stated, which is followed by a section devoted to deriving the model and its structure. The chapter then proceeds with the results section and some concluding remarks.

9.2 Related Work

Nonlinear approaches have been used in relation to slip estimation before, see [Carlson and Gerdes, 2005] for a nonlinear least-squares approach for longitudinal tire-stiffness estimation in automotive systems and [Ward and Iagnemma, 2008; Yi et al., 2009] for extended Kalman-filter solutions to slip estimation in mobile robotics. Nonlinear estimation schemes have also been used for motion estimation; see [Imsland et al., 2006] for estimation of vehicle speed via nonlinear observers that utilize a dynamic description of the tire-road interaction. However, no approach based on particle filters, which can accurately capture the nonlinear wheel-slip dynamics, has been reported. Moreover, slip estimation techniques are typically either rule based and modeled for straight-line driving [Ward and Iagnemma, 2008; Savaresi and Tanelli, 2010; Iagnemma and Ward, 2009]; assume kinematic constraints on the experimental setup [Yi et al., 2009]; restricted to two-wheel drive [Gustafsson, 1997; Carlson and Gerdes, 2005]; or assume knowledge of, or the need to, estimate physical vehicle parameters before being possible to use [Ward and Iagnemma, 2008].

Other related research on motion estimation includes [Reina et al., 2006], where visual information was introduced as a means to estimate wheel sinkage. Linearizations of the force-slip curve were used in [Miller et al., 2001] to estimate the longitudinal tire stiffness. Motor-current based wheel-slip detection was proposed in [Ojeda et al., 2005]. A fuzzy-logic approach for choosing Kalman gains was introduced in [Kobayashi et al., 1995]. Moreover, [Ray, 1997] used a nine degrees-of-freedom vehicle model in a Kalman filter to estimate vehicle speed, brake forces, wheel slip, and vehicle sideslip angle. In [Lghani, 2012] a roll-angle and road-bank estimation procedure using a proportional-integral observer was introduced, and [Grip et al., 2009] presented an experimentally verified roll-angle estimation scheme with accompanying stability proofs. Further, [Ryu and Gerdes, 2004] estimated the roll angle using a dynamic model. Another paper is [Lundquist et al., 2014], in which the effective wheel radius was estimated during normal driving conditions.

9.3 Preliminaries

Figure 9.2 contains the kinematics schematics for a four-wheeled ground vehicle and the corresponding notation, and Figure 9.3 shows the degrees of freedom and the notation for the wheels. We introduce the following assumptions:

- The vehicle has two rotational degrees of freedom, being the roll angle ϕ and yaw angle ψ. This is motivated by that roll and yaw are the rotational quantities of interest when considering vehicle-safety aspects [Tseng et al., 1999]. Further, the approximations $\cos(\phi) \approx 1$ and $\sin(\phi) \approx \phi$ are introduced since the roll angle typically is below 5 deg, even during extreme maneuvering [Lundahl et al., 2013a].[1]
- The vehicle moves in the plane—that is, vertical movements are assumed small. This is motivated by that vertical movements will have a minor impact on the rest of the states for small inclination and road bank angles.
- Measurements of the longitudinal and lateral accelerations $\boldsymbol{a} \in \mathbb{R}^2$, wheel angular velocities $\boldsymbol{\omega} \in \mathbb{R}^4$, wheel steer angles $\boldsymbol{\delta} \in \mathbb{R}^4$, and roll and rotation rates $\boldsymbol{\xi} \in \mathbb{R}^2$ are available. These measurements can be considered standard equipment for vehicles equipped with ABS, ESC, and rollover-avoidance system. Consumer cars are often

[1] Figure 1.5 seems to contradict this claim. However, since the estimates are intended to be used in safety systems, or situation-aware systems, it is feasible to assume that situations as those in Figure 1.5 do not occur.

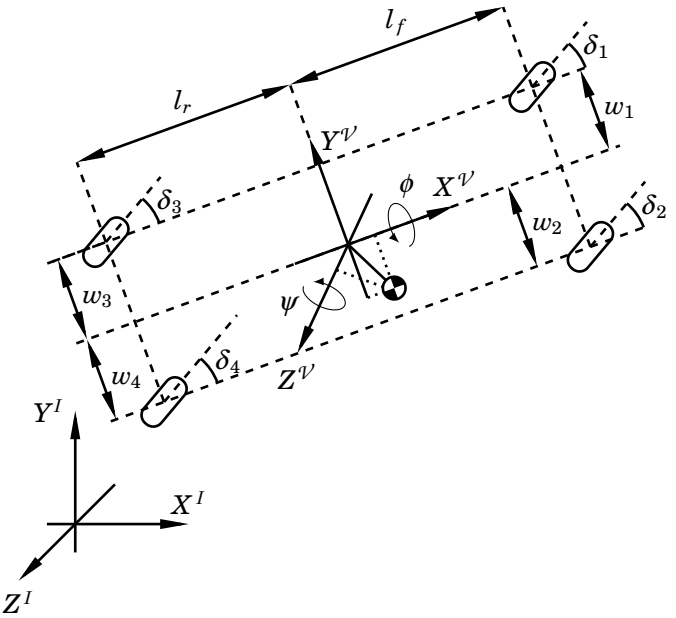

Figure 9.2 The vehicle model and its degrees of freedom. The wheels are numbered from the front left wheel to the rear right wheel. The coordinate axes of the vehicle-fixed and the inertial frames are denoted with capital letters.

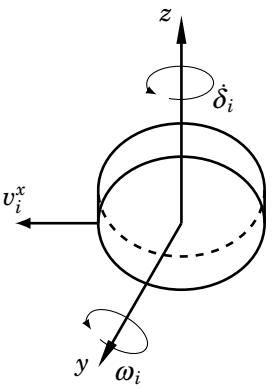

Figure 9.3 Wheel i and its two rotational degrees of freedom. The coordinate axes of the wheels' coordinate systems are denoted with small letters.

equipped with GPS—for example, cars with a navigation system. We do not assume, but can incorporate, GPS measurements in the presented framework.

With a slight modification of notation compared with Chapter 4, with $\boldsymbol{p}_k = \begin{bmatrix} p_k^X & p_k^Y \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^2$ we mean the longitudinal and lateral positions of the vehicle's mass center at time index k expressed in the inertial frame I, and $\dot{\boldsymbol{p}}_k = \boldsymbol{v}_k \in \mathbb{R}^2$ represents the corresponding velocity vector. The composite rotation between the inertial frame I and the body-fixed frame $\mathcal{B}$ is described by $\boldsymbol{\psi} = \begin{bmatrix} \phi & \psi \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^2$. This composite rotation is defined as a rotation ψ about Z^I, followed by a rotation ϕ about the resulting $X_{\mathcal{V}}$-axis. The rotation matrix $\boldsymbol{R}_{\mathcal{V}}^I \in \mathbb{R}^{2\times 2}$ from $\mathcal{V}$ to I is

$$\boldsymbol{R}_{\mathcal{V}}^I = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix}.$$

The superscript $\mathcal{V}$ on a vector, as in $\boldsymbol{v}^{\mathcal{V}}$, means $\boldsymbol{v}$ with respect to I, expressed in frame $\mathcal{V}$. With $\boldsymbol{u_a} \in \mathbb{R}^2$ and $\boldsymbol{u_\xi} \in \mathbb{R}^2$ we mean the accelerometer and gyroscope measurements, respectively, which are mounted (hence also expressed) in frame $\mathcal{B}$. When using the rotation matrix $\boldsymbol{R}_{\mathcal{B}}^{\mathcal{V}} \in \mathbb{R}^{3\times 3}$ (see Appendix A), the two-dimensional acceleration and rotational rate vectors are appended with zeros in the Z- and Y- components, respectively. $\boldsymbol{R}_{\mathcal{B}}^I$ denotes the rotation matrix from $\mathcal{B}$ to I.

9.4 Modeling

The mass center is assumed to be located at center of geometry (CoG). We do not impose restrictions on the wheels' steer or drive angles; that is, the wheels can steer and drive independently from each other. This allows for more general vehicle classes than standard cars, such as vehicles utilizing independent front and rear axle steering, or pseudo-omnidirectional mobile robots. Note that we in this chapter do not make any assumptions on whether the vehicle is of all-wheel drive type or not.

Let the velocity of CoG with respect to frame I expressed in frame $\mathcal{V}$ be $\boldsymbol{v}^{\mathcal{V}} = \begin{bmatrix} v^{\mathcal{V},X} & v^{\mathcal{V},Y} \end{bmatrix}^{\mathrm{T}}$. Then the longitudinal velocities at the wheel center contact points, $\{v_i^x\}_{i=1}^4$, are equal to

$$\begin{aligned}
v_1^x &= \cos(\delta_1)(v^{\mathcal{V},X} - w_1\dot{\psi}) - \sin(\delta_1)(v^{\mathcal{V},Y} + l_f\dot{\psi}) \\
v_2^x &= \cos(\delta_2)(v^{\mathcal{V},X} + w_2\dot{\psi}) - \sin(\delta_2)(v^{\mathcal{V},Y} + l_f\dot{\psi}) \\
v_3^x &= \cos(\delta_3)(v^{\mathcal{V},X} - w_3\dot{\psi}) - \sin(\delta_3)(v^{\mathcal{V},Y} - l_r\dot{\psi}) \\
v_4^x &= \cos(\delta_4)(v^{\mathcal{V},X} + w_4\dot{\psi}) - \sin(\delta_4)(v^{\mathcal{V},Y} - l_r\dot{\psi}).
\end{aligned} \tag{9.1}$$

There exists several definitions for wheel slip. The longitudinal wheel slip for wheel i is here defined as

$$\lambda_i := 1 - \frac{R_i \omega_i}{v_i^x}, \quad \forall i \in \{1, \ldots, 4\}, \tag{9.2}$$

where R_i is the wheel radius for wheel i. This definition implies that $\lambda_i \in (-\infty, 1]$, and is slightly different compared with (4.2) and (4.3). The reason for this definition is that it simplifies the modeling.

Continuous-Time Dynamic Model

The aim is to perform joint vehicle-motion and wheel-slip estimation. Hence, the minimum set of states to estimate are the position $\boldsymbol{p} \in \mathbb{R}^2$, rotation $\boldsymbol{\psi} \in \mathbb{R}^2$, velocity $\boldsymbol{v} \in \mathbb{R}^2$, and the wheel slip $\boldsymbol{\lambda} \in \mathbb{R}^4$. We model the acceleration and gyro measurements $\boldsymbol{u_a}$ and $\boldsymbol{u_\xi}$ as process inputs. Typically, low-cost inertial measurement units suffer from biased measurements [Woodman and Harle, 2008]. A common way to model the bias is to assume first-order Markov processes for both accelerometer, $\boldsymbol{b_a} = \begin{bmatrix} b_{\boldsymbol{a}}^X & b_{\boldsymbol{a}}^Y \end{bmatrix}^{\mathrm{T}}$, and gyroscope, $\boldsymbol{b_\xi} = \begin{bmatrix} b_{\boldsymbol{\xi}}^X & b_{\boldsymbol{\xi}}^Z \end{bmatrix}^{\mathrm{T}}$, as

$$\dot{b} = -\frac{1}{T_b} b + w_b, \tag{9.3}$$

where T_b denotes the time constant. The bias noise term w_b, which is modeled as white Gaussian with zero mean, can be determined from an Allan variance analysis [IEEE, 1996] for each coordinate axis of the accelerometer and gyroscope. For measurement series lasting less than a few minutes, the bias is typically dominated by a constant offset. For the experiments in this chapter, we therefore neglect the first term in (9.3). Thus, in our model the accelerometers and gyroscopes measure

$$\boldsymbol{u_a} = \boldsymbol{a}^{\mathcal{B}} + \boldsymbol{b_a} + \boldsymbol{g} + \boldsymbol{w_a}, \tag{9.4a}$$

$$\boldsymbol{u_\xi} = \boldsymbol{\xi}^{\mathcal{B}} + \boldsymbol{b_\xi} + \boldsymbol{w_\xi}, \tag{9.4b}$$

where $\boldsymbol{g} = \begin{bmatrix} 0 & g\phi \end{bmatrix}^{\mathrm{T}}$ is the component vector from gravity caused by the roll angle ϕ. Moreover, $\boldsymbol{w_a}$ and $\boldsymbol{w_\xi}$ are modeled as zero mean, white Gaussian noise terms. This gives that the kinematic equations for the position,

velocity, and bias states are

$$\dot{\boldsymbol{p}} = \boldsymbol{v}, \tag{9.5a}$$
$$\dot{\boldsymbol{v}} = \boldsymbol{R}_{\mathcal{B}}^{I}(\boldsymbol{u_a} - \boldsymbol{b_a} - \boldsymbol{g} + \boldsymbol{w_a}), \tag{9.5b}$$
$$\dot{\boldsymbol{b}}_{\boldsymbol{a}} = \boldsymbol{w}_{\boldsymbol{b}_a}, \tag{9.5c}$$
$$\dot{\boldsymbol{b}}_{\boldsymbol{\xi}} = \boldsymbol{w}_{\boldsymbol{b}_\xi}, \tag{9.5d}$$
$$\dot{\phi} = (u_{\boldsymbol{\xi}}^{X} - b_{\boldsymbol{\xi}}^{X} + w_{\boldsymbol{\xi}}^{X}), \tag{9.5e}$$
$$\dot{\psi} = u_{\boldsymbol{\xi}}^{Z} - b_{\boldsymbol{\xi}}^{Z} + w_{\boldsymbol{\xi}}^{Z}, \tag{9.5f}$$

The kinematic equations (9.5a)–(9.5d) describe the relation between the position $\boldsymbol{p}$ and velocity $\boldsymbol{v}$, expressed in the inertial frame I, and the acceleration $\boldsymbol{a}^{\mathcal{B}}$ and associated bias $\boldsymbol{b_a}$, expressed in the body frame $\mathcal{B}$. The kinematic equations (9.5e)–(9.5f) describe the relation between the rotation angles with respect to I and the rotation rates $\boldsymbol{\xi}^{\mathcal{B}}$ with associated bias $\boldsymbol{b}_{\boldsymbol{\xi}}$, expressed in the body frame $\mathcal{B}$.

To derive dynamic equations for the slip, differentiate (9.2):

$$\begin{aligned}
\dot{\lambda}_i &= \frac{\partial \lambda_i}{\partial v_i^x}\frac{\mathrm{d}v_i^x}{\mathrm{d}t} + \frac{\partial \lambda_i}{\partial \omega_i}\frac{\mathrm{d}\omega_i}{\mathrm{d}t} \\
&= \frac{\partial\left(1 - \frac{R_i\omega_i}{v_i^x}\right)}{\partial v_i^x} a_i^x + \frac{\partial\left(1 - \frac{R_i\omega_i}{v_i^x}\right)}{\partial \omega}\dot{\omega}_i \\
&= \frac{R_i\omega_i}{(v_i^x)^2} a_i^x - \frac{R_i}{v_i^x}\dot{\omega}_i.
\end{aligned} \tag{9.6}$$

Rearranging the slip definition (9.2) gives that

$$\frac{R_i\omega_i}{v_i^x} = 1 - \lambda_i, \tag{9.7}$$

and inserting (9.7) into the first term in (9.6), results in the following model of the slip dynamics:

$$\dot{\lambda}_i = \frac{1-\lambda_i}{v_i^x} a_i^x - \frac{R_i}{v_i^x}\dot{\omega}_i, \quad \forall i \in \{1,\ldots,4\}. \tag{9.8}$$

To find a_i^x, it is convenient to utilize that the relation

$$\boldsymbol{a}_i = \boldsymbol{a}^{\mathcal{V}} + \ddot{\boldsymbol{\psi}} \times \boldsymbol{d}_i^{\mathcal{V}} + \dot{\boldsymbol{\psi}} \times (\dot{\boldsymbol{\psi}} \times \boldsymbol{d}_i^{\mathcal{V}}), \tag{9.9}$$

holds, compare with (4.16) (page 76). In (9.9), $\boldsymbol{a}_i^{\mathcal{V}}$ is the acceleration of a point P_i with coordinates $\boldsymbol{d}_i^{\mathcal{V}}$ and $\boldsymbol{d}_i^{\mathcal{V}}$. Note that all quantities in (9.9)

Table 9.1 The states used in the dynamic model.

Notation	Description
$\boldsymbol{p} \in \mathbb{R}^2$	Position vector
$\boldsymbol{v} \in \mathbb{R}^2$	Velocity vector
$\boldsymbol{b_a} \in \mathbb{R}^2$	Accelerometer bias
$\boldsymbol{b_\xi} \in \mathbb{R}^2$	Gyroscope bias
$\phi \in \mathbb{R}$	Roll angle
$\psi \in \mathbb{R}$	Yaw angle
$\boldsymbol{\lambda} \in \mathbb{R}^4$	Wheel slip

should be interpreted as three dimensional for the cross product to be defined.

The first and third terms in (9.9) can be found using the accelerometer and gyroscope measurements and their associated bias states. The second term is, however, not easily available. One option is to differentiate the gyroscope signal to provide an estimate. This approach introduces additional noise and outliers, because it would mean differentiating an already noisy signal. Moreover, the introduced noise will make it harder to extract valuable information, if any. Hence, in the following we neglect this term. Using (9.9), the longitudinal acceleration at the wheel center contact points, expressed in the respective wheels' coordinate system $a_i^x,\ \forall i \in \{1,\dots,4\}$, is then written as

$$\begin{aligned}
a_1^x &= \cos(\delta_1)(a^{\mathcal{V},X} - w_1\dot{\psi}^2) - \sin(\delta_1)(v^{\mathcal{V},Y} + l_f\dot{\psi}^2) \\
a_2^x &= \cos(\delta_2)(a^{\mathcal{V},X} + w_2\dot{\psi}^2) - \sin(\delta_2)(v^{\mathcal{V},Y} + l_f\dot{\psi}^2) \\
a_3^x &= \cos(\delta_3)(a^{\mathcal{V},X} - w_3\dot{\psi}^2) - \sin(\delta_3)(v^{\mathcal{V},Y} - l_r\dot{\psi}^2) \\
a_4^x &= \cos(\delta_4)(a^{\mathcal{V},X} + w_4\dot{\psi}^2) - \sin(\delta_4)(v^{\mathcal{V},Y} - l_r\dot{\psi}^2),
\end{aligned} \tag{9.10}$$

where $\dot{\psi}$ is replaced with (9.5f) and the acceleration $a^{\mathcal{V},X}$ is replaced with (9.4a) together with a small-angle approximation. This yields

$$\dot{\psi} = u_{\boldsymbol{\xi}}^Z - b_{\boldsymbol{\xi}}^Z + w_{\boldsymbol{\xi}}^Z, \tag{9.11a}$$

$$a^{\mathcal{V},X} = u_{\boldsymbol{a}}^X - b_{\boldsymbol{a}}^X + w_{\boldsymbol{a}}^X. \tag{9.11b}$$

To summarize, the process model consists of (9.5) and (9.8), where (9.1) and (9.10) are used for computing (9.8). Table 9.1 summarizes the different states and the respective description.

Discrete-Time Dynamic Model

The dynamic model needs to be discretized to fit into the estimation framework. With the sampling period T_s, discretization of (9.5) and (9.8) yields

the discrete-time dynamic model

$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + T_s \boldsymbol{v}_k + \frac{T_s^2}{2} \boldsymbol{R}_{\mathcal{B}}^{I} (\boldsymbol{u}_{\boldsymbol{a},k} - \boldsymbol{b}_{\boldsymbol{a},k} - \boldsymbol{g}_k + \boldsymbol{w}_{\boldsymbol{a},k}), \tag{9.12a}$$

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + T_s \boldsymbol{R}_{\mathcal{B}}^{I} (\boldsymbol{u}_{\boldsymbol{a},k} - \boldsymbol{b}_{\boldsymbol{a},k} - \boldsymbol{g}_k + \boldsymbol{w}_{\boldsymbol{a},k}), \tag{9.12b}$$

$$\boldsymbol{b}_{\boldsymbol{a},k+1} = \boldsymbol{b}_{\boldsymbol{a},k} + T_s \boldsymbol{w}_{\boldsymbol{b}_{\boldsymbol{a}},k}, \tag{9.12c}$$

$$\boldsymbol{b}_{\boldsymbol{\xi},k+1} = \boldsymbol{b}_{\boldsymbol{\xi},k} + T_s \boldsymbol{w}_{\boldsymbol{b}_{\boldsymbol{\xi}},k}, \tag{9.12d}$$

$$\phi_{k+1} = \phi_k + T_s (u_{\boldsymbol{\xi},k}^{X} - b_{\boldsymbol{\xi},k}^{X} + w_{\boldsymbol{\xi},k}^{X}), \tag{9.12e}$$

$$\psi_{k+1} = \psi_k + T_s (u_{\boldsymbol{\xi},k}^{Z} - b_{\boldsymbol{\xi},k}^{Z} + w_{\boldsymbol{\xi},k}^{Z}), \tag{9.12f}$$

$$\lambda_{i,k+1} = \lambda_{i,k} + T_s \left(\frac{1 - \lambda_{i,k}}{v_{i,k}^{x}} a_{i,k}^{x} - \frac{R_i}{v_{i,k}^{x}} (\dot{\omega}_{i,k} + w_{\dot{\omega},i}) \right). \tag{9.12g}$$

In (9.12a) we have used an Euler discretization, with second order corrections for the acceleration components. In (9.12g), zero mean, white Gaussian noise terms $w_{\dot{\omega},i}$ for the wheel angular accelerations have been added. In total, the input state vector is

$$\boldsymbol{u}_k = \begin{bmatrix} \boldsymbol{u}_{\boldsymbol{a},k} \\ \boldsymbol{u}_{\boldsymbol{\xi},k} \\ \dot{\boldsymbol{\omega}}_k \end{bmatrix} \in \mathbb{R}^8.$$

The wheel angular accelerations, which are used as inputs, are not measured. These are estimated by a central difference approximation using the wheel angular velocities measured by the ABS wheel-speed sensors. Hence, the approximation $\hat{\dot{\boldsymbol{\omega}}}_k$ is calculated as

$$\hat{\dot{\boldsymbol{\omega}}}_k = \frac{\boldsymbol{\omega}(t_k) - \boldsymbol{\omega}(t_k - 2T_s)}{2T_s}.$$

Remark 9.1
The terms $a_{i,k}^{x}$ and $v_{i,k}^{x}$ in (9.12g) are computed with (4.15) and (9.10), which in its turn employ (9.11). Therefore the noise terms for the accelerations and the rotation rates also enter in (9.12g). □

Remark 9.2
The steer angles $\boldsymbol{\delta}_k$ are considered known. The motivation for this is that although it is possible to let the steer angles be used as inputs disturbed by noise, it will complicate the estimation algorithm while only giving minor, if any, tracking improvements. □

Measurement Model

The first two elements in the measurement vector consist of the longitudinal and lateral GPS position measurements. To incorporate velocity

information it is common to model the vehicle velocities in the body frame as measurements, either by using the wheel angular velocity measurements $\boldsymbol{\omega}_{m,k}$ together with the forward kinematics, see Chapter 6 and [Yi et al., 2009], or employment of GPS velocity measurements [Ward and Iagnemma, 2008; Iagnemma and Ward, 2009]. Both these approaches are valid in many scenarios, but to estimate the slip when it is incorporated in the state equations, we here use the wheel angular velocity measurements of each wheel independently as elements of the measurement vector. The rationale for this is that it reduces observability issues.

The relation between the GPS position measurement $\boldsymbol{p}_{m,k}$ and the position $\boldsymbol{p}_k$ is

$$\boldsymbol{p}_{m,k} = \boldsymbol{p}_k + \boldsymbol{e}_{\boldsymbol{p},k}, \tag{9.13}$$

where $\boldsymbol{e}_{\boldsymbol{p},k}$ is the GPS position measurement noise. To incorporate the wheel angular velocities, reuse of the slip definition (9.2) gives

$$\lambda_i = 1 - \frac{R_i \omega_i}{v_i^x} \iff v_i^x \lambda_i = v_i^x - R_i \omega_i \iff R_i \omega_i = v_i^x (1 - \lambda_i). \tag{9.14}$$

Thus, the wheel angular velocity measurements $\boldsymbol{\omega}_{m,k}$ are at each time instant related to the states as

$$R_i \omega_{m,i} = v_i^x (1 - \lambda_i) + e_{\omega,i}, \quad \forall i \in \{1, \dots, 4\}, \tag{9.15}$$

where $\{e_{\omega,i}\}_{i=1}^4$ are the measurement noise sources for the wheel-speed measurements. Note that (9.15) measures a combination of λ_i, $\boldsymbol{v}$, ψ, ϕ, and the bias terms through the relation (4.15). However, since the contribution from the roll angle ϕ is small, it is hard to estimate ϕ from (9.15) alone. Thus, to improve roll-angle estimation, start with noting that the acceleration of CoG is composed of a translational part and a rotational part; that is, it is possible to rewrite the Y-component of (9.4a) as

$$\begin{aligned} u_{\boldsymbol{a}}^Y &= a^{\mathcal{B},Y} + b_{\boldsymbol{a}}^Y + g\phi + w_{\boldsymbol{a}}^Y \\ &= \dot{v}^{\mathcal{B},Y} + \dot{\psi} v^{\mathcal{B},X} + b_{\boldsymbol{a}}^Y + g\phi + w_{\boldsymbol{a}}^Y \\ &\approx \dot{v}^{\mathcal{V},Y} + \dot{\psi} v^{\mathcal{V},X} + b_{\boldsymbol{a}}^Y + g\phi + w_{\boldsymbol{a}}^Y \\ &\approx \dot{\psi} v^{\mathcal{V},X} + b_{\boldsymbol{a}}^Y + g\phi + w_{\boldsymbol{a}}^Y, \end{aligned} \tag{9.16}$$

where we, besides $\sin(\phi) \approx \phi$ and $\cos(\phi) \approx 1$, have assumed that the acceleration owing to $\dot{v}^{\mathcal{V},Y}$ is small. Insertion of (9.5f) into (9.16) yields

$$u_{\boldsymbol{a}}^Y \approx \left(u_{\boldsymbol{\xi}}^Z - b_{\boldsymbol{\xi}}^Z + w_{\boldsymbol{\xi}}^Z \right) v^{\mathcal{V},X} + b_{\boldsymbol{a}}^Y + g\phi + w_{\boldsymbol{a}}^Y. \tag{9.17}$$

Splitting up (9.17) into the standard measurement equation form results in the measurement equation

$$u_{\boldsymbol{a}}^Y = \left(u_{\boldsymbol{\xi}}^Z - b_{\boldsymbol{\xi}}^Z \right) v^{\mathcal{V},X} + b_{\boldsymbol{a}}^Y + g\phi + \begin{bmatrix} v^{\mathcal{V},X} & 1 \end{bmatrix} \begin{bmatrix} w_{\boldsymbol{\xi}}^Z & w_{\boldsymbol{a}}^Y \end{bmatrix}^{\mathrm{T}}.$$

To summarize, the measurement vector is

$$\boldsymbol{y}_k = \begin{bmatrix} \boldsymbol{p}_{m,k} \\ R_1 \omega_{m,1,k} \\ \vdots \\ R_4 \omega_{m,4,k} \\ u^Y_{\boldsymbol{a},k} \end{bmatrix} \in \mathbb{R}^7$$

and is related to the states via

$$\boldsymbol{y}_k = \begin{bmatrix} \boldsymbol{p}_k \\ v^x_{1,k}(1-\lambda_{1,k}) \\ \vdots \\ v^x_{4,k}(1-\lambda_{4,k}) \\ \left(u^Z_{\boldsymbol{\xi},k} - b^Z_{\boldsymbol{\xi},k}\right) v^{\mathcal{V},X}_k + b^Y_{\boldsymbol{a},k} + g\phi_k \end{bmatrix} + \boldsymbol{e}_k, \tag{9.18}$$

where

$$\boldsymbol{e}_k = \begin{bmatrix} \boldsymbol{I}_{2\times 2} & \boldsymbol{0}_{2\times 4} & \boldsymbol{0}_2 & \boldsymbol{0}_2 \\ \boldsymbol{0}_{4\times 2} & \boldsymbol{I}_{4\times 4} & \boldsymbol{0}_4 & \boldsymbol{0}_4 \\ \boldsymbol{0}_2^{\mathrm{T}} & \boldsymbol{0}_4^{\mathrm{T}} & v^{\mathcal{B},X}_k & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_{\boldsymbol{p},k} \\ \boldsymbol{e}_{\boldsymbol{\omega},k} \\ w^Z_{\boldsymbol{\xi},k} \\ w^Y_{\boldsymbol{a},k} \end{bmatrix}. \tag{9.19}$$

9.5 Estimation Algorithm

To get the dynamics (9.12) and the measurement equations (9.18) on the same form as (3.20) (page 56), it is necessary to partition the states into a linear part and a nonlinear part. Naturally, it is computationally efficient to put as many states as possible in the linear part, because these can be estimated with a Kalman filter. The state space that the particle filter estimates is therefore smaller, hence demanding fewer particles [Lindsten et al., 2011]. Consequently, only those states that contribute the most to the nonlinearities are put into the nonlinear part. The remaining states are either linear or linearized, and then estimated by a Kalman filter.

Partitioning the States

The major nonlinearities are the velocities at the wheels, $v^x_{i,k}$, found in (9.12g). Since the wheel velocities depend on the yaw angle, longitudinal and translational vehicle velocites, the roll angle, and the bias terms, see (4.15) and (9.11a), all these states can be modeled as nonlinear. However,

with computational efficiency in mind, we only model the vehicle velocities and yaw angle as nonlinear. The reason is that these will dominate the contributions to $v^x_{i,k}$. The roll angle is multiplied with the gyro input and the bias term $b^Y_{\xi,k}$ in both (9.12f) and (9.18). This nonlinearity is mild and can be handled quite good with a linearization, especially since ϕ_k and $b^Y_{\xi,k}$ are small. Note that the yaw-rate measurement enters in $v^x_{i,k}$, that is, in the denominator in (9.12g), which is not supported in (3.20). Hence, this term has to be linearized in the measurement update step. The slip is multiplied with the wheel acceleration in (9.12g), where the wheel acceleration depends on the bias terms and the process inputs. The same type of nonlinearities occur in the second to fifth element in (9.18). These nonlinearities are rather mild compared with $1/v^x_{i,k}$, and introducing the four slip quantities as nonlinear states would drastically increase computational complexity. Thus $\{\lambda_{i,k}\}_{i=1}^4$ are considered to be linear.

In total there are three nonlinear states in $\boldsymbol{\eta}_k$ and 11 linear states in $\boldsymbol{z}_k$, which together form the total state vector $\boldsymbol{x}_k = \begin{bmatrix}\boldsymbol{z}_k^{\mathrm{T}} & \boldsymbol{\eta}_k^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{14}$, where

$$\boldsymbol{\eta}_k = \begin{bmatrix}\boldsymbol{v}_k^{\mathrm{T}} & \psi_k\end{bmatrix}^{\mathrm{T}}, \quad \boldsymbol{z}_k = \begin{bmatrix}\boldsymbol{p}_k^{\mathrm{T}} & \boldsymbol{b}_{\boldsymbol{a},k}^{\mathrm{T}} & \boldsymbol{b}_{\boldsymbol{\xi},k}^{\mathrm{T}} & \phi_k & \boldsymbol{\lambda}_k^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}. \tag{9.20}$$

Time Update

In the prediction steps of the RBPF, the dynamics (9.12) is used for propagating the estimates $\hat{\boldsymbol{x}}^i_k$ to time index $k+1$. To propagate the covariances in the Kalman filter, the states in $\boldsymbol{z}^i_k$ that actually are nonlinear need to be linearized. With the partitioning (9.20), the matrices needed for (3.20a) in the time update of the covariance matrices are

$$\begin{aligned}
\boldsymbol{A}_k &= \begin{bmatrix}
\boldsymbol{I}_{2\times 2} & \bar{\boldsymbol{P}}_{\boldsymbol{b}_a} & \boldsymbol{0}_{2\times 2} & \bar{\boldsymbol{P}}_{\phi} & \boldsymbol{0}_{2\times 4} \\
\boldsymbol{0}_{2\times 2} & \boldsymbol{I}_{2\times 2} & \boldsymbol{0}_{2\times 2} & 0 & \boldsymbol{0}_{2\times 4} \\
\boldsymbol{0}_{2\times 2} & \boldsymbol{0}_{2\times 2} & \boldsymbol{I}_{2\times 2} & 0 & \boldsymbol{0}_{2\times 4} \\
\boldsymbol{0}_2^{\mathrm{T}} & \boldsymbol{0}_2^{\mathrm{T}} & [-T_s\ 0] & 1 & \boldsymbol{0}_4^{\mathrm{T}} \\
\boldsymbol{0}_{2\times 2} & \boldsymbol{\Lambda}_{\boldsymbol{b}_a} & \boldsymbol{\Lambda}_{\boldsymbol{b}_\xi} & \boldsymbol{\Lambda}_{\phi} & \boldsymbol{\Lambda}_{\boldsymbol{\lambda}}
\end{bmatrix} \\
\boldsymbol{F}_k &= \begin{bmatrix}
\frac{T_s^2}{2}\boldsymbol{R}^I_{\mathcal{B}} & \boldsymbol{0}_{2\times 2} & \boldsymbol{0}_{2\times 2} & \boldsymbol{0}_{2\times 2} & \boldsymbol{0}_{2\times 4} \\
\boldsymbol{0}_{2\times 2} & T_s\boldsymbol{I}_{2\times 2} & \boldsymbol{0}_{2\times 2} & \boldsymbol{0}_{2\times 2} & \boldsymbol{0}_{2\times 4} \\
\boldsymbol{0}_{2\times 2} & \boldsymbol{0}_{2\times 2} & \boldsymbol{0}_{2\times 2} & T_s\boldsymbol{I}_{2\times 2} & \boldsymbol{0}_{2\times 4} \\
\boldsymbol{0}_2^{\mathrm{T}} & \boldsymbol{0}_2^{\mathrm{T}} & [T_s\ 0] & \boldsymbol{0}_2^{\mathrm{T}} & \boldsymbol{0}_4^{\mathrm{T}} \\
\boldsymbol{D}_1 & \boldsymbol{0}_{4\times 2} & \Lambda_{\boldsymbol{u}_\xi} & \boldsymbol{0}_{4\times 2} & \boldsymbol{D}_2
\end{bmatrix},
\end{aligned} \tag{9.21}$$

whereas the matrices that are to be inserted into (3.20b) for the covariance prediction are

$$\begin{aligned}\boldsymbol{B}_k &= \begin{bmatrix} \mathbf{0}_{2\times 2} & -T_s \boldsymbol{R}_{\mathcal{B}}^{I} & \mathbf{0}_{2\times 2} & 0 & \mathbf{0}_{2\times 4} \\ \mathbf{0}_2^{\mathrm{T}} & \mathbf{0}_2^{\mathrm{T}} & -[0 \; T_s] & 0 & \mathbf{0}_4^{\mathrm{T}} \end{bmatrix} \\ \boldsymbol{G}_k &= \begin{bmatrix} T_s \boldsymbol{R}_{\mathcal{B}}^{I} & \mathbf{0}_{2\times 2} \\ \mathbf{0}_2^{\mathrm{T}} & [0 \; T_s] \end{bmatrix}.\end{aligned} \tag{9.22}$$

In (9.21), $\bar{\boldsymbol{P}}_{\boldsymbol{b}_a} = \frac{\partial \boldsymbol{p}_{k+1}}{\partial \boldsymbol{b}_{a,k}}\Big|_{\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k}$ and similarly for Λ. Moreover,

$$\begin{aligned}\boldsymbol{D}_1 &= \begin{bmatrix} \frac{1-\lambda_{1,k}}{v_{1,k}^x}\cos(\delta_{1,k}) & \frac{1-\lambda_{1,k}}{v_{1,k}^x}\sin(\delta_{1,k}) \\ \vdots & \vdots \\ \frac{1-\lambda_{4,k}}{v_{4,k}^x}\cos(\delta_{4,k}) & \frac{1-\lambda_{4,k}}{v_{4,k}^x}\sin(\delta_{4,k}) \end{bmatrix}, \\ \boldsymbol{D}_2 &= -T_s \mathrm{diag}\left(\begin{bmatrix} \frac{R_1}{v_{1,k}^x} & \frac{R_2}{v_{2,k}^x} & \frac{R_3}{v_{3,k}^x} & \frac{R_4}{v_{4,k}^x} \end{bmatrix} \right).\end{aligned}$$

The noise terms corresponding to $\boldsymbol{w}_k^{\boldsymbol{z}}$ and $\boldsymbol{w}_k^{\boldsymbol{\eta}}$ in (3.20a) and (3.20b) are

$$\begin{aligned}\boldsymbol{w}_k^{\boldsymbol{z}} &= \begin{bmatrix} \boldsymbol{w}_{\boldsymbol{a},k} \\ \boldsymbol{w}_{\boldsymbol{b}_a,k} \\ \boldsymbol{w}_{\boldsymbol{\xi},k} \\ \boldsymbol{w}_{\boldsymbol{b}_\xi,k} \\ \boldsymbol{w}_{\dot{\boldsymbol{\omega}},k} \end{bmatrix} \in \mathbb{R}^{12} \\ \boldsymbol{w}_k^{\boldsymbol{\eta}} &= \begin{bmatrix} \boldsymbol{w}_{\boldsymbol{a},k} \\ \boldsymbol{w}_{\boldsymbol{\xi},k} \end{bmatrix} \in \mathbb{R}^4,\end{aligned} \tag{9.23}$$

where both $\boldsymbol{w}_k^{\boldsymbol{z}}$ and $\boldsymbol{w}_k^{\boldsymbol{\eta}}$ are white and Gaussian distributed with zero mean according to

$$\begin{aligned}\boldsymbol{w}_k^{\boldsymbol{z}} &\sim \mathcal{N}(\mathbf{0}_{12}, \boldsymbol{Q}^{\boldsymbol{z}}) \\ \boldsymbol{w}_k^{\boldsymbol{\eta}} &\sim \mathcal{N}(\mathbf{0}_4, \boldsymbol{Q}^{\boldsymbol{\eta}}).\end{aligned} \tag{9.24}$$

Both $\boldsymbol{Q}^{\boldsymbol{z}}$ and $\boldsymbol{Q}^{\boldsymbol{\eta}}$ in (9.24) are assumed to be constant and diagonal. Hence, they are modeled as

$$\begin{aligned}\boldsymbol{Q}^{\boldsymbol{z}} &= \mathrm{diag}\left(\begin{bmatrix} \boldsymbol{Q}_{\boldsymbol{a}} & \boldsymbol{Q}_{\boldsymbol{b}_a} & \boldsymbol{Q}_{\boldsymbol{\xi}} & \boldsymbol{Q}_{\boldsymbol{b}_\xi} & \boldsymbol{Q}_{\dot{\boldsymbol{\omega}}} \end{bmatrix} \right) \\ \boldsymbol{Q}^{\boldsymbol{\eta}} &= \mathrm{diag}\left(\begin{bmatrix} \boldsymbol{Q}_{\boldsymbol{a}} & \boldsymbol{Q}_{\boldsymbol{\xi}} \end{bmatrix} \right).\end{aligned} \tag{9.25}$$

The process noise terms $\boldsymbol{w}_k^{\boldsymbol{z}}$ and $\boldsymbol{w}_k^{\boldsymbol{\eta}}$ are correlated, as seen from (9.23). Thus, the cross term $\boldsymbol{Q}^{\boldsymbol{z\eta}}$ in (3.21) is nonzero and fulfills

$$(\boldsymbol{Q}^{\boldsymbol{z\eta}})^{\mathrm{T}} = \begin{bmatrix} \boldsymbol{Q}_{\boldsymbol{a}} & \mathbf{0}_{2\times 2} & \mathbf{0}_{2\times 2} & \mathbf{0}_{2\times 2} & \mathbf{0}_{4\times 4} \\ \mathbf{0}_{2\times 2} & \mathbf{0}_{2\times 2} & \boldsymbol{Q}_{\boldsymbol{\xi}} & \mathbf{0}_{2\times 2} & \mathbf{0}_{4\times 4} \end{bmatrix}. \tag{9.26}$$

The process inputs that correspond to the noise terms (9.23) are

$$\begin{aligned} \boldsymbol{u}_k^{\boldsymbol{z}} &= \begin{bmatrix} \boldsymbol{u}_{\boldsymbol{a},k} \\ \boldsymbol{b}_{\boldsymbol{a},k} \\ \boldsymbol{u}_{\boldsymbol{\xi},k} \\ \boldsymbol{b}_{\boldsymbol{\xi},k} \\ \dot{\boldsymbol{\omega}}_k \end{bmatrix} \\ \boldsymbol{u}_k^{\boldsymbol{\eta}} &= \begin{bmatrix} \boldsymbol{u}_{\boldsymbol{a},k} \\ \boldsymbol{u}_{\boldsymbol{\xi},k} \end{bmatrix}. \end{aligned} \tag{9.27}$$

The state prediction consists of propagating the dynamics (9.12) based on the estimates at the previous time step. The prediction of the covariance matrices for the linear states is done with the system matrices in (9.21) and (9.22) and the noise distributions (9.25)–(9.26).

Remark 9.3
The mixed Gaussian formulation in (3.20) assumes zero mean inputs, but the inputs, in this case (9.27), are straightforward to include. □

Measurement Update

In a similar manner to the prediction step, the measurement update step needs $\boldsymbol{C}_k$ in (3.20c), where some of the states have to be linearized to fit in the framework. With our choice of partitioning in (9.20), the measurement matrix becomes

$$\boldsymbol{C}_k = \begin{bmatrix} \boldsymbol{I}_{2\times 2} & \mathbf{0}_{2\times 2} & \mathbf{0}_{2\times 2} & 0 & \mathbf{0}_{2\times 4} \\ \mathbf{0}_2^{\mathrm{T}} & \mathbf{0}_2^{\mathrm{T}} & \boldsymbol{Y}_{1,\boldsymbol{b}_{\boldsymbol{\xi}}} & 0 & \boldsymbol{Y}_{1,\boldsymbol{\lambda}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0}_2^{\mathrm{T}} & \mathbf{0}_2^{\mathrm{T}} & \boldsymbol{Y}_{4,\boldsymbol{b}_{\boldsymbol{\xi}}} & 0 & \boldsymbol{Y}_{4,\boldsymbol{\lambda}} \\ \mathbf{0}_2^{\mathrm{T}} & 0 \quad 1 & 0 & -v_k^{\mathcal{V},X} & g & \mathbf{0}_4^{\mathrm{T}} \end{bmatrix}, \tag{9.28}$$

where

$$\boldsymbol{Y}_{1,\boldsymbol{b}_\xi} = \left.\frac{\partial v^x_{1,k}(1-\lambda_{1,k})}{\partial \boldsymbol{b}_{\boldsymbol{\xi},k}}\right|_{\hat{\boldsymbol{x}}_k,\boldsymbol{u}_k}$$
$$\boldsymbol{Y}_{4,\boldsymbol{b}_\xi} = \left.\frac{\partial v^x_{4,k}(1-\lambda_{4,k})}{\partial \boldsymbol{b}_{\boldsymbol{\xi},k}}\right|_{\hat{\boldsymbol{x}}_k,\boldsymbol{u}_k}$$

and similarly for the other partial derivatives.

Decorrelating the Noise

Note that the measurement noise (9.19) can be written as

$$\boldsymbol{e}_k = \boldsymbol{H}_k \bar{\boldsymbol{e}}_k \tag{9.29}$$

where

$$\bar{\boldsymbol{e}}_k = \begin{bmatrix} \boldsymbol{e}_{\boldsymbol{p},k} \\ \boldsymbol{e}_{\boldsymbol{\omega},k} \\ w^Z_{\boldsymbol{\xi},k} \\ w^Y_{\boldsymbol{a},k} \end{bmatrix}.$$

The noise sources for the wheel encoders and the Z- and Y-components of the gyroscope and accelerometer, respectively, enter both in (9.29) and (9.23). Hence, the process noise and measurement noise sources are correlated, which should be taken into account for improved performance. This can be done using a decorrelation procedure similar to the one that is used for decorrelating the noise (9.26) in the RBPF [Schön et al., 2005], and is not pursued further here.

9.6 Experimental Results

The presented method has been validated on five different test drives, ranging from aggressive maneuvering on race tracks to normal driving conditions in city traffic. The experimental results presented here are from a test drive that was performed at a race track in Linköping, Sweden.

To compare the performance with other methods, we have also implemented the approach to slip estimation in [Savaresi and Tanelli, 2010]. It is a rule-based velocity estimator that uses the angular velocities of the wheels and the longitudinal acceleration for estimating the velocity. Depending on if the vehicle is braking, accelerating, driving with approximately constant velocity, or if the velocity is very low, different sensor signals are used.

Experimental Setup

The testbed consists of a Volkswagen Golf V 2008, equipped with state-of-the-art sensors, see Figure 9.4 and [Lundahl et al., 2013b]. The sensors we use in the estimation algorithm are:

- The wheel angular velocities from the ABS wheel-speed sensors, sending measurements at 10 Hz.
- A GPS sensor, delivering position measurements at 4 Hz with an accuracy of approximately 2.5 m.
- An XSens IMU [Xsens Technologies B.V., 2010] that delivers measurements at 100 Hz. Note that only the planar accelerations and the yaw and roll rates are used in the estimation algorithm. The IMU is approximately located at the (unloaded) mass center, nearly aligned with the vehicle's coordinate system, during the experiments. No actions are taken to account for the location error or misalignment.

In addition, the vehicle is equipped with high-precision roll and pitch angle sensors (accuracy approximately 0.07 deg at 250 Hz), as well as an optical sensor for measuring the longitudinal velocity with high precision (accuracy 0.1% at 250 Hz). By using the wheel-speed measurements together with measurements of the longitudinal velocity, it is possible to extract information about longitudinal wheel slip with high accuracy.

The internal sensor measurements, such as the wheel-speed measurements, are accessible over the CAN bus [Johansson et al., 2005]. For the experiments in this chapter the data stream from the CAN bus was sent to a measurement computer, which synchronized the data with the external sensors and then converted the data to a MATLAB-readable format. For more information on the experimental setup, consult [Lundahl et al., 2013b].

REMARK 9.4
The effective tire radii $\{R_i\}_{i=1}^4$, which are important in both velocity and slip estimation, were determined by logging data during steady-state, straight-line driving and comparing the measured longitudinal velocity v^X with the wheel angular velocity ω_i. The effective wheel radius for each wheel was then chosen as $R_i = v^X/\omega_i$. Note that the datasets were not acquired on the same day, and thus the variation in tire pressure will add uncertainty to the wheel-radius estimate. □

Implementation Aspects

The RBPF in Algorithm 3.3 is used in the experiments. The algorithm is implemented in MATLAB and converted to `mex`-files using the Coder toolbox. No measures to code optimization have been taken.

Figure 9.4 The vehicle testbed used in the experiments. The velocity sensor is placed at the front end, aligned with the longitudinal axis. The roll and pitch angle sensors are situated at the front end and in front of the rear wheels. The vehicle was also equipped with a GPS receiver during the experiments.

The variances of the IMU measurements are chosen to be approximately the same as in the product specifications. The standard deviations of the GPS measurements are set to be in the order of the accuracy, and the noise parameters of the wheel speeds and the numeric differentiation of the wheel speeds are decided by a straightforward procedure for standard-deviation estimation [Johansson, 2009]. Moreover, to take into account that the noise sources are not truly Gaussian noise distributions, a roughening procedure is implemented [Gordon et al., 1993], which works as follows: For all particles $\{\boldsymbol{\eta}_{k+1}\}_{i=1}^{N}$, which are calculated from the dynamic equations in the time-update step, add an independent jitter c_i. The jitter is a sample from a Gaussian distribution with standard deviation σ. The standard deviation of the Gaussian jitter is computed as

$$\sigma = EN^{-1/n_\eta},$$

where E is the length of the interval between the maximum and minimum values of the particles. All test scenarios used the same noise distributions. Only a minor effort was put into tuning the filter parameters. It is therefore highly likely that better performance can be achieved by making more sensible choices. Further, choosing a better prior for updating the particle weights can also improve performance.

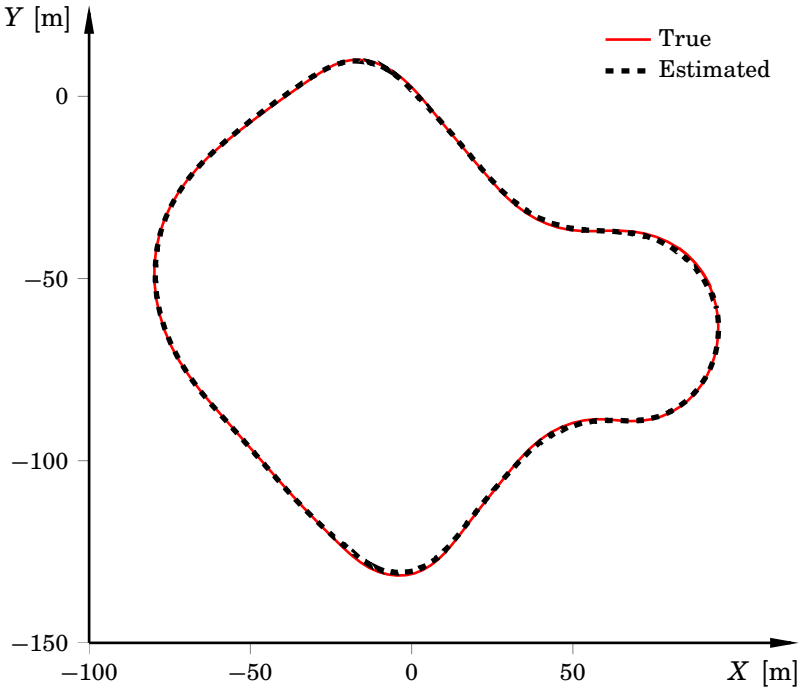

Figure 9.5 The estimated positions and GPS positions for one lap on the race track, using 1000 particles. The differences between GPS position measurements and the estimated positions are at most approximately 2 m, which is in the order of the measurement uncertainty.

Race-Track Scenario

This scenario was constructed by driving approximately seven laps on a small part of a race track. The dataset contains both straight-line driving and cornering, and both drive and brake torques are heavily utilized throughout. Figure 9.5 shows the GPS positions for a portion of the scenario together with the position estimates, corresponding to approximately one lap. The accelerations, as measured by the accelerometer, and the longitudinal velocity, measured by the optical sensor, are shown in Figure 9.6 for the whole test drive. It is clear that a major part of the available friction is used during cornering and that the braking behavior is very aggressive.

Figure 9.7 shows results for a 25 s excerpt of the maneuver, which corresponds to approximately one lap. The upper plot compares the estimated longitudinal velocity with the velocity measured by the optical sensor, which is used as ground truth. The roll angle estimate and the

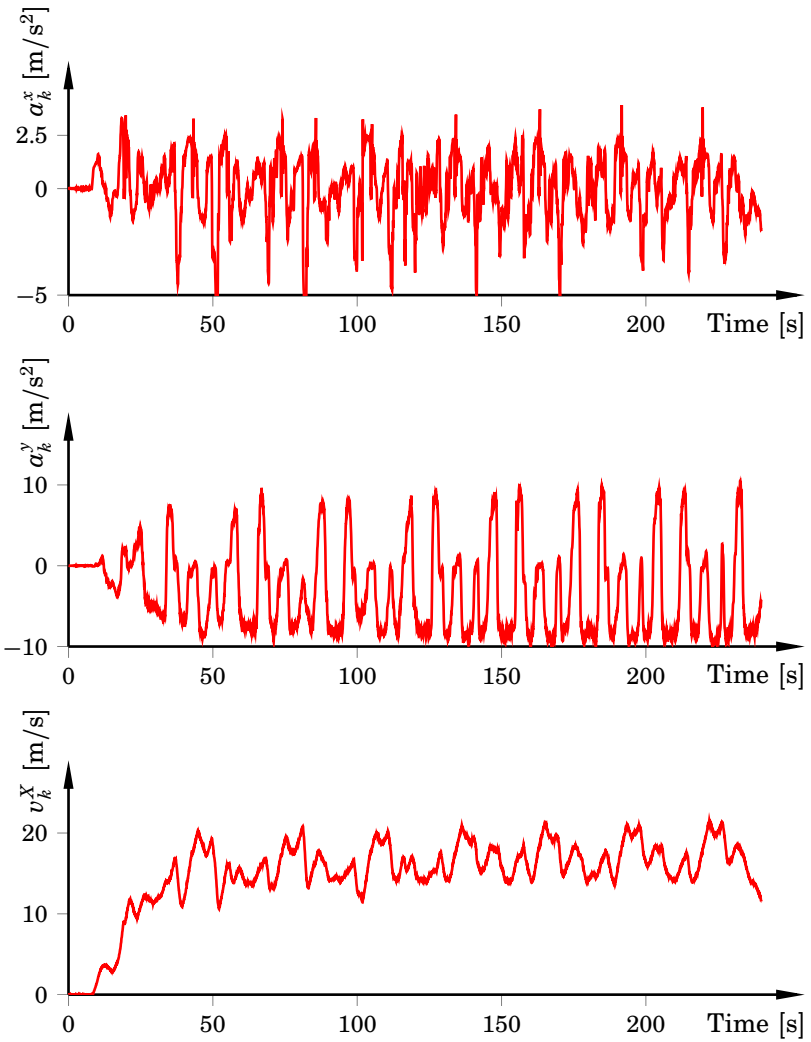

Figure 9.6 The accelerations as measured by the accelerometer, together with the longitudinal velocity as measured by the optical sensors. Note that the velocity measurement was not used in the estimation algorithm, but only as a ground-truth certificate. The driving behavior is aggressive and utilizes the available friction. The lateral acceleration peaks at approximately 10 m/s^2. The maneuver is performed on dry asphalt, where the friction coefficient μ typically is in the range $0.95 \leq \mu \leq 1.2$ [Svendenius, 2007].

corresponding ground truth are in the middle plot, while the lower plot shows the estimated wheel slip and true wheel slip (as measured by the external, optical sensor) of the second wheel. The maximum velocity error in this figure is approximately 0.6 m/s, and occurs at 83 s. The roll-angle estimation is consistent. The largest error is approximately 2 deg (occurring at 91 s). The roll-angle estimates are, however, more accurate than 2 deg for most parts of the maneuver. The slip behavior is also captured very well, with only minor discrepancies. The results for the other wheels are very similar in performance.

Figure 9.8 visualizes the true wheel slip and estimated wheel slip for the second and fourth wheel, respectively, together with corresponding results for the rule-based slip estimation algorithm in [Savaresi and Tanelli, 2010][2] for another 25 s excerpt. As shown in [Savaresi and Tanelli, 2010], and as we have also confirmed using other datasets, that algorithm performs well for straight-line driving scenarios. For this dataset, however, it is clear that the slip estimation with our method is superior, and also very rapid changes in the slip are handled well. The slip characteristics between the second and fourth wheels are different. This is explained by that the vehicle is front-wheel driven in combination with significant load transfer in both roll and pitch direction, owing to large acceleration/deceleration as well as aggressive cornering. The slip values exceed 0.1, which approximately corresponds to the slip value for where the force peak is attained [Solyom, 2004; Svendenius, 2007].

[2] The parameters in that algorithm were chosen as in [Savaresi and Tanelli, 2010].

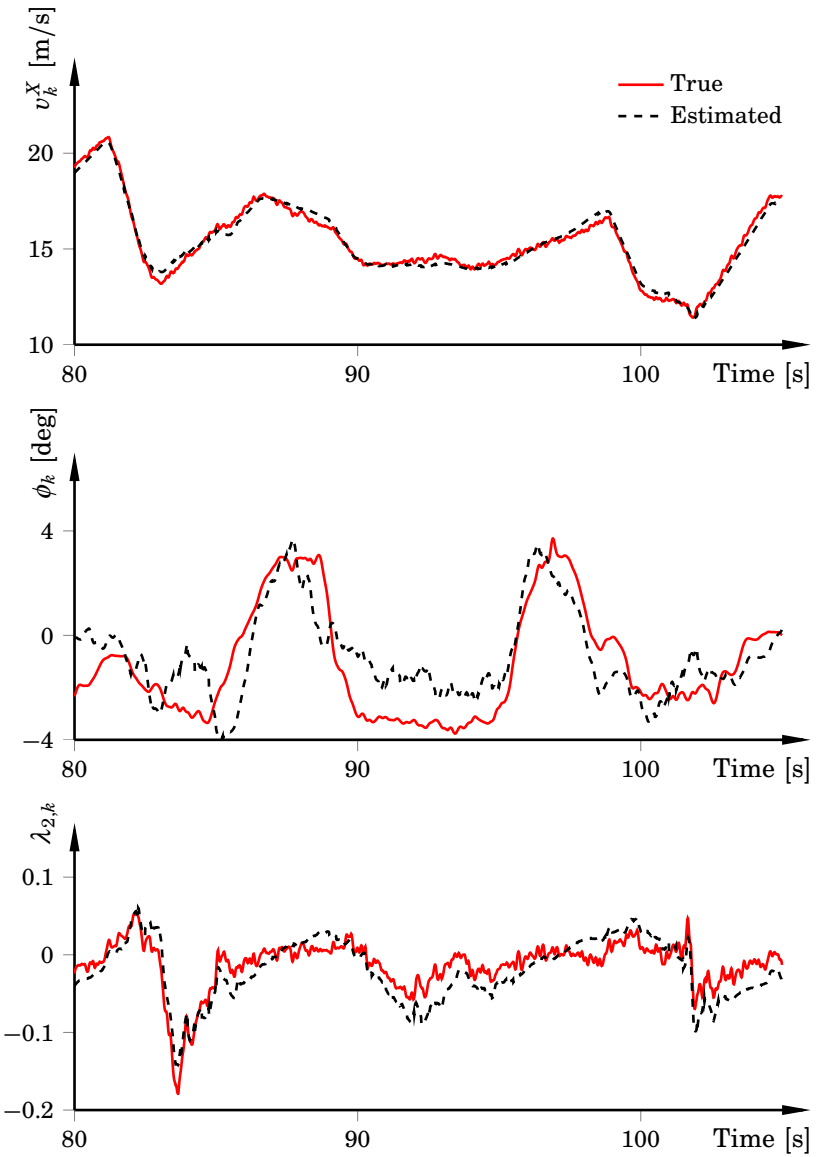

Figure 9.7 Estimated (black) and ground-truth (red) velocity, roll angle, and wheel slip for a 25 s excerpt from the whole dataset, using 1000 particles. The largest estimation error for the roll angle is approximately 2 deg (occurring at 91 s), but the estimation error is typically within one deg. The peak in the force curve occurs when the slip (λ) is about 0.1.

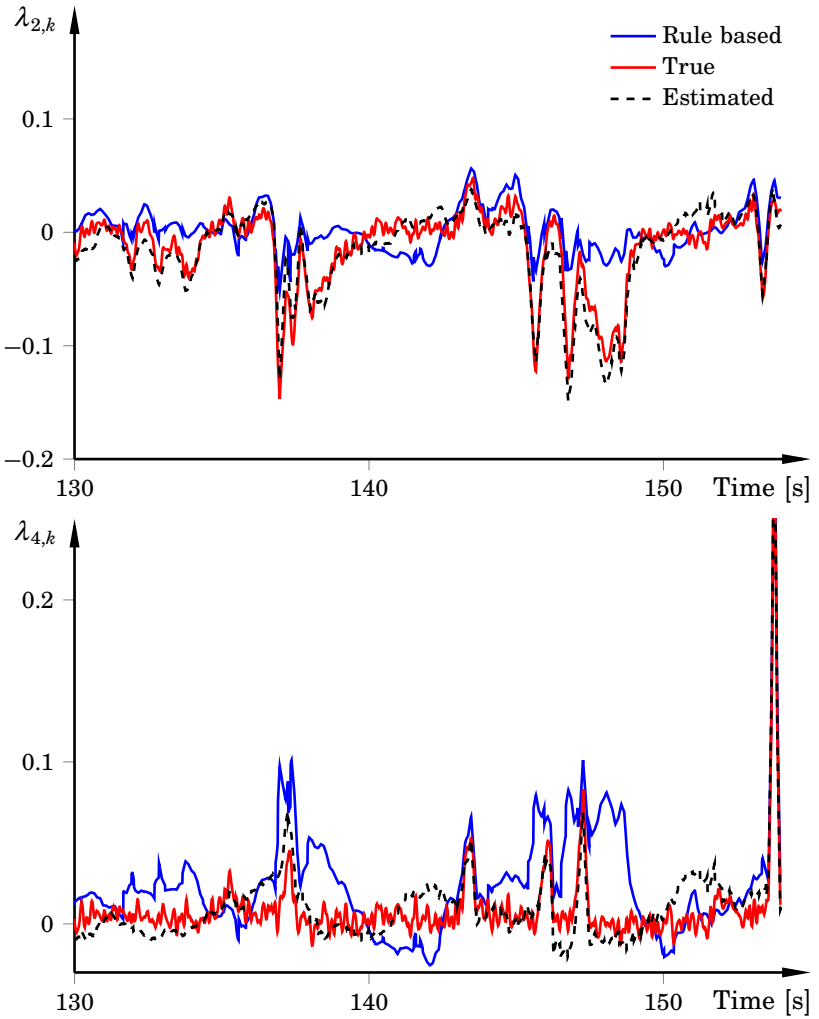

Figure 9.8 Estimated (black dashed) and ground truth (red) wheel slip for a 25 s excerpt from the whole dataset, using 1000 particles. For comparison, the results for the slip-estimation approach in [Savaresi and Tanelli, 2010] are also shown (blue). The considered dataset contains combined longitudinal and lateral movement, in addition to significant roll. This puts large demands on modeling coupling effects, something which the proposed method does remarkably well. The slip peak at $t \approx 154$ s reaches about 0.3, something which both schemes handle well.

Robustness in Tracking

To give a measure of how the number of particles affects the tracking performance, Figure 9.9 shows the likelihood of the time-averaged velocity error as a cumulative distribution function over the 100 executions, for 100, 200, 400, and 1000 particles. As an example, 93% of the executions yield less than 0.37 m/s velocity error when using 400 particles. The corresponding average error for λ_1 is 0.025 (and is in the same range for the other wheels). Moreover, the probability of achieving an average velocity error smaller than 1 m/s when using 400 particles is 94%, with the corresponding number for 1000 particles being 99%. Note that only one two of the 100 executions rendered an error of more than 0.7 m/s (corresponding to an error of 0.05 for λ_1) for 1000 particles. For 1000 particles, over 90% of the executions yield less than 0.27 m/s in velocity error and a slip error smaller than 0.02. When using very few particles, in round numbers less than 200, the performance deteriorates quickly.

9.7 Concluding Discussion

The proposed method performed well in all of the considered datasets, five in total. Some of the shorter datasets that were gathered (approximately 30 s long) did not contain GPS measurements. The method estimates velocities and slip quantities with high precision also in these experiments. Hence, the algorithm works also in the cases when occasional packet losses in the GPS communication occur. In some setups the GPS velocities are also available. This has not been utilized here, but it is likely that incorporating this information improves performance.

When it comes to statistical properties, the algorithm delivers excellent tracking accuracy in all but very few cases (see Figure 9.9). It is highly likely that better tuning of the filter, in combination with using more particles, can improve performance further. In addition, the IMU and the GPS receiver were assumed to be located at the mass center. This is an erroneous assumption for this particular experimental setup. There are therefore improvement possibilities when it comes to calibration. Note that the results presented in Figures 9.7–9.8 are representative of the performance that can be expected from a typical realization of the filter.

The dataset used in this chapter corresponds to very aggressive maneuvering. As such, it really tests the method's abilities to handle large slip in combination with significant roll. The largest longitudinal slip values that were measured were approximately $\lambda = 0.3$, which is beyond the value where the force peak occurs. The lateral acceleration peaks at approximately 10 m/s^2. It is interesting that the slip estimation is of high quality also in periods of aggressive cornering, because this is a scenario

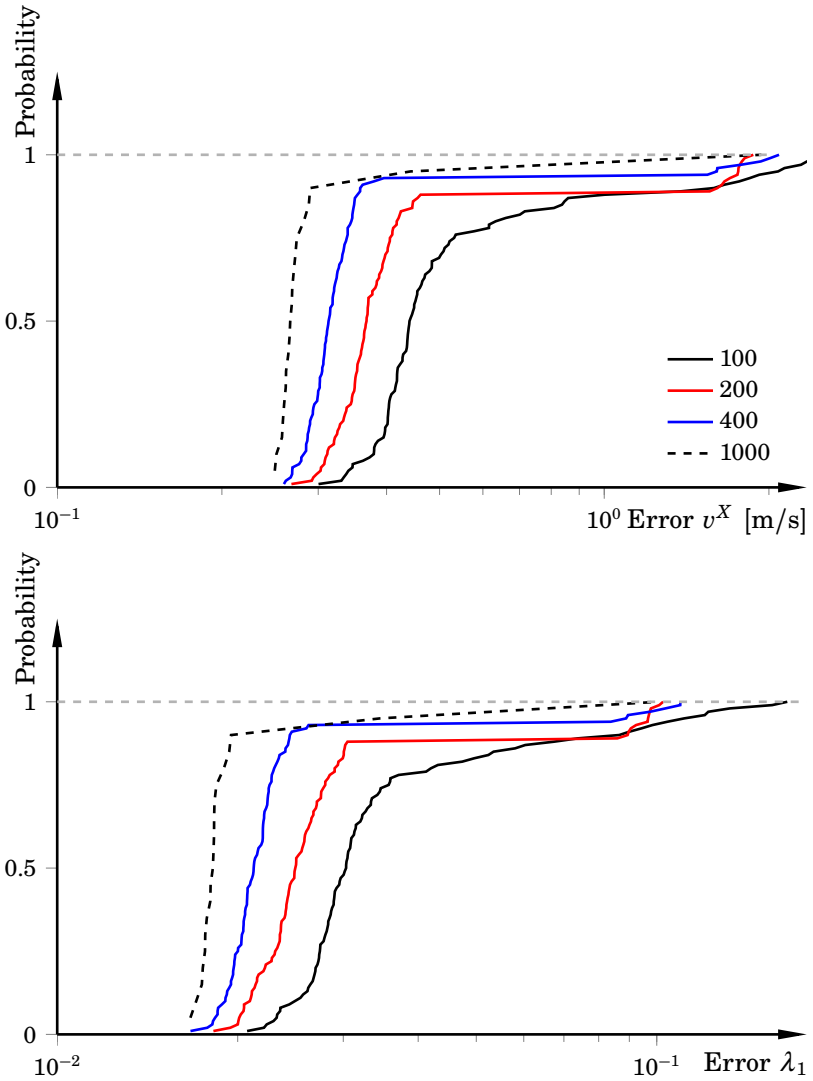

Figure 9.9 The probability of time-averaged velocity error for different number of particles, computed as a cumulative distribution function over 100 executions of the whole dataset. Out of the 100 executions, 98 give a velocity error smaller than 0.7 m/s when using 1000 particles. The results for the other wheels are similar. The corresponding number for 400 particles is 93%. Moreover, over 90% of the executions yield less than 0.27 m/s in velocity error for 1000 particles. Using less than 400 particles makes the algorithm quite sensitive.

that often causes deficient performance for many slip-estimation algorithms. Moreover, also the roll angle is precisely estimated in most parts of the experiments.

The average execution time for one iteration of the algorithm when using 400 particles is about 5 ms. Thus, even in a general-purpose MATLAB implementation, where no measures to code optimization have been taken, the algorithm almost executes in real time. The algorithm scales linearly with the number of particles [Karlsson et al., 2005], and for 1000 particles the average execution time is approximately 20 ms in the current implementation. Hence, with a dedicated implementation, the algorithm can execute in real time for a large amount of particles.

9.8 Summary

This chapter presented a novel method for combined wheel-slip and motion estimation of four-wheeled vehicles. One of the main novelties was a Bayesian approach to solve the combined wheel-slip and vehicle-motion estimation problem. The approach explicitly models the highly nonlinear slip dynamics within a mixed-Gaussian formulation. This makes the model suitable for Rao-Blackwellized particle filters. Another novelty was that traditional key variables in safety systems, such as roll angle, vehicle velocity, and wheel slip are estimated simultaneously. In reality, all these variables are correlated. Still, usually estimation algorithms found in literature concentrate on some of the variables. This is not the case in the proposed method. The drawbacks are that the model is highly nonlinear and that the computational complexity is higher than for linear approaches.

We evaluated the resulting algorithm on five datasets in total, and this chapter investigated results from one of them. The results showed that the estimation algorithm provides excellent tracking, even for aggressive maneuvering with large resulting slip values. A comparison with a slip estimation algorithm from the literature indicates clear improvements in terms of slip estimation, especially during combined cornering and acceleration/deceleration.

10

Dynamic Optimization for Automotive Systems

The previous chapter presented results on joint wheel-slip and motion estimation for vehicles. When the method is applied to automotive systems, the estimated states can be used in different active safety systems. Systems for active safety in automotive systems have been a hot topic for the last decades. Two famous examples are anti-lock braking systems (ABS) and electronic stability control systems (ESCs). There exist many active safety systems in current production cars—for example, anti-slip regulation (ASR) systems; active suspension systems (such as active body control) to eliminate rollover accidents; and active steering systems that serve as an additional actuator, see [Kiencke and Nielsen, 2005; Rajamani, 2006; Isermann, 2006; Schindler, 2007; Reif and Dietsche, 2011]. However, as pointed out in [Funke et al., 2012] and mentioned in Section 1.2, current state-of-the-art safety systems are still inferior to the maneuvering performance achievable by expert drivers in critical situations, who often manage to maintain the desired vehicle trajectory while being at the tire friction limits, see Figure 10.1 for an illustration. Moreover, despite improved safety, there are still more than 30 000 fatal vehicle traffic crashes per year in the USA alone, with an estimated cost of more than 230 Billion U.S. Dollars [NHTSA, 2011]. Hence, there is clearly a need for improved safety systems.

This chapter presents research on what impact different vehicle models and road surfaces have on the optimal trajectories in safety-critical maneuvers. Optimal control of automotive systems is in literature often done using a particular vehicle model on a specific surface. Here, we compare the optimal control inputs and corresponding trajectories for different tire and chassis models, four in total, in addition to investigating what impact the road surface has on the solutions. The tire-road interaction is modeled based on experimental data, and all relevant tire parameters

Figure 10.1 An example of an expert driver, who manages to follow a desired vehicle trajectory while pushing the tire-surface interaction to its limits. Source: Wikimedia Commons.

differ between the surfaces. This is in contrast to the standard approach, where typically only few parameters, such as the friction coefficient, are changed.

The presented results extend those in the publications [Berntorp et al., 2013; Olofsson et al., 2013; Lundahl et al., 2013a; Berntorp et al., 2014]. To gain insight in how commonly used vehicle models behave when utilized in at-the-limit maneuvers, we employ a minimum-time optimization criterion. The chapter presents optimal solutions for surfaces corresponding to dry asphalt, snow, and smooth ice in two different maneuvers; a 90 deg turn and a hairpin turn, using four different combinations of tire and chassis models. Of particular interest is to analyze the results from a safety-system perspective; that is, what driving behavior and model characteristics can be extracted from the results, and what differences are there between the different surfaces. A large part of the chapter is therefore devoted to investigating how the solutions differ, both with respect to the models and the surfaces.

10.1 Motivation

To improve active safety systems even further, they need to be designed to explicitly prevent lane departure and collisions with other vehicles. In-

volving optimal control in design of safety systems is appealing because it implies that a model-based, systematic approach to control design is used. A survey from 2011 on optimal control in automotive applications [Sharp and Peng, 2011] points out that finding the right balance between models, correct formulations, and optimization methods is the fundamental problem to be solved.

Optimal control often results in trajectories that utilize the achievable limits of the input and state regions. It is therefore crucial how, for example, the tires are modeled outside their normal range of operation. In addition, chassis dynamics such as roll and pitch are necessary to give a correct representation of load transfer and vehicle stability. The situation is further complicated by that the interaction between tire and road is complex, and that different tires have different characteristics. Even when only considering the longitudinal stiffness (i.e., the initial slope of the longitudinal force-slip curve in (4.7a)) the measurements differ up to 100% between experiments, see [Carlson and Gerdes, 2005]. In addition to the differences in stiffness there are also deviations between the characteristic shape of the curve at the maximum force, where the peak can be more or less accentuated (see Figure 4.3). This is particularly noticeable when comparing different surfaces, see for example [Braghin et al., 2006; Svendenius, 2007; Reimpell and Betzler, 2005].

A less complex model obviously provides for an easier system analysis, control design, and implementation, and is thus the preferred one if it *sufficiently captures* relevant dynamics. Simple models have been used in simulation and for design of control systems, with good results. Consequently, it is natural to use simple models also in optimal control. However, since optimal control problems typically result in control inputs that give aggressive maneuvering, proper model choice is pivotal. Erroneous modeling can be utilized by the optimization solver, which can result in inappropriate, and sometimes even unrealistic, behavior.

10.2 Related Work

Optimization for vehicles in time-critical situations has been studied in numerous publications previously. Different examples concerning collision mitigation and cornering are found in [Chakraborty et al., 2011; Chakraborty et al., 2013; Velenis and Tsiotras, 2005; Velenis, 2011]. Control laws for vehicle emergency-maneuvers were developed in [Dingle and Guzzella, 2010] based on an analytic optimal-control approach, using a one-track model (see Section 4.2, page 73). That approach neglects roll and pitch dynamics. [Yang et al., 2013] used optimal path control, to be activated after light collisions.

[Shiller and Sundar, 1998] discussed optimal lane-change maneuvers. Given the initial speed and additional assumptions, the minimum distance at which an approaching obstacle can be avoided was determined. The time-optimal race-car trajectories were investigated in [Kelly and Sharp, 2010; Casanova, 2000]. [Sharp and Peng, 2011] contains a survey on existing vehicle dynamics applications of optimal-control theory. The influence of the road surface and the car transmission layout in minimum-time cornering was investigated in [Tavernini et al., 2013] using a one-track chassis model. Methods for constraint-based trajectory planning for optimal maneuvers were given in [Anderson et al., 2010; Anderson et al., 2012]. In [Yi et al., 2012], the stability and agility of aggressive pendulum turn maneuvers were investigated using a one-track chassis model and a LuGre-based friction model [Wit et al., 1995]. Further, [Andreasson, 2009; Sundström et al., 2010] discussed optimal control of over-actuated vehicles using a two-track model and a friction-ellipse based tire model.

A method for optimal force allocation in yaw stabilization of automotive vehicles was proposed in [Tøndel and Johansen, 2005]. [Tjønnås and Johansen, 2010] contains an expansion of the previous paper, consisting of a two-level strategy for active steering and adaptive control allocation. A convex approach to force allocation was outlined in [Schofield, 2008]. Further, [Esmailzadeh et al., 2003] treated an optimal yaw control law for road vehicles.

Scaling of nominal tire models for different surfaces was discussed and experimentally verified in [Braghin et al., 2006]. Scaling factors were identified for different tires, together with uncertainty estimates. The investigation of the road-surface impact on the optimal trajectories in this chapter is based on the results in [Braghin et al., 2006].

10.3 Preliminaries

Throughout, we consider DAEs of the form (2.3)—that is,

$$\boldsymbol{f}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{w}(t), \boldsymbol{u}(t)) = 0, \tag{10.1}$$

where $\boldsymbol{f}$ is a vector-valued function and $\boldsymbol{x}(t)$, $\boldsymbol{w}(t)$, and $\boldsymbol{u}(t)$ contain the differential (state) variables, algebraic variables, and control variables, respectively. The time dependency of the variables is implicit in what follows. We assume that (10.1) is twice continuously differentiable. This is a standard assumption in the dynamic optimization literature, and means that we neglect hybrid behavior, for example stemming from gearboxes. This is not a restriction, since this chapter is concerned with comparing rather common chassis and tire models and discuss their qualitative behavior. The tires are assumed to stay in contact with the flat ground at all

times. The comparison is meant to give insight in model choice for active safety systems, where wheel lift obviously is not wanted.

10.4 Modeling

This section summarizes the tire and chassis models that are used in this chapter, where many already have been mentioned in Chapter 4. It also discusses how to calibrate the tire models to be able to compare them.

Wheel and Tire Modeling

The wheel dynamics are given by (4.1). The longitudinal slip λ is computed with (4.2). The lateral slip angle α used here is given by

$$\dot{\alpha}\frac{\sigma}{v^x} + \alpha = -\arctan\left(\frac{v^y}{v^x}\right), \tag{10.2}$$

where σ is the relaxation length. The difference compared with the geometric slip angle computed in (4.4) is that (10.2) uses a first-order filter, to account for that tire forces do not develop instantaneously. This approach was validated in [Clark et al., 1972], and is also discussed in [Pacejka, 2006].

Tire Forces The nominal longitudinal and lateral tire forces F_0^x and F_0^y (i.e., the forces under pure slip conditions) are computed with the Magic Formula model (4.8), repeated for convenience:

$$\begin{aligned} F_0^x &= \mu_x F^z \sin\Big(C_x \arctan\left(B_x\lambda - E_x(B_x\lambda - \arctan B_x\lambda)\right)\Big) \\ F_{y0} &= \mu_y F^z \sin\Big(C_y \arctan\left(B_y\alpha - E_y(B_y\alpha - \arctan B_y\alpha)\right)\Big), \end{aligned} \tag{10.3}$$

where μ_x and μ_y are the longitudinal and lateral friction coefficients, B is the stiffness factor, C is the shape factor, and E is the curvature factor.

As pointed out in Chapter 4 on page 70, under combined slip conditions—that is, when both λ and α are nonzero—the longitudinal and lateral tire forces depend on both slip quantities. How this coupling is described can have large impact on the vehicle dynamics. In an optimal maneuver, the computed control inputs will result in the best combination of longitudinal and lateral forces, and these forces are, of course, coupled via the tire physics. There are no experiments that measure the complete longitudinal-lateral tire interaction, but a variety of characteristics have been established, see [Reimpell and Betzler, 2005; Kiencke and Nielsen, 2005; Pacejka, 2006; Rajamani, 2006; Svendenius, 2007].

This study employs two models: the friction-ellipse approximation (4.9) and the weighting-functions model (4.11). The weighting-functions model

is referred to as WF. For the friction-ellipse based model it is customary to use the nominal force F_0^x directly as an input to the vehicle model, see [Wong, 2008]. Here, however, we use the drive/brake torque as input, see (4.1), since these are quantities that can be controlled in a physical setup of a vehicle. The main limitation with the friction-ellipse based model is that the longitudinal force does not explicitly depend on the lateral slip, which is not realistic. With longitudinal slip present, it is possible to use (4.10), which is a more realistic model. However, we use (4.9) since it represents the simplest combined-force model that is used in the literature, see [Andreasson, 2009; Sundström et al., 2010] for two examples. This model is hereafter denoted by FE.

Chassis Models

We use two chassis models of different complexity. The most complex model is a two-track model with roll and pitch dynamics and both longitudinal and lateral load transfer. This chassis model is illustrated in Figure 4.7 and was derived in Section 4.2. Throughout the chapter, the indices f, r and $1, 2, 3, 4$ denote the respective wheel pair and wheel, respectively. The dynamic equations for the longitudinal load transfer are given by

$$(F_1^z + F_2^z)l_f - (F_3^z + F_4^z)l_r = K_\theta\theta + D_\theta\dot{\theta}, \quad \sum_{i=1}^{4} F_i^z = mg, \tag{10.4}$$

where F_i^z, $i \in \{1, 2, 3, 4\}$, denote the time-dependent normal forces; l_f, l_r are defined in Figure 4.7; and g is the gravitational constant. The lateral load transfer is determined by the relations

$$-w(F_1^z - F_2^z) = K_{\phi,f}\phi + D_{\phi,f}\dot{\phi},$$
$$-w(F_3^z - F_4^z) = K_{\phi,r}\phi + D_{\phi,r}\dot{\phi},$$

where the half track width $w = w_f = w_r$ is defined in Figure 4.7. This load-transfer model will fit the true load transfer well for modest combined roll and pitch angles.

The second model is the one-track model (see Figure 4.6), with the dynamics given by (4.12). This model lumps together the left and right wheel on each axle and neglects roll and pitch dynamics. The model has three degrees of freedom, two translational and one rotational (the yaw angle ψ). Table A.1 on page 324 provides the vehicle parameters used for the results.

Tire-Force Characteristics and Model Calibration

It is not obvious how to model different surfaces, and the modeling choice can have a large impact on the obtained results. This is especially true

in aggressive maneuvering when the tires perform close to their friction limits. Given a set of tire parameters for WF (4.11) and the nominal forces (10.3) on a nominal surface, [Pacejka, 2006] proposes to use scaling factors s_j to describe different road conditions. This method was used in [Braghin et al., 2006], where the scaling factors representing surfaces corresponding to dry asphalt, wet asphalt, snow, and smooth ice were estimated based on experimental data. That study included a set of different tire brands and models; hence, the results presented in [Braghin et al., 2006] can be seen as guidelines on how the tire characteristics will vary.

The tire calibration in this chapter uses the scaling factors from [Braghin et al., 2006] as a basis for calibrating tire models approximately corresponding to the force characteristics on the different surfaces. The nominal tire parameters used in that paper are not public domain. Thus, we use the parameters from [Pacejka, 2006], which have been estimated from experimental data, to represent dry asphalt. We introduce the relative scaling factors with respect to dry asphalt as

$$s_{\text{dry}} = 1, \quad s_{\text{snow}} = \frac{s^*_{\text{snow}}}{s^*_{\text{dry}}}, \quad s_{\text{ice}} = \frac{s^*_{\text{ice}}}{s^*_{\text{dry}}}, \tag{10.5}$$

where s_j are the scaling factors used here and s^*_j are the scaling factors presented in [Braghin et al., 2006]. There are uncertainties in the original scaling factors because of the experimental data, which causes inconsistencies for the snow and ice models. This is especially true for larger slip values, where measurements typically are highly uncertain, if possible to gather at all; for example, it is for technical reasons hard to obtain valid experimental data for slip larger than where the force peak is attained. The original snow model produces a longitudinal force F^x that changes sign for large slip, which is avoided by adjusting the scaling factor for C_α in (4.11b). Tables A.2 and A.3 on page 325 provide the complete set of tire parameters. A subset of these parameters are dependent on the normal force acting on the wheel. The front and rear parameter values differ in some respects. We therefore determine the parameters from the normal forces present when the vehicle is at rest. This is, of course, an approximation, but modeling the dependence on the time-varying normal force is nontrivial. Further, it complicates convergence in the optimization.

The parameters we use correspond to dry asphalt, packed snow, and smooth ice: in the following, asphalt, snow, and ice denote dry asphalt, packed snow, and smooth ice, respectively.

Calibrating Tire Models for Comparison

There are several approaches to calibrate two different tire models to get comparable solutions to the optimal control problem. As an example,

Figure 10.2 shows the resultant of the tire forces, $F_{\text{res}} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, defined as

$$F_{\text{res}} := \sqrt{\left(F^x(\lambda, \alpha)\right)^2 + \left(F^y(\alpha, \lambda)\right)^2},$$

for two tire models. The resultant is expressed in terms of the normal force to easier enable comparison of models with and without dynamic load transfer. The first plot in Figure 10.2 is parametrized using WF and the second using FE. To equalize these models in comparative studies, one way would be to have the same average resultant force, whereas another approach would be to equalize the longitudinal stiffness. We have calibrated the particular tire models considered here to agree for pure slip conditions; that is, the parameters in (10.3) are the same for both FE and WF.

Studying Figure 10.2 gives a basis for discussion of the behavior of the tire models in an optimal maneuver and complements the discussion in Chapter 4; for example, the models give different force characteristics for combined slip, where one major difference is that FE predicts a larger force for large combined slip than WF does. Further, the characteristic peaks in F_{res} obviously influence the behavior of the tire force model significantly. As mentioned in Section 4.1, WF has been experimentally verified. Figure 4.3 showed that one difference between FE and WF is that it is always possible to fully utilize the available friction with FE, something that is not possible in reality.

Section 10.1 mentioned that parametrizations based on experimental data have uncertainties, and the differences between experiments can be large. Hence, the tire-force characteristics in Figure 10.2 typically differ, even for the same surface, depending on which particular tire that is used.

10.5 Dynamic Optimization Problem

The optimal trajectories are determined for the 90 deg turn and the hairpin turn as the solution to an optimization problem of the form (2.4). Considering the physical setup of the problem, it is clear that a solution exists for reasonable initial values. However, the resulting optimization problem is numerically challenging since the time optimality implies that the tire-friction models partly operate at large slip values. Since the force curves/surfaces, see Figures 4.3 and 10.2, are rather flat in these regions, the numerical sensitivity is large. Also, it is more demanding to solve dynamic optimization problems where the time horizon is free than when it is fixed. Further, the optimization needs proper initialization of the model trajectories to converge. To this purpose the chapter provides an initialization procedure based on driver models, see page 214. To make

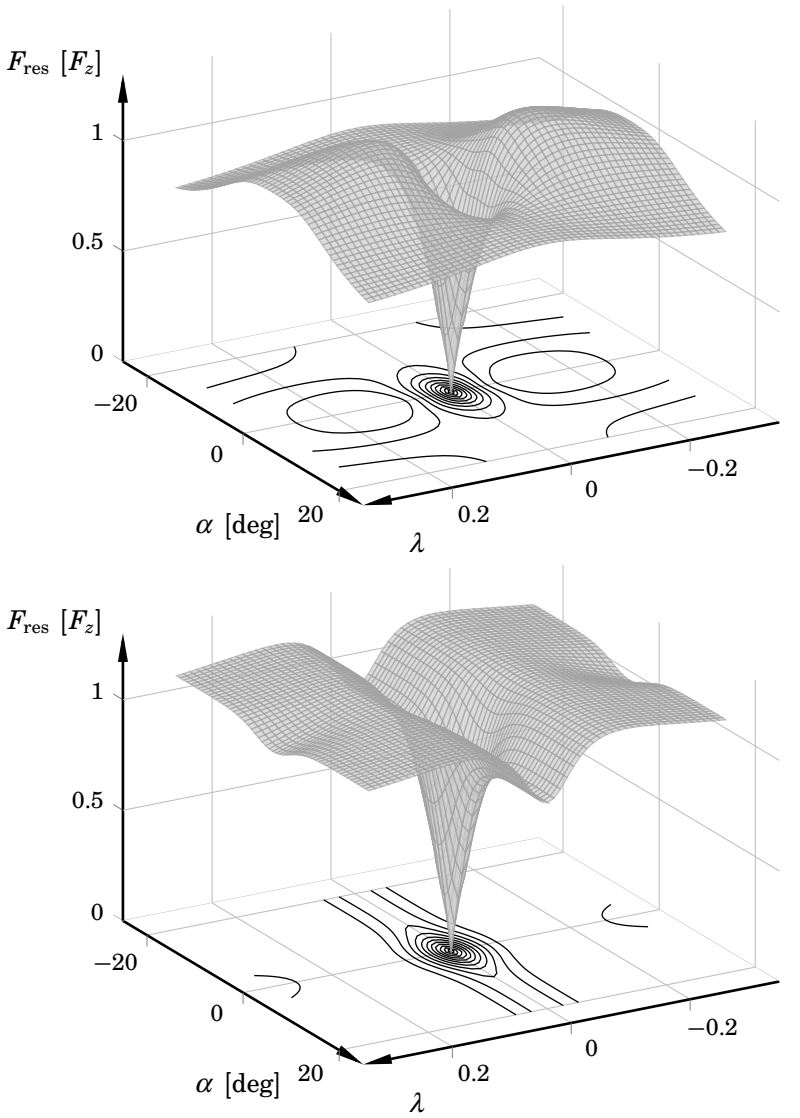

Figure 10.2 Resultant tire force F_{res} for the front wheel with WF (upper) and FE (lower), with experimental parameters from [Pacejka, 2006] according to the second column in Table A.2. The level curves are shown in the $\lambda\alpha$-planes for $F_{\text{res}} = 0$. The characteristics of WF differ depending on tire and road surface, but the typical shape of the force surface is qualitatively the same.

the convergence robust from a numerical point of view, scaling of the optimization variables to the same nominal interval is essential.

Note that, as for all nonconvex optimization problems, there are no guarantees that the obtained optimal solutions are globally optimal. One way to see if there are other solutions to the optimization problem, is to use different initial solutions. The results presented in this chapter have been obtained using several different initializations. All initializations give the same optimal solution, up to numerical tolerances.

Formulation of the Dynamic Optimization Problem

The models presented in the previous section describe a DAE system according to $\boldsymbol{f}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u}) = \boldsymbol{0}$, as discussed in Section 10.3. The wheel drive and brake torques for the front and rear axles, τ_f and τ_r, respectively, as well as the steer angle δ are control inputs. We assume that the front wheels have the same steer angle in the two-track models. Moreover, to make a fair comparison with the single-track models and for simplicity in the optimization, the two-track models have one wheel-torque input for each axle. The inputs are equally distributed between the wheels at the respective axle—that is, $\tau_1 = \tau_2 = \tau_f/2$ and $\tau_3 = \tau_4 = \tau_r/2$. The optimization problem is formulated over the time horizon $t \in [0, t_f]$ and the objective is to minimize t_f. Accordingly, we state the dynamic optimization problem as

$$
\begin{aligned}
\underset{\dot{\boldsymbol{x}},\boldsymbol{x},\boldsymbol{w},\boldsymbol{u},t_f}{\text{minimize}} \quad & t_f \\
\text{subject to} \quad & \tau_{i,\min} \leq \tau_i \leq \tau_{i,\max}, \; i \in \{1,2,3,4\} \\
& |\dot{\tau}_i| \leq \dot{\tau}_{i,\max}, \; i \in \{f, r\} \\
& |\delta| \leq \delta_{\max} \\
& |\dot{\delta}| \leq \dot{\delta}_{\max} \\
& \boldsymbol{x}(0) = \boldsymbol{x}_0 \\
& \boldsymbol{x}(t_f) = \boldsymbol{x}_{t_f} \\
& \Gamma(p^X, p^Y) \leq 0 \\
& \boldsymbol{f}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u}) = 0,
\end{aligned}
\tag{10.6}
$$

where $\boldsymbol{x}_0$ is the initial state, $\boldsymbol{x}_{t_f}$ is the desired state at the final time $t = t_f$, and (p^X, p^Y) is the position of the vehicle's mass center in the respective maneuver. In practice, the terminal constraints are only applied to a subset of the model variables. Further, $\Gamma(p^X, p^Y)$ is a mathematical description of the road constraint for the mass center. These constraints in the geometric two-dimensional XY-plane are formulated as super-ellipses with different radii and degrees.

Implementation and Solution

We implement the chassis and tire dynamics using the modeling language Modelica [Modelica Association, 2014]. The implementation of the optimization problem (10.6) is straightforward by utilizing Optimica [Åkesson, 2008], which is an extension of Modelica for high-level description of optimization problems based on Modelica models.

Because of the complex nature of the nonlinear and nonconvex optimization problem (10.6), analytical solutions are intractable. Instead, we utilize numerical methods based on direct transcription of the continuous-time problem (10.6), see Section 2.3, page 40, for a short introduction. The collocation procedure and solution of the optimization problem are performed using the open-source software platform JModelica.org [Åkesson et al., 2010; JModelica.org, 2014]. JModelica.org uses orthogonal collocation, where *Lagrange polynomials* are used for representation of the state profiles in each element. The location of the collocation points (i.e., the points within each element) are chosen as the corresponding *Radau points* [Magnusson and Åkesson, 2012]. For the optimization problems described here, $N_e = 150$ discretization elements are used and each element contains $N_c = 3$ collocation points. This procedure results in an NLP on the form (2.5). The resulting NLP is solved internally in JModelica.org using the numerical solver IPOPT [Wächter and Biegler, 2006] in combination with the solver MA57 for the linear equation systems [HSL, 2014]. IPOPT is a solver based on interior-point methods opted for large and sparse optimization problems (see Section 2.3). JModelica.org applies symbolic transformations to the DAE and transforms it into an ordinary differential equation before using the interior-point method, as described in [Magnusson et al., 2014]. This leads to a drastically reduced number of system variables and hence improved convergence speed as the algebraic variables are eliminated from the equation system. The number of variables in the NLP decreases from approximately 49 000 to 23 000 variables for the two-track models when applying the symbolic transformations. For the one-track models, the number of variables decrease from approximately 20 000 to 15 000. This transformation procedure is common practice in simulation of DAEs in Modelica tools [Cellier and Kofman, 2006] but is traditionally not used in the context of DAE-constrained optimization, where the DAE is instead usually retained in its natural semi-explicit or implicit form.

First- and second-order derivatives related to the problem are required in the iterative numerical optimization procedure. JModelica.org uses CasADi [Andersson et al., 2012] (Computer algebra system with Automatic Differentiation) to obtain these derivatives. CasADi is a low-level tool for efficiently computing derivatives while preserving sparsity using

algorithmic differentiation, and is custom-tailored for dynamic optimization. This approach significantly reduces convergence times and provides increased numerical stability compared with the use of numerical approximations based on quasi-Newton methods, such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [Dennis and Schnabel, 1983].

Initialization Procedure

Robust convergence to a solution of the NLP in IPOPT relies on proper initialization. Here, a driver model in combination with the vehicle model is used for obtaining initial trajectories to the optimization. The driver model is designed such that the vehicle tracks the middle of the road while following a predefined velocity profile. The driver model uses the steer angle δ and the rear-wheel torque τ_r, and is based on a lane-keeping controller described in [Rajamani, 2006]:

$$\begin{aligned}\delta &= \delta_{ss} - k_1 e - k_2 \dot{e} - k_3 \xi - k_4 \dot{\xi},\\ \tau_r &= \tau_{r,ff} - k_5 (v^X - v^X_{\text{nom}}),\end{aligned}$$

where δ_{ss} is the steady-state steer angle, e is the lateral deviation from the desired path, ξ is the angular deviation from the desired heading direction, $\tau_{r,ff}$ is the feedforward term for the rear torque input, v_{nom} is the desired velocity profile, and $\{k_i\}_{i=1}^5$ are driver-model parameters. The controller parameters k_1–k_4 are chosen such that the eigenvalues of the closed-loop system are placed in the same manner as suggested in [Rajamani, 2006]. The desired velocity v^X_{nom} is tracked by controlling the rear-wheel torque with the feedforward part $\tau_{r,ff}$ and a term proportional to the speed-profile error. The feedforward term is computed from $\dot{v}^X_{\text{nom}}$.

10.6 Optimization Results

This section presents optimization results obtained by solving (10.6) for different model configurations. As mentioned previously, two maneuvers are used in the evaluation. The first maneuver is a 90 deg turn, see Figure 10.3 on page 217, which is important in, for example, evaluation of ESCs in lane-keeping scenarios. The second maneuver is a hairpin turn, see Figure 10.12 on page 229, which is selected because of its extremity and that it tests all aspects of the tire and chassis modeling.

The following chassis and tire configurations are evaluated:

- *OTFE*: The one-track chassis model (4.12) in combination with the friction-ellipse based tire model (4.9).
- *OTWF*: The one-track chassis model (4.12) in combination with the weighting-functions tire model (4.11).

- *TTFE*: The two-track chassis model (4.22)–(4.27) in combination with the friction-ellipse based tire model (4.9).
- *TTWF*: The two-track chassis model (4.22)–(4.27) in combination with the weighting-functions tire model (4.11).

Optimization Preliminaries

The maximum allowed wheel angle $\delta_{\max}$ and wheel-angle change rate $\dot{\delta}_{\max}$ are set to 30 deg and 60 deg/s, respectively, which are reasonable parameters, both seen from physical and driver limitations. The start (p_0^X, p_0^Y) and final vehicle position $(p_{t_f}^X, p_{t_f}^Y)$ are set to be in the middle of the road for both maneuvers. Further, the lower and upper constraints on the torque inputs are

$$\begin{aligned} \tau_{i,\min} &= -\mu_{x,i} R_w F_i^z, \ i \in \{f, r\} \text{ or } \{1,2,3,4\} \\ \tau_{r,\max} &= \mu_{x,r} F_{0,r}^z \\ \tau_{f,\max} &= 0, \end{aligned} \tag{10.7}$$

which implies a rear-wheel driven vehicle. The steer-angle constraint is based on the approximate maximum wheel-steer angle that is possible to achieve for a standard passenger vehicle. The constraints on the derivative of the torque inputs are $\dot{\tau}_{i,\max} = 2.5\mu_{x,i} R_w mg, \ i \in \{f, r\}$. Note that the friction coefficients and the other tire parameters on the left and right wheels on the respective axle are assumed to be equal in the two-track models. The choice of torque limitations originates from that the maximum brake torque that can be applied to the wheel is much larger than the corresponding acceleration torque. Further, the drive-torque limit is set to prevent excessive wheel spin, corresponding to large slip ratios. The reason is that the empirical tire models we use are based on tire-force measurements that for experimental reasons are only possible to obtain for a limited area in the $\lambda\alpha$-plane. The reasoning behind having constraints on the derivatives of the input torques is that the driver cannot change the acceleration or deceleration instantly, and in addition the engine or motor time constant limits the change rate of the torque in a physical vehicle. Note, however, that the choice of limitations is less restrictive than the typical values measured for a combustion engine. Furthermore, to aid the solver, and without loss of optimality in the original problem formulation (10.6), the wheel velocities are constrained to be nonnegative.

Solution Times

Tables 10.1 and 10.2 show the solution times and number of iterations for IPOPT in the 90 deg turn and hairpin maneuver, respectively, using the tire parameters corresponding to asphalt. Obviously, the number of

Table 10.1 Solution times and number of iterations required for solving the time-optimal maneuver problem in the 90 deg turn on asphalt. The optimizations were executed on a standard desktop PC with an Intel i7 CPU, using IPOPT version 3.11.4.

Model	Solution Time [s]	Iterations
OTFE	8.6	159
OTWF	5.3	96
TTFE	27.9	110
TTWF	44.3	144

Table 10.2 Solution times and number of iterations required for solving the time-optimal maneuver problem in the hairpin turn on asphalt. The optimizations were executed on a standard desktop PC with an Intel i7 CPU, using IPOPT version 3.11.4.

Model	Solution Time [s]	Iterations
OTFE	4	78
OTWF	3.7	74
TTFE	219.6	510
TTWF	82.4	283

iterations and computation times are dependent on the complexity of the model configuration and the maneuver. The solution times for *TTWF* and *TTFE* are much longer than for *OTWF* and *OTFE*. Note that the increase in model complexity does not translate directly to the increase in solution time.

The solution times for the 90 deg turn are sometimes longer than the corresponding solution times for the hairpin maneuver, despite the seemingly simpler geometry in the 90 deg turn. This is partly explained by that IPOPT relies on proper initialization, and the generated initial trajectories are not equally suitable for the respective maneuver.

Optimization Results in 90 deg Turn—Dry Asphalt

Here, we discuss the results when using the tire parameters that correspond to asphalt. The vehicle start position is set to $(p_0^X, p_0^Y) = (37.5, 0)$ m in the 90 deg turn and the vehicle is aligned with the road direction, $\psi_0 = \pi/2$. The target vehicle position is set to $(p_{t_f}^X, p_{t_f}^Y) = (0, 37.5)$ m, again with the vehicle heading in the road direction, $\psi_{t_f} = \pi$. The initial velocities are $v_0 = 70$ km/h (approximately 19.5 m/s) and the control inputs at t_0 are set to zero. Figure 10.3 displays the geometric paths. The start position $(p_0^X, p_0^Y) = (37.5, 0)$ corresponds to the lower right corner.

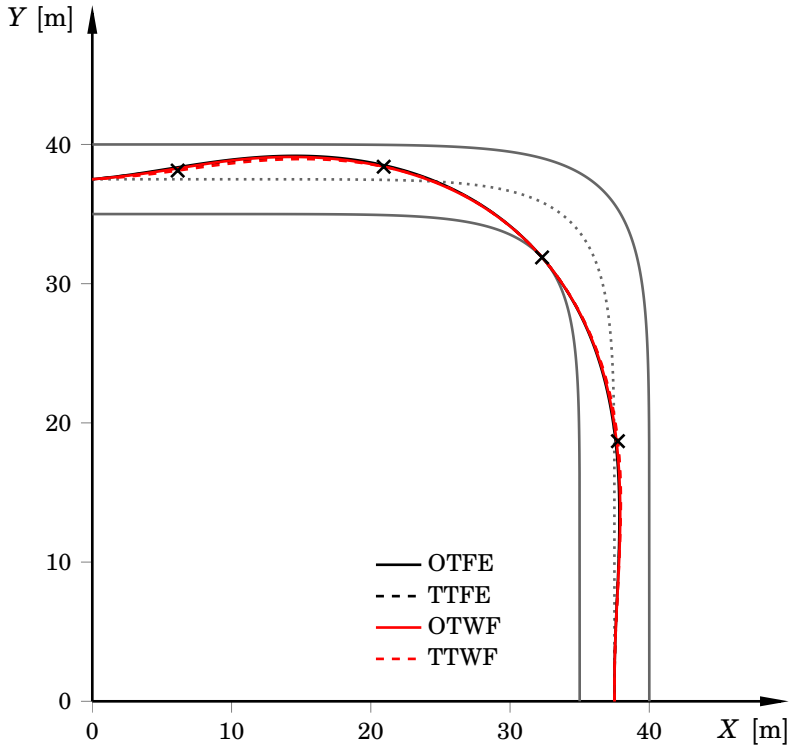

Figure 10.3 Time-optimal geometric paths in the 90 deg turn for tire parameters corresponding to dry asphalt. The vehicles start in the lower right corner. The difference is largest at approximately $(p^X, p^Y) = (15, 39)$, and is roughly 0.3 m. The dots indicate the middle of the road and the crosses indicate the vehicle position at each second.

The paths are very similar to each other throughout the maneuver. The largest difference is approximately 0.3 m.

Table 10.3 provides the execution times for the maneuver with the respective model configuration. The execution times vary approximately 1.6% at most, which occurs between the one-track models and *TTWF*. *TTWF* and *TTFE* exhibit larger discrepancies in execution times for the respective tire model than *OTWF* and *OTFE*. This is a result of the load transfer incorporated in the former models, which results in large variations in the normal load on the wheels during the maneuver. Using *TTFE* results in shorter execution time than using *TTWF*. The reason is that the resulting force for FE is always larger than that for WF when combined slip is present, see Figure 10.2. In this maneuver combined slip is devel-

Table 10.3 Time for executing the maneuver with each model configuration in the 90 deg turn.

Model	Maneuver Execution Time t_f [s]
OTFE	4.28
OTWF	4.28
TTFE	4.30
TTWF	4.35

oped; hence, FE results in larger forces and thus increased acceleration and deceleration. However, the differences in execution time are minor.

Trajectories The first observation when investigating the optimal trajectories in Figure 10.4 closer is that the slip behavior is much more excessive for the models using FE, which can be observed in the plot for β. An explanation is that the characteristics of FE (compare FE and WF in Figure 10.2) leads to that the largest forces are attained when the vehicle both accelerates and slides. The effects of these characteristics are pronounced when modeling load transfer, which can be seen by comparing β between *TTFE* and *OTFE*. Nevertheless, the models yield very similar solutions for v^X and ψ, and β is similar for *OTWF* and *TTWF*. Approximately the same lateral forces are generated at each axle, but the longitudinal forces differ considerably between the one-track and two-track models (Figure 10.5). The reason is the load-transfer modeling in the two-track models, and it is clear that this has a large impact on the available tire forces. Moreover, Figure 10.6 clearly shows that the pitch and roll angles, and thus the load transfer, are very similar between *TTWF* and *TTFE*.

Control Inputs Figure 10.7 shows the optimal control inputs. One observation is that the different models result in optimal control inputs with characteristics that are similar in several aspects. However, the torques differ significantly in magnitude between the one-track and two-track models, because of the lack of load transfer in the one-track models. The two-track models reduce front-wheel braking slightly earlier than the one-track models, see τ_f in Figure 10.7. This might be a consequence of τ_f being equally distributed between the front wheels for the two-track models. Thus, when braking while cornering, the inner wheels will have less load and thus risk to lock up for large brake torques.

The steer angle varies considerably between the models. At $t \approx 0.7$ s, a smaller δ is obtained for *TTWF* than for the remaining model configurations. Shortly after, between $t \approx 0.8$–1.3 s, the steer angle increases sharply in the solutions for *TTWF*. The other models do not give rise to

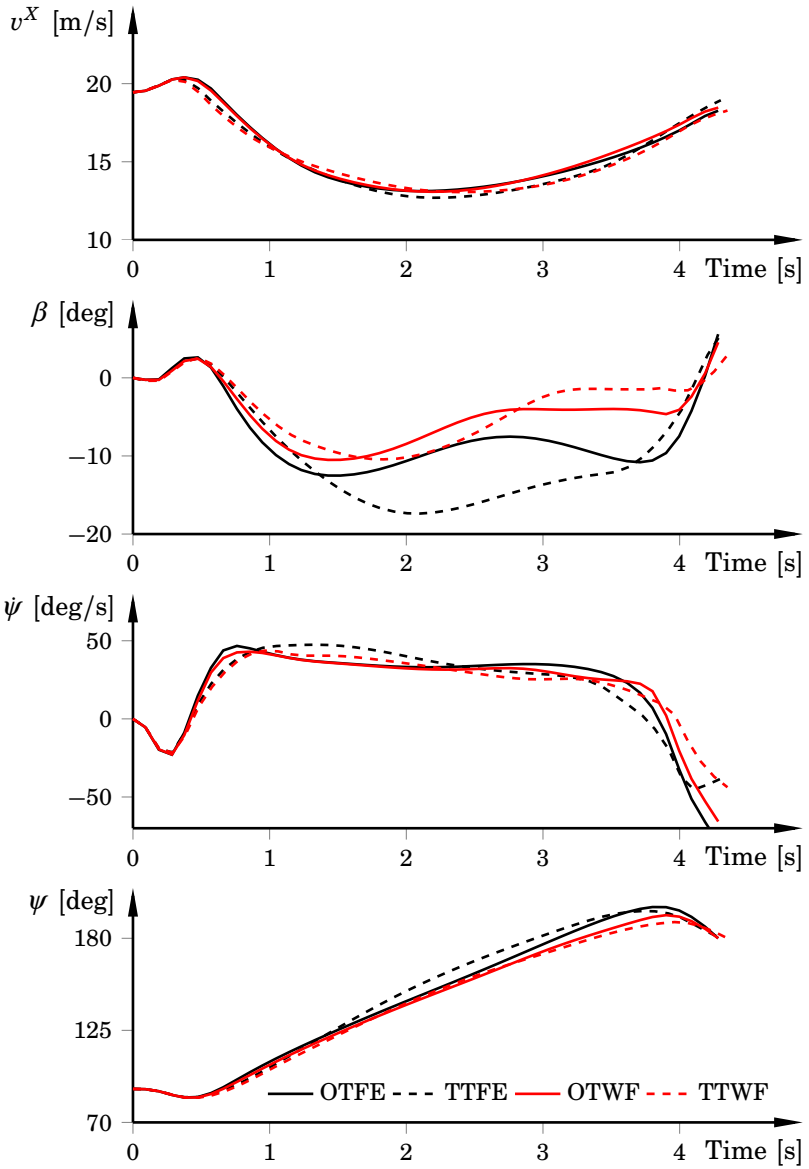

Figure 10.4 Time-optimal trajectories in the 90 deg turn for tire parameters corresponding to asphalt. The models that incorporate the friction ellipse differ from each other, but the models based on the weighting functions are similar for all variables.

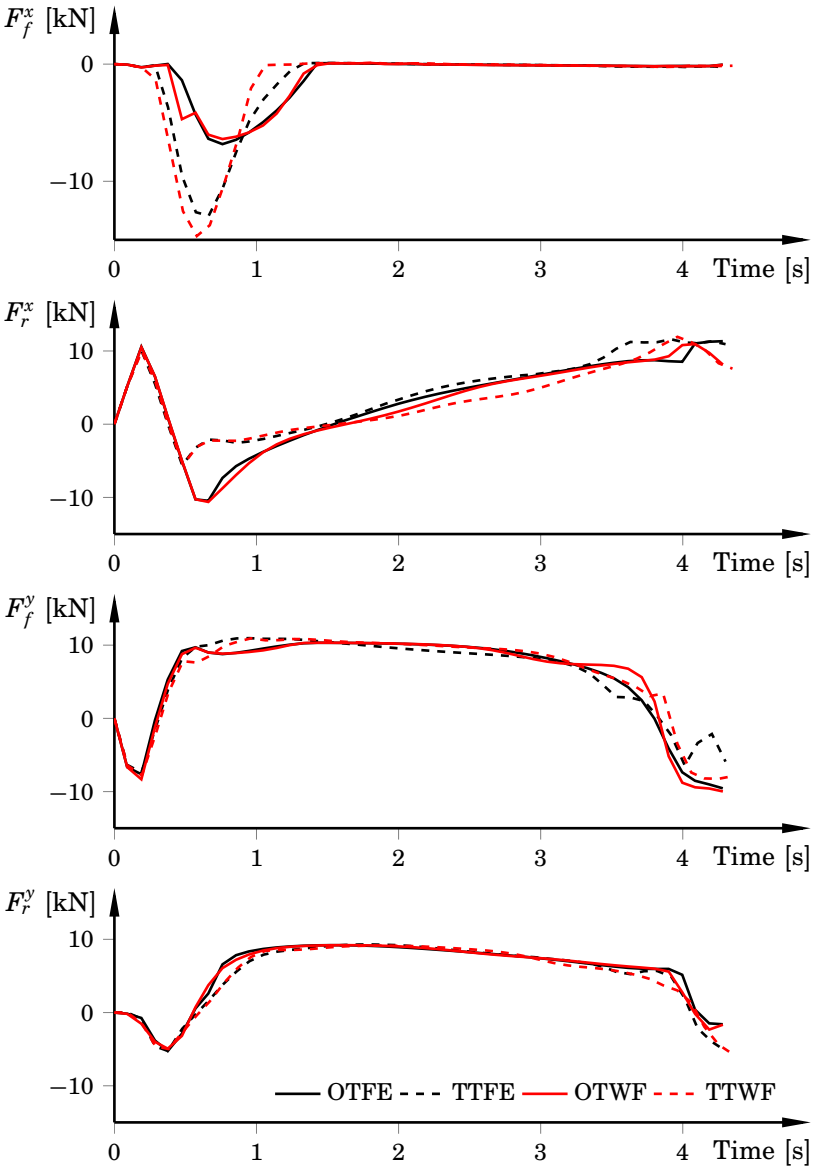

Figure 10.5 Time-optimal longitudinal and lateral forces for each axle in the 90 deg turn. The tire parameters correspond to asphalt. The lateral forces are similar for all models, whereas the longitudinal forces differ because of the different load transfer models.

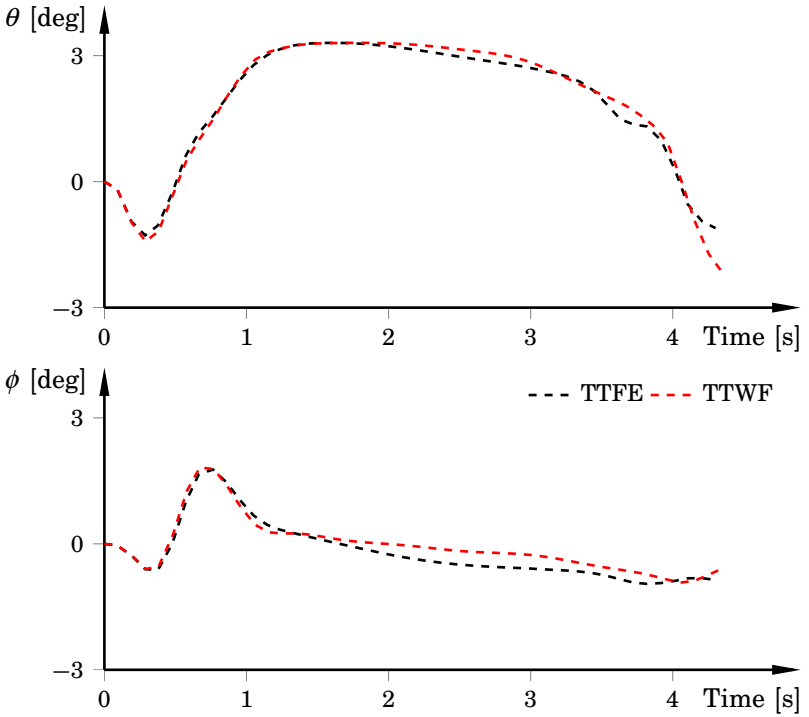

Figure 10.6 Time-optimal pitch and roll angles in the 90 deg turn for tire parameters corresponding to asphalt. The one-track models do not include pitch and roll dynamics, and are therefore not shown.

this behavior. However, the impact on the tire forces seems to be negligible (Figure 10.5).

It is interesting that despite the optimal control inputs differing significantly between the one-track and two-track models in parts of the maneuver, especially when entering the turn, the variables in Figures 10.3 and 10.4 are very similar for *OTWF* and *TTWF* (i.e., the models that use the weighting functions).

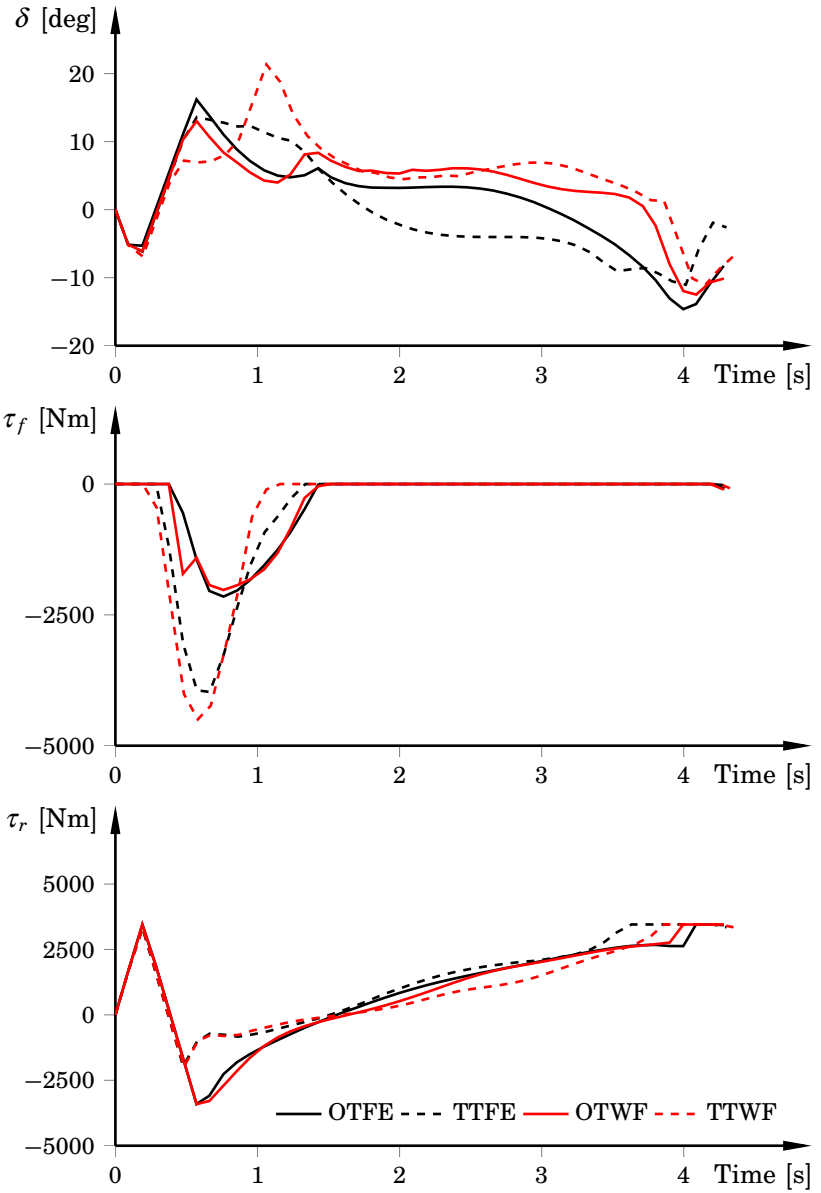

Figure 10.7 Time-optimal control inputs in the 90 deg turn for tire parameters corresponding to asphalt. Both the steer angles and torques differ considerably in parts of the maneuver.

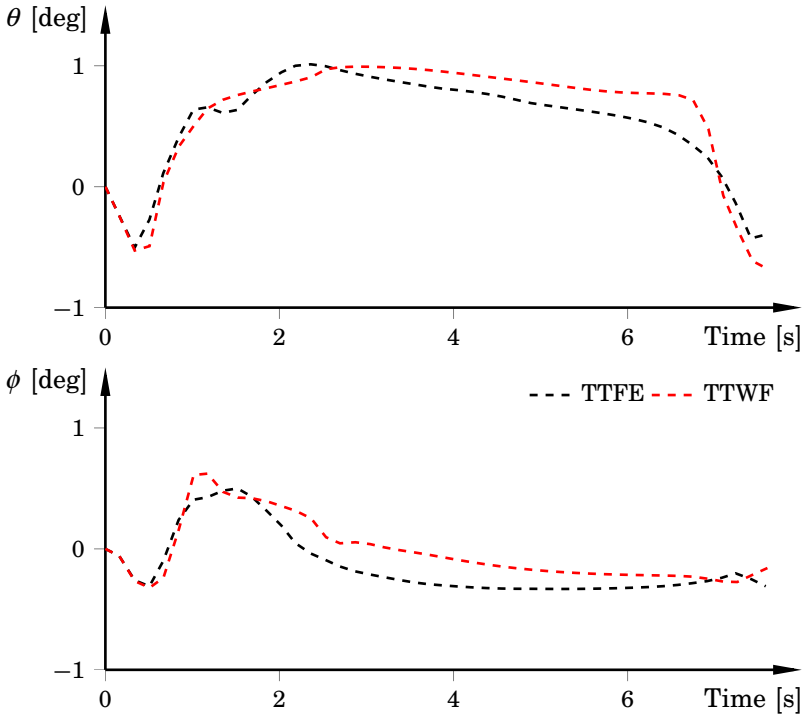

Figure 10.8 Time-optimal pitch and roll angles in the 90 deg turn for tire parameters corresponding to snow (see Tables A.2–A.3). The angles are much smaller when compared with the results for dry asphalt in Figure 10.6, implying less load transfer. Note that the scale is different from Figure 10.6.

Optimization Results in 90 deg Turn—Snow

The initial velocity is $v_0 = 40$ km/h (approximately 11 m/s), which is smaller than for asphalt. The rest of the variables have the same initial values as for asphalt. Intuitively the differences between the models should be suppressed on low friction surfaces compared with high friction surfaces, because load transfer has less impact on low-friction surfaces. Figure 10.8 displays the time-optimal pitch and roll angles for *TTFE* and *TTWF*, and it verifies this claim; the angles are smaller than those for asphalt in Figure 10.6. Hence, the load transfer has less impact on snow than on asphalt, as expected.

Figures 10.9 and 10.10 show corresponding optimal trajectories and control inputs, respectively, for the 90 deg turn on snow. For *OTWF* and *TTWF*, the differences are suppressed for most variables. Both the lon-

gitudinal velocities and the yaw rates are virtually inseparable for most parts of the maneuver, and the body-slip angles are more similar than in Figure 10.4. This is, however, not the case for *OTFE* and *TTFE*. The differences are even more pronounced, with, for example, the sideslip angle β differing with approximately 20 deg. Hence, the difference in body slip between *TTFE* and *OTFE* on snow is actually larger than the peak values for *TTWF* and *TTFE* on asphalt. The reason for this is most probably that the longitudinal force does not depend on the lateral force for FE; that is, it is always possible to achieve the desired longitudinal force, irrespective of the lateral slip. As mentioned earlier, this is an approximation, and when performing aggressive maneuvering this feature is probably utilized to achieve large forces.

The yaw rates and velocities are smaller for all models compared with the solutions for asphalt. The reason is the lower friction coefficients on snow. The slip behavior for the WF-based models is similar compared with asphalt. For *OTWF*, the peaks in β are approximately 10 and 14 deg on asphalt and snow, respectively, whereas the value for *TTWF* is 10 deg for both asphalt and snow.

Figure 10.11 contains the resulting longitudinal and lateral slip for the rear wheels when using *TTWF* and *TTFE*, respectively. It is clear that FE results in much larger lateral slip. This is consistent with the large β in Figure 10.9. We see that FE yields less longitudinal slip. However, this is partly owing to the torque constraints (10.7). These constraints decrease the possibilities for FE to reach large λ when α is nonzero. Without them, it is likely that larger λ would be obtained with FE. The torque constraints do not impose large restrictions for WF, because the resultant force decreases for combined slip.

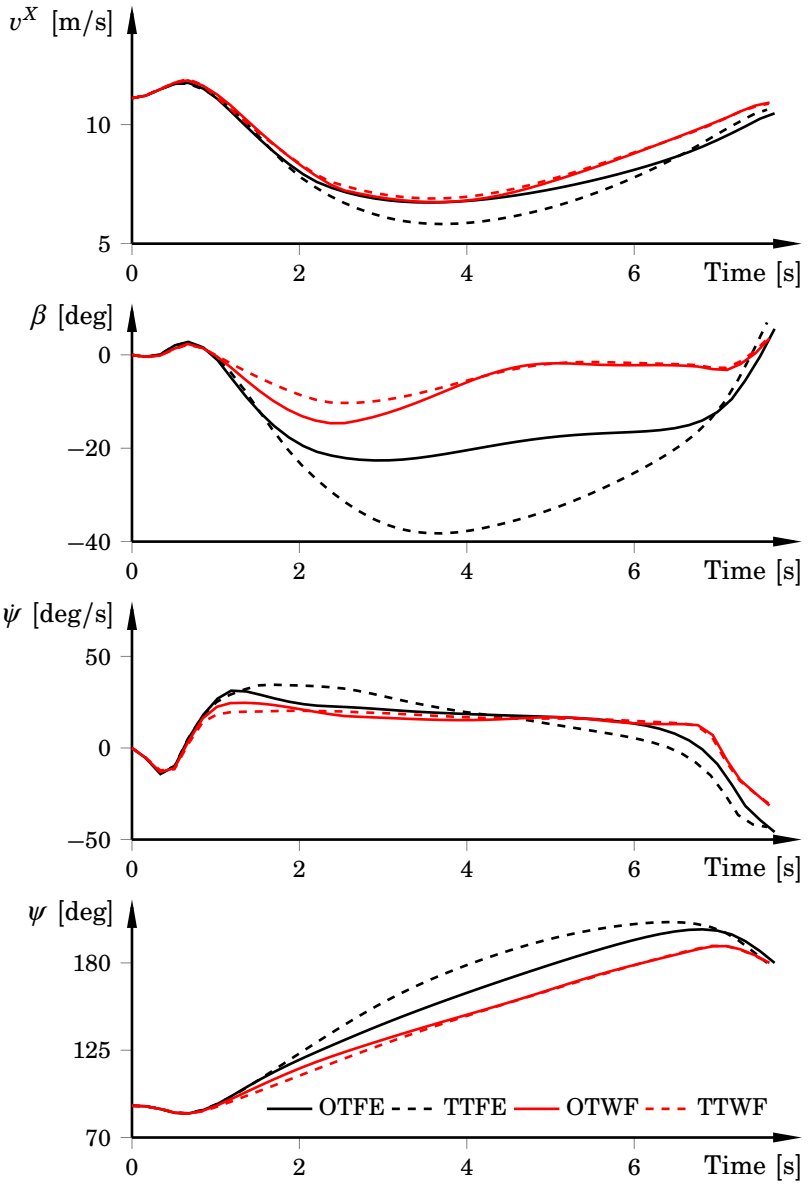

Figure 10.9 Time-optimal solutions in the 90 deg turn for tire parameters corresponding to snow (see Tables A.2–A.3). The models using WF are more similar than in the corresponding figure for asphalt (Figure 10.4), which is expected. *TTFE* and *OTFE* differ considerably from each other.

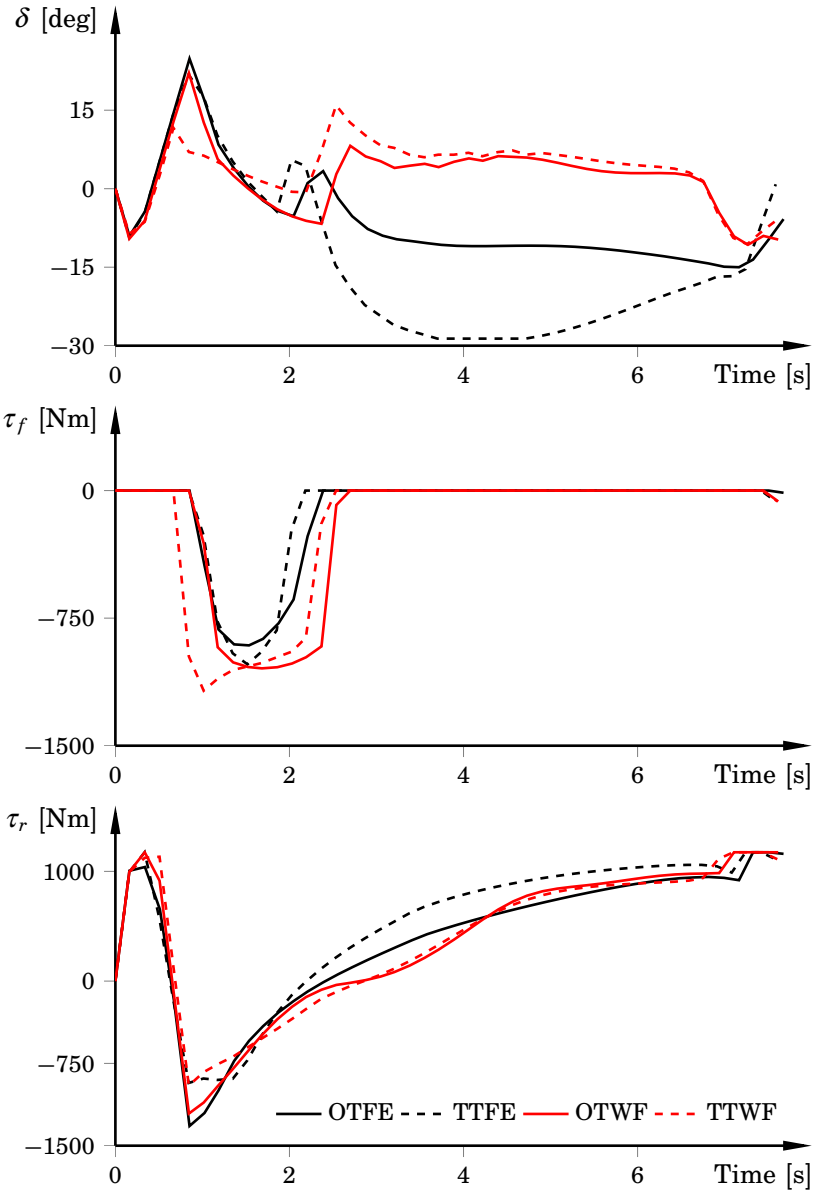

Figure 10.10 Time-optimal control inputs in the 90 deg turn for tire parameters corresponding to snow (see Tables A.2–A.3). The load-transfer effect is suppressed compared with asphalt, at least for *TTWF* and *OTWF*.

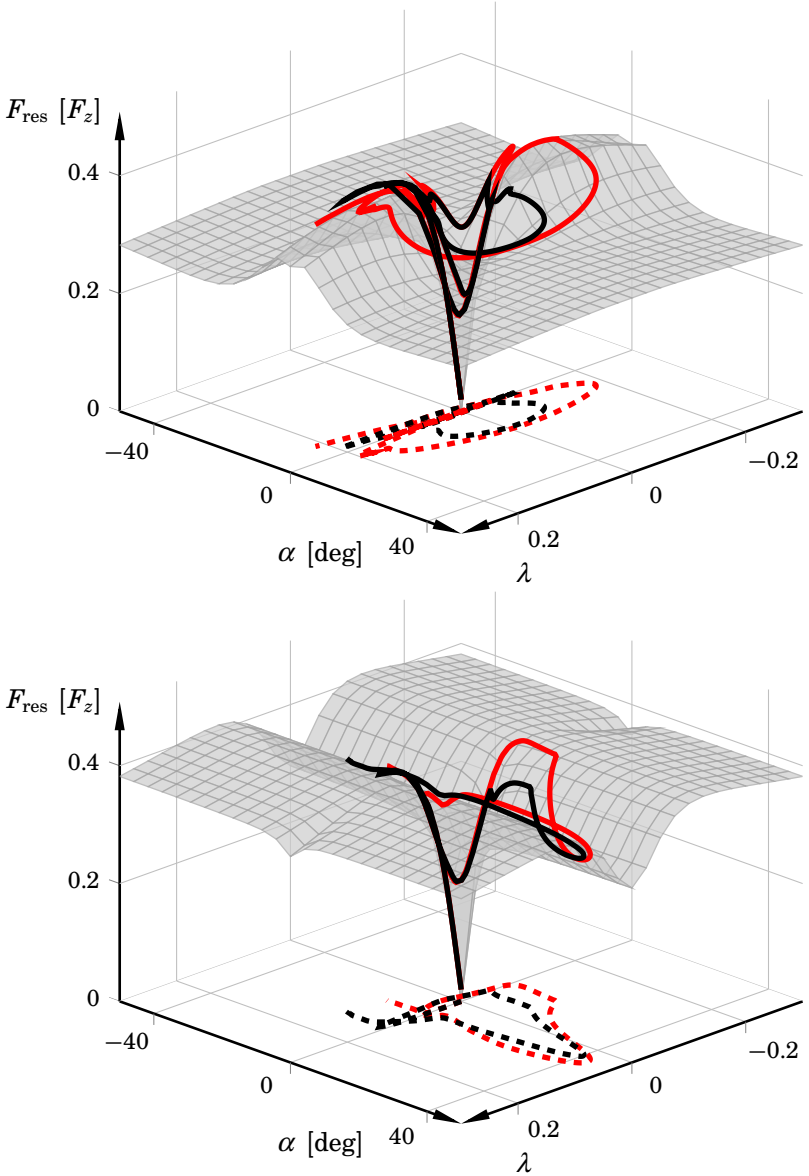

Figure 10.11 Resultant rear tire forces for *TTWF* (upper) and *TTFE* (lower) in the 90 deg turn for snow (black - left wheel, red - right wheel). The friction ellipse yields larger lateral slip. The resulting combined slip is projected in the $\lambda\alpha$-plane.

Optimization Results in Hairpin Turn—Surface Comparison

We will now discuss and compare time-optimal vehicle control for asphalt, snow, and ice, using the hairpin turn as test scenario. FE does not have parameters that describe combined slip; that is, the behavior for combined slip is entirely decided by the nominal tire forces. Therefore, the surface comparison is done using WF as tire model.

The results for the 90 deg turn indicate that the differences between chassis models when using WF are suppressed on low-friction surfaces compared with the results for high friction. This is in agreement with intuition and the pitch and roll angle plots in Figures 10.6 and 10.8. Consequently, we will perform the comparison using WF in combination with the one-track chassis model, based on parametrizations as described on page 208.

Solution Times Table 10.4 shows the solution times for the respective optimization problems. Asphalt gives the shortest solution times. This is no surprise, given that asphalt has force characteristics with more pronounced peaks, implying that the gradient search in the interior-point solver will take larger steps.

Optimal Solutions The start and final positions in the hairpin turn are set to $(p_0^X, p_0^Y) = (-5, 0)$, $(p_{t_f}^X, p_{t_f}^Y) = (5, 0)$ for all three surfaces, and the heading angles are aligned with the road direction. The initial velocity is $v_0 = 25$ km/h (approximately 7 m/s). Table 10.5 displays the execution times for the different surfaces. Obviously, the time for completing the maneuver is longer for snow and ice than for asphalt. The time-optimal

Table 10.4 Solution times and number of iterations required for solving the minimum-time control problem in the hairpin turn for the different surfaces.

Surface	Solution Time [s]	Iterations
Asphalt	3.7	74
Snow	6.18	103
Ice	47.5	608

Table 10.5 Time for executing the maneuver for the different surfaces, using *OTWF* in the hairpin turn.

Surface	Maneuver Execution Time t_f [s]
Asphalt	8.55 s
Snow	13.88 s
Ice	19.21 s

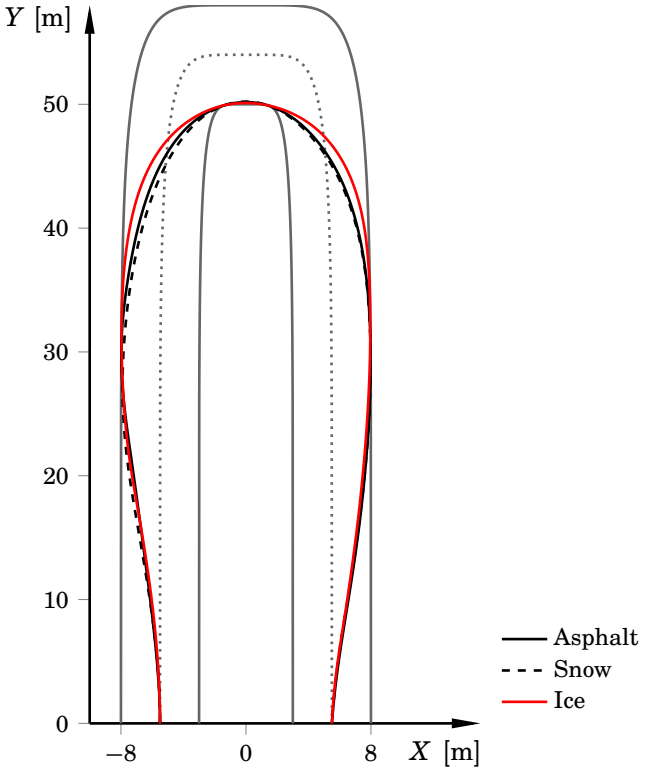

Figure 10.12 Time-optimal geometric paths in the hairpin turn for the different surfaces. The vehicles start in the lower left corner. The dots indicate the middle of the road.

geometric paths of the respective mass center, shown in Figure 10.12, are similar for asphalt and snow. All three paths are rather symmetric, but the vehicle path is wider on ice. A possible explanation is that there is less available friction on ice. Thus, the vehicle cannot enter the turn as aggressively as for the other surfaces.

Optimal Trajectories The maneuver execution time differs significantly between the surfaces. Thus, for easier comparison, Figure 10.13 visualizes the optimal trajectories as functions of driven distance rather than time. The differences in velocity and yaw rate are expected since the friction coefficients, and thus the available forces, are different.

The slip behavior between the different surfaces is fundamentally different. The optimal trajectories have large slip in the critical part of the maneuver on asphalt and snow, but not on ice. This is coupled to the wide

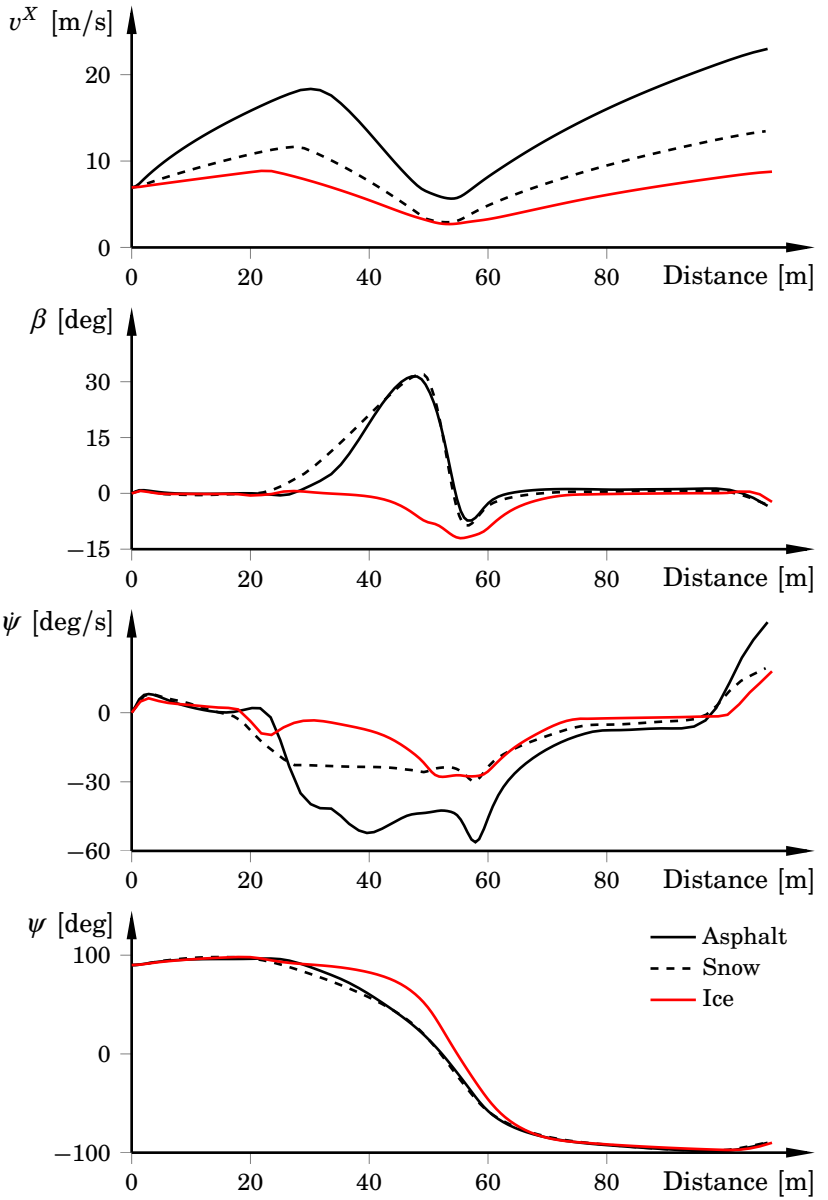

Figure 10.13 Time-optimal trajectories in the hairpin turn on different surfaces, plotted as functions of the driven distance. The slip behavior is fundamentally different on ice.

vehicle path on ice; it is easier to approach and exit a turn narrowly if sliding through it. The reason for the difference can be explained by the force characteristics of the ice model compared with the other surfaces. Figures 10.14–10.16 contain the resulting longitudinal and lateral slip for all three surfaces. The resultant tire forces for ice exhibit a considerably sharper peak and thus decay faster, with respect to combined slip, than for both asphalt and snow. Thus, to achieve the desired time optimality on ice, it is natural to choose a small-slip control strategy. Note that the slip values are smaller for the front wheels, because the vehicle is rear-wheel driven.

Optimal Control Inputs The differences between asphalt, snow, and ice when considering the control inputs are fundamental, as seen in Figure 10.17. First, the optimal control inputs on snow and ice both apply braking and use the steer angle δ earlier when approaching the hairpin. This is most certainly an effect of the much reduced tire forces that can be realized on these surfaces compared with asphalt. Second, the steer angle differs between ice and the other surfaces. The reason is that the vehicle employs counter steering when it starts to slip on asphalt and snow as it approaches the hairpin.

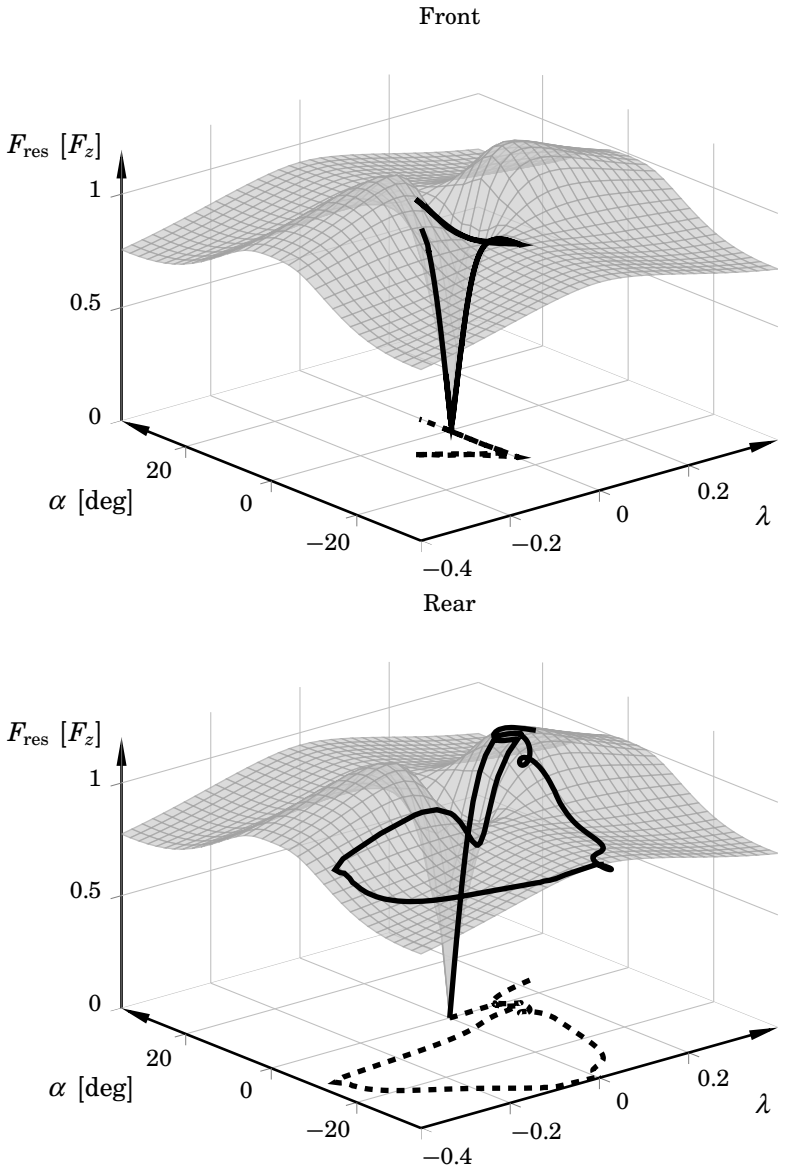

Figure 10.14 Resultant tire forces for the front and rear wheel in the hairpin turn for asphalt, using *OTWF*. The resulting combined slip is projected in the $\lambda\alpha$-plane.

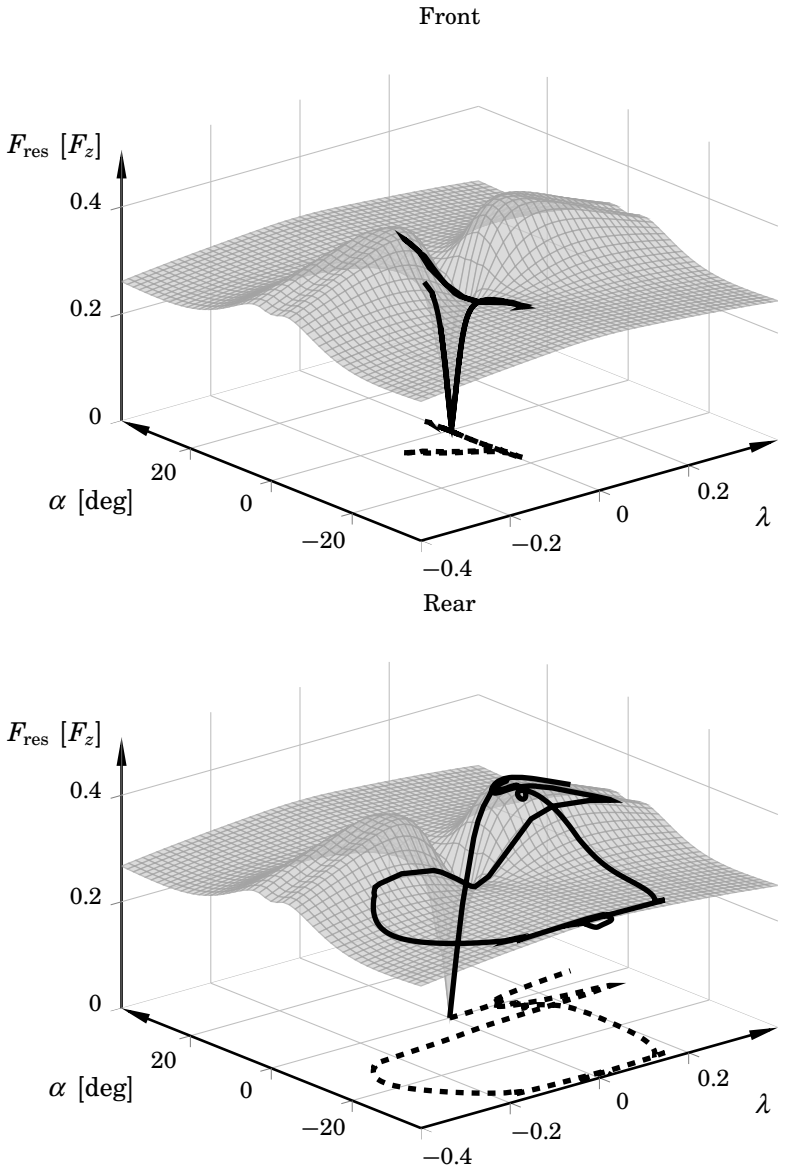

Figure 10.15 Resultant tire forces for the front and rear wheel in the hairpin turn for snow, using *OTWF*. The resulting combined slip is projected in the $\lambda\alpha$-plane.

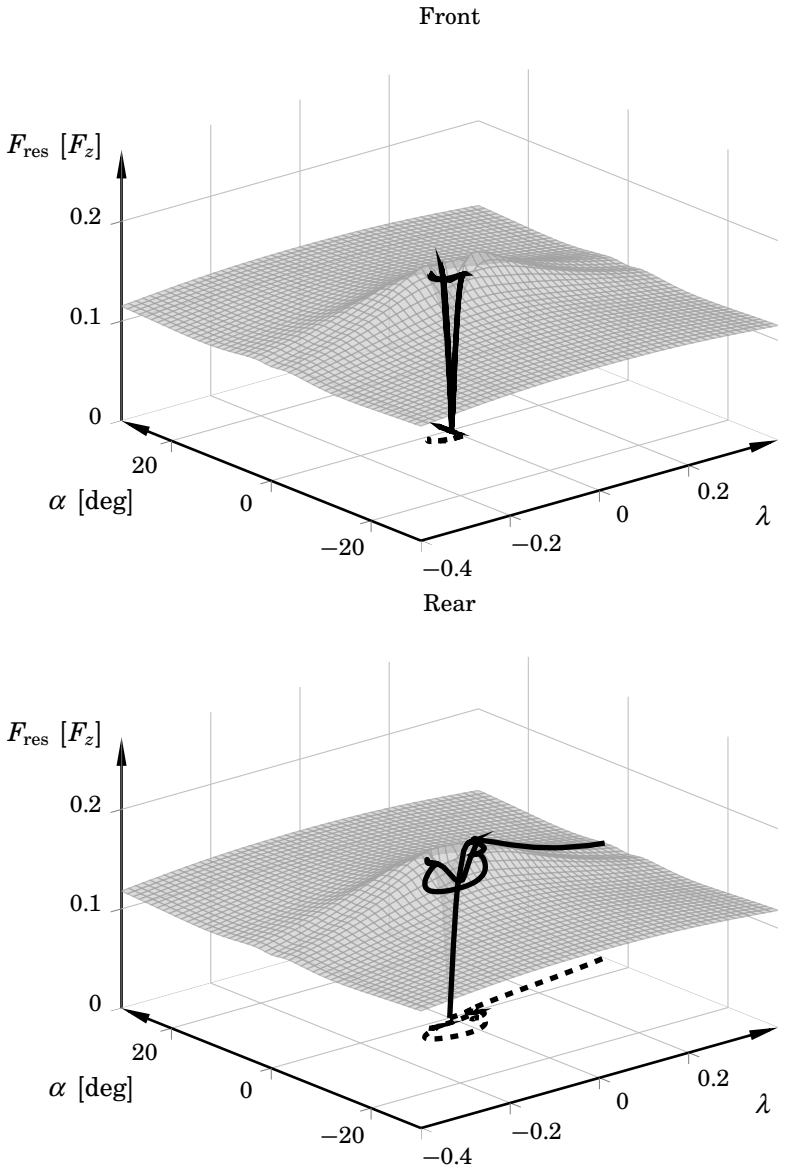

Figure 10.16 Resultant tire forces for the front and rear wheel in the hairpin turn for ice, using *OTWF*. The resulting combined slip is projected in the $\lambda\alpha$-plane. The forces decay faster for combined slip when compared with Figures 10.14 and 10.15.

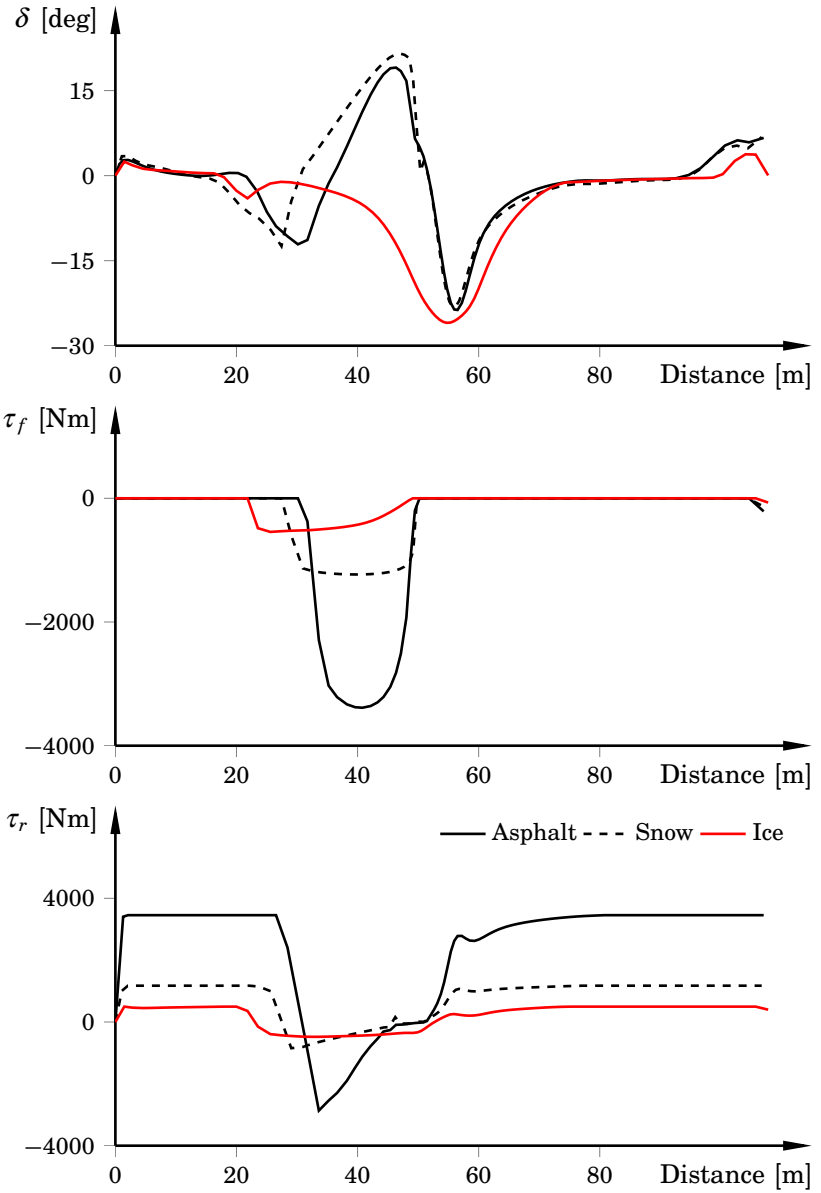

Figure 10.17 Time-optimal control inputs in the hairpin turn for *OTWF* on asphalt, snow, and ice. The solutions apply braking earlier on snow and ice. The steer angle is kept rather steady on ice, whereas the solutions on snow and asphalt apply counter steering.

Naive Approach to Tire-Model Calibration An integral part of the vehicle model is the tire characteristics. One approach to account for different surfaces is to only scale the friction coefficients μ_x and μ_y, as done in, for example, [Chakraborty et al., 2011]. However, the peaks in the tire-force surfaces occur at different lateral and longitudinal slip combinations, see Figures 10.14–10.16. Also, the sharpness and width of the maxima and minima differ depending on the surface. Thus, only changing the friction coefficients will render different force characteristics (and thus different optimal solutions) compared with when changing the complete set of parameters.

That this can have a large impact on the optimal trajectories is verified by constructing a tire-force model where the parameters corresponding to asphalt are used together with the friction coefficients for ice. Solving the dynamic optimization problem gives that the optimal solution has significant slip, in contrast to the results obtained for the empirical ice model. See Figure 10.18 for the geometric path obtained by scaling the friction coefficients only.

10.7 Concluding Discussions

It is clear from Section 10.6 that the model choice can potentially lead to fundamentally different control strategies, where the choice between WF and FE seems crucial, see Figures 10.7 and 10.10. The characteristics of the two tire models are fundamentally different, where WF results in much smaller forces for combined slip than when only one of the slip quantities is nonzero. This is in contrast to FE, where the largest forces are attained for combined slip. The behavior of FE is entirely predetermined by the nominal force parameters. The characteristics of FE is a result of that the longitudinal force is not affected by the lateral slip, which is not reasonable. This results in very extreme maneuvering compared with WF, especially for low-friction surfaces, see Figure 10.9. WF has been experimentally verified in several independent studies [Pacejka, 2006; Braghin et al., 2006]. Still, experimental data for large combined slip have large uncertainties, if available at all, which implies that the tire parameters are more uncertain in those regions. This makes it difficult to draw any decisive conclusions. However, WF generally leads to solutions with relatively small body slip. This is consistent with the behavior observed when monitoring the control strategies exerted by expert drivers [Tavernini et al., 2013] on asphalt.

Whether to use a one-track or two-track model has large impact on the resulting control strategy. If good feedback-control performance is an aim, the obvious choice is to use a two-track model since it is more ad-

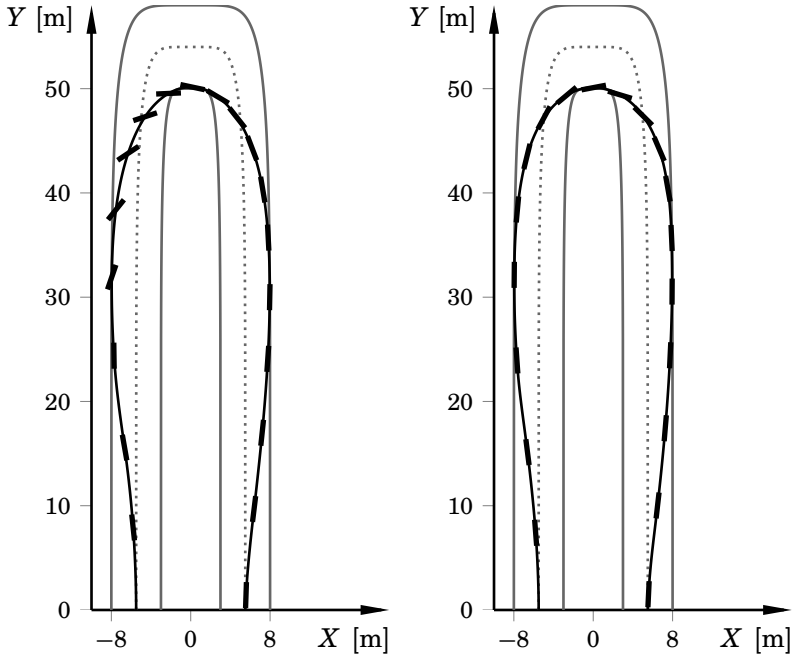

Figure 10.18 Time-optimal paths on ice for two model parametrizations, scaling of friction coefficients only (left) and empirical tire model (right). The black bars indicate the vehicle heading each second. When only scaling the friction coefficients, the solution has significant slip.

vanced. However, the results in this chapter indicate that the two-track model is highly dependent on the choice of tire model, where the different characteristics for combined slip has large impact on the solutions. To exemplify: The one-track model with FE behaves rather similarly to the models with WF in several aspects, indicating that FE is valid. If FE is combined with the two-track model, however, the results are fundamentally different, especially on low-friction surfaces. This is not in agreement with intuition. It also indicates that FE should be used with care, and under situations where nonphysical modes in the vehicle modeling are sure not to be triggered.

The WF-based vehicle models, *OTWF* and *TTWF*, behave similarly in several key aspects; for example, variables traditionally used for detecting loss of maneuvering stability, such as the yaw rate and the body-slip angle, only exhibit minor discrepancies. The input torques and steer angles differ significantly during parts of the maneuver, but the differences do not have much impact on the other variables, which is interesting. The minimum-

time optimization problem gives very aggressive control signals and large resulting slip values; hence, the vehicle variables are likely to be even more similar for other cost functions.

Regarding tire-force modeling of different road surfaces, a conclusion is that only scaling the friction coefficients can lead to fundamentally different solutions compared with adjusting all involved tire parameters. The tire modeling in this chapter is based on experimental data, gathered from controlled experiments. Nevertheless, the data have uncertainties, especially for large slip. In addition, it is not obvious that using scaling factors is the most sensible way to model different surfaces. Still, the results show that when large longitudinal and lateral slip are present, more careful tire modeling might be required. The slip angles are similar for asphalt and snow, but fundamentally different on ice. However, the optimal driving techniques—that is, the control actions—are different depending on tire-road characteristics. An example is that the solutions on low-friction surfaces start to brake earlier.

Consequences for Safety Systems

The results presented in this chapter can be seen as an estimate of the largest differences that can occur between the models, because a minimum-time optimization problems typically fully utilize the available actuation. The problem formulation when considering a safety system is clearly not a minimum-time formulation. Rather, variables such as velocity and yaw rate are more likely to be part of the cost function.

The vehicle variables are rather insensitive to the choice of chassis model when using the weighting functions, even for a minimum-time formulation. The implication of this is that state trajectories may be generated by optimization using chassis models with low complexity, such as the one-track model. These state trajectories can then be utilized as inputs to an allocation algorithm for distributing the desired torque to the respective wheel.[1] This unfortunately does not hold for FE, where the vehicle variables in parts differ considerably.

The fundamentally different vehicle behavior for different tire-road characteristics is interesting, because it indicates that to further improve safety systems, they need to account for the surface interaction in informed ways. Sometimes it is enough to only consider a few of the tire parameters when modeling different surfaces, but it can also lead to very different behavior compared with using more of the available tire parameters. Many of the safety systems currently in production handle different surfaces. Even so, there is much to be gained in that respect.

[1] Sometimes it is the longitudinal slip that should be distributed, depending on if an ABS handles the torque control or not.

10.8 Summary

This chapter investigated time-optimal maneuvers in a 90 deg turn and a hairpin maneuver. Four different vehicle models were compared in two different maneuvers, using three different road-surface parametrizations. The compared models are of the same type that is frequently encountered in automotive literature. The results were thoroughly analyzed and discussed. One conclusion was that the optimal control inputs differ considerably between the models, but the corresponding optimal trajectories have relatively smaller differences. Further, the results showed that the weighting-functions tire model produces similar behavior between the chassis models.

The tire-road interaction was parametrized based on experimental data. We used the hairpin maneuver as a test scenario for the surface comparison. One conclusion here was that the different parametrizations gave distinct results; for example, the slip behavior differed between the surfaces, and braking was initiated earlier on the low-friction surfaces.

11

Closed-Loop Optimal Control for Vehicle Autonomy

Currently, one of the main trends in automotive-related research is improving situation awareness in vehicles, which we saw one example of in Chapter 9. The enabler for this is, as stated already in Chapter 1, the increased sensing and computing capabilities in modern passenger vehicles [Lundquist, 2011]. Examples of sensing information are cameras, radar systems, satellite positioning systems, and inertial sensors. The combination of improved sensing and actuation capabilities makes it possible to improve current safety systems for at-the-limit maneuvers, where autonomous, or at least semiautonomous, lane-keeping systems are natural extensions to ESCs.

More autonomy is not only desirable for reducing fatalities. Autonomy can also decrease the number of light collisions—for example, caused by nondeterministic or inattentive driving behavior. In total there are about 10 million (in the USA alone) traffic-related accidents every year [NHTSA, 2011], of which approximately 30 000 are fatal. This implies that autonomy under normal driving conditions is also an important area. Another area is highway driving, where platooning and advanced cruise-control techniques can improve fuel efficiency [Alam et al., 2010].

In this chapter we outline a hierarchical, high- and low-level optimal-control approach to lane keeping and trajectory generation for road vehicles, which already at the high level takes into account nonlinear chassis and tire dynamics. In the previous chapter, one of the main conclusions was that one-track and two-track models have similar behavior when using the weighting functions for tire modeling, but that the input torques and steer angles are significantly different. This conclusion is utilized here, where a high-level trajectory-generation problem is cast as a dy-

namic optimization problem over a horizon that is dependent on the road curvature. More specifically, the dynamics is already at the high level based on a nonlinear vehicle model, unlike other approaches. We use the one-track model combined with the experimentally verified weighting-functions tire model, which incorporates combined-slip behavior. The low-level control-input allocator is formulated as a nonlinear model predictive control (NMPC) problem [Del Re et al., 2010; Mayne et al., 2000] over a part of the high-level references. Nonlinear optimization problems have the disadvantage that they sometimes fail to converge, or the convergence is slow. By combining the approach with linear MPC (LMPC), we provide a suboptimal control solution in the cases for when the NMPC fails to converge in a timely manner.

11.1 Related Work

Application of optimal control in automotive systems is a popular research topic. In [Gao et al., 2010], a hierarchical approach for automated highway driving was introduced. In that approach, the trajectory-generation problem is solved with an LMPC at the high level. The references are then allocated to the wheels via a low-level NMPC. The approach in [Gao et al., 2010] uses a simple point-mass representation of the vehicle at the high level. Although this might work well for steady-state conditions, the reference trajectories that are generated may not be feasible in more aggressive maneuvering, because there is little physical coupling between the high-level and low-level controllers. Moreover, the low-level NMPC allocates forces to the wheels. This means that the dynamics of the torque actuation is neglected. Other work on MPC in vehicle dynamics control is [Falcone et al., 2008], where an MPC was used for path following. MPC was also used in [Ali et al., 2013] for predictive prevention of loss of vehicle control, where the operation was restricted to the linear region of the vehicle dynamics. MPC has also been used for idle-speed control [Di Cairano et al., 2012].

Mitigation of collision impact has been explored in a series of papers, see [Chakraborty et al., 2011; Chakraborty et al., 2013] for two examples. Optimization of emergency maneuvers have also been treated in the literature, see [Dingle and Guzzella, 2010; Frasch et al., 2013; Shiller and Sundar, 1998] for three examples. A framework for threat assessment and trajectory planning was described in [Anderson et al., 2010]. A method for optimal control allocation in yaw stabilization of automotive vehicles was proposed in [Tøndel and Johansen, 2005], and an expansion of the work comprising a two-level strategy for active steering and adaptive control allocation was presented in [Tjønnås and Johansen, 2010]. In [Schofield,

2008] a convex formulation of the brake-force allocation problem was derived, which extended the work in [Härkegård, 2003]. Further, an optimal yaw control law was discussed in [Esmailzadeh et al., 2003].

11.2 Assumptions

We assume that the vehicle's position and velocity are known. The road geometry and distance to surrounding vehicles are also assumed known; that is, road-preview information is available. In addition, all necessary vehicle parameters are assumed available. The position and velocity can be estimated using wheel-speed measurements, an accelerometer, and measurements from a global positioning system (GPS). The road geometry and information of surrounding vehicles can be estimated using radar measurements and cameras in combination with GPS information. Estimates of vehicle parameters on different surfaces are possible to obtain, as discussed already, [Braghin et al., 2006; Svendenius, 2007; Schofield, 2008]. The results in Chapter 9 in combination with [Ray, 1997; Carlson and Gerdes, 2005; Imsland et al., 2006; Solmaz et al., 2008; Lundquist, 2011] indicate that these assumptions are reasonable.

Regarding actuation, the assumption is that individual wheel torques and steer angle can be controlled. This is a reasonable assumption: ABS, ASR[1], and steer-by-wire have these capabilities [Johansen et al., 2003; Rajamani, 2006; Schofield, 2008; Berntorp, 2008; Tjønnås and Johansen, 2010; Jonasson et al., 2011].

11.3 Vehicle Modeling

The models we use in this chapter are a subset of those discussed in Chapter 4 and used in Chapter 10, and are summarized next. The main differences are that we in this chapter model wheel-torque and steer actuation dynamics, in addition to having individual drive and brake actuation on all wheels.

To account for that the internal vehicle controllers for the steer angle and brake/drive torques do not achieve instantaneous tracking, we incorporate first-order models from reference to achieved value for this dynamics in both the high- and low-level models according to

$$T\dot{\delta} = -\delta + \delta_{\text{ref}}, \tag{11.1}$$

and similarly for the torques, where T in (11.1) is the time constant of the respective control loop. The parameter values that are used here cor-

[1] ASR: Anti-slip regulation.

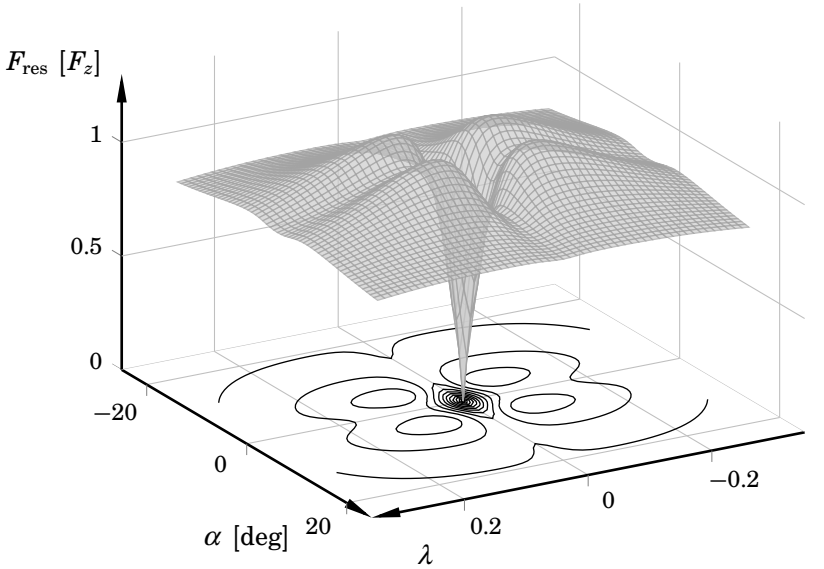

Figure 11.1 Resultant tire force F_{res} for the weighting-functions tire model used in this chapter. The parameters are given in Table A.5. The level curves are shown in the $\lambda\alpha$-plane for $F_{\text{res}} = 0$.

respond to a medium-sized passenger car on asphalt. They differ from those that were used in Chapter 10 in that the considered car is smaller, with less stiff suspension dynamics. In addition, the tire parameters are slightly different. The parameter values are summarized in Tables A.4 and A.5. The resultant force of the tire model is shown in Figure 11.1. When comparing with the parameters used in Chapter 10, see Figure 10.2, there are now four maxima, shifted in α and λ. Hence, if maximum force for some reason is desired, a nonzero α should be chosen.

High-Level Model

For the high-level trajectory generator, we use the nonlinear one-track model (4.12) in combination with the Magic formula (4.8) (see also (10.3)) for computing the tire forces under pure slip, with the longitudinal slip defined as in (4.2). Similar to what was done in Chapter 10, the alternative lateral slip definition

$$\dot{\alpha}\frac{\sigma}{v^x} + \alpha_i := -\arctan\left(\frac{v^y}{v^x}\right)$$

is used. For combined slip, we use the weighting functions (4.11). The motivation for using this vehicle model on the high level is that it provides

a few-state vehicle model that still captures the relevant characteristics of more complex models, as long as the wheel-torque and steer inputs are not used directly as control inputs. There are two reasons for using the weighting functions instead of the model based on the friction ellipse. First, the results in Chapter 10 indicate that the weighting functions give more predictable optimization results. Second, Tables 10.1 and 10.4 suggest that the weighting functions give more predictable—and often also shorter—solution times, which is crucial in an online implementation.

Low-Level Model

The model used in the low-level control layer is the two-track model, which consists of (4.22)–(4.27). Naturally, we use the weighting-functions based tire model here as well, with the same definition of wheel slip as for the high-level model.

11.4 Proposed Control Structure

The suggested control structure is shown in Figure 11.2. It consists of a high-level optimizer that uses information about the road geometry and surrounding vehicles as inputs, as well as estimates of the position $\boldsymbol{p}$, velocity $\boldsymbol{v}$, yaw angle ψ, and yaw rate $\dot{\psi}$. Based on this information the high-level optimizer computes reference trajectories for the position, velocity, yaw angle, and yaw rate. These references are then fed to an NMPC, which computes desired wheel torques $\boldsymbol{\tau}$ and steer angle δ. If the NMPC fails to converge, or if the convergence is deemed too slow, the references are instead sent to an LMPC. The LMPC uses a linearization of the two-track model, and computes desired wheel torques and steer angle. We will next go through the high-level optimizer.

High-Level Trajectory Generation

The goal of the high-level optimizer is to find a path and corresponding state trajectories that minimize a suitable cost function J while staying in lane. In a lane-keeping scenario it is natural to introduce the deviation e from the middle of the lane as a part of the cost function. A common measure of vehicle stability is the vehicle sideslip angle β, defined in (4.13) and repeated for convenience:

$$\beta := \arctan\left(\frac{v^Y}{v^X}\right).$$

A large β indicates poor maneuverability for the average driver. It is traditionally used as a performance measure in ESCs. In theory, an

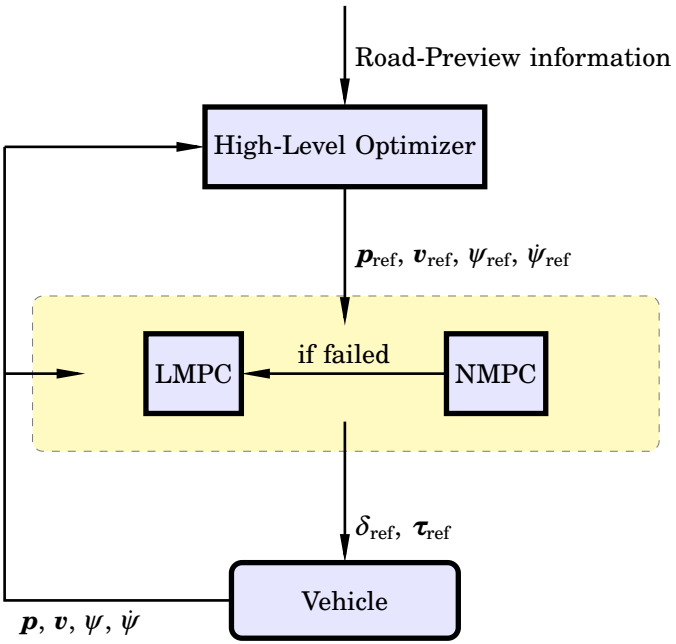

Figure 11.2 The proposed control structure. The high-level optimizer finds velocity and position references over the considered horizon, which is dependent on the road geometry and the available information. The inputs are measurements and/or estimates of the relevant vehicle states. Based on the high-level trajectories, the NMPC aims to find wheel torques and steer angle. If the NMPC fails to converge in time, according to some predefined rule, the references are instead sent to an LMPC. The LMPC uses a linearized version of the two-track model dynamics.

optimization-based safety system does not suffer from a large β, because it by definition finds the optimal solutions and can thus operate in the unstable regions of the tire-ground interaction. In practice, however, model errors will lead to deviations from the computed trajectories. Therefore, it is still desired to keep the vehicle in the small-slip region, if possible. Furthermore, a large β is a measure of driver and passenger discomfort. Thus, we let the cost function depend on the mid-lane deviation e and the body-slip angle β according to

$$\mathcal{J} := \int_{t_0}^{t_f} (\kappa_1 e^2 + \kappa_2 \beta^2)\, \mathrm{d}t,$$

where κ_1 and κ_2 are positive, constant scalar weights, t_0 is the given start time, and t_f is the free final time. This cost function is a tradeoff between

lane following and body slip.

The prediction horizon, or look-ahead, is chosen depending on the road geometry and sample periods. It must be chosen such that the reference trajectories span over the control horizon of the MPC, for the sample periods of the MPC, before a new high-level optimization is performed. However, the horizon cannot be made arbitrarily large. First, a larger horizon implies longer optimization times. Second, if the road curvature is steep, the available look-ahead information prevents a prediction horizon that is too large. In practice the prediction horizon is determined by the terminal constraint on the mass center's position.

Constraints on input torques and the steer angle are also introduced. The single-track dynamics (4.12) in combination with the tire dynamics, consisting of (4.8), (4.11), and the slip definitions, can be written as an index-one system of differential-algebraic equations (DAEs) (in a similar fashion to (10.1)) according to

$$\boldsymbol{f}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{w}, \delta_{\text{ref}}, \tau_{f,\text{ref}}, \tau_{r,\text{ref}}) = 0, \tag{11.2}$$

where $\boldsymbol{x}$ contains the differential (state) variables, $\boldsymbol{w}$ contains the algebraic variables, and $\tau_{f,\text{ref}}$, $\tau_{r,\text{ref}}$ are the desired wheel drive/brake torques on the front and rear axle, respectively. The dynamic optimization problem is then formulated over the time horizon $t \in [t_0, t_f]$, with free final time, as

$$\begin{aligned}
\underset{\delta_{\text{ref}},\tau_{f,\text{ref}},\tau_{r,\text{ref}}}{\text{minimize}} \quad & \int_{t_0}^{t_f} (\kappa_1 e^2 + \kappa_2 \beta^2)\, \mathrm{d}t \\
\text{subject to} \quad & \boldsymbol{f}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{w}, \delta_{\text{ref}}, \tau_{f,\text{ref}}, \tau_{r,\text{ref}}) = \boldsymbol{0} \\
& |\tau_{i,\text{ref}}| \leq \tau_{i,\max}, \ \forall i \in \{f, r\} \\
& |\delta_{\text{ref}}| \leq \delta_{\max} \\
& \|\boldsymbol{p}(t_f) - \boldsymbol{p}_f\| \leq \epsilon \\
& \boldsymbol{\Gamma}(\boldsymbol{p}) \leq \boldsymbol{0} \\
& \boldsymbol{x}(t_0) = \boldsymbol{x}_0.
\end{aligned} \tag{11.3}$$

where $\boldsymbol{x}_0$ is the initial state and $\boldsymbol{p}_f$ is the terminal position constraint. Further, $\boldsymbol{\Gamma}(\boldsymbol{p})$ is a mathematical description of the road constraint for the vehicle's mass center and its endpoints. Note that it is possible to express collision-avoidance tasks in $\boldsymbol{\Gamma}(\boldsymbol{p})$. We have introduced a slack ϵ in the terminal constraint for the position, since exact tracking is typically not crucial. Moreover, it improves convergence since exact terminal constraints are harder to fulfill. To generate an initial guess for the nonconvex problem (11.3), we simulate the system with a constant steer angle and zero input torques.

When (11.3) has been solved, the optimal trajectories for $\boldsymbol{p}$, $\boldsymbol{v}$, ψ, $\dot{\psi}$, are sent to the low-level layer for allocation to the wheel and steer actuators.

The optimization problem (11.3) is solved repeatedly, with a sampling period of $T_{s,h}$ s. The idea is that a new optimization is started directly after sending the trajectory references to the low-level layer.

Low-Level Control-Input Allocation

The aim of the low-level controller is to track the state references, which are computed by the high-level trajectory generator. This is done by allocating appropriate wheel-torque and steer-angle references to the vehicle's internal controllers using an MPC.

To this end, introduce the notation

$$\boldsymbol{r} = \begin{bmatrix} \boldsymbol{p}_{\text{ref}}^{\text{T}} & \boldsymbol{v}_{\text{ref}}^{\text{T}} & \psi_{\text{ref}} & \dot{\psi}_{\text{ref}} \end{bmatrix}^{\text{T}} \tag{11.4}$$

for the (time varying) references to the MPC. Further, let

$$\boldsymbol{u} = \begin{bmatrix} \delta_{\text{ref}} & \tau_{1,\text{ref}} & \tau_{2,\text{ref}} & \tau_{3,\text{ref}} & \tau_{4,\text{ref}} \end{bmatrix}^{\text{T}} \tag{11.5}$$

denote the control input vector to the vehicle, which the MPC will compute. The references in (11.4), on the other hand, are computed from the high-level optimal-control problem. Note that there are four torque references in (11.5), since we use a two-track model for torque allocation.

The chassis dynamics for the two-track model and the tire dynamics are, in a similar way to (11.2), formulated as the DAE system

$$\boldsymbol{f}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u}) = 0, \tag{11.6}$$

where $\boldsymbol{x}$ are the state variables for the combined two-track and tire model and $\boldsymbol{w}$ are the corresponding algebraic variables. Note that the function and variables in (11.6) are not the same as in (11.2).

For the low-level control-input allocation, the aim is to track the references computed by the high-level trajectory generator. Hence, the reference vector (11.4) must be a part of the cost function. For better tracking, it is typically advantageous to include a specific cost term on the terminal position. In addition, we include a terminal constraint on the position. Tracking of state references is not the only objective, since driver comfort also needs attention. This is accommodated by introducing a penalty on the control signals as well. The low-level NMPC problem formulation is

in each time step k stated as (with a slight abuse of notation)

$$\underset{\boldsymbol{u}}{\text{minimize}} \quad \int_{\bar{t}_0}^{\bar{t}_f} \left(\|\boldsymbol{x} - \boldsymbol{r}\|^2_{\boldsymbol{W}_x} + \|\boldsymbol{u}\|^2_{\boldsymbol{W}_u} \right) \mathrm{d}t + \|\boldsymbol{p}(\bar{t}_f) - \bar{\boldsymbol{p}}_f\|^2_{\boldsymbol{W}_f} \tag{11.7a}$$

$$\text{subject to} \quad \boldsymbol{f}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u}) = \boldsymbol{0} \tag{11.7b}$$

$$-\boldsymbol{u}_{\max} \leq \boldsymbol{u} \leq \boldsymbol{u}_{\max} \tag{11.7c}$$

$$\|\boldsymbol{p}(\bar{t}_f) - \bar{\boldsymbol{p}}_f\| \leq \epsilon \tag{11.7d}$$

$$\boldsymbol{\Gamma}(\boldsymbol{p}) \leq \boldsymbol{0} \tag{11.7e}$$

$$\boldsymbol{x}(\bar{t}_0) = \bar{\boldsymbol{x}}_0 \tag{11.7f}$$

where $\bar{t}_0 = t_0 + (k+1)T_{s,l}$ is the initial time, $\bar{t}_f = t_0 + (k + H_l)T_{s,l} \leq t_f$ is the final time, $T_{s,l}$ is the sampling period of the MPC, and H_l is the prediction horizon of the MPC. Moreover, $\boldsymbol{W}_x$, $\boldsymbol{W}_u$, and $\boldsymbol{W}_f$ in (11.7a) are the weight matrices, $\|\boldsymbol{x}\|^2_{\boldsymbol{W}_x} = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{W}_x\boldsymbol{x}$, $\bar{\boldsymbol{p}}_f$ in (11.7a) and (11.7d) is the position reference at time $\bar{t}_f$, $\boldsymbol{u}_{\max}$ contains the input-reference bounds, and $\bar{\boldsymbol{x}}_0$ in (11.7f) is the initial state vector at time $\bar{t}_0$, given by estimates and/or measurements. Note that the path constraints are also included at the low level, in (11.7e). In each time step, we solve (11.7) with the constraint that the control-input vector is piecewise constant over the sampling periods. To generate an initial guess for (11.7), we simulate the system with optimal control inputs from the previous time step. When (11.7) has been solved, the control inputs from the first sampling period are sent to the internal vehicle controllers.

The highly nonlinear dynamics (11.7b) will sometimes cause the convergence of (11.7) to be too slow, or even fail. To address this, we design an additional control-input allocator, an LMPC, which is based on repeated linearizations of the dynamics. The resulting LMPC formulation becomes a quadratic program with linear dynamics on the form (2.10), for which several efficient solvers exist. To reduce the problem size and thereby increase efficiency, we start with noting that (11.6) is a DAE system that can be reformulated as an ordinary differential equation (ODE) system. The algebraic variables $\boldsymbol{w}$ arise from the longitudinal slip and the tire-force equations (4.8) and (4.11). These can be solved for, see Section 11.5, and the result is the set of ODEs

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}). \tag{11.8}$$

By discretizing (11.8) using a forward-difference approximation, we end up with the system

$$\boldsymbol{x}_{k+1} = \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k). \tag{11.9}$$

Now, by introducing

$$\boldsymbol{A}_k = \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_k, \boldsymbol{u}_k}, \quad \boldsymbol{F}_k = \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}}\right|_{\boldsymbol{x}_k, \boldsymbol{u}_k}, \tag{11.10}$$

the resulting LMPC formulation becomes

$$\begin{aligned}
\underset{\mathcal{U}_k}{\text{minimize}} \quad & \sum_{m=1}^{H_l} \|\boldsymbol{x}_{k+m} - \boldsymbol{r}_{k+m}\|^2_{\boldsymbol{W}_x} + \sum_{m=1}^{H_l-1} \|\boldsymbol{u}_{k+m}\|^2_{\boldsymbol{W}_u} + \|\boldsymbol{p}_{k+H_l} - \bar{\boldsymbol{p}}_f\|^2_{\boldsymbol{W}_f} \\
\text{subject to} \quad & \boldsymbol{x}_{k+m+1} = \boldsymbol{A}_{k+m}\boldsymbol{x}_{k+m} + \boldsymbol{F}_{k+m}\boldsymbol{u}_{k+m} \\
& \boldsymbol{u}_{\min} \leq \boldsymbol{u}_{k+m} \leq \boldsymbol{u}_{\max} \\
& \Delta\boldsymbol{u}_{\min} \leq \boldsymbol{u}_{k+m} - \boldsymbol{u}_{k+m-1} \leq \Delta\boldsymbol{u}_{\max} \\
& m = 1, \ldots, H_l - 1 \\
& \boldsymbol{x}_{t_0} = \bar{\boldsymbol{x}}_0,
\end{aligned} \tag{11.11}$$

where $\mathcal{U}_k := \{\boldsymbol{u}_{k+1}, \ldots, \boldsymbol{u}_{k+H_l-1}\}$ is the set of control inputs to be determined, $\bar{\boldsymbol{p}}_f$ is the terminal position reference, and $\bar{\boldsymbol{x}}_0$ is the measured and/or estimated initial position. Compared with (11.7), (11.11) involves a discretized and linearized version of the dynamics, which introduces approximation errors. Hence, (11.11) is only executed when (11.7) fails to converge or when the convergence rate is too slow. The weight matrices in (11.11) need not be the same as those in (11.7). Note that the control input at time step $k+1$ is applied to the vehicle, because the computation time is in the order of the sampling period.

The complete algorithm is summarized in Algorithm 11.1, where `conv` is an indicator of whether the NMPC has converged or not and where H_h is the resulting horizon of the high-level optimal-control problem.

Algorithm 11.1
Input: State estimates $\boldsymbol{x}(t_0)$ and road-preview information.
1: Solve (11.3) and form

$$\boldsymbol{r} = \begin{bmatrix} \boldsymbol{p}^{\mathrm{T}} & \boldsymbol{v}^{\mathrm{T}} & \psi & \dot{\psi} \end{bmatrix}^{\mathrm{T}}$$

for the time period $t \in [t_0, t_f]$.
2: Set $j = 1$.
3: **while** $j \leq \lceil T_{s,h}/T_{s,l} \rceil$ **do**
4: Solve (11.7).
5: **if** `conv` $\neq$ True **then**
6: Compute (11.9) and (11.10).
7: Solve (11.11).
8: **end if**
9: Apply $\boldsymbol{u}(t_0 + (k+j)T_l)$.
10: Set $j = j + 1$.
11: **end while**
12: Go to step 1.

Line 1 in Algorithm 11.1 executes with the sampling period $T_{s,h}$ s, and the while-loop executes with the sampling period $T_{s,l}$ s. The convergence condition `conv` on line 5 in Algorithm 11.1 is based on an analysis of mean convergence time of the LMPC: Assume that the mean solution time of the LMPC is h s. Then the NMPC is terminated and conv is set to false if the execution time is larger than $T_l - h + \Delta$, where Δ is a slack that is introduced to provide robustness for variations in execution time.

Note that the state estimates are only used in the first iteration of the high-level optimization problem. Thereafter, the initial state is chosen as the state of the previous optimal solution at the next sampling instant (i.e., the optimal solution at $T_{s,h}$ s forward in time). This approach assumes that the low-level controller manages to keep the vehicle close to its references, and implies that the high-level state trajectories are continuous.

11.5 Implementation

Implementation of the high-level trajectory generator and the NMPC are similar to what was done in Chapter 10 (page 213), and is only briefly described here.

The high-level trajectory generation and the NMPC are implemented using the open-source software platform JModelica.org [Åkesson et al., 2010]. The DAE-constrained optimization problems (11.3) and (11.7) are first transformed into ODE-constrained optimization problems using the procedure described in [Magnusson et al., 2014]. They are then discretized using first-order direct local Radau collocation with the implementation described in [Magnusson and Åkesson, 2012], where Lagrange polynomials are used for representation of the state profiles in each element and the location of the collocation points are chosen as the corresponding Radau points. The results in this chapter are obtained by using $N_e = 10$ elements and $N_c = 2$ collocation points for the high-level trajectory generator, and $N_e = 5$, $N_c = 1$ for the NMPC. Since the NMPC has piecewise-constant control inputs, the number of collocation points can in general be kept small compared with the cases for continuous control inputs.

The resulting nonlinear program (NLP) is solved using IPOPT [Wächter and Biegler, 2006] in combination with the linear solver MA27 [HSL, 2014]. JModelica.org uses CasADi [Andersson et al., 2012] to obtain the relevant first- and second-order derivatives of the NLP functions.

The symbolic transformations to ODE-constrained optimization problems lead to drastically reduced number of system variables and hence improved convergence speed, as the algebraic variables are eliminated from the equation system. The high-level optimal-control problem contains ap-

proximately 700 variables (1250 before the symbolic transformations) in total, whereas the low-level problem contains 400 variables. Moreover, the transformations provide solution times that enable online implementations. For the considered examples, the convergence is much more robust compared with using a DAE formulation. This is important for an online implementation. Note, however, that the framework is partly implemented in Python, and is as such in the current state not optimized for speed.

The LMPC is implemented in C using CVXGEN [Mattingley and Boyd, 2012] and exploits sparsity. To link the LMPC with Python, the ctypes library [Python Software Foundation, 2014] is used.

11.6 Simulation Study

To show how the method performs under aggressive maneuvering, this section contains simulation results from a road segment with a road curvature radius of 30 m. The road is 4 m wide. The initial velocity is $v_0 = 70$ km/h (approximately 19.5 m/s). The maximum entrance velocity to be able to stay in the lane is approximately 75 km/h [Lundahl et al., 2014].[2] The input constraints are set to

$$\begin{aligned} \delta_{\max} &= 0.5, \\ \tau_{i,\max} &= F_i^z \mu_x / R_w, \quad \forall i \in \{f, r\} \text{ or } \{1, 2, 3, 4\}. \end{aligned}$$

This corresponds to an all-wheel drive vehicle. The constraint values are similar to the constraints (10.7) in Chapter 10. Note that we model actuator dynamics. Thus, no constraints on the derivatives on the inputs are used. This differs to the optimization problem (10.6) in Chapter 10.

For the high-level optimization problem (11.3), the weights are $\kappa_1 = 1$ and $\kappa_2 = 15$, leading to that large body slip is penalized more than deviation from the mid-road segment. The sampling period of the high-level optimal-control problem is set to $T_{s,h} = 0.4$ s. The look-ahead used in this example is between 10–20 m, corresponding to a resolution in position of approximately 0.5–1 m for the high-level control problem.

For the low-level MPCs, the prediction horizons are $H_l = 5$ samples and the sampling period is $T_{s,l} = 0.04$ s. The choice for when to terminate the NMPC is decided based on estimations of how long execution time the LMPC needs to converge. With the settings used in this example, the LMPC typically converges within 10 ms (corresponding to 15–20 iterations). Thus, when the NMPC has been executing more than approximately 30 ms without converging, the LMPC is turned on. In the actual implementation, however, to facilitate reproducibility we instead use the number of iterations as the termination criterion.

[2] Achieved by solving an optimization problem where the entrance speed is maximized.

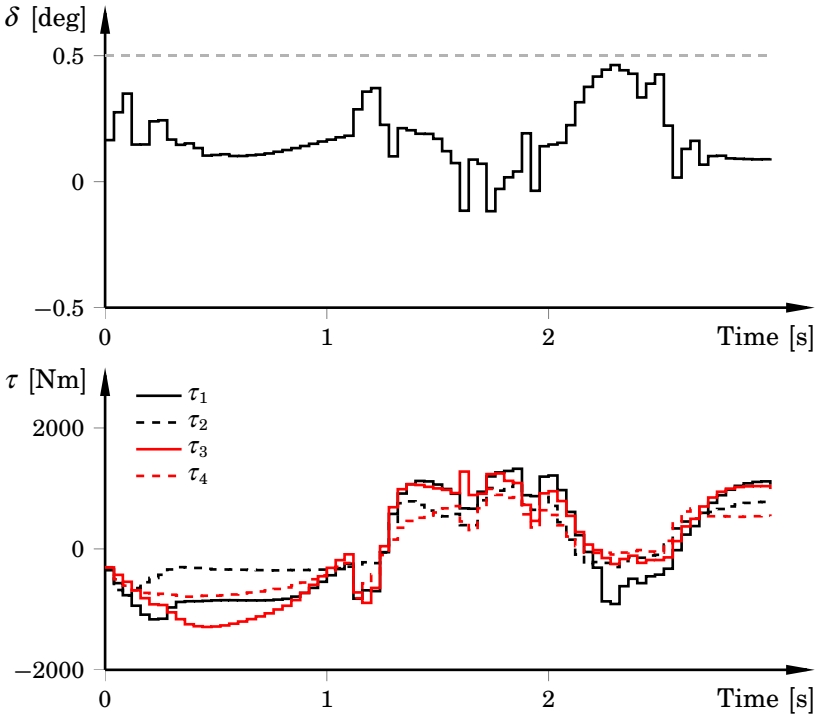

Figure 11.3 Control inputs as computed from the low-level control architecture. The controller tuning is rather conservative, which means that driver comfort is more emphasized than perfect trajectory tracking.

Results

The control signals for the maneuver are visualized in Figure 11.3. The choice of controller parameters is a tradeoff between driver comfort and reference tracking, and here the emphasis is on driver comfort. Figure 11.4 shows which of the MPCs that is active. As seen, the NMPC converges in the vast majority of the iterations. Nevertheless, there are 13 cases where the NMPC fails to converge in time for this particular scenario. By investigating Figures 11.3 and 11.4, it is clear that the NMPC gives rise to smoother control signals. The horizons and sampling periods are the same in both MPCs, but the tuning is different. In this example, both MPCs have been tuned with driver comfort in mind. Still, the LMPC often (but not always) computes more aggressive control signals. This probably owes to the choice of discretization method in (11.9) and the linearizations in (11.10).

Figure 11.5 contains the position references and actual positions.

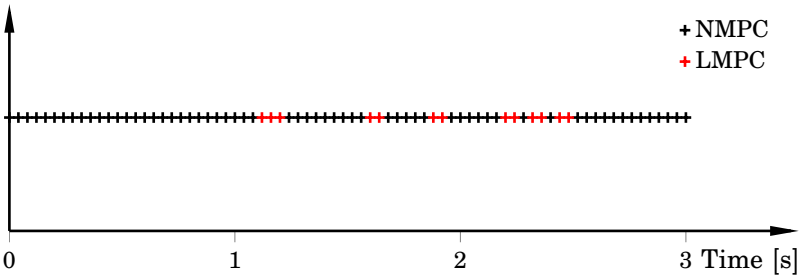

Figure 11.4 The figure shows when which MPC is active. The NMPC converges before the computation-time limit in 62 out of the approximately 75 optimizations.

There is a steady-state error in the position. This is caused by the controller tuning, which penalizes large control signals more than it aims to achieve good tracking. Moreover, the terminal position constraint slack in the NMPC is $\epsilon = 1$, which is conservative. The slack implies that the optimal solution allows for a predicted error of 1 m.

Figure 11.6 shows some of the state-trajectory references that often are connected to vehicle stability. These references are tracked closely for most of the maneuver, except for the discrepancy in ψ between approximately 1.8–2 s. The discontinuities in the references are an effect of the data extraction.

Aggressive Tuning

To show that the proposed approach is able to track the position references accurately, Figure 11.7 displays the position references and the actual positions with a more aggressive controller tuning. The vehicle now tracks the position references with high precision. The difference compared with Figure 11.5 is that the bound for the terminal constraint (11.7d) is now $\epsilon = 0.1$, whereas it in the previous example was $\epsilon = 1$. Moreover, the weights on the control inputs are decreased. This is, of course, a tradeoff with aggressive control signals and thus comfort. Depending on if the system is truly autonomous or if it should only intervene when absolutely necessary, different tunings should be used. The controller tuning could also be dependent on predictions of how severe the situation is.

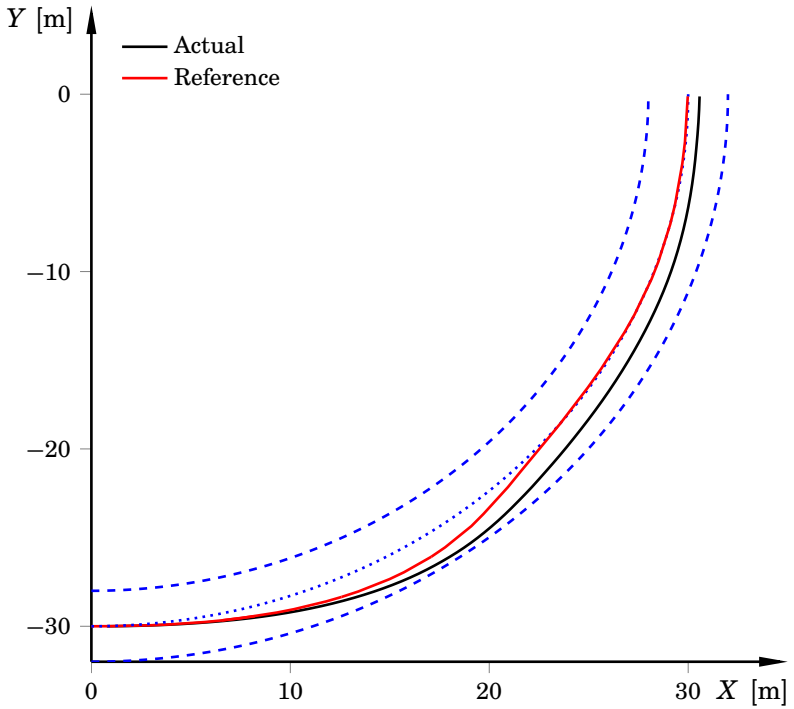

Figure 11.5 Position references (red) from the high-level optimizer and actual positions (black). The mid-line segment is shown as blue dots, and the road constraints are the blue dashed lines. The position error is approximately 1 m at most. Note that the controller tuning is not done with perfect tracking in mind.

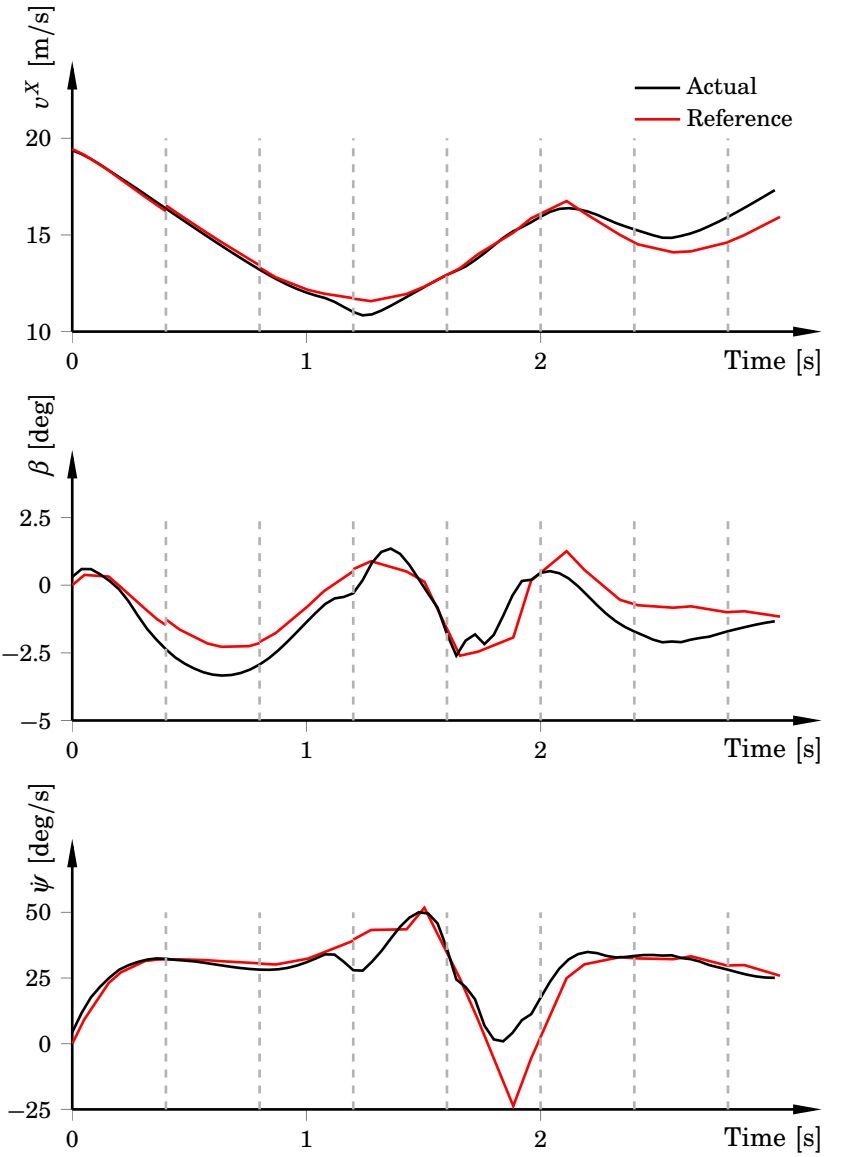

Figure 11.6 State-trajectory references (red) and actual values (black). The sampling instants of the high-level trajectory generator are shown as dashed gray vertical lines. The low-level MPCs track the high-level references well in most parts. Note that the reference tracking is a tradeoff with driver comfort, and this tuning corresponds to conservative tracking. The discontinuities in the references are an effect of the data extraction.

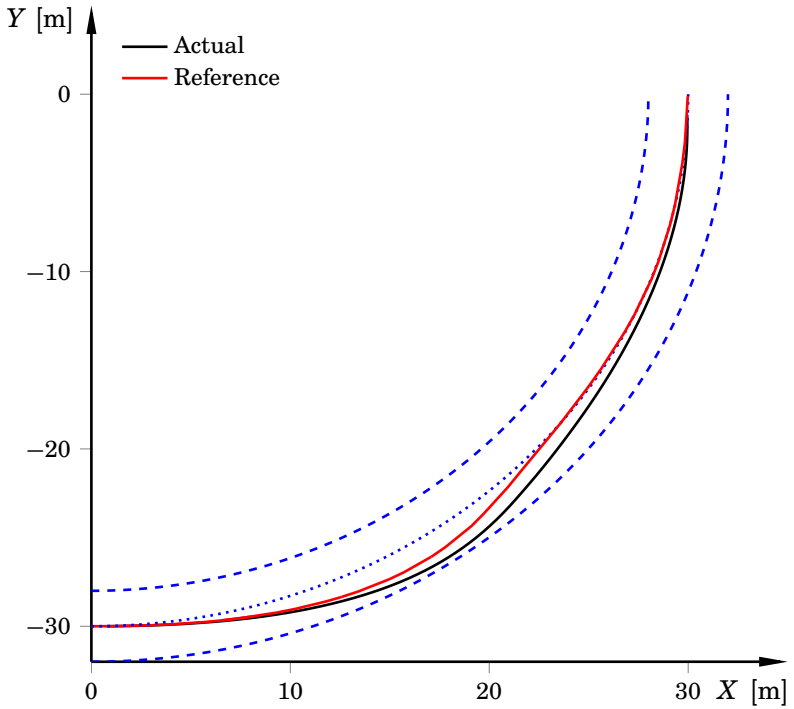

Figure 11.7 Position references (red) from the high-level optimizer and actual positions (black). When compared with Figure 11.5, the reference trajectory is more closely tracked, where the only deviation is in the middle part of the maneuver. This is, of course, a tradeoff with aggressive control signals and thus comfort. The remaining error is partly owing to the model discrepancies between the high- and low-level models. However, it is also dependent on the prediction horizon and the discretization resolution.

11.7 Discussion

The proposed control structure has been tested using various initial velocities and controller parameters. The typical convergence time for the high-level problem was approximately 100–150 ms, corresponding to 7–10 Hz. For other scenarios, the computations might take longer time. To account for this, we used an update rate of 2.5 Hz in the considered scenario. For the different tested configurations, if using a feasible entry speed, the high-level trajectory generator never failed to converge. In a physical setup, however, it has to be accompanied with an additional reference generator, similar to what we have implemented for the low-level controller. This could, for example, be done using a point-mass description of the vehicle, leading to a convex optimization problem.

The low-level controller is executed with a sample rate of 25 Hz. This corresponds well to the average execution times for the combined MPCs. An easy-to-use, general-purpose implementation was used. More specialized implementations are possible, and it is highly likely that the solution times can be decreased significantly. A way to reduce the computation time further is to generate C code for evaluation of the NLP functions and their derivatives, something which is not done in the current implementation. Based on experience, this can reduce the solution time with approximately 30% on average, and sometimes with as much as 50%. Also the LMPC can be improved in terms of execution time—for example, by using algorithms custom-tailored for LMPC [Giselsson, 2014].

The parameter tunings of the MPCs differ slightly. The LMPC relies on repeated linearizations of the dynamics, and the predictions over the horizon will thus be less reliable than in the NMPC. Using the same tuning for the LMPC as for the NMPC can cause unnecessarily aggressive control behavior, and perhaps even instability. Hence, the controller tuning is preferably set more conservatively for the LMPC than what is done in the NMPC.

Autonomous or Semiautonomous Vehicles

We have not discussed whether the system is supposed to be active always (i.e., a truly autonomous system) or only intervene when deemed necessary. If the system is assumed to be autonomous, the algorithm basically consists of having the MPCs tracking the high-level trajectories in the same way as was done in Section 11.6.

In the case of a system that only intervenes when necessary, it first has to be decided what is meant by necessary. In traditional ESCs, driver commands are compared with a linearized one-track model. When the driver's intentions differ significantly from the model behavior, the ESC is activated to maintain vehicle stability [Berntorp, 2008]. This idea can also

be used for the proposed architecture. The high-level trajectory generator minimizes a combination of deviation from the mid-road segment and body slip. This is also, at least implicitly, what the average driver aims to do. Moreover, the average driver is typically most comfortable with the driving experience when the body slip is small.[3] Hence, the driver intention can also here be compared with a reference model, in this case the high-level trajectory generator.

Future Work

It remains some work before the architecture can be used online in a physical setup. The first problem is that the framework used for the high-level trajectory generator and NMPC is neither designed for real-time scenarios (e.g., the memory footprint is relatively large) nor optimized for execution speed in the current implementation. Despite this, as we saw in Section 11.6, the current solver speed is feasible for online implementation, and as mentioned, there are quite large speed improvements to be made (30–50%) with relatively little effort.

Another issue is that the high-level trajectory generator is based on nonconvex optimization, where nonlinear equation systems have to be solved. As is the case for NMPC, this can sometimes cause convergence failure, or at least slow convergence. Hence, also the high-level layer should be accompanied with a backup algorithm that computes feasible trajectories when necessary. One solution is to compute the trajectories based on geometric reasoning and closed-loop simulation of the one-track model, which provides feasible trajectories. Another possibility is to use a point-mass representation of the vehicle. This allows for a convex problem formulation, and should be easy to solve in a timely manner [Gao et al., 2010]. However, such a formulation should only be used in rare cases, because a point-mass approximation is only suitable in (nearly) steady-state driving conditions.

11.8 Summary

This chapter presented a novel two-level approach to robust, optimal trajectory generation for vehicles. The high-level layer computes state-trajectory references. These references are then used in the low-level controller, which allocates control inputs. The first novelty was the use of a nonlinear vehicle model with tire modeling in the optimization problem at the high level. This provides for better coupling with the low-level

[3] With small β we mean values somewhere in the range of approximately 3–5 deg, depending on road surface and vehicle type.

control-input allocator, especially for aggressive maneuvering. The second novelty was a low-level control structure that uses an NMPC for allocating the torques to the wheels, incorporating a nonlinear two-track model with suspension dynamics as well as rotations in space. To increase robustness we combined this with an LMPC, to be used in those cases when the NMPC fails to find a solution within a prescribed time limit. Results showed that the control structure is robust and that viable computation times are achieved, even when using a general framework for implementation.

12

Path Tracking and Obstacle Avoidance: A Design Flow

Path planning and trajectory generation have been studied for several decades in robotics. Trajectory generation deals with finding state trajectories between two given points in space. In some cases only the endpoints are of interest in the task specification. However, in many situations the path between the points is important in itself—for example, in laser cutting, milling, and for transporting goods in factories using mobile robots. It is common that a high-level path planner provides the geometric path. Thenceforth, a subsequent trajectory generation is made. To this purpose, the *decoupled approach* to trajectory generation has been established in literature [LaValle, 2006; Van Loock et al., 2013]. Naturally, reliable path tracking is essential. In addition, to react upon environmental uncertainties, real-time obstacle avoidance is a desired feature of the path tracker [Khatib, 1986]. As mentioned already in Section 1.2, for high productivity a near time-optimal solution to the path-tracking problem is desired.

This chapter describes a hierarchical design flow for performing online, minimum-time trajectory generation for four-wheeled vehicles, combined with real-time obstacle avoidance. A hierarchical control structure, which combined nonconvex and convex optimization problems, was used in the previous chapter for feedback-based control in automotive applications. There, the motivation for using nonconvex methods was that the relatively complex vehicle dynamics do not allow for a convex formulation without resorting to severe model simplifications. Mobile robots have similar kinematics as automotive systems. However, since the velocities typically are smaller in robotic applications in combination with the exclusion of complex suspension systems, the dynamics can be accurately captured using simple models that, for example, do not model load transfer or nonlinear tire characteristics. These simplifications enable convex and efficient formulations of the optimal-control problems. Another dif-

ference with the method in Chapter 11 is that we in this chapter assume that the path is given. The control structure is validated both in simulation and in an experimental setup, where the simulation study focuses on the impact of the model simplifications that are made.

12.1 Motivation

Using a dynamic model of the robot, [Verscheure et al., 2009c] showed how the time-optimal path-tracking problem for stationary robot manipulators, see [Shin and McKay, 1985], can be reformulated as a convex problem. In this chapter we show that the theory in [Verscheure et al., 2009c] can be extended to the case of four-wheeled vehicles, especially pseudo-omnidirectional mobile robots,[1] which have fundamentally different dynamics and kinematics compared with stationary robots. In particular, scenarios are considered where a nominal path to be tracked is planned using the available static map information. High-level feedback from the estimated position and orientation of the vehicle is achieved using linear MPC [Mayne et al., 2000; Maciejowski, 2002]. Moreover, obstacles that are detected during runtime are avoided by using online local reoptimizations of the trajectories with a scheme that is integrated in the MPC.

Another feature with the method proposed here is the implementation in an integrated framework. To reduce the complexity of the programming phase, large efforts have been put in developing software services for mobile robots. Two examples of software services are the Robot Operating System (ROS) [ROS, 2014] and Orocos [Orocos, 2014]. Our implementation utilizes ROS for easy integration with different algorithms, such as simultaneous localization and mapping (SLAM) algorithms and the internal robot controllers.

Trajectory generation and online collision avoidance for mobile robots have been studied before, see [Khatib, 1986; Quinlan and Khatib, 1993; Fox et al., 1997; Qu et al., 2004; Iagnemma and Dubowsky, 2004; Choi et al., 2009; Anderson et al., 2012] for a few examples. However, there seems to be few, if any, approaches to time-optimal path tracking based on convex optimization for wheeled vehicles. In addition, some of the previously proposed methods are only based on the kinematic relations of the robot, and do not consider a nonlinear dynamic model incorporating friction. Moreover, differential-drive mobile robots are often considered in literature, yielding less complex models. The characteristics of differential-drive robots are considerably different compared with four-wheeled vehicles with independent steering; for example, significant constraints are

[1] The mobile robot in Chapter 6 is an example of a pseudo-omnidirectional mobile robot.

enforced on the maneuverability and the required slip modeling for the former. The trajectory-tracking controller in this chapter incorporates both predicting and optimizing characteristics in a convex optimization formulation, allowing fast real-time solutions even when operating at high sample frequencies. In addition, constraints derived from dynamic obstacles and map information can be introduced during runtime, owing to the receding-horizon principle of the MPC.

12.2 Previous Work

The solution to the time-optimal path-tracking problem was derived already in the 1980's [Bobrow et al., 1985; Shin and McKay, 1985; Pfeiffer and Johanni, 1987] for industrial manipulators. By utilizing the special structure of the dynamic model and a parametrization of the spatial path in a path coordinate, the minimum-time problem was reformulated to an optimal-control problem with fixed horizon for the independent variable and significantly reduced number of states. Further investigations, for example, with respect to model parameter uncertainties, were made in [Shin and McKay, 1987; Chen and Desrochers, 1989; Shiller and Lu, 1992]. Note that solutions to these optimal-control problems were found offline. The obtained trajectories were combined with feedback, thus taking care of model uncertainties and disturbances in the online task execution [Dahl and Nielsen, 1989; Dahl, 1992].

Utilizing recent advancements in convex optimization [Boyd and Vandenberghe, 2008], [Verscheure et al., 2009c] showed how the time-optimal path-tracking problem for stationary industrial manipulators can be solved efficiently using convex optimization techniques. However, certain constraints on the dynamic model, such as neglecting the viscous friction in the joints, need to be imposed. [Verscheure et al., 2009b] proposed an online path-tracking algorithm. In this algorithm, the path to be tracked is delivered online to the trajectory generator, which computes the trajectories in real time. Further investigation of the method in [Verscheure et al., 2009c] was presented in [Lipp and Boyd, 2014], and [Ardeshiri et al., 2011] outlined a method for approximation of velocity-dependent constraints in the convex optimization formulation. This has been further elaborated on for convex-concave constraints using sequential convex programming in [Debrouwere et al., 2013] and using a convex relaxation in [Reynoso-Mora et al., 2013].

MPC for trajectory tracking is not new, see, for example, [Kanjanawanishkul and Zell, 2009; Howard et al., 2009; Klančar and Škrjanc, 2007]. However, these approaches do not combine the MPC with a time-optimal trajectory-generation procedure and the trajectories are assumed known

a priori. Nonlinear MPC for obstacle avoidance in the case of autonomous vehicles has been investigated in [Norén, 2013]. Another approach to trajectory generation in robotics is the Reflexxes motion library, see [Kröger, 2011].

12.3 Modeling

The enabler for the convex, high-level trajectory generator is a dynamic model that captures the relevant characteristics while not being too complex. Neither the one-track model nor the two-track model in Section 4.2 fulfill the necessary criteria. The one-track model fails in that it models the vehicle as having two wheels, which is not enough for the considered application. Moreover, the two-track model is too complex. Hence, we in this section derive a third chassis model. In the spirit of the original formulations of the path-tracking problem, the derivation is done using an Euler-Lagrange modeling approach. The section also contains a derivation of the inverse kinematics.

Deriving the Chassis Model

For modeling the vehicle, the first assumption is that the motor dynamics can be neglected. This is motivated by that the motor dynamics is inherently fast compared with the other dynamics of the vehicle. Also, the motor-current controllers, which operate at the lowest level, ensure fast torque tracking. We assume planar movement, thus neglecting vertical dynamics and rotational coupling such as roll dynamics. Figure 12.1 contains a sketch of the vehicle.

Given a set of *generalized coordinates* $\{q_i\}_{i=1}^n$, the Euler-Lagrange equations, [Spong and Hutchinson, 2006], state that

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathscr{L}}{\partial \dot{q}_i} - \frac{\partial \mathscr{L}}{\partial q_i} = \mathscr{F}_i, \quad \forall i \in \{1, \ldots, n\}, \tag{12.1}$$

where $\mathscr{F}_i$ is the ith external torque and $\mathscr{L} = \mathscr{T} - \mathscr{V}$ is the difference between the kinetic and potential energy (the Lagrangian). We assume planar movement, implying that $\mathscr{V} = 0$. The kinetic energy is the sum of the translational and rotational velocities in the system—that is,

$$\mathscr{T} = \frac{1}{2}\left(m\left((\dot{p}^X)^2 + (\dot{p}^Y)^2\right) + I_{ZZ}\dot{\psi}^2 + \sum_{j=1}^{4}\left(I_\vartheta \omega_j^2 + I_\delta \dot{\delta}_j^2\right)\right), \tag{12.2}$$

where I_ϑ is the wheel moment of inertia in the drive direction and I_δ is the wheel moment of inertia in the steer direction, see Figure 12.2. A

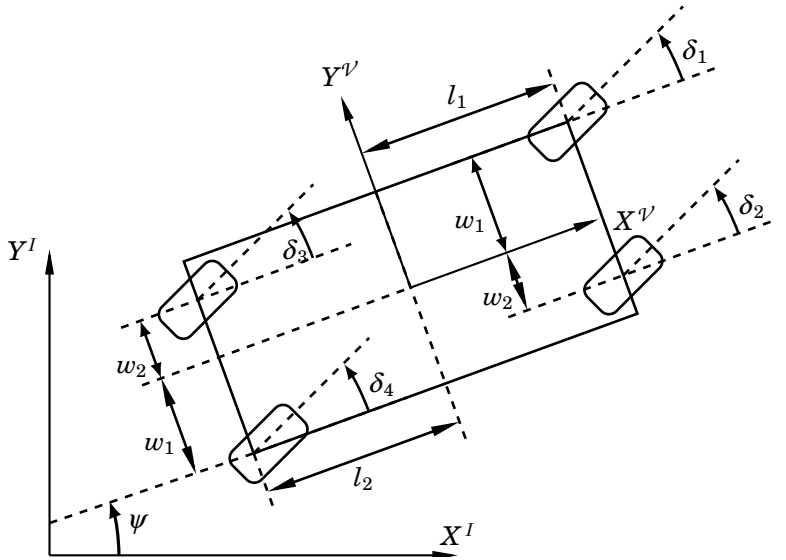

Figure 12.1 The four-wheeled vehicle and the coordinate systems used for modeling. The vehicle has inertia I_{ZZ} about the $Z^{\mathcal{V}}$-axis and mass m.

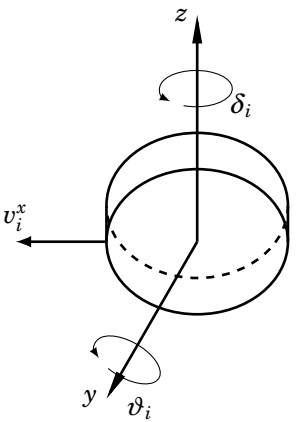

Figure 12.2 An illustration of the wheels' rotational degrees of freedom. The drive angle is introduced. Since ω denotes the wheel's angular velocity in the drive direction, $\dot{\vartheta} = \omega$.

natural choice of generalized coordinates would be

$$\boldsymbol{q} = \begin{bmatrix} p^X & p^Y & \psi & \vartheta_1 & \cdots & \vartheta_4 & \delta_1 & \cdots & \delta_4 \end{bmatrix}^{\mathrm{T}}.$$

A more convenient choice, however, is to express the dynamics in the vehicle's coordinate system using the base velocities, because velocity references and torque commands are by convention given in this frame. To this end, we make use of the following set of generalized coordinates:

$$\boldsymbol{q}^* = \begin{bmatrix} v^{\mathcal{V},X} & v^{\mathcal{V},Y} & \psi & \vartheta_1 & \cdots & \vartheta_4 & \delta_1 & \cdots & \delta_4 \end{bmatrix}^{\mathrm{T}},$$

where a coordinate transformation between $\dot{\boldsymbol{p}} = \boldsymbol{v}$ and $\boldsymbol{v}^{\mathcal{V}}$ is given by

$$\boldsymbol{v}^{\mathcal{V}} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \dot{\boldsymbol{p}} = \boldsymbol{R}_{\mathcal{V}}^{I} \dot{\boldsymbol{p}}. \tag{12.3}$$

The following transformation from global to local coordinates can be established:

$$\begin{aligned} \frac{\partial \mathscr{T}}{\partial \dot{p}^X} &= \frac{\partial \mathscr{T}}{\partial v^{\mathcal{V},X}} \cos(\psi) - \frac{\partial \mathscr{T}}{\partial v^{\mathcal{V},Y}} \sin(\psi) \\ \frac{\partial \mathscr{T}}{\partial \dot{p}^Y} &= \frac{\partial \mathscr{T}}{\partial v^{\mathcal{V},X}} \sin(\psi) + \frac{\partial \mathscr{T}}{\partial v^{\mathcal{V},Y}} \cos(\psi). \end{aligned} \tag{12.4}$$

Insertion of (12.4) into (12.1) and premultiplying with

$$\boldsymbol{R}_{I}^{\mathcal{V}} = (\boldsymbol{R}_{\mathcal{V}}^{I})^{-1} = (\boldsymbol{R}_{\mathcal{V}}^{I})^{\mathrm{T}}$$

gives that the following two modified Euler-Lagrange equations are obtained:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial \mathscr{T}}{\partial v^{\mathcal{V},X}} - \dot{\psi} \frac{\partial \mathscr{T}}{\partial v^{\mathcal{V},Y}} &= \mathscr{F}^X \\ \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial \mathscr{T}}{\partial v^{\mathcal{V},X}} + \dot{\psi} \frac{\partial \mathscr{T}}{\partial v^{\mathcal{V},Y}} &= \mathscr{F}^Y. \end{aligned} \tag{12.5}$$

Using the partial derivatives of the kinetic energy and (12.5), the following set of dynamic equations are found:

$$m\dot{v}^{\mathcal{V},X} - m\dot{\psi} v^{\mathcal{V},Y} = \mathscr{F}^X, \tag{12.6a}$$

$$m\dot{v}^{\mathcal{V},Y} + m\dot{\psi} v^{\mathcal{V},X} = \mathscr{F}^Y, \tag{12.6b}$$

$$I_{ZZ}\ddot{\psi} = \mathscr{F}^{\psi}, \tag{12.6c}$$

$$I_{\vartheta,i}\ddot{\vartheta}_i = \mathscr{F}_i^{\vartheta}, \tag{12.6d}$$

$$I_{\delta,i}\ddot{\delta}_i = \mathscr{F}_i^{\delta}, \quad \forall i \in \{1,\ldots,4\}. \tag{12.6e}$$

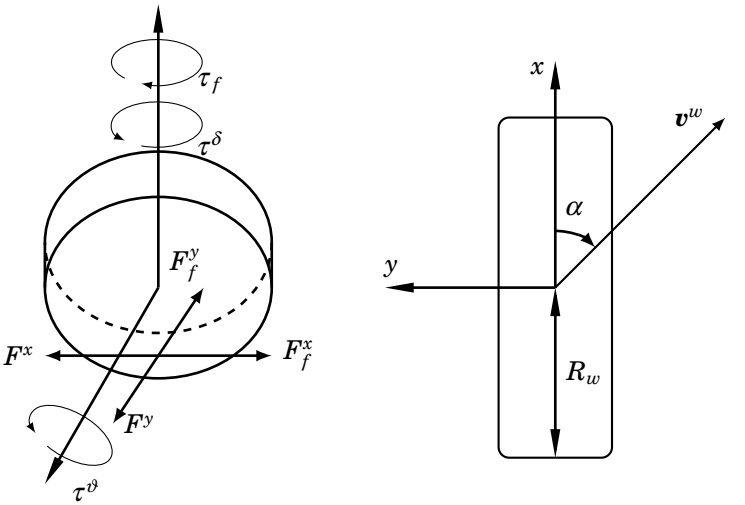

Figure 12.3 An illustration of the forces and torques acting on each wheel (left), and the wheel together with its coordinate system seen from above (right).

The forces and torques that act on the wheels are shown in Figure 12.3. We also model friction forces F_f and a friction torque τ_f in addition to the standard tire forces. The reason is that the forces and torques are much smaller than in automotive systems. The motor torques acting on each wheel are modeled as two independent torques for driving and steering, τ^ϑ and τ^δ, respectively. Hence, the forces acting along the robot's longitudinal and lateral directions, corresponding to the right-hand sides in (12.6a)–(12.6b), are

$$\begin{aligned} \mathscr{F}^X &= F^X = \sum_{i=1}^{4} \Big(\cos(\delta_i)(F_i^x - F_{f,i}^x) - \sin(\delta_i)(F_i^y - F_{f,i}^y) \Big) \\ \mathscr{F}^Y &= F^Y = \sum_{i=1}^{4} \Big(\sin(\delta_i)(F_i^x - F_{f,i}^x) + \cos(\delta_i)(F_i^y - F_{f,i}^y) \Big) . \end{aligned} \tag{12.7}$$

The generalized torque $\mathscr{F}^\psi$ is equal to the yaw moment M^Z and is

straightforwardly computed as

$$\begin{aligned} M^Z = &(l_1 \sin(\delta_1) - w_1 \cos(\delta_1))(F_1^x - F_{f,1}^x) \\ &+ (l_1 \sin(\delta_2) + w_2 \cos(\delta_2))(F_2^x - F_{f,2}^x) - (l_1 \sin(\delta_3) + w_2 \cos(\delta_3))(F_3^x - F_{f,3}^x) \\ &- (l_1 \sin(\delta_4) - w_1 \cos(\delta_4))(F_4^x - F_{f,4}^x) + (l_1 \cos(\delta_1) + w_1 \sin(\delta_1))(F_1^y - F_{f,1}^y) \\ &+ (l_1 \cos(\delta_2) - w_2 \sin(\delta_2))(F_2^y - F_{f,2}^y) - (l_1 \cos(\delta_3) - w_2 \sin(\delta_3))(F_3^y - F_{f,3}^y) \\ &- (l_1 \cos(\delta_4) + w_1 \sin(\delta_4))(F_4^y - F_{f,4}^y). \end{aligned} \tag{12.8}$$

Moreover, the generalized forces in the wheels' drive and steer directions, (12.6d) and (12.6e), are

$$\begin{aligned} \mathscr{F}_i^\vartheta &= \tau_i^\vartheta - R_w(F_i^x - F_{f,i}^x) \\ \mathscr{F}_i^\delta &= \tau_i^\delta - \tau_{f,i}, \quad \forall i \in \{1, \dots, 4\}. \end{aligned} \tag{12.9}$$

Wheel-Force Modeling

The friction forces are modeled using Coulomb friction. This means that

$$\begin{aligned} F_f^x &= F_C^x \text{sign}(v^x) \\ F_f^y &= F_C^y \text{sign}(v^y) \\ \tau_f &= \tau_C \text{sign}(\dot{\delta}), \end{aligned} \tag{12.10}$$

where F_C^x, F_C^y, and τ_C are the Coulomb-friction constants. The reason for only considering Coulomb friction is that velocities necessary for, for example, viscous friction to dominate will not be reached under normal operating conditions for the considered type of vehicles. For large velocities air drag becomes a factor, and will typically dominate over Coulomb and viscous friction. This is straightforwardly modeled within this framework, as pointed out in [Lipp and Boyd, 2014].

For the types of maneuvers the considered class of four-wheeled vehicles perform in general, it is safe to assume that the longitudinal and lateral tire forces are proportional to the respective slip quantity.[2] The wheel forces caused by wheel slip are then computed as in (4.6), together with the slip definitions (4.3) and (4.4). This yields

$$\begin{aligned} F^x &= C_\lambda \lambda \\ F^y &= C_\alpha \alpha, \end{aligned} \tag{12.11}$$

where C_λ, C_α are the (constant) longitudinal and lateral stiffness, respectively, found from experiments, dependent on the robot mass, wheel

[2] Vehicles performing under extreme conditions are excluded since they require detailed tire-force and slip modeling in the case of time-optimal maneuvers, see Chapter 10.

material, and surface conditions. With this wheel-force modeling, the dynamic equations of the vehicle are constituted by (12.6)–(12.11). Note that (12.10)–(12.11) hold for each wheel individually.

Model for Optimization

Two options exist for formulating the time-optimal path-tracking problem. Either the tracking is enforced at the Cartesian level, similar to what was done at the high level in Chapter 11. Then these Cartesian forces and torques have to be used together with the mobile platform dynamics and inverse kinematics for determining the wheel inputs. Another possibility is to use the wheel dynamics (12.6d)–(12.6e) directly with the wheel torques as inputs. Since the actuator constraints are formulated on the wheel level (originating from motor constraints), it is advantageous to formulate the problem based on the wheel dynamics and is consequently the choice made here.

For simplification, we invoke the no-slip assumption. This implies that the applied wheel torques directly influence the vehicle movement. Hence, the wheel dynamic equations, derived from (12.6d)–(12.6e) and (12.9) by assuming that the longitudinal forces are zero, are written as

$$\begin{bmatrix} \boldsymbol{\tau}^{\vartheta} \\ \boldsymbol{\tau}^{\delta} \end{bmatrix} = \boldsymbol{\tau}^{\rho} = \boldsymbol{I}(\boldsymbol{\rho})\ddot{\boldsymbol{\rho}} + \mathbf{F}_{\boldsymbol{\rho},C}\mathrm{sign}(\dot{\boldsymbol{\rho}}), \tag{12.12}$$

where

$$\boldsymbol{\rho} = \begin{bmatrix} \vartheta_1 & \cdots & \vartheta_4 & \delta_1 & \cdots & \delta_4 \end{bmatrix}^{\mathrm{T}},$$

$\boldsymbol{I}(\boldsymbol{\rho})$ is the inertia matrix, and $\mathbf{F}_{\boldsymbol{\rho},C}$ is the Coulomb-friction vector.

Kinematics

The geometric vehicle path is, by convention, determined in Cartesian space by a high-level path planner. A method is therefore needed for transferring Cartesian path coordinates $\{\boldsymbol{p}, \psi\}$ to wheel-space coordinates $\boldsymbol{\rho}$. Thus, we want to find a transformation

$$\Omega : \{\boldsymbol{p}, \psi\} \rightarrow \boldsymbol{\rho}.$$

Because wheels exhibit slip, a closed-form transformation is in general not possible. To derive analytic expressions, we again impose the no-slip assumption, which means that $v^x = R_w\omega$, $v^y = 0$. This assumption does not hold during acceleration and deceleration, but the deviations are suppressed using high-level feedback, see Section 12.5, and the low-level wheel control loops. Conceptually, the transformation for the drive angles can be derived as follows: We are given a path for the geometric

center (CoG) of the robot for K grid points as $\{\boldsymbol{p}(k), \psi(k)\}_{k=1}^{K}$. This implies knowledge of the path at the wheel center point $\{\boldsymbol{p}^w(k)\}_{k=1}^{K}$, for all wheels. Thus, given the Cartesian wheel position $\boldsymbol{p}^w(k)$ and $\boldsymbol{p}^w(k-1)$ in two neighboring elements, the wheel drive angle for each wheel at each grid point k is found as

$$\vartheta(k) = \vartheta(k-1) + \frac{\|\Delta \boldsymbol{p}(k)^w\|}{R_w}, \tag{12.13}$$

for small enough $\Delta \boldsymbol{p}^w(k) = \boldsymbol{p}^w(k) - \boldsymbol{p}^w(k-1)$. To find the steer angles we apply trigonometry, yielding

$$\delta(k) = \arctan2\big(\Delta p^{w,Y}(k), \Delta p^{w,X}(k)\big) - \psi(k), \tag{12.14}$$

where arctan2 is the four-quadrant inverse tangent function. Relations (12.13) and (12.14) hold for small enough differences—that is, for large enough number of grid points K. Note that ψ_k is subtracted since we want to know δ_k with respect to the vehicle.

REMARK 12.1
The inverse kinematics derivation assumes that the vehicle velocity expressed in the inertial frame is positive and that $|\psi| \leq \pi/2$. The other cases are straightforward but tedious to derive and are omitted here. □

12.4 Trajectory Generation

This section describes the approach for generating time-optimal trajectories given a nominal geometric path. In addition, this section considers the discretization of the continuous-time optimization problem for enabling online numerical solutions.

Time-Optimal Trajectory Generation

The trajectory generation problem is formulated as a convex optimization problem, given a geometric path $\boldsymbol{f}$. This is different from the approach in Chapter 11, where the path and trajectories were found simultaneously. The formulation considers the constraints on the actuators in terms of realizable torques. We neglect slip, as discussed on page 268, which means that the considered dynamic system is given by (12.12). The path to be tracked is parametrized in a path coordinate $s := s(t)$, according to[3]

$$\boldsymbol{f}(s) = \begin{bmatrix} f_1(s) & \cdots & f_n(s) \end{bmatrix}^{\mathrm{T}}, \quad s \in [s_0, s_f], \tag{12.15}$$

[3] The derivation in this section is performed using the more general assumption on n vehicle coordinates. For the dynamic model that is considered in Section 12.3, it holds that $n = 8$.

where n is the dimension of $\boldsymbol{\rho}$, and s_0 and s_f are the path coordinates at the path's endpoints, respectively. From the relation $\boldsymbol{\rho}(t) = \boldsymbol{f}(s(t))$, which should hold for all t, the following relations are established by using the chain rule:

$$\begin{aligned} \dot{\boldsymbol{\rho}} &= \boldsymbol{f}'(s)\dot{s}(s) \\ \ddot{\boldsymbol{\rho}} &= \boldsymbol{f}'(s)\ddot{s}(s) + \boldsymbol{f}''(s)\dot{s}(s)^2, \end{aligned} \tag{12.16}$$

where

$$\boldsymbol{f}'(s) := \frac{\mathrm{d}}{\mathrm{d}s}\boldsymbol{f}(s).$$

Utilizing the derivatives in (12.16), the dynamic equations in (12.12) can be reformulated in the path coordinate, [Bobrow et al., 1985; Shin and McKay, 1985; Pfeiffer and Johanni, 1987; Dahl, 1993], according to

$$\boldsymbol{\tau}^{\boldsymbol{\rho}}(s) = \boldsymbol{\Gamma}_1(s)\ddot{s}(s) + \boldsymbol{\Gamma}_2(s)\dot{s}(s)^2 + \boldsymbol{\Gamma}_3(s), \tag{12.17}$$

where

$$\begin{aligned} \boldsymbol{\Gamma}_1(s) &= \boldsymbol{I}(\boldsymbol{\rho}(s))\boldsymbol{f}'(s), \\ \boldsymbol{\Gamma}_2(s) &= \boldsymbol{I}(\boldsymbol{\rho}(s))\boldsymbol{f}''(s), \\ \boldsymbol{\Gamma}_3(s) &= \mathbf{F}_{\boldsymbol{\rho},C}\mathrm{sign}(\boldsymbol{f}'(s)). \end{aligned}$$

The aim of the trajectory generation is to minimize the execution time of the path tracking. The optimal-control problem is thus formulated over the time horizon $t \in [0, t_f]$, with the cost function chosen as the final time t_f. Utilizing the path coordinate and its time derivatives, the cost function is reformulated as

$$t_f = \int_0^{t_f} 1\,\mathrm{d}t = \int_{s_0}^{s_f} \frac{\mathrm{d}t}{\mathrm{d}s}\,\mathrm{d}s = \int_{s_0}^{s_f} \frac{1}{\dot{s}}\,\mathrm{d}s.$$

The state variable $\beta(s)$ and the algebraic variable $w(s)$ are introduced as

$$\begin{aligned} \beta(s) &= \dot{s}(s)^2, \\ w(s) &= \ddot{s}(s). \end{aligned}$$

Moreover, the relation

$$\left.\begin{aligned} \dot{\beta}(s) &= 2w(s)\dot{s} \\ \dot{\beta}(s) &= \beta'(s)\dot{s} \end{aligned}\right\} \quad \Longrightarrow \beta'(s) = 2w(s)$$

holds. The continuous-time optimal control problem to be solved is now stated in a similar way to [Verscheure et al., 2009c] as

$$\begin{aligned}
\underset{\alpha(s),\beta(s),\boldsymbol{\tau}(s)}{\text{minimize}} \quad & \int_{s_0}^{s_f} \left(\frac{1}{\sqrt{\beta(s)}} + \kappa \sum_{i=1}^{n} \left| \left(\tau_i^{\boldsymbol{\rho}}\right)'(s) \right| \right) \mathrm{d}s && \text{(12.18a)} \\
\text{subject to} \quad & \boldsymbol{\tau}^{\boldsymbol{\rho}}(s) = \boldsymbol{\Gamma}_1(s) w(s) + \boldsymbol{\Gamma}_2(s)\beta(s) + \boldsymbol{\Gamma}_3(s) && \text{(12.18b)} \\
& \beta(s_0) = \beta(s_f) = 0 && \text{(12.18c)} \\
& \beta'(s) = 2w(s) && \text{(12.18d)} \\
& \beta(s) \geq 0 && \text{(12.18e)} \\
& \boldsymbol{\tau}^{\boldsymbol{\rho}}_{\min} \leq \boldsymbol{\tau}^{\boldsymbol{\rho}}(s) \leq \boldsymbol{\tau}^{\boldsymbol{\rho}}_{\max}, && \text{(12.18f)}
\end{aligned}$$

where (12.18c) uses the assumption that the vehicle starts and stops in rest. The cost function (12.18) utilizes regularization of the torque derivatives. The weight κ is a tuning parameter that decides the impact of the torques in the cost function. An explicit time dependency is recovered from the solution by using the relation

$$t(s) = \int_{s_0}^{s} \frac{1}{\sqrt{\beta(\zeta)}} \, \mathrm{d}\zeta, \quad s_0 \leq s \leq s_f,$$

which is used for determining the input trajectories as functions of time.

The cost function (12.18a) is a convex function of the state variable and the input torques, and the model dynamics (12.18b) is affine in the optimization variables and control inputs. Hence, the optimal control problem (12.18) is a convex problem (see (2.9) on page 43). If the control problem would have been stated in the time domain, the model variables would have been decision variables, as is the case for the problems in Chapter 10 (see, e.g., page 212). Here, however, only one state (i.e., $\beta(s)$) and one algebraic variable, $w(s)$, are required for formulation of the optimal control problem, in contrast to originally $2n$ states required for the dynamics in (12.12).

Discretization and Numerical Solution

For numerical solution of the trajectory-generation problem, we discretize the continuous-time optimization problem (12.18) using direct transcription (see page 40) according to the procedure in [Verscheure et al., 2008; Verscheure et al., 2009c], where it is assumed that $w(s)$ is piecewise constant. With this assumption, it follows from (12.18d) that $\beta(s)$ is piecewise linear. Using a grid with L elements in the interval $[s_0, s_f]$, setting $\kappa = 0$, and eliminating the torques $\boldsymbol{\tau}^{\boldsymbol{\rho}}(s)$ and the algebraic variable $w(s)$ from

(12.18), results in the convex optimization problem

$$\begin{aligned} \underset{\beta_1,\ldots,\beta_{L-1}}{\text{minimize}} \quad & \sum_{k=0}^{L-1} \frac{2\Delta s_{k+1}}{\sqrt{\beta_{k+1}}+\sqrt{\beta_k}} && \text{(12.19a)} \\ \text{subject to} \quad & \beta_k \geq 0, \ \forall k \in \{1,\ldots,L-1\} && \text{(12.19b)} \\ & \boldsymbol{\tau}_{\min}^{\boldsymbol{\rho}} \leq \boldsymbol{g}(s_{k+1/2}) \leq \boldsymbol{\tau}_{\max}^{\boldsymbol{\rho}}, \ \forall k \in \{0,\ldots,L-1\}, && \text{(12.19c)} \end{aligned}$$

where

$$\boldsymbol{g}(s_{k+1/2}) = \boldsymbol{\Gamma}_1(s_{k+1/2})\frac{\beta_{k+1}-\beta_k}{2\Delta s_{k+1}} + \boldsymbol{\Gamma}_2(s_{k+1/2})\frac{\beta_{k+1}+\beta_k}{2} + \boldsymbol{\Gamma}_3(s_{k+1/2}),$$

with $\Delta s_k = (s_k - s_{k-1})$ and $s_{k+1/2} = (s_{k+1}+s_k)/2$. The cost function (12.19a) is found by analytic integration of (12.18a) with $\beta(s)$ piecewise linear as:

$$\begin{aligned} \int_{s_k}^{s_{k+1}} \frac{1}{\sqrt{\beta(s)}}\,\mathrm{d}s &= \int_{s_k}^{s_{k+1}} \frac{1}{\sqrt{\beta_k + \dfrac{\Delta\beta_{k+1}}{\Delta s_{k+1}}(s-s_k)}}\,\mathrm{d}s \\ &= 2\left(\sqrt{\beta_{k+1}} - \sqrt{\beta_k}\right)\frac{\Delta s_{k+1}}{\Delta\beta_{k+1}}, \end{aligned}$$

which gives (12.19a).

Implementation It is straightforward to solve (12.19) for the values of $\beta(s)$ at the discretization points s_k, $k \in \{0,\ldots,L-1\}$, using general-purpose tools for convex optimization, such as CVX [Grant and Boyd, 2014]. However, to find the solution online in a real-time setting, we compute an approximate solution to (12.19). This is done using a method inspired by the approach suggested in [Verscheure et al., 2009a; Verscheure et al., 2009b; Wang and Boyd, 2010], where an unconstrained approximate version of (12.19) is formulated by employing logarithmic barrier-functions [Boyd and Vandenberghe, 2008] for the constraints in the cost function. This gives that (12.19) approximates to

$$\underset{\beta_0,\ldots,\beta_{L-1}}{\text{minimize}} \quad \sum_{k=0}^{L-1} \bar{g}(\beta_k, \beta_{k+1}), \qquad \text{(12.20)}$$

where

$$\begin{aligned} \bar{g}(\beta_k,\beta_{k+1}) = &\frac{2\Delta s_{k+1}}{\sqrt{\beta_{k+1}}+\sqrt{\beta_k}} \\ &- \frac{\mu}{2nL}\sum_{i=1}^{n}\Big(\log\left(\tau_{i,\min} - g_i(s_{k+1/2})\right) + \log\left(g_i(s_{k+1/2}) - \tau_{i,\max}\right)\Big). \end{aligned}$$

The parameter μ is the log-barrier parameter. With the unconstrained optimization problem (12.20), we determine an approximate globally optimal solution using Newton's method. This requires computation of the first- and second-order derivatives (Jacobian and Hessian) related to the problem. For our model, the problem structure allows for analytic expressions of these quantities, which are used in the implementation. The Hessian in the Newton iterations is tridiagonal, which follows from that the Hessian only depends on β_{k-1}, β_k, and β_{k+1}. Hence, the time complexity for solving the inherent linear equation system is linear in the number of discretization elements [Golub and Van Loan, 1996], which enables fast solutions also in the case when high grid density is required—for example, for intricate geometric paths where high resolution is important.

12.5 High-Level Feedback Controller

We assume that both static and dynamic obstacles, which are not encoded in the map a priori, are present. Hence, an online obstacle-avoidance scheme is required. This section outlines a scheme for obstacle avoidance that is integrated with a linear MPC. The considered approach provides feedback from global coordinates for robustness to the model uncertainties that are present, both when determining the time-optimal trajectories in the optimization and when executing the trajectory generator. Naturally, it also accounts for disturbances such as uneven ground, slippery floor, and doorsteps.

Obstacle Avoidance

The trajectory generator described in Section 12.4 computes time-optimal trajectories given a geometric path. The path is predetermined and is based on a static map of the environment. When objects that are not part of the map—for example, humans, other vehicles, open/closed doors, and moving obstacles—are present, the given path may not be collision free during runtime. One alternative to remedy this is to plan a new path once new sensor data are available, thus taking the obstacles into account. There are two problems with this approach. First, the dimension and shape of the object might be uncertain depending on the available sensor data, and the replanned path will thus possibly also render a collision. Second, if the object is moving, several replanning and reoptimization steps are required. This hinders smooth vehicle movement and most certainly increases path-traversal time. Moreover, in certain applications where a mobile platform carries a robot manipulator that performs a task, two examples being milling and painting, it is important to maintain the nominal path for the manipulator tool even when perfect tracking of the

platform path is impossible. The aim is then to deviate from the vehicle's nominal path when dictated by dynamic obstacles, and subsequently rejoin in a graceful manner.

Local Replanning of Trajectory

For the obstacle avoidance we assume that online range measurements are available, stemming from laser scanners, sonar sensors, or online vision data. The goal of the obstacle-avoidance scheme is to enforce that the vehicle is outside the obstructed area. Note that for an optimization problem, the constraint that the vehicle path should be outside of the obstacle is nonconvex. Thus, to approximate the object's shape, we use a linearization around the N closest points on the obstacle seen from the vehicle. Consequently, if the distance to an object (that is not modeled in the predefined static map) is decreasing and reaches a predefined threshold, we compute a hyperplane π_o from the N closest points of the object edge, see Figure 12.4 for $N = 1$. Then the normal vector $\boldsymbol{v}_\perp$, which is orthogonal to the vector between the object plane π_o and the N closest points on the vehicle's hull, $\boldsymbol{v}_o$, is determined as

$$\boldsymbol{v}_\perp = \pm \boldsymbol{v}_o \times \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{\mathrm{T}}. \tag{12.21}$$

The velocity vector's sign is determined based on the relative position between vehicle and obstacle, aiming to keep the deviation from the nominal path small. The idea for determining the velocity vector is similar to the convex-concave procedure (also known as sequential convex programming) for solving optimization problems [Yuille and Rangarajan, 2003], where the concave part of the constraint is approximated using a linearization about the current solution. With the proposed approach, the obstacle avoidance scheme then computes a new normal vector $\boldsymbol{v}_\perp$ at each time step. Note that the shape of the object is not important, since the velocity vector is recomputed in each sample based on the new sensor data. The norm of the velocity vector $\boldsymbol{v}_\perp$ is chosen as the corresponding norm of the time-optimal velocity trajectory, $\|\boldsymbol{v}_{\mathrm{opt}}\|$, for the point on the path that is closest to the vehicle.

To keep track of the current point along the nominal trajectories, the path coordinate s is updated based on the path traversal. When activating the obstacle-avoidance scheme and thus leaving the nominal path, the point at the nominal path closest to the vehicle's current point is used for computing the path coordinate. The modified velocity vector $\boldsymbol{v}_{\mathrm{mod}}$ is thereby computed as a convex combination of the normal velocity vector $\boldsymbol{v}_\perp$, determined by the obstacle-avoidance scheme, and the nominal time-optimal velocity vector $\boldsymbol{v}_{\mathrm{opt}}$ according to

$$\boldsymbol{v}_{\mathrm{mod}} = \theta \boldsymbol{v}_\perp + (1-\theta)\boldsymbol{v}_{\mathrm{opt}}, \quad \theta \in [0,1], \tag{12.22}$$

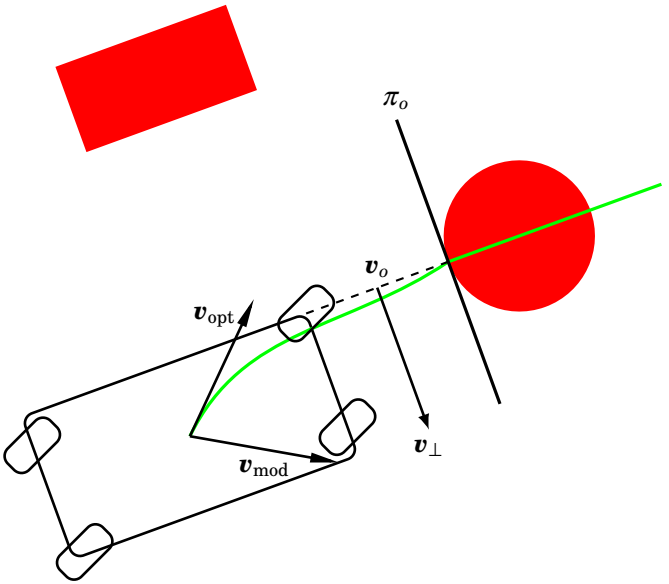

Figure 12.4 A sketch of how the modified velocity reference is generated in each time step in the case of dynamic obstacles. The robot initially follows the green line (nominal geometric path). An object, in this case the red circle, enters the planned path of the vehicle. By estimating the distance from the vehicle to the object and finding the hyperplane π_o, a new vehicle-motion direction $\boldsymbol{v}_{\perp}$ is calculated. The modified velocity vector $\boldsymbol{v}_{\text{mod}}$ is then found as a convex combination between $\boldsymbol{v}_{\text{opt}}$ (the nominal time-optimal trajectory) and $\boldsymbol{v}_{\perp}$. The vector of the minimal distance between the robot and the object is denoted by $\boldsymbol{v}_o$. The sketch shows the case for when the modified velocity reference is based on the minimal distance to one point. In the implementation, we use the N closest points on the object seen from the robot.

where θ is the weight. The weight θ is increased when the distance from the obstacle decreases, and vice versa. This also leads to a natural rejoining procedure after passing the obstacle when the weight is decreased again and finally, after a certain safety distance, the robot only tracks the nominal time-optimal trajectory. The position reference $\boldsymbol{p}_{\text{mod}}$ corresponding to $\boldsymbol{v}_{\text{mod}}$ is straightforward to compute based on the velocity vector.

Model Predictive Controller

The computed time-optimal trajectories for the wheel torques can be applied directly to the vehicle. However, model uncertainties and sensor imperfections will lead to deviations from the position and velocity trajectories, which are computed by the time-optimal trajectory generation

and collision-avoidance schemes, if sent directly to the internal low-level wheel torque or velocity controllers. In addition, wheel slip will cause a discrepancy between the wheel coordinates and the corresponding global Cartesian position and orientation of the vehicle.

We use MPC to introduce feedback from the estimated vehicle pose on a high level in a hierarchical control architecture. The MPC objective is to track the computed time-optimal trajectories and suppress disturbances and model uncertainties. The computed trajectories can here be considered as time-optimal feedforward control inputs. Consequently, the cost function in the MPC is chosen as the squared deviations from the desired trajectory, and the control signal from the MPC is the global velocity of the vehicle. In practice, these are realized with torque-resolved wheel-velocity controllers. When no obstacles are present, the MPC computes references for the global velocity based on the time-optimal trajectories. When there are no disturbances or model errors, and with appropriate MPC tuning, this results in the computed time-optimal control.

The MPC approach has the benefit that it naturally admits obstacle avoidance. The velocity references determined by the MPC are subsequently applied on each wheel using the vehicle's inverse differential kinematics. The main purpose of the MPC is to account for model errors and disturbances by minimizing the reference tracking-error; hence, we consider a kinematic model in discrete time on the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{F}\boldsymbol{u}_k, \tag{12.23}$$

with

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{p}_k \\ \boldsymbol{v}_k \\ \psi_k \\ \dot{\psi}_k \end{bmatrix}, \quad \boldsymbol{u}_k = \begin{bmatrix} \boldsymbol{v}_{k,\mathrm{ref}} \\ \dot{\psi}_{k,\mathrm{ref}} \end{bmatrix},$$

where $\boldsymbol{u}_k \in \mathbb{R}^3$ contains the corresponding control inputs (reference values to the internal control loops in the robot). All variables are expressed with respect to an inertial frame I in line with the notation throughout the thesis, see Figure 12.1. Moreover,

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & \eta_1 & 0 & 0 & 0 \\ 0 & 1 & 0 & \eta_1 & 0 & 0 \\ 0 & 0 & \eta_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \eta_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \eta_1 \\ 0 & 0 & 0 & 0 & 0 & \eta_2 \end{bmatrix}, \quad \boldsymbol{F} = \begin{bmatrix} \eta_3 & 0 & 0 \\ 0 & \eta_3 & 0 \\ \eta_4 & 0 & 0 \\ 0 & \eta_4 & 0 \\ 0 & 0 & \eta_3 \\ 0 & 0 & \eta_4 \end{bmatrix},$$

where

$$\begin{aligned}\eta_1 &= -T_s(T_s\sigma - 1),\\ \eta_2 &= 1 - T_s\sigma,\\ \eta_3 &= \frac{T_s^2\sigma}{2},\\ \eta_4 &= T_s\sigma,\end{aligned}$$

T_s denotes the sampling period and σ represents the time constant of the low-level wheel control loops. The rationale behind using (12.23) is that the wheel control loops ensure accurate velocity-reference tracking (if possible, depending on the maximum realizable motor torque). In addition, the assumptions on mechanical properties of the vehicle type imply that the translational and rotational dynamics are almost independent from each other. A decoupled, kinematic model is therefore enough on this level.

The quadratic cost function[4] is written as

$$\mathcal{J}_k = \sum_{m=1}^{H_p} \|\boldsymbol{x}_{k+m} - \boldsymbol{r}_{k+m}\|_{\boldsymbol{W}_x}^2 + \sum_{m=0}^{H_c-1} \|\boldsymbol{u}_{k+m}\|_{\boldsymbol{W}_u}^2,$$

where $\boldsymbol{r}_m$ denotes the position and velocity reference vector at time step m, computed by the time-optimal trajectory generator or the local trajectory replanner described earlier, starting at page 274. The convex optimization problem to be solved in the MPC at each sample $k > 0$ is

$$\begin{aligned}
\underset{\mathcal{U}_k}{\text{minimize}} \quad & \sum_{m=1}^{H_p} \|\boldsymbol{x}_{k+m} - \boldsymbol{r}_{k+m}\|_{\boldsymbol{W}_x}^2 + \sum_{m=0}^{H_c-1} \|\boldsymbol{u}_{k+m}\|_{\boldsymbol{W}_u}^2 && \text{(12.24a)}\\
\text{subject to} \quad & \boldsymbol{x}_{k+m+1} = \boldsymbol{A}\boldsymbol{x}_{k+m} + \boldsymbol{F}\boldsymbol{u}_{k+m} && \text{(12.24b)}\\
& \boldsymbol{p}_{k+m,\min} \leq \boldsymbol{p}_{k+m} \leq \boldsymbol{p}_{k+m,\max} && \text{(12.24c)}\\
& \boldsymbol{v}_{\min} \leq \boldsymbol{v}_{k+m} \leq \boldsymbol{v}_{\max} && \text{(12.24d)}\\
& \dot{\psi}_{\min} \leq \dot{\psi}_{k+m} \leq \dot{\psi}_{\max}, \quad \forall m \in \{0,\ldots,H_p\} && \text{(12.24e)}\\
& \boldsymbol{u}_{\min} \leq \boldsymbol{u}_{k+i} \leq \boldsymbol{u}_{\max} && \text{(12.24f)}\\
& \Delta\boldsymbol{u}_{\min} \leq \boldsymbol{u}_{k+i} - \boldsymbol{u}_{k+i-1} \leq \Delta\boldsymbol{u}_{\max} && \text{(12.24g)}\\
& \boldsymbol{x}_k = \bar{\boldsymbol{x}} && \text{(12.24h)}\\
& \forall i \in \{0,\ldots,H_c-1\},
\end{aligned}$$

[4] In the MPC cost function we exclude weights on the final state in the prediction horizon. This is caused by the nature of the a priori unknown dynamic obstacles. Given the slowly changing robot motion, this does not imply stability violations. This is in contrast to what was done in Chapter 11. The terminal constraint is easily added if desired.

where $\mathcal{U}_k = \{\boldsymbol{u}_k, \ldots, \boldsymbol{u}_{k+H_c-1}\}$ is the set of control inputs to be determined, $\bar{\boldsymbol{x}}$ is the initial state at time step k, H_p is the prediction horizon, and H_c is the control horizon. If $H_c < H_p$, the control signal $\boldsymbol{u}_i$ is assumed to be constant and equal to $\boldsymbol{u}_{k+H_c-1}$ for all $i \geq k + H_c$. In (12.24), the position constraints (12.24c) are given by rectangular approximations of the surrounding, a priori known obstacles in the map; (12.24d)–(12.24g) are implied by the robot's physical properties; and (12.24h) is estimated by a particle filter.

12.6 Simulation Study

The outlined method uses torque-resolved velocity controllers at the lowest level instead of applying the torques found from optimization directly. The reason for doing this is that even for the slightest error in the dynamic model, the torques will propagate in the dynamics and give large errors in position as, for example, the nonlinear motor dynamics and the wheel slip are not accounted for in the dynamic model. To illustrate this, we will discuss results from a simulation study using the dynamic model with slip, which was derived in Section 12.3. We discuss results from two simulations; one simulation where the optimal torques found from the trajectory generator in Section 12.4 are used as inputs, and one simulation where the optimized velocity references from the trajectory generator are used as inputs to the dynamic model. For the second simulation, cascaded PI controllers are designed for wheel position and velocity control, to mimic the control loops in the experimental setup. The masses and inertias are set to values reasonable for the mechanical parts involved. The maximum torques are set to reproduce what can be achieved from the motors that are mounted on the physical mobile base (see page 281).

Simulation Results

Figure 12.5 shows the torques for wheel 2 generated in the optimization (red) and the torques generated by the cascaded PI controllers when using the optimized velocities as inputs (black). Figure 12.6 contains the desired path (blue), the path followed by using the optimized torques as inputs (red), and the path followed by using the optimized velocities as inputs (black). Moreover, Figure 12.7 visualizes the longitudinal and lateral slip for wheel 2 using the torque-resolved velocity controllers. The longitudinal slip values hardly exceed 0.015, and the lateral slip is at most roughly 0.5 deg. Thus, for the maneuvers and robot characteristics considered, neglecting slip in the model aimed at optimization is indeed a valid assumption. The largest discrepancies in the torques occur where the robot changes velocity rapidly, and hence where the slip is largest.

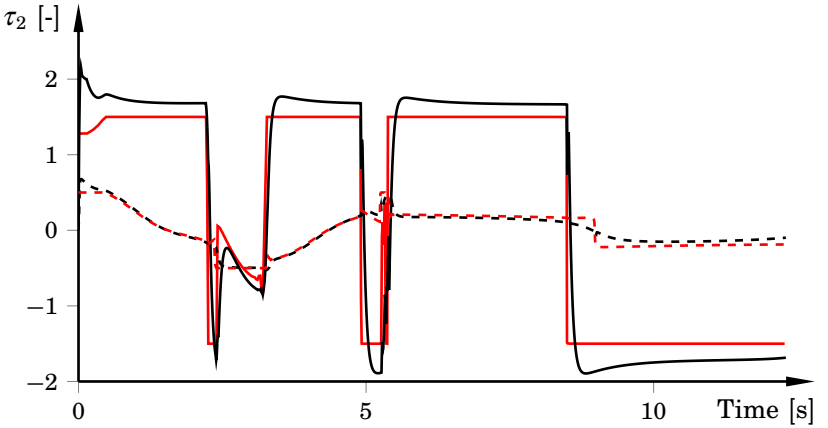

Figure 12.5 Torques for wheel 2 generated by optimization (red) and by using the optimized velocities (black) as inputs to cascaded PI controllers. The steer torques are shown as dashed. The torques are similar in shape and size, but the resulting trajectories differ significantly, see Figure 12.6. Note that the differences in the torques are largest in the transient phases, which is expected since it is during acceleration the slip gives the largest impact.

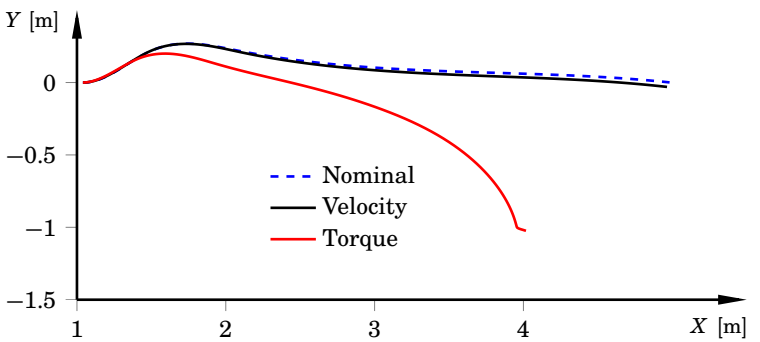

Figure 12.6 Simulation results for the path corresponding to Figures 12.5 and 12.7. The nominal path (blue dashed), the path using velocity-reference inputs (black), and the path using the torque inputs (red) are shown. The maximum deviation for the path using velocity references is a few centimeters.

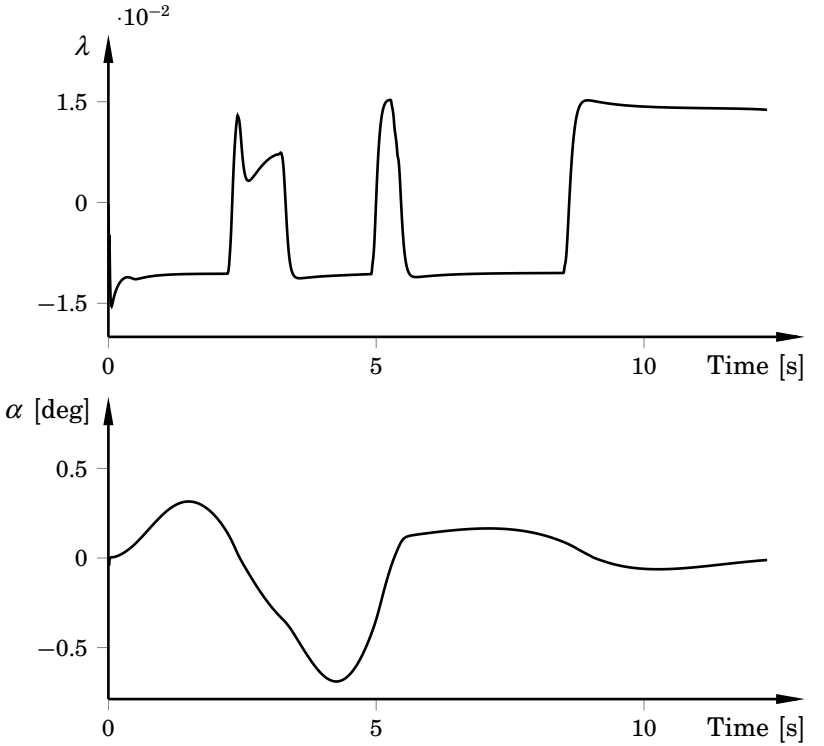

Figure 12.7 The longitudinal and lateral slip for wheel 2 developed by the torques (black) in Figure 12.5 for the path in Figure 12.6 (black). The slip values are close to zero throughout the simulation, a verification that neglecting slip in the optimization model can be justified. The figure only shows the slip for one wheel, because the considered maneuver gives similar slip on all wheels.

Although the torques are similar in size and shape (Figure 12.5), the trajectory when using the torques as inputs quickly deteriorates. This motivates why using the wheel velocities as references is advantageous. Because of the model errors in the initial and final transient phases, the maximum torques used in the optimization are in practical implementations chosen to be slightly smaller than the physical torque constraints. Further, it is clear that feeding the velocity PI controllers with the optimal velocity trajectories produces almost the same torques as computed in the optimization, for most parts of the path. Thus, time optimality is fulfilled.

12.7 Experimental Results

This section presents experimental results, using a pseudo-omnidirectional mobile robot.

Experimental Setup

The robot used for the experimental validation is a four-wheeled pseudo-omnidirectional mobile robot equipped with eight motors, two for each wheel realizing the steering and driving, see Figure 12.8. The mobile robot, which was built and designed at Fraunhofer IPA in Stuttgart, is the successor to the Care-O-Bot 3 mobile base [Reiser et al., 2009a; Connette et al., 2009]. It is equipped with two SICK s300 laser scanners that deliver laser-range measurements from the robot's front and rear corners. The robot is controlled and sensor data are acquired using the ROS software package [ROS, 2014]. The wheel-encoder position and velocity measurements are extracted with a rate of 100 Hz. The individual wheels are controlled with torque-resolved cascaded position and velocity controllers executing at 100 Hz (implemented in C++) and executed internally in ROS. Identification of the required robot parameters is described next.

Parameter Identification To estimate the parameters in the matrix $\boldsymbol{I}(\boldsymbol{\rho})$ and Coulomb-friction parameter vector $\mathbf{F}_{\boldsymbol{\rho},C}$ required for the dynamics in (12.12), experimental data were collected. Under the assumption that the translational motion of the robot is significantly larger than the rotational motion, the mass matrix can be assumed to be diagonal with inertia elements

$$\boldsymbol{I}(\boldsymbol{\rho}) = \operatorname{diag}\left(\begin{bmatrix} I_{\vartheta,1} & I_{\vartheta,2} & I_{\vartheta,3} & I_{\vartheta,4} & I_{\delta,1} & I_{\delta,2} & I_{\delta,3} & I_{\delta,4} \end{bmatrix}\right). \tag{12.25}$$

The parameters were estimated by linearly increasing the velocity references to the wheel controllers, starting at rest. A linearly increasing velocity corresponds to a constant applied torque in the robot model. Hence, the inertia elements in the mass matrix can be estimated as the ratio between the applied torque and the corresponding angular acceleration. The motor-current measurements are accessible within the mobile robot platform via ROS. For estimating the Coulomb-friction parameters $\mathbf{F}_{\boldsymbol{\rho},C}$ for the respective wheel, a series of constant low-velocity references, with different signs, were applied to the wheels while monitoring the motor torques. Consequently, the robot was moving both forward and backward at low velocities, thus providing the necessary data for the friction parameter estimation with a method similar to [Bittencourt et al., 2010].

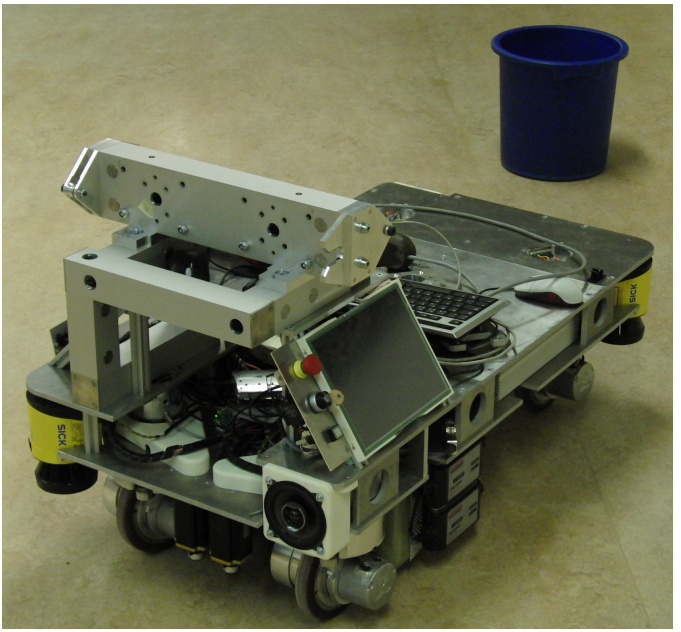

Figure 12.8 The mobile platform that is used for the experimental validation, together with one of the items (garbage bin in the background) that served as obstacles during the experiments. The yellow laser scanners attached to two of the robot's corners are used for obstacle detection, localization, and map building.

Implementation and Software Architecture

Figure 12.9 gives a schematic of the control structure. The path planner generates a feasible geometric path given a predefined static map of the environment, using Dijkstra's algorithm [LaValle, 2006]. The map is determined prior to the experiment based on data from the two laser scanners, which are attached to the corners of the mobile robot platform, combined with the wheel odometry. This is here done using the SLAM algorithm described in [Grisetti et al., 2005; Grisetti et al., 2007], but it is possible to use other algorithms. The generated path is then sent to the trajectory generator, which provides time-optimal wheel position trajectories, velocity trajectories, and input torques. The wheel velocity references given by the trajectory generator are subsequently transformed to Cartesian velocity references using the forward differential kinematics. These velocity trajectories and the corresponding position trajectories are then sent to the MPC together with the current estimates of the state vector (pose and velocity), estimated by a particle filter based on the static map,

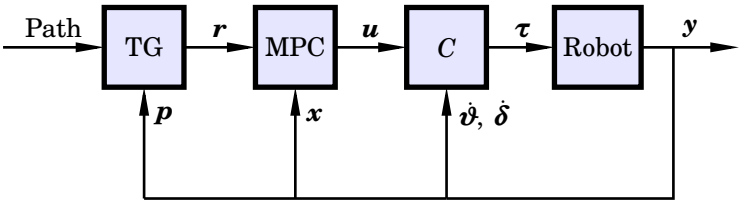

Figure 12.9 A schematic of the control structure. A path planner provides the trajectory generator (TG) with a desired path. The trajectory generator computes corresponding time-optimal trajectories, which are sent to the MPC. The MPC then computes desired velocities in the robot's coordinate system. The internal wheel controller, C, transforms the Cartesian velocities to the wheel level. Based on these references, wheel torques are computed and applied to the robot. Note that the trajectory generator only computes new trajectories when a new path is available.

the laser-scanner data, and the wheel odometry. The MPC computes the input sequence $\mathcal{U}_k$. The Cartesian velocities and yaw rate from the first element in $\mathcal{U}_k$ are transformed to local coordinates and then sent to the internal velocity control loops.

Figure 12.10 shows a schematic representation of the implementation structure. The path planner is implemented internally in ROS, together with the torque-resolved velocity controllers (C in Figure 12.9). The obstacle avoidance is implemented in ROSPy, which is a Python abstraction of ROS.

The log-barrier solver presented in Section 12.4 is implemented in MATLAB and transformed to C code using the Coder toolbox in MATLAB and compiled. The MPC is implemented in C using CVXGEN [Mattingley and Boyd, 2012]. This results in an average solution time of 1 ms for the model at hand and the prediction and control horizons considered ($H_p = H_c = 10$, corresponding to a time horizon of 0.4 s, results in desired tracking behavior). To link the developed controllers with ROS, we use ROSPy, which executes with a rate of 25 Hz. The ctypes library [Python Software Foundation, 2014] is used for executing the trajectory generator and the MPC, both implemented in C, from Python.

Path-Tracking Scenario

The results originates from experiments in a room with an area of approximately 75 m^2. The map of the room, as estimated by the SLAM algorithm, is shown in Figure 12.11. Based on the map, in this scenario a geometric path is planned from the coordinate $(0, 1.5)$ m to $(7, -1.5)$ m; the robot's orientation along the path is computed such that the robot heading is in the direction of the tangent of the path. The resulting velocity trajectories

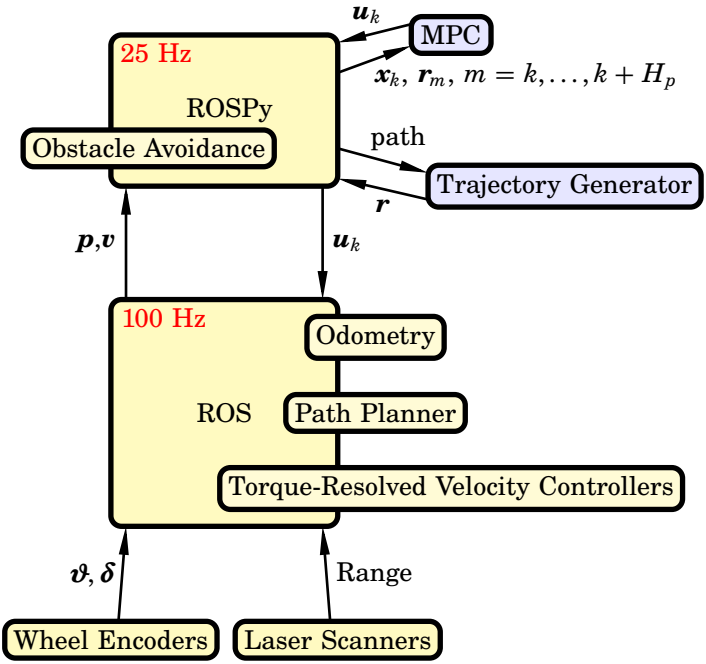

Figure 12.10 A sketch of the implementation structure. The odometry, path planner, and torque-resolved velocity controllers can be interacted with through ROS. They consist of compiled C++ code. We use a Python abstraction of ROS, named ROSPy, and connect the MPC and log-barrier solver to ROSPy using compiled C code via the ctypes library. The MPC executes at 25 Hz, but the trajectory geenrator only executes when a new path is available.

in global Cartesian coordinates, obtained from the time-optimal trajectory generation, are displayed in Figure 12.12. The corresponding time-optimal torques for the drive and steer motors are shown in Figure 12.13. The constraints on the drive and steer actuators are $\tau^{\vartheta}_{\max} = 0.31$ and $\tau^{\delta}_{\max} = 0.27$, and symmetrically for the lower bounds. The constraints are chosen based on the physical properties of the wheel motors in the mobile robot. They are slightly conservative, to provide actuation capability in the MPC for handling disturbances during runtime and unmodeled dynamics present when determining the time-optimal trajectories.

Note that at least one actuator is at the limit at each time instance, indicating the desired time optimality. In addition, it is clear that the driving motors are the limiting factors. This is expected, because the geometry of the path gives relatively small orientation changes. However, there is

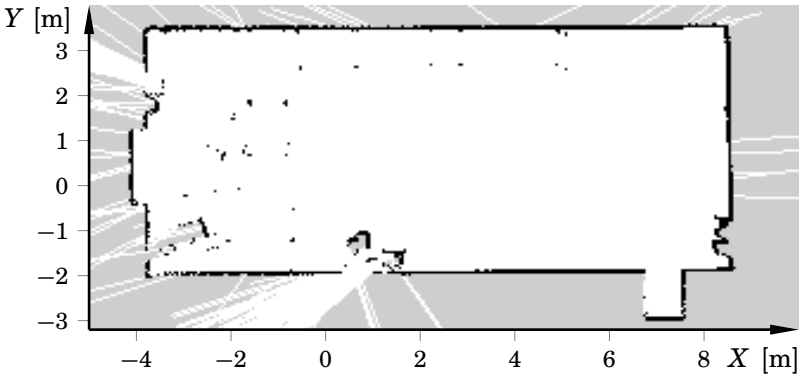

Figure 12.11 The room where the experiments were performed, as estimated by the SLAM algorithm. White color indicates free space and black indicates objects. The small black marks are reflections from table legs and the white lines outside of the room are caused by reflections.

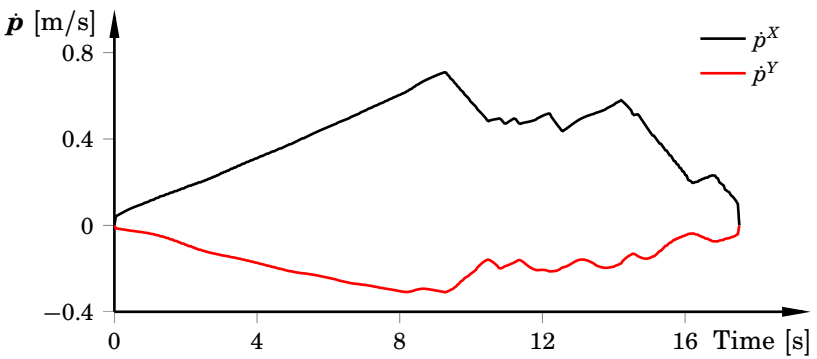

Figure 12.12 Time-optimal velocity-trajectory references obtained from the trajectory generator, expressed in global coordinates.

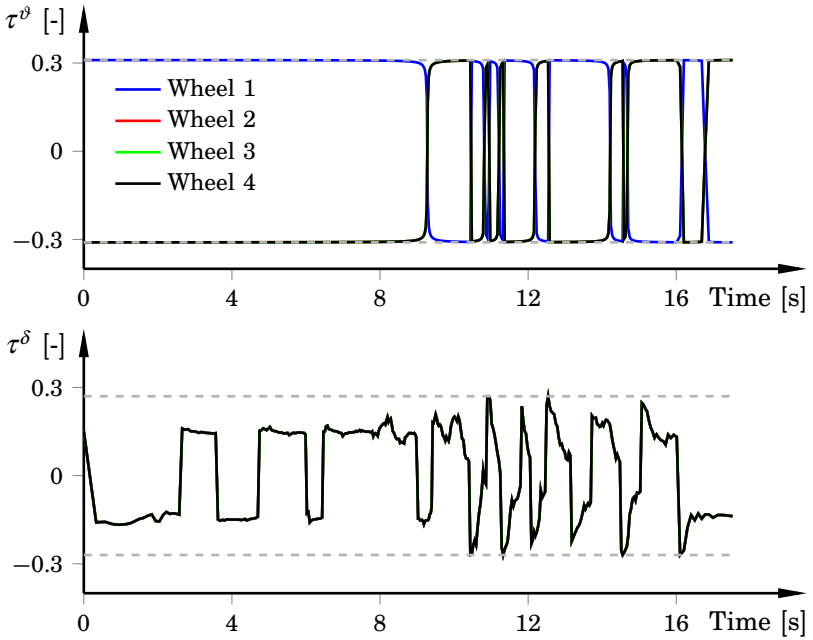

Figure 12.13 Time-optimal motor-torque trajectories, obtained from the trajectory generator. The torque constraints are indicated by the horizontal dashed gray lines. Because of the mechanical design of the robot, the wheel drive torques are pairwise equal. Moreover, the steer torques are equal for all wheels since the robot's orientation is fixed along the tangent of the path.

significant motion perpendicular to the orientation of the platform, resulting in the steer torques observed in the lower plot in Figure 12.13.

Initially, in the absence of dynamic obstacles, the mobile robot tracks the nominal path and the time-optimal trajectories using the MPC. The controller parameters (i.e., the weight matrices in (12.24a)) are

$$\begin{aligned} \boldsymbol{W}_{\boldsymbol{x}} &= \operatorname{diag}\left(\begin{bmatrix} 1 & 1 & 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix}\right), \\ \boldsymbol{W}_{\boldsymbol{u}} &= \operatorname{diag}\left(\begin{bmatrix} 0.1 & 0.1 & 1 \end{bmatrix}\right). \end{aligned}$$

The constraint on the velocity reference vector from the MPC is

$$\begin{bmatrix} \boldsymbol{v}_{\mathrm{ref,max}} \\ \dot{\psi}_{\mathrm{ref,max}} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0.05 \end{bmatrix},$$

whereas the slew-rate limits are

$$\begin{bmatrix} \Delta \boldsymbol{v}_{\mathrm{ref,max}} \\ \Delta \dot{\psi}_{\mathrm{ref,max}} \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.3 \\ 0.05 \end{bmatrix}.$$

The lower limits are symmetric to the upper limits. The units are in m/s and rad/s, respectively.

After the robot has started to move along the nominal path, two new obstacles are placed such that they cross the nominal geometric path (the obstacle locations are not encoded in the static map defined previously). These obstacles are detected during runtime using the laser-scan sensors. To minimize the path deviation because of the obstacles, the threshold value where the obstacle-avoidance scheme should be activated is set to 300 mm, which results in a large enough safety zone while not deviating too much from the nominal path. Figure 12.14 displays the executed path together with the map. The corresponding Cartesian velocity references computed by the MPC are shown in Figure 12.15, where also the time instants at which the obstacle-avoidance scheme is activated are displayed. It is clear that the robot is detecting and subsequently avoiding the new obstacles and also that when following the nominal path, the time-optimal velocity trajectories are tracked closely (compare Figure 12.15 with Figure 12.12). Further, the lower plot in Figure 12.15 indicates that the coupling between translational and rotational movement is small, implying that the model (12.23) is indeed valid.

12.8 Discussion

A key feature of the proposed method when compared with most approaches in literature, is that it combines a trajectory-reference generator (based on a nonlinear model of the vehicle that incorporates friction) that computes time-optimal trajectories with a feedback controller, where the problems of finding the references and control signals are posed as convex optimization problems. This implies that an optimal solution will be found quickly, thus enabling high sampling frequencies in the control architecture. Typically, the solution time for the trajectory generator is well below 1 s for paths of approximately 10 m (corresponding to $L = 500$ discretization elements in the optimization problem (12.19)), and the solution time for the MPC is almost always within 1 ms (corresponding to 5–10 iterations). Further, considering different optimization criteria than time, the trajectory generator can be modified to minimize combinations of path execution time and energy consumption.

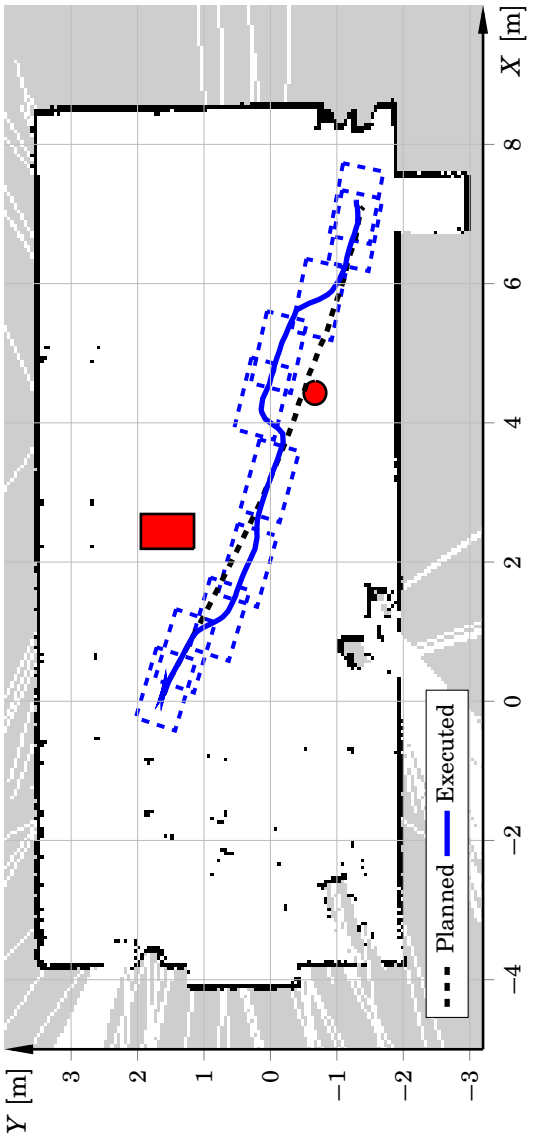

Figure 12.14 The nominal planned path together with the actual traversed path, the latter resulting because of the a priori unknown obstacles (red). The mobile robot hull is displayed (dashed blue) every other second.

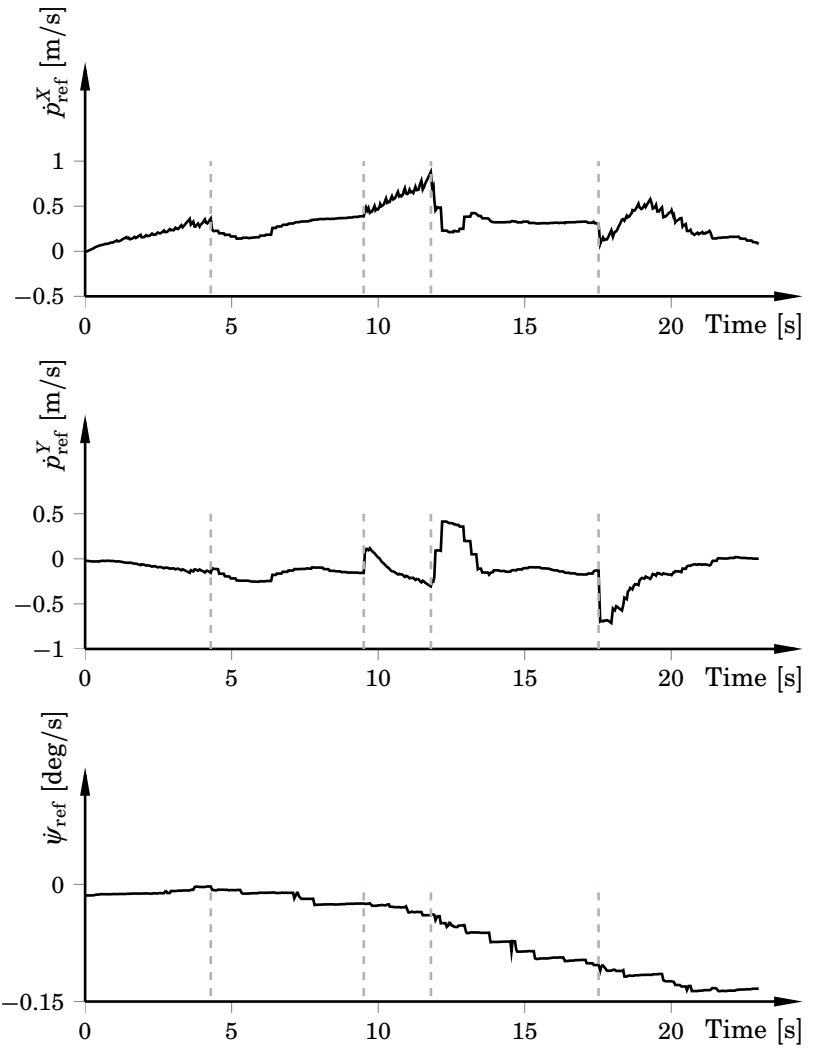

Figure 12.15 Cartesian velocity commands, measured in global space, computed by the MPC based on the time-optimal trajectories and the obstacle-avoidance scheme. The instances at which a switch between path tracking and obstacle avoidance occur are indicated by the vertical dashed gray lines. The time-optimal trajectories in Figure 12.12 are closely tracked when the obstacle avoidance is inactive. The tuning is aggressive, which can be seen, for example, in the upper plot at 10 s. The yaw-rate constraint corresponds to approximately 3 deg/s.

The MPC suppresses the effects of model errors. We showed in Section 12.6 that the model errors caused by slip are small. Still, there are errors present. By combining the proposed approach for generating time-optimal trajectories, using a model without slip, with an MPC, these errors are suppressed. Moreover, there are other imperfections present as well, such as uncertainties in the geometry. The MPC uses the global position and orientation estimates for feedback; thus, it effectively removes effects of model uncertainties when combined with the low-level wheel controllers. This is different from the current standard in mobile robotics, where a reference trajectory is generated and then fed to the low-level loops without global feedback.

As mentioned before, a motivation for using the online trajectory regeneration rather than replanning the path when an obstacle is encountered, is that it is sometimes desired to stay close to the original (nominal) path. Another motivation is that successive replanning and trajectory generation prevents task effectiveness. To demonstrate this, we use the same scenario as considered in Section 12.7, but now executing the robot's internal navigation module instead of method proposed in this chapter. The navigation module has been developed by the robot manufacturer. It is written in C++ and takes full advantage of the omnidirectional characteristics of the considered robot, but only incorporates constraints on a kinematic level. Thus, it is not using the full potential of the motors. Figure 12.16 shows the path traversed by the robot and Figure 12.17 visualizes the Euclidean norm of the velocity vector along the path. The same scenario that takes approximately 25 s to complete with the proposed approach now demands about 60 s. The shorter execution time with our approach is expected since it is time optimal. Furthermore, the velocity constraints in the internal navigation module are restrictive. It is more interesting that the robot stops and finds new feasible paths three times in total, with each replanning lasting about 1 s. Thus, it is clear that an approach that performs online collision avoidance based on local replanning of the trajectory is advantageous when task effectiveness is desired. By inspection of Figure 12.16 it is also obvious that the resulting geometric path using replanning differs significantly from the nominal path, shown in Figure 12.14. In addition, note that if it really is desired to replan the path and recompute the trajectories when an obstacle is encountered. This is easily achievable with the trajectory generator proposed in this chapter, and enables fast solution times for the reoptimization of the trajectory.

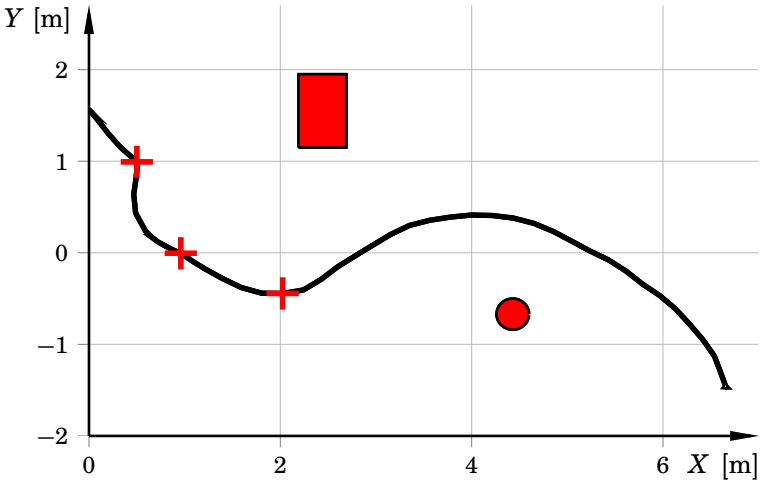

Figure 12.16 Resulting geometric path for the CoG when using the robot's internal navigation module. The locations at which replanning of the path occurs are marked with red +. The motion is performed from the coordinate $(0, 1.5)$ m to $(7, -1.5)$ m.

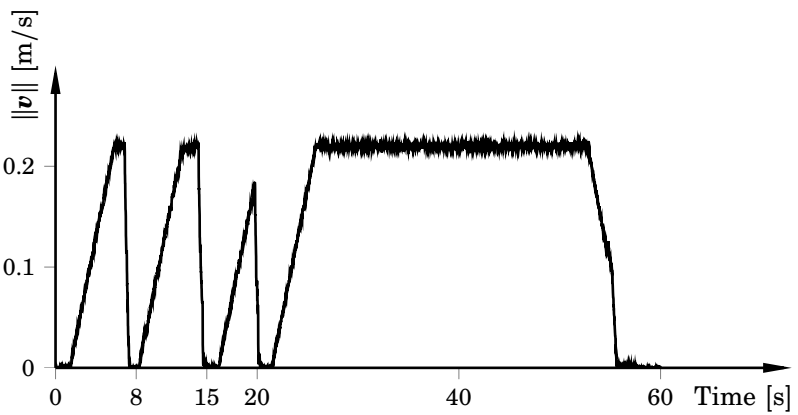

Figure 12.17 Euclidean norm of the velocity vector when using the robot's internal navigation module for the scenario in Figure 12.14. The robot stops for replanning purposes at $t = 8$, 15, and 20 s.

Outlook

Although the trajectory-generation approach was verified on a specific mobile robot, it is rather general. The algorithm allows for other four-wheeled vehicle types, as long as they are equipped with independent steer and drive actuation. This opens up for other applications, not necessarily including mobile robots.

As an example, automated highway driving for automotive systems is another potential application. In that scenario the path is likely to be given beforehand, typically based on road-map information. Because the intended driving scenario is that of traversing a path under normal driving conditions, the no-slip assumptions that we made in this chapter will approximately hold. As was shown in Chapter 9 and discussed in Chapter 11, it is not too optimistic to envision scenarios where the vehicle has knowledge about the relevant states (i.e., enough situation awareness).

12.9 Summary

This chapter outlined a design flow for time-optimal trajectory generation combined with obstacle avoidance for four-wheeled vehicles with independent steer and drive actuation. The proposed approach is entirely based on convex optimization, allowing fast computations both for trajectory generation and online control. To enable the convex formulation, a number of approximations were made, such as neglecting wheel slip. We investigated these assumptions in a simulation study, and the obtained results imply that the no-slip assumption is indeed valid. The obstacle-avoidance scheme was integrated in a high-level feedback controller based on MPC. The proposed method was fully implemented on a pseudo-omnidirectional mobile platform and evaluated in experiments in a demanding path-tracking scenario. The method performed well. The experimental results highlighted several advantages in comparison with a reference method, especially in terms of traversal time and velocity smoothness.

13

Summary and Conclusions

This thesis deals with various key elements in vehicle estimation and control. The underlying motivation is that the advancements in computing power and sensing abilities have led to an increased interest in providing vehicles that do not only react on operator commands, but also perform real-time decision making by themselves. To enable this, the vehicles need to be equipped with advanced estimation algorithms that can handle all available sensing information. In addition, the control algorithms need to be both advanced and fast enough to make informed decisions based on the provided information. This chapter summarizes the contributions of the thesis.

13.1 Estimation

Out-of-Sequence Measurements

Current vehicle systems have much available sensing information. With the sensors possibly executing at different update rates, it is likely that the sensor-fusion center will encounter out-of-sequence measurements (OOSMs).

Chapter 7 outlined a particle-filter method for handling situations when multiple OOSMs arrive simultaneously. The proposed method adapted an algorithm that is optimal for linear systems, and integrated this into an existing OOSM particle-filtering algorithm. The rationale for this was that the linear algorithm will also yield performance improvements when adapted to nonlinear systems. The results in showed that for particular sensor setups, there are quite significant estimation improvements if utilizing algorithms that specifically handle multiple out-of-sequence measurements. Hence, for certain applications, the method should be useful for improved tracking performance. It is likely that the performance gains will be larger when the uncertainties in the process dynamics are more significant than the measurement uncertainties.

It is common that dynamic systems are neither strictly linear nor nonlinear. Rather, the system consists of a mixture of linear and nonlinear components. One way to describe this is by using mixed Gaussian state-space models. In Chapter 8, two algorithms that utilize the model structure for improved handling of out-of-sequence measurements were developed. Both algorithms use approximations. Still, for the three considered simulation examples, the proposed algorithms always achieve better tracking performance than those that do not utilize model structure. In some cases, the performance is similar to what is theoretically achievable. In addition, an outlier and variance comparison indicate that the proposed algorithms also generate fewer outliers, and therefore more predictable behavior. In online tracking systems, it is important to have few outliers; not only because it improves tracking performance, but also because it simplifies outlier detection. When compared with a filter that is based on exact Bayesian recursions, the algorithms provide a notable speedup. That the computation time is limited implies that the algorithms are feasible alternatives in online implementations.

Ground-Vehicle Estimation

Chapter 9 derived a model-based combined wheel-slip and motion-estimation method for four-wheeled ground vehicles. The rationale for the approach is that, unlike what is traditionally done in the automotive industry, better estimation performance can be achieved by utilizing that there are coupling effects between different dynamic states. High-precision slip estimation is imperative for active safety systems, and improved performance in this respect can also implicitly improve control performance. We used a Bayesian approach. The model was written on mixed-Gaussian state-space form, which enabled the use of the Rao-Blackwellized particle filter.

One feature with the method is that among the 14 estimated states, several key variables in safety systems, such as wheel slip and roll angle, are estimated with high accuracy. The enabler for this is that coupling effects are incorporated into the model. The estimation accuracy is excellent even in very aggressive maneuvering scenarios, as the results showed. We compared the performance with a rule-based algorithm, which has complexity in the order of current slip-estimation schemes in production. The proposed method drastically improved performance, especially during heavy cornering.

The method is robust for the considered test scenarios, even without resorting to individual tuning of the noise parameters. This is interesting, particularly since the experimental setup contained uncertainties. The inertial measurement unit, for example, was assumed to be located at the

mass center, which is not true. The effective wheel radius was estimated for one of the test scenarios, which was then used in all scenarios. This also introduces errors.

The algorithm executes in real time, even for a MATLAB implementation that is not optimized for performance. Thus, the algorithm is certainly viable in online implementations.

13.2 Vehicle Control

Chapter 6 contained a realistic mobile-manipulation application in which a stationary manipulator picked up objects and placed them on a mobile robot. This scenario corresponds to what is desired in a factory setting, where a stationary robot typically has its own, predefined workstation. It is then up to the mobile robots to deliver objects across the workstations. There exist several mobile manipulation systems. However, they all have limited possibilities for allowing that the mobile base and robot manipulator move at the same time. The example used inertial and wheel-encoder measurements in combination with a vision system, which resulted in out-of-sequence measurements. We accounted for the out-of-sequence measurements by adopting a suboptimal linear-estimation algorithm. Owing to the estimation algorithm, and in combination with force sensing and appropriate controls, the manipulator successfully managed to place objects on the mobile robot while the mobile based was moving. In the considered example the vision algorithm had slow update rates, but this is realistic when there are limited resources. To further improve estimation accuracy it is necessary to model dynamic coupling effects between mobile base and robot manipulator. Also, faster update rates in the control loops would be useful, especially for the mobile base.

Impact of Vehicle Models in Optimal Control

To make sensible choices of which models to use in a specific application, proper understanding of the models is essential. This is particularly important for optimal-control based algorithms, because solutions to optimization problems tend to utilize a large portion of the available state space.

We compared and analyzed different combinations of chassis and tire models and their behavior in aggressive, time-optimal maneuvering scenarios in Chapter 10. In total four different model combinations were compared on two different maneuvers, on three different surfaces. The motivation for the study was to gain insight in at-the-limit maneuvers. Most models have been used in automotive applications before. The results showed that there are large differences in control inputs (i.e., torques

and steer angle) between the one-track and two-track models, but that the high-level variables, such as yaw rate and longitudinal velocity, are more similar. In particular, when restricting the discussion to assuming the more complex tire model, the one-track and two-track models are very similar in key aspects; this observation is even more true for low-friction surfaces. This has implications for safety systems, where a high-level trajectory generator can use a relatively simple model, which is then propagated to a more complex model for producing the control inputs. For the simpler (friction-ellipse based) tire model, however, this is not true. On the contrary, this model gave rise to large differences for low-friction surfaces, where the effects of load transfer intuitively should be suppressed. Another observation was that the behavior of the friction-ellipse based models is not consistent with that of expert drivers in similar scenarios.

We based the tire models used in the study on optimal-control solutions for different road surfaces on experimental data. A comparison of solution behavior on three different surfaces showed that there are significant differences between the surfaces, where, for example, the slip behavior differs. In addition, a scenario was given where only the friction coefficient was adjusted between the surfaces. Solving the optimization problem using that parametrization gave that the optimal solution had large slip, contrary to the results obtained for the empirical smooth-ice model. It is difficult to gather reliable experimental data and for large combined slip, it is hard to obtain data at all. Nevertheless, the results indicate that it is crucial how the tire-road interaction is modeled. This implies that safety systems have to be even more aware of the road conditions than what is typically done in current safety systems.

Closed-Loop Control for Autonomous Vehicles

The conclusions from Chapter 10 came to use in Chapter 11, where we derived a two-level hierarchical control structure for improved vehicle autonomy. The main idea is that a trajectory generator is responsible for generating high-level trajectory references to a low-level MPC, which distributes the corresponding torques and steer angle to the vehicle. Unlike other two-level approaches, this provides a chain of optimal controllers with a tight physical coupling in between. Results showed that the method is robust. In addition, the execution times indicate that it is feasible for real-time implementations. The low-level controller was able to follow the trajectory references even in aggressive maneuvering. This implies that the approach is valid. Moreover, it shows that the conclusions from Chapter 10 hold. Because the low-level layer contained a linear MPC in addition to the nonlinear MPC, the approach is robust to slow convergence, or even lack of convergence, in the nonlinear MPC.

The last contribution treated a design flow (Chapter 12), from modeling to implementation, for online path tracking with obstacle avoidance. We proposed a hierarchical control structure similar to that in Chapter 11. The main difference with Chapter 11 is that the approach in Chapter 12 assumes that the path is given. Also, it only uses convex problem formulations, which is possible since the intended application is path tracking under normal driving conditions. The convex formulations of the optimal-control problems give control structures that are possible to use with update rates of approximately 100 Hz. The validation was done on a specific robot, but it is easy to envision other scenarios for other types of vehicles, for example, automated highway driving for passenger cars.

14

Directions for Future Work

The aim with this chapter is to give directions for future work, both short term and long term. We divide the discussion into four categories:

1. Mobile robotics and manipulation
2. Out-of-sequence measurement processing
3. Vehicle localization and perception
4. Optimal control in automotive systems

Obviously, there are many ways to provide improvements in these four areas. We will only mention a few things that are tightly connected to the topics in this thesis.

14.1 Mobile Robotics and Manipulation

Chapters 6 and 12 discussed mobile robotics from two slightly different views. Chapter 6 was concerned with the localization and manipulation approach. The navigation of the mobile robot was not mentioned. Instead, the stationary manipulator was given responsibility for controlling the behavior. In contrast, Chapter 12 assumed a strict navigation problem. Thus, the obvious extension is to combine the two. By doing this, several issues immediately arise.

First, the motion of the two robots must be coordinated in a sensible way. One way is to do what we did in Chapter 6: allocate full responsibility to the manipulator. Intuitively, this is not the best way coordinate movements. One reason is that the mechanical construction is not fully utilized with this approach. Another reason is that the mobile robot's work space is much larger than that of the stationary robot. Hence, better performance is possible if allowing both robots to react on references, and

how to do this coordination is a path of research that is by no means fully explored.

Second, the approach to trajectory generation and obstacle avoidance in Chapter 12 is in the current state only designed for wheeled vehicles. However, the convex approach to trajectory generation was originally designed for stationary robots, and it would be interesting to investigate the simultaneous trajectory-generation and collision-avoidance problem.

14.2 Out-of-Sequence Measurement Processing

One of the things that remains doing is to actually use the OOSM algorithms in an online, physical setup. The results indicate that the algorithms are robust and provide good tracking performance. Nevertheless, that the algorithms perform well in simulation does not automatically mean that the same is true in experimental setups, so this has to be verified.

An obvious extension of the algorithms in Chapter 8 is to adapt them to the multi-OOSM problem. Currently, the algorithms only update the most recent estimate. For the backward-simulation approach, a straightforward approximate solution is to compute the updated weights and linear estimates at each time step. A problem with this approach is that it is computationally heavy; therefore, more efficient methods should be derived.

14.3 Vehicle Estimation

The combined slip and motion estimation scheme in Chapter 9 would be interesting to test in other vehicle setups as well. The method directly applies to four-wheeled mobile robots. Thus, it could be useful for, for example, mobile robots on outdoor, low-traction terrain.

It would be particularly interesting to see whether the improved slip estimation also leads to improved performance of existing safety systems. The control performance of braking systems, for example, is typically deficient when the road conditions change rapidly. Typical cases are when a subset of the wheels encounter ice patches.[1] It is possible, or even likely, that the proposed method can at least partly remedy this.

There are several ways to extend the method. In its current state, the method assumes flat ground. On banked roads, it is an advantage to separate the vehicle's roll angle from the road bank angle. Currently, this is not done in the method. An extension in this direction would therefore

[1] μ-split, or split friction.

be useful. Another extension is to allow for translations and rotations in three-dimensional space. This could be useful for, for example, planetary exploration, where the vehicles typically encounter different terrain types.

14.4 Optimal Control in Automotive Systems

The type of investigatory studies that were discussed in Chapter 10 can be adapted to other scenarios—for example, using other cost functions and maneuvers, in addition to other models. We only discussed results for a small subset of the available models, so there is much more to be done in terms of model comparisons.

The study in Chapter 10 on the impact of different road surfaces relied on a specific set of tire parameters. As mentioned several times, the tire-road interaction is complex, and it is likely that the parameters will be different for other tire configurations. An indication of this is that the force surfaces in Chapter 11 differ from those in Chapter 10. Hence, more studies have to be done using other experimental data, to investigate whether the results can be generalized.

Regarding the closed-loop vehicle control in Chapter 11, real vehicle trials have not been carried out. For allowing real tests, a real-time friendly implementation is necessary. In addition, the high-level trajectory generator should be accompanied with a method that steps in when the nonconvex optimization problem fails to converge. Although the method seems to be robust, a backup is needed for when it fails to converge.

The path tracking and obstacle avoidance in Chapter 12 is also suited for automotive systems, if assuming that the slip is small. How the method performs in automated highway driving scenarios, such as platooning, is an open question. In those cases, the path is extracted from road-map data. Extensions of the method in terms of velocity-dependent constraints is important for this to work, but this can easily be merged into the proposed method. The combination of the methods in Chapters 11 and 12 could also be investigated. One possibility is to use the convex trajectory generator in Chapter 12 as a backup for the trajectory generator in Chapter 11.

Bibliography

Agnoli, A., A. Chiuso, P. D'errico, A. Pegoraro, and L. Schenato (2008). "Sensor fusion and estimation strategies for data traffic reduction in rooted wireless sensor networks". In: *3rd International Symposium on Communications, Control, and Signal Processing*. La Valletta, Malta.

Åkesson, J. (2008). "Optimica—an extension of Modelica supporting dynamic optimization". In: *6th International Modelica Conference*. Bielefeld, Germany.

Åkesson, J., K.-E. Årzén, M. Gäfvert, T. Bergdahl, and H. Tummescheit (2010). "Modeling and optimization with Optimica and JModelica.org—Languages and tools for solving large-scale dynamic optimization problems". *Computers and Chemical Engineering* **34**:11, pp. 1737–1749.

Åkesson, L., G. Hedin, S. G. Robertz, and B. Magnusson (2012). "Instance-aware assemblies of services in pervasive computing". In: *27th Annual ACM Symposium on Applied Computing*. Trento, Italy.

Alam, A., A. Gattami, and K. Johansson (2010). "An experimental study on the fuel reduction potential of heavy duty vehicle platooning". In: *13th International IEEE Conference on Intelligent Transportation Systems*. Madeira Island, Portugal.

Albanesi, C., M. Bossi, L. Magni, J. Paderno, F. Pretolani, P. Kuehl, and M. Diehl (2006). "Optimization of the start-up procedure of a combined cycle power plant". In: *45th IEEE Conference on Decision and Control*. San Diego, California.

Ali, M., P. Falcone, C. Olsson, and J. Sjöberg (2013). "Predictive prevention of loss of vehicle control for roadway departure avoidance". *IEEE Transactions on Intelligent Transportation Systems* **14**:1, pp. 56–68.

Anderson, B. D. O. and J. B. Moore (1979). *Optimal Filtering*. Prentice Hall, Englewood Cliffs, New Jersey.

Anderson, S., S. Peters, T. Pilutti, and K. Iagnemma (2010). "An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios". *International Journal of Vehicle Autonomous Systems* **8**, pp. 190–216.

Anderson, S., S. Karumanchi, and K. Iagnemma (2012). "Constraint-based planning and control for safe, semi-autonomous operation of vehicles". In: *2012 IEEE Intelligent Vehicles Symposium*. Alcalá de Henares, Spain.

Andersson, J., J. Åkesson, and M. Diehl (2012). "CasADi – A symbolic package for automatic differentiation and optimal control". In: Forth, S. et al. (Eds.). *Recent Advances in Algorithmic Differentiation*. Lecture Notes in Computational Science and Engineering. Springer Verlag, Berlin Heidelberg, Germany.

Andreasson, J. (2009). "Enhancing active safety by extending controllability — How much can be gained?" In: *2009 IEEE Intelligent Vehicles Symposium*. Xi'an, Shaanxi, China.

Ardeshiri, T, M Norrlöf, J Löfberg, and A Hansson (2011). "Convex optimization approach for time-optimal path tracking of robots with speed dependent constraints". In: *18th IFAC World Congress*. Milan, Italy.

Armesto, L., S. Chroust, M. Vincze, and J. Tornero (2004). "Multi-rate fusion with vision and inertial sensors". In: *2004 IEEE International Conference on Robotics and Automation*. New Orleans, Louisiana.

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

Åström, K. J. and T. Hägglund (2005). *Advanced PID Control*. ISA - The Instrumentation, Systems, and Automation Society, Research Triangle Park, North Carolina.

Åström, K. J. and R. M. Murray (2008). *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, Princeton, Oxford.

Åström, K. J. and B. Wittenmark (1997). *Computer-Controlled Systems*. 3rd ed. Prentice Hall, Upper Sadle River, New Jersey.

Bar-Shalom, Y. (2002). "Update with out-of-sequence measurements in tracking: exact solution". *IEEE Transactions on Aerospace and Electronic Systems* **38**:3, pp. 769 –777.

Bar-Shalom, Y., X. Li, and T. Kirubarajan (2001). *Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software*. John Wiley & Sons, Inc., Hoboken, New Jersey.

Bar-Shalom, Y., H. Chen, and M. Mallick (2004). "One-step solution for the multistep out-of-sequence-measurement problem in tracking". *IEEE Transactions on Aerospace and Electronic Systems* **40**:1, pp. 27–37.

Berntorp, K. (2008). *ESP for Suppression of Jackknifing in an Articulated Bus*. Master's Thesis ISRN LUTFD2/TFRT--5831--SE. Department of Automatic Control, Lund University, Sweden.

Berntorp, K. (2013). *Derivation of a Six Degrees-of-Freedom Ground-Vehicle Model for Automotive Applications*. Technical Report ISRN LUTFD2/TFRT--7627--SE. Department of Automatic Control, Lund University, Sweden.

Berntorp, K. and J. Nordh (2014). "Rao-Blackwellized particle smoothing for occupancy-grid based SLAM using low-cost sensors". In: *19th IFAC World Congress*. Cape Town, South Africa. Accepted.

Berntorp, K., K.-E. Årzén, and A. Robertsson (2011). "Sensor fusion for motion estimation of mobile robots with compensation for out-of-sequence measurements". In: *11th International Conference on Control, Automation and Systems*. Gyeonggi-do, Korea.

Berntorp, K., K.-E. Årzén, and A. Robertsson (2012). "Mobile manipulation with a kinematically redundant manipulator for a pick-and-place scenario". In: *2012 IEEE Multi-Conference on Systems and Control*. Dubrovnik, Croatia.

Berntorp, K., B. Olofsson, K. Lundahl, B. Bernhardsson, and L. Nielsen (2013). "Models and methodology for optimal vehicle maneuvers applied to a hairpin turn". In: *2013 American Control Conference*. Washington, DC.

Berntorp, K., B. Olofsson, K. Lundahl, and L. Nielsen (2014). "Models and methodology for optimal trajectory generation in safety-critical road-vehicle maneuvers". *Vehicle System Dynamics*. Submitted.

Beskos, A., D. Crisan, and A. Jasra (2011). "On the stability of sequential Monte Carlo methods in high dimensions". *ArXiv e-prints*. arXiv: 1103.3965.

Biegler, L. T. (2010). *Nonlinear programming : concepts, algorithms and applications to chemical processes*. MOS-SIAM series on optimization. Society for industrial and applied mathematics, Philadelphia, Pennsylvania.

Binder, T., L. Blank, H. Bock, R. Bulirsch, W. Dahmen, M. Diehl, T. Kronseder, W. Marquardt, J. Schlöder, and O. Stryk (2001). "Introduction to model based optimization of chemical processes on moving horizons". In: Grötschel, M. et al. (Eds.). *Online Optimization of Large*

Scale Systems: State of the Art. Springer Verlag, Berlin Heidelberg, Germany, pp. 295–340.

Biswas, K. and A. Mahalanabis (1973). "Suboptimal algorithms for nonlinear smoothing". *IEEE Transactions on Aerospace and Electronic Systems* **9**:4, pp. 529–534.

Bittencourt, A. C., E. Wernholt, S. Sander-Tavallaey, and T. Brogårdh (2010). "An extended friction model to capture load and temperature effects in robot joints". In: *2010 IEEE/RJS International Conference on Intelligent Robots and Systems*. Taipei, Taiwan.

Blackman, S. and R. Popoli (1999). *Design and analysis of modern tracking systems*. Artech House, Norwood, Massachusetts.

Blackwell, D. (1947). "Conditional expectation and unbiased sequential estimation". *The Annals of Mathematical Statistics* **18**:1, pp. 105–110.

Blomdell, A., G. Bolmsjö, T. Brogårdh, P. Cederberg, M. Isaksson, R. Johansson, M. Haage, K. Nilsson, M. Olsson, T. Olsson, A. Robertsson, and J. Wang (2005). "Extending an industrial robot controller–implementation and applications of a fast open sensor interface". *IEEE Robotics and Automation Magazine* **12**:3, pp. 85–94.

Blomdell, A., I. Dressler, K. Nilsson, and A. Robertsson (2010). "Flexible application development and high-performance motion control based on external sensing and reconfiguration of ABB industrial robot controllers". In: *ICRA 2010 Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications*. Anchorage, Alaska.

Bobrow, J. E., S Dubowsky, and J. S. Gibson (1985). "Time-optimal control of robotic manipulators along specified paths". *The International Journal of Robotics Research* **4**:3, pp. 3–17.

Boyd, S. and L. Vandenberghe (2008). *Convex Optimization*. 6th ed. Cambridge University Press, New York.

Braghin, F., F. Cheli, and E Sabbioni (2006). "Environmental effects on Pacejka's scaling factors". *Vehicle System Dynamics* **44**:7, pp. 547–568.

Burton, D., A. Delaney, S. Newstead, D. Logan, and B. Fildes (2004). *Evaluation of Anti-lock Braking Systems Effectiveness*. Research Report No. 04/01. Royal Automobile Club of Victoria Ltd.

Carlson, C. R. and J. C. Gerdes (2005). "Consistent nonlinear estimation of longitudinal tire stiffness and effective radius". *IEEE Transactions on Control Systems Technology* **13**:6, pp. 1010–1020.

Carriker, W., P. Khosla, and B. Krogh (1989). "An approach for coordinating mobility and manipulation". In: *1989 International Conference on Systems Engineering*. Dayton, Ohio.

Casanova, D (2000). *On minimum time vehicle manoeuvring: The theoretical optimal lap*. PhD thesis. Cranfield University, Cranfield, United Kingdom.

Cellier, F. E. and E. Kofman (2006). *Continuous System Simulation*. Springer, New York.

Cervantes, A. and L. T. Biegler (2009). "Optimization strategies for dynamic systems". In: *Encyclopedia of Optimization*, pp. 2847–2858.

Chakraborty, I., P. Tsiotras, and J. Lu (2011). "Vehicle posture control through aggressive maneuvering for mitigation of T-bone collisions". In: *50th IEEE Conference on Decision and Control*. Orlando, Florida.

Chakraborty, I., P. Tsiotras, and R. S. Diaz (2013). "Time-optimal vehicle posture control to mitigate unavoidable collisions using conventional control inputs". In: *2013 American Control Conference*. Washington, DC.

Chen, Y. and A. A. Desrochers (1989). "Structure of minimum-time control law for robotic manipulators with constrained paths". In: *1989 IEEE International Conference on Robotics and Automation*. Scottsdale, Arizona.

Chib, S. and E. Greenberg (1995). "Understanding the Metropolis-Hastings algorithm". *The American Statistician* **49**:4, pp. 327–335.

Choi, J.-W., R. E. Curry, and G. H. Elkaim (2009). "Obstacle avoiding real-time trajectory generation and control of omnidirectional vehicles". In: *2009 American Control Conference*. St. Louis, Michigan.

Choi, M., W. K. Chung, S. Yi, W. Choi, and K.-S. Chang (2007). "State estimation with delayed observations considering uncertainty in time". In: *4th International Conference on Ubiquitous Robots and Ambient Intelligence*. Pohang, Korea.

Clark, S., R. Dodge, and G. Nybakken (1972). *An evaluation of string theory for the prediction of dynamic tire properties using scale model aircraft tires*. NASA contractor report 2058. National Aeronautics and Space Administration.

Connette, C. P., C. Parlitz, M. Hägele, and A. Verl (2009). "Singularity avoidance for over-actuated, pseudo-omnidirectional, wheeled mobile robots". In: *2009 IEEE International Conference on Robotics and Automation*. Kobe, Japan.

Corke, P., J. Lobo, and J. Dias (2007). "An introduction to inertial and visual sensing". *The International Journal of Robotics* **26**, pp. 519–535.

Dahl, O. (1992). *Path Constrained Robot Control*. PhD thesis ISRN LUTFD2/TFRT--1038--SE. Department of Automatic Control, Lund University, Sweden.

Dahl, O. (1993). “Path constrained motion optimization for rigid and flexible joint robots”. In: *1993 IEEE International Conference on Robotics and Automation*. Atlanta, Georgia.

Dahl, O. and L. Nielsen (1989). “Torque limited path following by on-line trajectory time scaling”. In: *1989 IEEE International Conference on Robotics and Automation*. Scottsdale, Arizona.

De Schutter, J., T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx (2007). “Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty”. *The International Journal of Robotics Research* **26**:5, pp. 433–455.

Debrouwere, F., W. Van Loock, G. Pipeleers, Q. Tran Dinh, M. Diehl, J. De Schutter, and J. Swevers (2013). “Time-optimal path following for robots with convex-concave constraints using sequential convex programming”. *IEEE Transactions on Robotics* **29**:6, pp. 1485–1495.

Del Re, L., F. Allgöwer, L. Glielmo, C. Guardiola, and I. E. Kolmanovsky (2010). *Automotive Model Predictive Control: Models, Methods and Applications*. Lecture notes in Control and Information Sciences. Springer Verlag, Berlin Heidelberg, Germany.

Dennis, J. and R. Schnabel (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Classics in Applied Mathematics, Society for Industrial and Applied Mathematics.

DeSouza, G. N. and A. C. Kak (2002). “Vision for mobile robot navigation: a survey”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**:2, pp. 237–267.

Di Cairano, S., D. Yanakiev, A. Bemporad, I. Kolmanovsky, and D. Hrovat (2012). “Model predictive idle speed control: design, analysis, and experimental evaluation”. *IEEE Transactions on Control Systems Technology* **20**:1, pp. 84–97.

Dingle, P. and L. Guzzella (2010). “Optimal emergency maneuvers on highways for passenger vehicles with two- and four-wheel active steering”. In: *2010 American Control Conference*. Baltimore, Maryland.

Douc, R. (2005). “Comparison of resampling schemes for particle filtering”. In: *4th International Symposium on Image and Signal Processing and Analysis*. Zagreb, Croatia.

Douc, R., A. Garivier, E. Moulines, and J. Olsson (2012). “Sequential Monte Carlo smoothing for general state space hidden Markov models”. *Annals of Applied Probability* **21**, pp. 2109–2145.

Doucet, A., S. Godsill, and C. Andrieu (2000). “On sequential Monte Carlo sampling methods for Bayesian filtering”. *Statistics and Computing* **10**:3, pp. 197–208.

Ellis, J. R. (1994). *Vehicle Handling Dynamics*. Mechanical Engineering Publications, London, United Kingdom.

Esmailzadeh, E., A. Goodarzi, and G. Vossoughi (2003). "Optimal yaw moment control law for improved vehicle handling". *Mechatronics* **13**:7, pp. 659–675.

Falcone, P., F. Borrelli, J. Asgari, H. Tseng, and D. Hrovat (2007). "Predictive active steering control for autonomous vehicle systems". *IEEE Transactions on Control Systems Technology* **15**:3, pp. 566–580.

Falcone, P., H. Eric Tseng, F. Borrelli, J. Asgari, and D. Hrovat (2008). "MPC-based yaw and lateral stabilisation via active front steering and braking". *Vehicle System Dynamics* **46**, pp. 611–628.

Fong, W., S. J. Godsill, A. Doucet, and M. West (2002). "Monte Carlo smoothing with application to audio signal enhancement." *IEEE Transactions on Signal Processing* **50**:2, pp. 438–449.

Fors, D. S. (2009). *Assemblies of Pervasive Services*. PhD thesis. Department of Computer Science, Lund University.

Fox, D., W. Burgard, and S. Thrun (1997). "The dynamic window approach to collision avoidance". *IEEE Robotics & Automation Magazine* **4**:1, pp. 23–33.

Fox, D., S. Thrun, F. Dellaert, and W. Burgard (2000). "Particle filters for mobile robot localization". In: Doucet, A. et al. (Eds.). *Sequential Monte Carlo Methods in Practice*. Springer Verlag, New York.

Frasch, J., A. Gray, M. Zanon, H. Ferreau, S. Sager, F. Borrelli, and M. Diehl (2013). "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles". In: *2013 European Control Conference*. Zurich, Switzerland.

Funke, J., P. Theodosis, R. Hindiyeh, G. Stanek, K. Kritatakirana, C. Gerdes, D. Langer, M. Hernandez, B. Muller-Bessler, and B. Huhnke (2012). "Up to the limits: Autonomous Audi TTS". In: *2012 IEEE Intelligent Vehicles Symposium*. Alcalá de Henares, Spain.

Gäfvert, M. (2003). *Topics in Modeling, Control, and Implementation in Automotive Systems*. PhD thesis ISRN LUTFD2/TFRT--1066--SE. Department of Automatic Control, Lund University, Sweden.

Gao, Y., T. Lin, F. Borrelli, E. Tseng, and D. Hrovat (2010). "Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads". In: *ASME 2010 Dynamic Systems and Control Conference*. Cambridge, Massachusetts.

Gauss, C. F. and C. H. Davis (1857). *Theory of the motion of the heavenly bodies moving about the sun in conic sections*. Making of America Project. Little, Brown and company, Boston, Massachusetts.

Giselsson, P. (2014). “Improved fast dual gradient methods for embedded model predictive control”. In: *19th IFAC World Congress*. Cape Town, South Africa. Accepted.

Godsill, S. J., A. Doucet, and M. West (2004). “Monte Carlo smoothing for non-linear time series”. *Journal of the American Statistical Association* **99**, pp. 156–168.

Golub, G. H. and C. F. Van Loan (1996). *Matrix Computations*. 3rd ed. The Johns Hopkins University Press, Baltimore, Maryland.

Gordon, N. J., D. J. Salmond, and A. F. M. Smith (1993). “Novel approach to nonlinear/non-Gaussian Bayesian state estimation”. *IEE Proceedings F, Radar and Signal Processing* **140**:2, pp. 107–113.

Govaers, F. and W. Koch (2012). “A generalized solution to smoothing and out-of-sequence processing”. In: *15th International Conference on Information Fusion*. Singapore.

Grant, M. and S. Boyd (2014). *CVX: matlab software for disciplined convex programming, version 2.0*. http://cvxr.com/cvx.

Grimm, G., M. J. Messina, S. E. Tuna, and A. R. Teel (2005). “Model predictive control: for want of a local control Lyapunov function, all is not lost”. *IEEE Transactions on Automatic Control* **50**:5, pp. 546–558.

Grip, H., L. Imsland, T. Johansen, J. Kalkkuhl, and A. Suissa (2009). “Estimation of road inclination and bank angle in automotive vehicles”. In: *2009 American Control Conference*. St. Louis, Missouri.

Grisetti, G., C. Stachniss, and W. Burgard (2005). “Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling”. In: *2005 IEEE International Conference on Robotics and Automation*. Barcelona, Spain.

Grisetti, G., C. Stachniss, and W. Burgard (2007). “Improved techniques for grid mapping with Rao-Blackwellized particle filters”. *IEEE Transactions on Robotics* **23**, pp. 34–46.

Grover, P. and C. Andersson (2012). “Optimized three-body gravity assists and manifold transfers in end-to-end lunar mission design”. In: *22nd AAS/AIAA Space Flight Mechanics Meeting*. Charleston, South Carolina.

Grüne, L. and A. Rantzer (2008). “On the infinite horizon performance of receding horizon controllers”. *IEEE Transactions on Automatic Control* **53**:9, pp. 2100–2111.

Gustafsson, F. (2009). “Automotive safety systems”. *IEEE Signal Processing Magazine* **26**:4, pp. 32–47.

Gustafsson, F. (2010a). “Particle filter theory and practice with positioning applications”. *IEEE Aerospace and Electronics Systems Magazine* **25**:7, pp. 53–82.

Gustafsson, F. (2010b). *Statistical Sensor Fusion*. Utbildningshuset/Studentlitteratur, Lund, Sweden.

Gustafsson, F., F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund (2002). “Particle filters for positioning, navigation, and tracking”. *IEEE Transactions on Signal Processing* **50**:2, pp. 425–437.

Gustafsson, F. (1997). “Slip-based tire-road friction estimation”. *Automatica* **33** (6), pp. 1087–1099.

Hähnel, D, W Burgard, D Fox, and S Thrun (2003). “A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements”. In: *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Las Vegas, Nevada.

Hamilton, W. R. (1844). “On Quaternions, or on a new system of imaginaries in algebra”. *Philosophical Magazine* **25**:3, pp. 489–495.

Hammersley, J. M. and K. W. Morton (1954). “Poor man’s Monte Carlo”. *Journal of the Royal Statistical Society. Series B (Methodological)* **16**:1, pp. 23–38.

Hanebeck, U., C. Fischer, and G. Schmidt (1997). “Roman: a mobile robotic assistant for indoor service applications”. In: *1997 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Grenoble, France.

Härkegård, O. (2003). *Backstepping and Control Allocation with Applications to Flight Control*. PhD thesis 820. Department of Electrical Engineering, Linköping University, Sweden.

Henningsson, M. (2012). *Data-Rich Multivariable Control of Heavy-Duty Engines*. PhD thesis ISRN LUTFD2/TFRT--1092--SE. Department of Automatic Control, Lund University, Sweden.

Hermans, R. M., M. Lazar, I. V. Kolmanovsky, and S. Di Cairano (2013). “Horizon-1 predictive control of automotive electromagnetic actuators”. *IEEE Transactions on Control Systems Technology* **21**:5, pp. 1652–1665.

Hogan, N. (1984). “Impedance control: an approach to manipulation”. In: *1984 American Control Conference*. San Diego, California.

Hol, J. D., T. B. Schön, and F. Gustafsson (2006). “On resampling algorithms for particle filters”. In: *2006 IEEE Nonlinear Statistical Signal Processing Workshop*. Cambridge, United Kingdom.

Howard, T., C. Green, and A. Kelly (2009). "Receding horizon model-predictive control for mobile robot navigation of intricate paths". In: *7th International Conference on Field and Service Robotics*. Cambridge, Massachusetts.

HSL (2014). *A collection of Fortran codes for large scale scientific computation*. `http://www.hsl.rl.ac.uk`.

Hsu, F.-Y. and L.-C. Fu (2000). "Intelligent robot deburring using adaptive fuzzy hybrid position/force control". *IEEE Transactions on Robotics and Automation* **16**:4, pp. 325–335.

Iagnemma, K. and S. Dubowsky (2004). *Mobile Robots in Rough Terrain - Estimation, Motion Planning, and Control with Application to Planetary Rovers*. Vol. 12. Springer Tracts in Advanced Robotics. Springer.

Iagnemma, K. and C. C. Ward (2009). "Classification-based wheel slip detection and detector fusion for mobile robots on outdoor terrain." *Autonomous Robots* **26**:1, pp. 33–46.

IEEE (1996). *IEEE Standard Specification Format Guide and Test Procedure for Single-axis Laser Gyros*. IEEE Std.

Imsland, L., T. A. Johansen, T. I. Fossen, H. F. Grip, J. Kalkkuhl, and A. Suissa (2006). "Vehicle velocity estimation using nonlinear observers". *Automatica* **42**:12, pp. 2091–2103.

Isermann, R. (2006). *Fahrdynamik-Regelung: Modellbildung, Fahrerassistenzsysteme, Mechatronik*. Vieweg-Verlag, Wiesbaden, Germany.

Jeon, J., R. Cowlagi, S. Peters, S. Karaman, E. Frazzoli, P. Tsiotras, and K. Iagnemma (2013). "Optimal motion planning with the half-car dynamical model for autonomous high-speed driving". In: *2013 American Control Conference*. Washington, DC.

Jia, Z., A. Balasuriya, and S. Challa (2008). "Sensor fusion-based visual target tracking for autonomous vehicles with the out-of-sequence measurements solution". *Robotics and Autonomous Systems* **56**:2, pp. 157–176.

JModelica.org (2014). URL: http://www.jmodelica.org.

Johansen, T. A., I. Petersen, J. Kalkkuhl, and J. Lüdemann (2003). "Gain-scheduled wheel slip control in automotive brake systems." *IEEE Transactions on Control Systems Technology* **11**:6, pp. 799–811.

Johansson, K. H., M. Törngren, and L. Nielsen (2005). "Vehicle application of controller area network". In: In The Handbook of Networked and Embedded Control Systems, edited by D. Hristu-Varsakelis and W. S. Levine. Springer Basel AG, Basel, Switzerland.

Johansson, R. (2009). *System Modeling and Identification*. 2nd ed. Prentice Hall, Englewood Cliffs, New Jersey.

Jonasson, M., J. Andreasson, S. Solyom, B. Jacobson, and A. Stensson Trigell (2011). "Utilization of actuators to improve vehicle stability at the limit : from hydraulic brakes toward electric propulsion". *Journal of Dynamic Systems Measurement, and Control* **133**:5, pp. 051003–1–10.

Julier, S. and J. Uhlmann (2004). "Unscented filtering and nonlinear estimation". *Proceedings of the IEEE* **92**:3, pp. 401–422.

Kalman, Rudolph, and Emil (1960). "A New Approach to Linear Filtering and Prediction Problems". *Transactions of the ASME–Journal of Basic Engineering* **82**:Series D, pp. 35–45.

Kanjanawanishkul, K. and A. Zell (2009). "Path following for an omnidirectional mobile robot based on model predictive control". In: *2009 IEEE International Conference on Robotics and Automation*. Kobe, Japan.

Karlsson, R., T. Schön, and F. Gustafsson (2005). "Complexity analysis of the marginalized particle filter". *IEEE Transactions on Acoustics, Speech, and Signal Processing* **53**:11, pp. 4408–4411.

Karlsson, R. and F. Gustafsson (2001). "Monte Carlo data association for multiple target tracking". In: *IEE Target Tracking: Algorithms and Applications*. Enschede, The Netherlands.

Kelly, D. P. and R. S. Sharp (2010). "Time-optimal control of the race car: a numerical method to emulate the ideal driver". *Vehicle System Dynamics* **48**:12, pp. 1461–1474.

Khatib, O., K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, and A. Casal (1996). "Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation". In: *1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Osaka, Japan.

Khatib, O. (1986). "Real-time obstacle avoidance for manipulators and mobile robots". *The International Journal of Robotics Research* **5**:1, pp. 90–98.

Khatib, O. (1999). "Mobile manipulation: the robotic assistant." *Robotics and Autonomous Systems* **26**, pp. 175–183.

Kiencke, U. and L. Nielsen (2005). *Automotive Control Systems—For Engine, Driveline and Vehicle*. 2nd ed. Springer-Verlag, Berlin Heidelberg, Germany.

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

Klančar, G. and I. Škrjanc (2007). "Tracking-error model-based predictive control for mobile robots in real time". *Robotics and Autonomous Systems* **55**:6, pp. 460–469.

Kobayashi, K., K. C.Cheok, and K. Watanabe (1995). “Estimation of absolute vehicle speed using fuzzy logic rule-based Kalman filter”. In: *1995 American Control Conference*. Seattle, Washington.

Koch, W. (2009). “On accumulated state densities with applications to out-of-sequence measurement processing”. In: *12th International Conference on Information Fusion*. Seattle, Washington.

Kock, S., T. Vittor, B. Matthias, H. Jerregard, M. Kallman, I. Lundberg, R. Mellander, and M. Hedelind (2011). “Robot concept for scalable, flexible assembly automation: a technology study on a harmless dual-armed robot”. In: *International Symposium on Assembly and Manufacturing*. Tampere, Finland.

Kröger, T. (2011). “Opening the door to new sensor-based robot applications—the reflexxes motion libraries”. In: *2011 IEEE International Conference on Robotics and Automation*. Shanghai, China.

Kullback, S. and R. A. Leibler (1951). “On information and sufficiency”. *Annals of Mathematical Statistics* **22**, pp. 49–86.

Laplace, P. S. (1986). “Memoir on the probability of the causes of events”. *Statistical Science* **1**:3, pp. 364–378.

Larsson, P.-O. (2011). *Optimization of Low-Level Controllers and High-Level Polymer Grade Changes*. PhD thesis ISRN LUTFD2/TFRT--1088--SE. Department of Automatic Control, Lund University, Sweden.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press, Cambridge, United Kingdom.

Lghani, M. (2012). “Road Bank and Vehicle Roll Angles Estimation Based on Proportional-Integral Observer”. In: *8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*. Mexico City, Mexico.

Lie, A., C. Tingvall, M. Krafft, and A. Kullgren (2005). “The effectiveness of Electronic Stability Control in reducing real life crashes and injuries”. In: *19th International Conference on the Enhanced Safety of Vehicles Conference*. Gothenburg, Sweden.

Liebemann, K., K. Meder, J. Schuh, and G. Nenninger (2005). “Safety and performance enhancement: the Bosch Electronic Stability Control”. In: *SAE Convergence International Congress & Exposition On Transportation Electronics*. Detroit, Michigan.

Linderoth, M. (2013). *On Robotic Work-Space Sensing and Control*. PhD thesis ISRN LUTFD2/TFRT--1098--SE. Department of Automatic Control, Lund University, Sweden.

Lindsten, F. (2013). *Particle filters and Markov Chains for Learning of Dynamical Systems*. PhD thesis 1530. Department of Electrical Engineering, Linköping University, Sweden.

Lindsten, F. and T. B. Schön (2010). "Identification of mixed linear/nonlinear state-space models". In: *49th IEEE Conference on Decision and Control*. Grand Wailea, Maui, Hawaii.

Lindsten, F. and T. B. Schön (2011). *Rao-Blackwellised Particle Smoothers for Mixed Linear/Nonlinear State-Space Models*. Technical Report LiTH-ISY-R-3018. Department of Electrical Engineering, Linköping University, Sweden.

Lindsten, F. and T. B. Schön (2013). "Backward simulation methods for Monte Carlo statistical inference". *Foundations and Trends in Machine Learning* **6**:1, pp. 1–143.

Lindsten, F., T. B. Schön, and J. Olsson (2011). "An explicit variance reduction expression for the Rao-Blackwellised particle filter". In: *18th IFAC World Congress*. Milan, Italy.

Lindsten, F., P. Bunch, S. J. Godsill, and T. B. Schön (2013). "Rao-Blackwellized particle smoothers for mixed linear/nonlinear state-space models". In: *38th International Conference on Acoustics, Speech, and Signal Processing*. Vancouver, Canada.

Lipp, T. and S. Boyd (2014). "Minimum-time speed optimization over a fixed path". *International Journal of Control* **87**:6, pp. 1297–1311.

Liu, X., B. Oreshkin, and M. Coates (2010). "Efficient delay-tolerant particle filtering through selective processing of out-of-sequence measurements". In: *13th International Conference on Information Fusion*. Edinburgh, United Kingdom.

Lundahl, K., K. Berntorp, B. Olofsson, J. Åslund, and L. Nielsen (2013a). "Studying the influence of roll and pitch dynamics in optimal road-vehicle maneuvers". In: *23rd International Symposium on Dynamics of Vehicles on Roads and Tracks*. Qingdao, China.

Lundahl, K., J. Åslund, and L. Nielsen (2013b). *Vehicle Dynamics Platform, Experiments, and Modeling Aiming at Critical Maneuver Handling*. Technical Report LiTH-R-3064. Department of Electrical Engineering, Linköping University, Sweden.

Lundahl, K., B. Olofsson, K. Berntorp, J. Åslund, and L. Nielsen (2014). "Towards lane-keeping electronic stability control for road-vehicles". In: *19th IFAC World Congress*. Cape Town, South Africa. Accepted.

Lundquist, C. and T. B. Schön (2011). "Joint ego-motion and road geometry estimation". *Information Fusion* **12**:4, pp. 253–263.

Lundquist, C., R. Karlsson, E. Ozkan, and F. Gustafsson (2014). "Tire radii estimation using a marginalized particle filter". *IEEE Transactions on Intelligent Transportation Systems* **15**:99, pp. 663–672.

Lundquist, C. (2011). *Sensor Fusion for Automotive Applications*. PhD thesis 1409. Department of Electrical Engineering, Linköping University, Sweden, Sweden.

Maciejowski, J. (2002). *Predictive Control with Constraints*. Prentice-Hall, Englewood Cliffs, New Jersey.

Magnusson, F. and J. Åkesson (2012). "Collocation methods for optimization in a Modelica environment". In: *9th International Modelica Conference*. Munich, Germany.

Magnusson, F., K. Berntorp, B. Olofsson, and J. Åkesson (2014). "Symbolic transformations of dynamic optimization problems". In: *10th International Modelica Conference*. Lund, Sweden.

Mallick, M. and A. Marrs (2003). "Comparison of the KF and particle filter based out-of-sequence measurement filtering algorithms". In: *6th International Conference on Information Fusion*. Cairns, Queensland, Australia.

Mallick, M., T. Kirubarajan, and S. Arulampalam (2002). "Out-of-sequence measurement processing for tracking ground target using particle filters". In: *IEEE Aerospace Conference*. Big Sky, Montana.

Marshall, A. W. (1956). *The Use of Multistage Sampling Schemes in Monte Carlo Computations*. Ed. by H. A. Meyer. Symposium on Monte Carlo Methods. John Wiley & Sons, Inc., Hoboken, New Jersey.

Maskell, S. R., R. G. Everitt, R. Wright, and M. Briers (2006). "Multi-target out-of-sequence data association: tracking using graphical models". *Information Fusion* **7**:4, pp. 434–447.

Mattingley, J. and S. Boyd (2012). "CVXGEN: a code generator for embedded convex optimization". *Optimization and Engineering* **13**:1, pp. 1–27.

Mauthner, M., W. Elmenreich, A. Kirchner, and D. Boesel (2006). "Out-of-sequence measurement treatment in sensor fusion applications: buffering versus advanced algorithms". In: *Workshop Fahrerassistenzsysteme*. Löwenstein/Hößlinsülz, Germany.

Mayne, D. Q., J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert (2000). "Constrained model predictive control: Stability and optimality". *Automatica* **36**:6, pp. 789–814.

Meeussen, W., M. Wise, S. Glaser, S. Chitta, C. McGann, P. Mihelich, E. Marder-Eppstein, M. Muja, V. Eruhimov, T. Foote, J. Hsu, R. B. Rusu, B. Marthi, G. Bradski, K. Konolige, B. P. Gerkey, and E. Berger (2010). "Autonomous door opening and plugging in with a personal robot". In: *2010 IEEE International Conference on Robotics and Automation*. Anchorage, Alaska.

Miller, S., B. Youngberg, A. Millie, S. Patrich, and J. C. Gerdes (2001). "Calculating longitudinal wheel slip and tire parameters using GPS velocity". In: *2001 American Control Conference*. Arlington, Virginia.

Modelica Association (2014). URL: http://www.modelica.org.

Montemerlo, M. and S. Thrun (2003). "Simultaneous localization and mapping with unknown data association using fastSLAM". In: *2003 IEEE International Conference on Robotics and Automation*. Taipei, Taiwan.

Muntzinger, M. M., M. Aeberhard, S. Zuther, M. Maehlisch, M. R. Schmid, and K. Dietmayer (2010). "Reliable automotive pre-crash system with out-of-sequence measurement processing". In: *2010 IEEE Intelligent Vehicles Symposium*. San Diego, California.

Neumann, J. von (1951). "Various techniques used in connection with random digits". *Journal of Research of the National Bureau of Standards. Applied Mathematics Series* **12**, pp. 36–38.

NHTSA (2011). *Traffic Safety Facts 2010, A Compilation of Motor Vehicle Crash Data from the Fatality Analysis Reporting System and the General Estimates System*. DOT HS 811 659. National Highway Traffic Safety Administration.

Norén, C. (2013). *Path Planning for Autonomous Heavy Duty Vehicles using Nonlinear Model Predictive Control*. Master's Thesis LiTH-ISY-EX–13/4707–SE. Department of Electrical Engineering, Linköping University, Sweden, Linköping, Sweden.

Nuetzi, G, S Weiss, D Scaramuzza, and R Siegwart (2011). "Fusion of IMU and vision for absolute scale estimation in monocular SLAM". *Journal of Intelligent and Robotic Systems* **61**, pp. 287–299.

Ojeda, L., D. Cruz, G. Reina, J. Borenstein, and S. Member (2005). "Current-based slippage detection and odometry correction for mobile robots and planetary rovers". *IEEE Transactions on Robotics* **22**:2, pp. 366–378.

Olofsson, B., K. Lundahl, K. Berntorp, and L. Nielsen (2013). "An investigation of optimal vehicle maneuvers for different road conditions". In: *7th IFAC Symposium on Advances in Automotive Control*. Tokyo, Japan.

Olsson, T. (2007). *High-Speed Vision and Force Feedback for Motion-Controlled Industrial Manipulators*. PhD thesis ISRN LUTFD2/TFRT--1078--SE. Department of Automatic Control, Lund University, Sweden.

Oreshkin, B., X. Liu, and M. Coates (2011). "Efficient delay-tolerant particle filtering". *IEEE Transactions on Signal Processing* **59**:7, pp. 3369–3381.

Orguner, U. and F. Gustafsson (2008). "Storage efficient particle filters for the out of sequence measurement problem". In: *11th International Conference on Information Fusion*. Cologne, Germany.

Orocos (2014). *Orocos–Open robot control software*. http://www.orocos.org.

Orton, M. and A. Marrs (2001). "A Bayesian approach to multi-target tracking and data fusion with out-of-sequence measurements". In: *IEE Target Tracking: Algorithms and Applications*. Enschede, The Netherlands.

Orton, M. and A. Marrs (2005). "Particle filters for tracking with out-of-sequence measurements". *IEEE Transactions on Aerospace and Electronic Systems* **41**:2, pp. 693 –702.

Pacejka, H. B. (2006). *Tyre and Vehicle Dynamics*. 2nd ed. Butterworth-Heinemann, Oxford, United Kingdom.

Petersen, I. (2003). *Wheel Slip Control in ABS Brakes using Gain Scheduled Optimal Control with Constraints*. PhD Thesis ISBN 82-471-5593-1. Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway.

Pfeiffer, F. and R. Johanni (1987). "A concept for manipulator trajectory planning". *IEEE Journal of Robotics and Automation* **3**:2, pp. 115–123.

Pontryagin, L. S., V. G. Boltianski, R. V. Gamkrelidze, E. F. Mishchenko, and D. E. Brown (1964). *The mathematical theory of optimal processes*. A Pergamon Press book. London, Paris.

Python Software Foundation (2014). *Ctypes — A foreign function library for Python*. http://docs.python.org/2/library/ctypes.html.

Qu, Z., J. Wang, and C. E. Plaisted (2004). "A new analytical solution to mobile robot trajectory generation in the presence of moving obstacles". *IEEE Transactions on Robotics* **20**:6, pp. 978–993.

Quinlan, S. and O. Khatib (1993). "Elastic bands: connecting path planning and control". In: *1993 IEEE International Conference on Robotics and Automation*. Atlanta, Georgia.

Rajamani, R (2006). *Vehicle Dynamics and Control*. Springer-Verlag, Berlin Heidelberg, Germany.

Ranganathan, A., M. Kaess, and F. Dellaert (2007). "Fast 3D pose estimation with out-of-sequence measurements". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. San Diego, California.

Rauch, H. E., C. T. Striebel, and F. Tung (1965). "Maximum likelihood estimates of linear dynamic systems". *Journal of the American Institute of Aeronautics and Astronautics* **3**:8, pp. 1445–1450.

Rawlings, J. and D. Mayne (2009). *Model Predictive Control: Theory and Design*. Nob Hill Publishing, Madison, Wisconsin.

Ray, L. R. (1997). "Nonlinear tire force estimation and road friction identification: simulation and experiments," *Automatica* **33**:10, pp. 1819–1833.

Rebeschini, P. and R. van Handel (2013). "Can local particle filters beat the curse of dimensionality?" *ArXiv e-prints*. arXiv: 1301.6585.

Reif, K. and K.-H. Dietsche (2011). *Bosch Automotive Handbook*. 8th. Bosch Invented for life. Robert Bosch GmbH, Plochingen, Germany.

Reimpell, J. and J. Betzler (2005). *Fahrwerktechnik: Grundlagen*. 5th ed. Vogel Fachbuch. Vogel Verlag Und Druck, Würzburg, Germany.

Reina, G., L. Ojeda, A. Milella, J. Borenstein, and S. Member (2006). "Wheel slippage and sinkage detection for planetary rovers". *IEEE/ASME Transactions on Mechatronics* **11**, pp. 185–195.

Reiser, U., C. Connette, J. Fischer, J. Kubacki, A. Bubeck, F. Weisshardt, T. Jacobs, C. Parlitz, M. Hagele, and A. Verl (2009a). "Care-o-bot 3 - creating a product vision for service robot applications by integrating design and technology". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. St. Louis, Missouri.

Reiser, U., R. Klauser, C. Parlitz, and E. Verl (2009b). "Desire web 2.0-integration management and distributed software development for service robots". In: *14th International Conference on Advanced Robotics*. Munich, Germany.

Reynoso-Mora, P., W. Chen, and M. Tomizuka (2013). "On the time-optimal trajectory planning and control of robotic manipulators along predefined paths". In: *2013 American Control Conference*. Washington, DC.

Robert, C. and G. Casella (2004). *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer, New York.

Rong Li, X. and V. P. Jilkov (2001). "A survey of maneuvering target tracking - part III: measurement models". In: *SPIE Conference on Signal and Data Processing of Small Targets*. Baltimore, Marlyland.

Rong Li, X. and V. P. Jilkov (2003). “Survey of maneuvering target tracking - Part I: dynamic models”. *IEEE Transactions on Aerospace and Electronic Systems* **39**:4, pp. 1333–1364.

ROS (2014). *Robot Operating System*. http://www.ros.org.

Ryu, J. and J. C. Gerdes (2004). “Estimation of vehicle roll and road bank angle”. In: *2004 American Control Conference*. Boston, Massachusetts.

Sällberg, E., A. Lind, S. p. Velut, J. Å kesson, S. Gallardo Yances, and K. Link (2012). “Start-up optimization of a combined cycle power plant”. In: *9th International Modelica Conference*. Munich, Germany.

Särkkä, S., A. Vehtari, and J. Lampinen (2004). “Rao-Blackwellized Monte Carlo data association for multiple target tracking”. In: *7th International Conference on Information Fusion*. Stockholm, Sweden.

Särkkä, S., P. Bunch, and S. J. Godsill (2012). “A backward-simulation based Rao-Blackwellized particle smoother for conditionally linear Gaussian models”. In: *16th IFAC Symposium on System Identification*. Brussels, Belgium.

Savaresi, S. M. and M. Tanelli (2010). *Active Braking Control Systems Design for Vehicles*. Springer Verlag Gmbh, Wiesbaden, Germany.

Schindler, E. (2007). *Fahrdynamik: Grundlagen Des Lenkverhaltens Und Ihre Anwendung Für Fahrzeugregelsysteme*. Expert-Verlag, Renningen, Germany.

Schofield, B. (2008). *Model-Based Vehicle Dynamics Control for Active Safety*. PhD thesis ISRN LUTFD2/TFRT--1083--SE. Department of Automatic Control, Lund University, Sweden.

Schön, T. B. (2006). *Estimation of Nonlinear Dynamic Systems : Theory and Applications*. PhD thesis 998. Department of Electrical Engineering, Linköping University, Sweden.

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

Schön, T. B., F. Gustafsson, and R. Karlsson (2006). “The marginalized particle filter in practice”. In: *IEEE Aerospace Conference*. Big Sky, Montana.

Sharp, R. S. and H. Peng (2011). “Vehicle dynamics applications of optimal control theory”. *Vehicle System Dynamics* **49**:7, pp. 1073–1111.

Shen, X., Y. Zhu, E. Song, and Y. Luo (2009). “Optimal centralized update with multiple local out-of-sequence measurements”. *IEEE Transactions on Signal Processing* **57**:4, pp. 1551 –1562.

Shiller, Z. and Y.-R. Gwo (1991). "Dynamic motion planning of autonomous vehicles". *IEEE Transactions on Robotics and Automation* **7**:2, pp. 241–249.

Shiller, Z. and H.-H. Lu (1992). "Computation of path constrained time optimal motions with dynamic singularities". *Journal of Dynamic Systems, Measurement, and Control* **114**:1, pp. 34–40.

Shiller, Z. and S. Sundar (1998). "Emergency lane-change maneuvers of autonomous vehicles". *Journal of Dynamic Systems, Measurement, and Control* **120**:1, pp. 37–44.

Shin, K. G. and N. D. McKay (1985). "Minimum-time control of robotic manipulators with geometric path constraints". *IEEE Transactions on Automatic Control* **30**:6, pp. 531–541.

Shin, K. G. and N. D. McKay (1987). "Robust trajectory planning for robotic manipulators under payload uncertainties". *IEEE Transactions on Automatic Control* **32**:12, pp. 1044–1054.

Siciliano, B. and L. Villani (1999). *Robot Force Control*. Kluwer international series in engineering and computer science: Robotics: vision, manipulation and sensors. Springer, New York.

Solmaz, S., M. Akar, R. Shorten, and J. Kalkkuhl (2008). "Realtime multiple-model estimation of center of gravity position in automotive vehicles." *Vehicle System Dynamics* **46**:9, pp. 763–788.

Solyom, S. (2004). *Control of Systems with Limited Capacity*. PhD thesis ISRN LUTFD2/TFRT--1069--SE. Department of Automatic Control, Lund University, Sweden.

Spong, M. and S. Hutchinson (2006). *Robot Modeling and Control*. John Wiley & Sons, Inc., Hoboken, New Jersey.

Stigler, S. M. (1981). "Gauss and the invention of least squares". *The Annals of Statistics* **9**:3, pp. 465–474.

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2012). "Robotic assembly using a singularity-free orientation representation based on quaternions". In: *10th International IFAC Symposium on Robot Control*. Dubrovnik, Croatia.

Sundström, P., M. Jonasson, J. Andreasson, A. Stensson Trigell, and B. Jacobsson (2010). "Path and control optimisation for over-actuated vehicles in two safety-critical maneuvers". In: *10th International Symposium on Advanced Vehicle Control*. Loughborough, United Kingdom.

Svendenius, J. (2007). *Tire modeling and Friction Estimation*. PhD thesis ISRN LUTFD2/TFRT--1077--SE. Department of Automatic Control, Lund University, Sweden.

Tavernini, D., M. Massaro, E. Velenis, D. I. Katzourakis, and R. Lot (2013). "Minimum time cornering: the effect of road surface and car transmission layout". *Vehicle System Dynamics* **51**:10, pp. 1533–1547.

Thrun, S., W. Burgard, and D. Fox (2006a). *Probabilistic Robotics*. The MIT Press, London, England.

Thrun, S., M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney (2006b). "Stanley: the robot that won the DARPA grand challenge". *Journal of Field Robotics* **23**:9, pp. 661–692.

Tjønnås, J. and T. Johansen (2010). "Stabilization of automotive vehicles using active steering and adaptive brake control allocation". *IEEE Transactions on Control Systems Technology* **18**:3, pp. 545–558.

Tøndel, P. and T. Johansen (2005). "Control allocation for yaw stabilization in automotive vehicles using multiparametric nonlinear programming". In: *2005 American Control Conference*. Portland, Oregon.

Törnqvist, D., T. Schön, R. Karlsson, and F. Gustafsson (2009). "Particle Filter SLAM with High Dimensional Vehicle Model". *Journal of Intelligent and Robotic Systems* **55**:4, pp. 249–266.

Tseng, H. E., B. Ashrafi, D. Madau, Allen, and D. Recker (1999). "The development of vehicle stability control at Ford". *IEEE/ASME Transactions on Mechatronics* **4**:3, pp. 223–234.

Van Loock, W., G. Pipeleers, and J. Swevers (2013). "Time-optimal path planning for flat systems with application to a wheeled mobile robot". In: *9th International Workshop on Robot Motion and Control*. Poznan, Poland.

Velenis, E. and P. Tsiotras (2005). "Minimum time vs. maximum exit velocity path optimization during cornering". In: *2005 IEEE International Symposium on Industrial Electronics*. Dubrovnik, Croatia.

Velenis, E. (2011). "FWD vehicle drifting control: the handbrake-cornering technique". In: *50th IEEE Conference on Decision and Control*. Orlando, Florida.

Verscheure, D., B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl (2008). "Time-energy optimal path tracking for robots: a numerically efficient optimization approach". In: *10th International Workshop on Advanced Motion Control*. Trento, Italy.

Verscheure, D., M. Diehl, J. De Schutter, and J. Swevers (2009a). "On-line time-optimal path tracking for robots". In: *2009 IEEE International Conference on Robotics and Automation*. Kobe, Japan.

Verscheure, D., M. Diehl, J. De Schutter, and J. Swevers (2009b). "Recursive log-barrier method for on-line time-optimal robot path tracking". In: *2009 American Control Conference*. St. Louis, Michigan.

Verscheure, D., B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl (2009c). "Time-optimal path tracking for robots: a convex optimization approach". *IEEE Transactions on Automatic Control* **54**:10, pp. 2318–2327.

Wächter, A. and L. T. Biegler (2006). "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". *Mathematical Programming* **106**:1, pp. 25–57.

Wang, W., D. E. Rivera, and K. G. Kempf (2007). "Model predictive control strategies for supply chain management in semiconductor manufacturing". *International Journal of Production Economics* **107**:1, pp. 56–77.

Wang, Y. and S. Boyd (2010). "Fast model predictive control using online optimization". *IEEE Transactions on Control Systems Technology* **18**:2, pp. 267–278.

Ward, C. C. and K. Iagnemma (2008). "A dynamic-model-based wheel slip detector for mobile robots on outdoor terrain". *IEEE Transactions on Robotics* **24**:4, pp. 821–831.

Westenberger, A., M. Gabb, M. Muntzinger, M. Fritzsche, and K. Dietmayer (2013). "State and existence estimation with out-of-sequence measurements for a collision avoidance system". In: *2013 IEEE Intelligent Vehicles Symposium*. Gold Coast, Australia.

Widd, A. (2012). *Physical Modeling and Control of Low Temperature Combustion in Engines*. PhD thesis ISRN LUTFD2/TFRT--1090--SE. Department of Automatic Control, Lund University, Sweden.

Wit, C. C. de, H. Olsson, K. J. Åström, and P. Lischinsky (1995). "A new model for control of systems with friction". *IEEE Transactions on Automatic Control* **40**:3, pp. 419–425.

Wong, J. (2008). *Theory of Ground Vehicles*. John Wiley & Sons, Inc., Hoboken, New Jersey.

Woodman, O. and R. Harle (2008). "Pedestrian localisation for indoor environments". In: *10th International Conference on Ubiquitous Computing*. Seoul, Korea.

Wyrobek, K. A., E. H. Berger, H. F. M. V. D. Loos, and J. K. Salisbury (2008). "Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot". In: *2008 International Conference on Robotics and Automation*. Pasadena, California.

Xsens Technologies B.V., N. (2010). *MTi and MTx User Manual and Technical Documentation*. http://www.xsens.com.

Yang, D., T. Gordon, B. Jacobson, and M. Jonasson (2013). “An optimal path controller minimizing longitudinal and lateral deviations after light collisions”. In: *16th International IEEE Conference on Intelligent Transportation Systems*. The Hauge, The Netherlands.

Yi, J., J. Zhang, D. Song, and S. Jayasuriya (2007). “IMU-based localization and slip estimation for skid-steered mobile robots”. In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. San Diego, California.

Yi, J., H. Wang, J. Zhang, D. Song, S. Jayasuriya, and J. Liu (2009). “Kinematic modeling and analysis of skid-steered mobile robots with applications to low-cost inertial-measurement-unit-based motion estimation”. *IEEE Transactions on Robotics* **25**:5, pp. 1087–1097.

Yi, J., J. Li, J. Lu, and Z. Liu (2012). “On the stability and agility of aggressive vehicle maneuvers: a pendulum-turn maneuver example”. *IEEE Transactions on Control Systems Technology* **20**:3, pp. 663–676.

Yilmaz, A., O. Javed, and M. Shah (2006). “Object tracking: A survey”. *ACM Computing Surveys* **38**:4.

Yuille, A. L. and A. Rangarajan (2003). “The concave-convex procedure”. In: *Neural Computation*. Vol. 15. 4. The MIT Press, Cambridge, Massachusetts, pp. 915–936.

Zhang, K., X. Rong Li, and Y. Zhu (2005). “Optimal update with out-of-sequence measurements.” *IEEE Transactions on Signal Processing* **53**:6, pp. 1992–2004.

Zhang, S. and Y. Bar-Shalom (2012a). “Optimal update with multiple out-of-sequence measurements with arbitrary arriving order”. *IEEE Transactions on Aerospace and Electronic Systems* **48**:4, pp. 3116–3132.

Zhang, S. and Y. Bar-Shalom (2012b). “Out-of-sequence measurement processing for particle filter: exact Bayesian solution”. *IEEE Transactions on Aerospace and Electronic Systems* **48**:4, pp. 2818–2831.

Zhang, S., Y. Bar-Shalom, and G. Watson (2010). “Tracking with multisensor out-of-sequence measurements with residual biases”. In: *13th International Conference on Information Fusion*. Edinburgh, United Kingdom.

A

Coordinate Systems and Automotive Parameters

This appendix provides descriptions of the coordinate frames that are used throughout the thesis. In addition, it contains the parameters that are used for the automotive applications in Chapters 10 and 11.

A.1 Coordinate Systems

The thesis employs three moving coordinate systems throughout, see Figure A.1. The first coordinate system, $\mathcal{V}$, is rotated with an angle ψ, the yaw, about the Z-axis of the inertial, earth-fixed frame I, yielding the rotation matrix

$$\boldsymbol{R}_{\mathcal{V}}^{I} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Moreover, the pitch angle θ is defined as a rotation about the Y-axis of $\mathcal{V}$, giving the coordinate system $\mathcal{C}$, with the rotation matrix

$$\boldsymbol{R}_{\mathcal{C}}^{\mathcal{V}} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}. \tag{A.1}$$

Finally, the third coordinate system $\mathcal{B}$ is defined by a rotation of an angle ϕ, the roll, about the X-axis of $\mathcal{C}$:

$$\boldsymbol{R}_{\mathcal{B}}^{\mathcal{C}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}. \tag{A.2}$$

Let $\mathbf{p}^{\mathcal{B}}$ denote a position $\mathbf{p} \in \mathbb{R}^3$ expressed in and with respect to $\mathcal{B}$. The expression in the earth-fixed frame I is $\mathbf{p}^{I} = \boldsymbol{R}_{\mathcal{V}}^{I} \boldsymbol{R}_{\mathcal{C}}^{\mathcal{V}} \boldsymbol{R}_{\mathcal{B}}^{\mathcal{C}} \mathbf{p}^{\mathcal{B}}$.

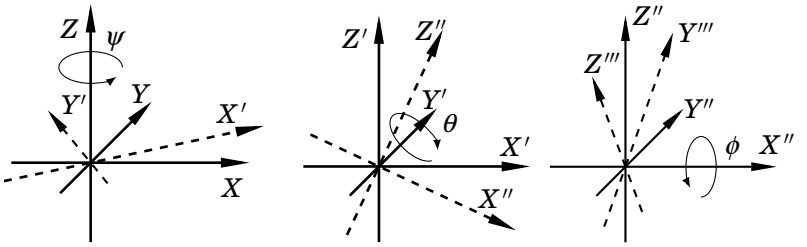

Figure A.1 An illustration of the three principal coordinate axis, where the rotations are taken in the order ψ–θ–ϕ. Note that the rotations are made with respect to the moving axes.

Table A.1 Vehicle-model parameters that are used in Chapter 10.

Parameter	Value	Unit
l_f	1.3	m
l_r	1.5	m
w	0.8	m
m	2 100	kg
I_{XX}	765	kgm^2
I_{YY}	3 477	kgm^2
I_{ZZ}	3 900	kgm^2
R_w	0.3	m
I_w	4.0	kgm^2
σ	0.3	m
g	9.82	ms^{-2}
h	0.5	m
K_ϕ	178 000	Nm(rad)^{-1}
D_ϕ	16 000	Nms(rad)^{-1}
K_θ	363 540	Nm(rad)^{-1}
D_θ	30 960	Nms(rad)^{-1}

A.2 Tire and Vehicle Parameters

The tire and vehicle parameters that are used in Chapter 10 are given in Tables A.1–A.3. The chassis parameters correspond to a car of sedan type. Tables A.4–A.5 contain the parameters that are used for the automotive application in Chapter 11. The parameters are slightly different to those in Chapter 10, but correspond to approximately the same surface and vehicle type.

Table A.2 Front-wheel tire-model parameters, which are used in Chapter 10. The parameters correspond to asphalt, snow, and ice.

Parameter	Asphalt	Snow	Ice
μ_x	1.20	0.407	0.172
B_x	11.7	10.2	31.1
C_x	1.69	1.96	1.77
E_x	0.377	0.651	0.710
μ_y	0.935	0.383	0.162
B_y	8.86	19.1	28.4
C_y	1.19	0.550	1.48
E_y	-1.21	-2.10	-1.18
C_α	1.09	1.09	1.02
$B_{\alpha 1}$	12.4	15.4	75.4
$B_{\alpha 2}$	-10.8	-10.8	-43.1
C_λ	1.08	1.08	0.984
$B_{\lambda 1}$	6.46	4.19	33.8
$B_{\lambda 2}$	4.20	4.20	42.0

Table A.3 Rear-wheel tire-model parameters, which are used in Chapter 10. The parameters correspond to asphalt, snow, and ice.

Parameter	Asphalt	Snow	Ice
μ_x	1.20	0.409	0.173
B_x	11.1	9.71	29.5
C_x	1.69	1.96	1.77
E_x	0.362	0.624	0.681
μ_y	0.961	0.394	0.167
B_y	9.30	20.0	30.0
C_y	1.19	0.550	1.48
E_y	-1.11	-1.93	-1.08
C_α	1.09	1.09	1.02
$B_{\alpha 1}$	12.4	15.4	75.4
$B_{\alpha 2}$	-10.8	-10.8	-43.1
C_λ	1.08	1.08	0.984
$B_{\lambda 1}$	6.46	4.19	33.8
$B_{\lambda 2}$	4.20	4.20	42.0

Table A.4 Vehicle-model parameters that are used in Chapter 11. The parameters approximately correspond to a medium-sized passenger car.

Parameter	Value	Unit
l_f	1.2	m
l_r	1.5	m
w	0.75	m
m	1600	kg
I_{XX}	900	kgm^2
I_{YY}	2000	kgm^2
I_{ZZ}	2700	kgm^2
R_w	0.32	m
I_w	1	kgm^2
σ	0.3	m
g	9.81	ms^{-2}
h	0.29	m
K_ϕ	120 000	Nm(rad)^{-1}
D_ϕ	8 000	Nms(rad)^{-1}
K_θ	165 000	Nm(rad)^{-1}
D_θ	14 000	Nms(rad)^{-1}
T	0.1	s

Table A.5 Tire-model parameters that are used in Chapter 11. The parameters correspond to asphalt.

Parameter	Value
μ_x	1.06
B_x	19.7
C_x	1.63
E_x	0.5
μ_y	0.92
B_y	13.06
C_y	1.28
E_y	-1.1
C_α	1.13
$B_{\alpha 1}$	9
$B_{\alpha 2}$	-8.6
C_λ	1.16
$B_{\lambda 1}$	6.4
$B_{\lambda 2}$	7.91

B

Code

Listing B.1 contains the MATLAB code that is used for Example 3.1 in Chapter 3. Note that the code is neither written with efficiency nor generality in mind, but only serves to give hands-on experience. The resampling scheme that is used here is the systematic resampling algorithm, see [Arulampalam et al., 2002].

Listing B.1 RBPF example code used in Example 3.1.

```
function [xmean,xparts] = RBPF(N,y,xp,w,P,A,B,C,R,Q,Tfinal)
xmean = zeros(4,Tfinal);
xparts = zeros(2,N,Tfinal);
for t=1:Tfinal
    for i=1:N                       %PF measurement update
        e = y(:,t) - [0.1*xp(4,i)^2*sign(xp(4,i));0] ...
            - C*xp(1:3,i);
        S = C*P(:,:,i)*C' + R; Sinv = S\eye(2);
        w(i) = w(i)/sqrt(det(S))*exp(-0.5*(e'*Sinv*e));
    end

    w = w/sum(w);                   %Normalize weights
    xparts(2,:,t) = w;
    xparts(1,:,t) = xp(4,:);
    if 1/sum(w.^2) < 0.67*N         %Conditional Resampling
        index = resample(w,N);
        xp = xp(:,index); P = P(:,:,index); w = ones(1,N)/N;
    end
    xf = xp;
    xmean(4,t) = sum(w.*xf(4,:),2); %Weighted mean estimate
    for i=1:N                       %KF Measurement update
        e = y(:,t) - [0.1*xf(4,i)^2*sign(xf(4,i));0] ...
            - C*xp(1:3,i);
        S = C*P(:,:,i)*C' + R; Sinv = S\eye(2);
```

```
        K = P(:,:,i)*C'*Sinv;
        P(:,:,i) = P(:,:,i) - K*S*K';
        xf(1:3,i) = xp(1:3,i) + K*e;
    end
    xmean(1:3,t) = sum(repmat(w,length(xf(1:3,1)),1)...
        .*xf(1:3,:),2);           %Weighted mean estimate
    for i=1:N                     %PF Time Update
        xp(4,i) = atan(xf(4,i) ) + B*xf(1:3,i)...
            + sqrt(B*P(:,:,i)*B' + Q(4,4))*randn(1);
    end
    for i=1:N                     %KF Time Update
        M = B*P(:,:,i)*B' + Q(4,4); Minv = 1/M;
        L = P(:,:,i)*B'*Minv;
        e = xp(4,i) - atan(xf(4,i)) - B*xf(1:3,i);
        xp(1:3,i) = A*(xf(1:3,i) + L*e);
        P(:,:,i) = A*(P(:,:,i) - L*M*L')*A' + Q(1:3,1:3);
    end
end
    function i=resample(w,n)      %Systematic resampling
        wc = cumsum(w); i = zeros(1,n); k = 1;
        u = (0:n-1+rand(1))/n;
        for j=1:n
            while (wc(k)<u(j))
                k = k + 1;
            end
            i(j) = k;
        end
    end


end
```

Printed by Media-Tryck, Lund University, Sweden

Department of Automatic Control
P.O. Box 118, 221 00 Lund, Sweden
www.control.lth.se

ISRN LUTFD2/TFRT--1101--SE
ISBN 978-91-7473-947-3
ISSN 0280–5316