



# LUND UNIVERSITY

## A Convex Approach to Path Tracking with Obstacle Avoidance for Pseudo-Omnidirectional Vehicles

Olofsson, Björn; Berntorp, Karl; Robertsson, Anders

2015

*Document Version:*

Publisher's PDF, also known as Version of record

[Link to publication](#)

*Citation for published version (APA):*

Olofsson, B., Berntorp, K., & Robertsson, A. (2015). *A Convex Approach to Path Tracking with Obstacle Avoidance for Pseudo-Omnidirectional Vehicles*. (Technical Reports TFRT-7643). Department of Automatic Control, Lund Institute of Technology, Lund University.

*Total number of authors:*

3

### General rights

Unless other specific re-use rights are stated the following general rights apply:

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: <https://creativecommons.org/licenses/>

### Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117  
221 00 Lund  
+46 46-222 00 00

# A Convex Approach to Path Tracking with Obstacle Avoidance for Pseudo-Omnidirectional Vehicles

Björn Olofsson, Karl Berntorp, and Anders Robertsson



**LUND**  
UNIVERSITY

Department of Automatic Control

Technical Report  
ISRN LUTFD2/TFRT--7643--SE  
ISSN 0280-5316

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

© 2015 by Björn Olofsson, Karl Berntorp, and Anders Robertsson. All rights reserved.  
Printed in Sweden by Media-Tryck.  
Lund 2015

# Abstract

This report addresses the related problems of trajectory generation and time-optimal path tracking with online obstacle avoidance. We consider the class of four-wheeled vehicles with independent steering and driving on each wheel, also referred to as pseudo-omnidirectional vehicles. Appropriate approximations of the dynamic model enable a convex reformulation of the path-tracking problem. Using the precomputed trajectories together with model predictive control that utilizes feedback from the estimated global pose, provides robustness to model uncertainty and disturbances. The considered approach also incorporates avoidance of a priori unknown moving obstacles by local online replanning. We verify the approach by successful execution on a pseudo-omnidirectional mobile robot, and compare it to an existing algorithm. The result is a significant decrease in the time for completing the desired path. In addition, the method allows a smooth velocity trajectory while avoiding intermittent stops in the path execution.

## Authors Contact Information

Björn Olofsson, Karl Berntorp<sup>1</sup>, & Anders Robertsson  
Department of Automatic Control, LTH, Lund University  
SE-221 00 Lund  
Sweden  
E-mail: `firstname.lastname@control.lth.se`

---

<sup>1</sup> Present address: Mitsubishi Electric Research Laboratories, Cambridge, MA 02139.



# 1

## Introduction

During the past decade, the interest for mobile robots in production scenarios as well as in domestic applications has increased as a result of the development of algorithms, computing power, and sensors. In addition, to reduce the complexity of the programming phase and to increase the learning capabilities by cognitive functionalities, large efforts have been put in the area of software services for mobile robots. Two examples of this are the Robot Operating System (ROS) [WillowGarage, 2015] and Orocos [The Orocos Project, 2015]. In a production scenario with small batch sizes, combination of a mobile robot platform with conventional robot manipulators 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.

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 start and end points except avoiding known obstacles [LaValle, 2006]. 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 global path planner provides the path, and a subsequent trajectory generation is to be made [Verscheure et al., 2009c]. To this purpose, the decoupled approach to trajectory generation has been established in literature [LaValle, 2006; Kant and Zucker, 1986]. To utilize as much as possible of the capacity of the robot in a path-tracking application in industry where the actuators are the limiting factors, a near time-optimal path tracking, where the constraints on the actuators are considered and robustness to modeling errors and disturbances is taken into account, is desirable. Note here that time-optimal does not per se imply high velocities, only that the maximum capacity of the actuators is used. In addition, avoidance of dynamic obstacles online [Khatib, 1986] is a desired characteristic of the path tracking.

In this report, we discuss a system architecture for trajectory generation and path tracking with online obstacle avoidance, applied to four-wheeled ve-

hicles with independent steering and driving on each wheel. A dynamic model of the vehicle is derived using the Euler-Lagrange approach. The trajectory-generation, online path tracking, and collision-avoidance problems are formulated as convex optimization problems, with various degrees of model approximations in a hierarchical control structure. In particular, we consider scenarios where a nominal path is planned using a predefined static map of the environment (accounting for all a priori known obstacles). A time-optimal trajectory is then generated, and model predictive control (MPC) [Mayne et al., 2000; Maciejowski, 1999] is used for online high-level feedback from the estimated vehicle state. Moreover, the vehicle avoids new emerging obstacles, detected during runtime, using online local replanning of the path and corresponding trajectory with a scheme integrated in the MPC.

## 1.1 Previous Research

Trajectory generation and collision avoidance for mobile robots are well-studied areas, see [Khatib, 1986; Fox et al., 1997; Choi et al., 2009; Qu et al., 2004; Quinlan and Khatib, 1993] for a few examples. A two-level hierarchical architecture was proposed in [Kant and Zucker, 1986], where a local replanning of the nominal trajectory was performed online to avoid moving obstacles. A solution to the time-optimal path-tracking problem was derived in the 1980's [Bobrow et al., 1985; Shin and McKay, 1985] for robot manipulators. By utilizing the special structure of the model resulting from the Euler-Lagrange formulation and a parametrization of the spatial path in a path coordinate, the minimum-time problem can be reformulated to an optimal control problem (OCP) with fixed horizon for the independent variable and with a significantly reduced number of states. Solutions to these OCPs were found offline. The obtained trajectories were in later research combined with feedback, thus accounting for model uncertainties and disturbances in the online task execution [Dahl and Nielsen, 1990; Dahl, 1992]. In the feedback, it is necessary to maintain the synchronization between the degrees-of-freedom of the system in order to achieve accurate path tracking. By utilizing convex optimization [Boyd and Vandenberghe, 2004], the paper [Verscheure et al., 2009c] showed how the time-optimal path-tracking problem for robot 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, were imposed. A subsequent online path-tracking algorithm was proposed in [Verscheure et al., 2009a; Verscheure et al., 2009b], where the geometric path was delivered online to the trajectory generator. Further investigation of applications areas and extensions of the method in [Verscheure et al., 2009c] was presented in [Lipp and Boyd, 2014; Castro et al., 2014], including applications to vehicles and aircrafts.

Trajectory optimization for electric vehicles based on the same convex optimization methods was proposed in [Castro et al., 2014]. Velocity-dependent constraints in the convex optimization formulation were handled in [Ardeshiri et al., 2011]. This was later elaborated for convex-concave constraints in [Debrouwere et al., 2013]. Time-optimal path planning for two-wheeled robots was investigated in [Van Loock et al., 2013]. MPC [Mayne et al., 2000; Maciejowski, 1999] has been developed for trajectory tracking and path tracking for mobile robots previously, see, for example, [Kanjanawanishkul and Zell, 2009; Howard et al., 2009; Klančar and Škrjanc, 2007; Connette et al., 2010]. The differences compared to the approach in this report are in the modeling and in the integration of the obstacle-avoidance functionality. Time-optimal path following for mobile robots with independent steering and driving was considered in [Oftadeh et al., 2014], based on kinematic models. Nonlinear MPC for obstacle avoidance in the case of autonomous vehicles has been investigated in [Norén, 2013], and [Berntorp and Magnusson, 2015] proposed a hierarchical predictive-control architecture for solving the motion-planning problem for road vehicles.

## 1.2 Contributions and Relations to Previous Research

The contributions of this report are as follows: First, an architecture for minimum-time trajectory generation and online path tracking for four-wheeled vehicles with independent steering and driving of each wheel, incorporating online obstacle-avoidance functionality, is developed. Second, we consider an approach to how the convex time-optimal trajectory-generation algorithms for robot manipulators with a fixed base proposed in [Verscheure et al., 2009c] can be applied to the case of pseudo-omnidirectional mobile robots, which have significantly different dynamics, using a limited modeling effort for the mobile robot. The third and major contribution of the report is the implementation of the methods in an integrated framework and the subsequent validation of the approach with successful experiments in a challenging scenario, employing a recently developed pseudo-omnidirectional mobile robot platform [Connette et al., 2009; Weisshardt and Garcia, 2014].

Several approaches to time-optimal trajectory generation and online obstacle avoidance for wheeled vehicles have been reported in the literature during the past decades as reviewed in Section 1.1. There seems to be only a few such methods for mobile robots or wheeled vehicles in general that are based on online convex optimization. Comparing the architecture in this report to the convex approaches presented in [Lipp and Boyd, 2014; Castro et al., 2014], our methods rely on different modeling assumptions and are therefore requiring less dynamic modeling. This enables more straightforward application of the methods at the cost of limiting the application area



slightly. The mechanical design of the considered types of vehicles allows for models with reduced complexity in the problem formulation; for example, the omnidirectional characteristics make it possible to turn without significant slipping and/or skidding. This differs from most types of mobile robots, such as differential-drive robots, where slip is a necessity for rotational movements. Lateral friction is required for turning at nonzero velocities with the considered type of vehicles due to the centripetal acceleration. However, for the velocities and the nature of the paths that are typically used for mobile robots, other dynamics dominate. This implies that lateral friction is not an essential feature of pseudo-omnidirectional vehicles.

A key feature of the developed architecture is that it combines a time-optimal trajectory-reference generator (based on a nonlinear vehicle model with friction) with a feedback controller, where the problems of finding the references and control signals are posed as convex optimization problems. This implies that the optimal solution will be found quickly, enabling high sampling frequencies in the control, as well as with guaranteed convergence to a global optimum. In addition, some of the previously proposed methods were only based on the kinematic relations of the robot, and did not consider a nonlinear dynamic model incorporating friction, see Section 1.1. Moreover, differential-drive mobile robots are often considered in literature. The characteristics of differential-drive robots are considerably different compared to four-wheeled vehicles with independent steering and driving—for example, significant constraints are enforced on the maneuverability and the required slip modeling for the former. The main advantages of the considered formulation of the trajectory-tracking controller is that it incorporates both prediction and optimizing characteristics in a convex optimization formulation, allowing fast real-time solutions. In addition, constraints derived from dynamic obstacles and map information can be introduced during runtime by utilizing the receding-horizon principle of the MPC.

A preliminary version of the trajectory generator was presented in [Berntorp et al., 2014a]. This report extends the results of [Berntorp et al., 2014a] in several respects: First, this report provides an improved and more rigorous presentation of the method. Second, we incorporate a global path planner into our algorithm. Third, we use an MPC approach for online feedback from global position and orientation (pose) estimates with collision-avoidance functionality. Fourth, the experimental scenario used for evaluation is more demanding in terms of the obstacle shapes and the length of the path.

### 1.3 Outline

The outline is as follows: Chapter 2 establishes an Euler-Lagrange model and the kinematic relations of the considered class of four-wheeled vehicles. This

derivation is the basis for the formulation of the convex optimization problem for trajectory generation presented in Chapter 3. High-level feedback using MPC and obstacle avoidance are discussed in Chapter 4. Further, Chapter 5 discusses the experimental setup and the implementation structure for the considered path-tracking scenario. Moreover, the same section presents experimental results for path tracking with a mobile robot platform in a realistic and demanding scenario. We elaborate on the applicability and generality of the proposed method in Chapter 6, and compare the results achieved with the proposed algorithms to corresponding results obtained with a reference method previously implemented by the robot manufacturer on the considered robot platform. Finally, the report is summarized and conclusions are drawn in Chapter 7.



# 2

## Modeling

Here, we derive a model of the vehicle dynamics and the inverse kinematic relations. We also discuss model approximations for obtaining a model that is suitable for convex optimization.

### 2.1 Modeling of the Dynamics

For modeling of the vehicle, we assume that the motor torques are controlled directly; that is, we neglect the motor dynamics. This is motivated by the fact that the motor dynamics is inherently fast compared to the other dynamics of the vehicle. Also, the inner motor-current controllers used in this kind of vehicles ensure fast torque tracking. Further, we assume planar movement, thus neglecting vertical dynamics and rotational coupling such as roll dynamics.

The Euler-Lagrange equations [Spong et al., 2006] state that

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = \tau_k, \quad k = 1, \dots, n, \quad (2.1)$$

where  $\frac{d}{dt}$  is the time-derivative with respect to an earth-fixed inertial frame. In (2.1),  $q_k$  is the  $k$ th generalized coordinate,  $\tau_k$  is the  $k$ th external torque,  $\dot{q}$  is the time-derivative of  $q$ ,  $L = T - V$  is the difference between the kinetic and potential energy, and  $n$  is the number of generalized coordinates. Since we assume planar movement,  $V = 0$ . Denote the coordinates of the center-of-geometry (CoG) of the vehicle with respect to an earth-fixed inertial frame  $XY$  as  $(p_X, p_Y)$ , see Figure 2.1. Further, denote the heading angle of the vehicle with respect to the inertial frame with  $\psi$ . Then, the kinetic energy is

$$T = \frac{1}{2} \left( m(\dot{p}_X^2 + \dot{p}_Y^2) + I_r \dot{\psi}^2 + \sum_{j=1}^4 (I_{\phi,j} \dot{\phi}_j^2 + I_{\delta,j} \dot{\delta}_j^2) \right), \quad (2.2)$$

where  $m$  is the mass of the vehicle,  $I_r$  is the vehicle moment-of-inertia,  $I_\phi$  is the wheel moment-of-inertia in the drive direction,  $I_\delta$  is the wheel moment-of-inertia in the steer direction, and  $\phi_i$ ,  $\delta_i$  are the drive and steer angles for wheel  $i$ , respectively. A natural choice of generalized coordinates is  $q = (p_X \ p_Y \ \psi \ \phi_i \ \delta_i)^T$ ,  $i = 1, \dots, 4$ . A more convenient choice than  $p_X$  and  $p_Y$ , however, is to express the dynamics in local coordinates and linear base velocities, since velocity references and torque commands are, according to convention, given in this frame. Therefore, the quasi coordinates in the vehicle frame is used instead, see [Pacejka, 2006]. A coordinate transformation between  $(\dot{p}_X, \dot{p}_Y)$  and the CoG velocities in the vehicle frame,  $(v_x, v_y)$ , is given by

$$\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} c_\psi & s_\psi \\ -s_\psi & c_\psi \end{pmatrix} \begin{pmatrix} \dot{p}_X \\ \dot{p}_Y \end{pmatrix} = R(\psi) \begin{pmatrix} \dot{p}_X \\ \dot{p}_Y \end{pmatrix}, \quad (2.3)$$

where  $c_\psi = \cos \psi$  and  $s_\psi = \sin \psi$ . The following transformation from global to local coordinates can be established using the chain rule:

$$\frac{\partial T}{\partial \dot{p}_X} = \frac{\partial T}{\partial v_x} c_\psi - \frac{\partial T}{\partial v_y} s_\psi, \quad (2.4)$$

$$\frac{\partial T}{\partial \dot{p}_Y} = \frac{\partial T}{\partial v_x} s_\psi + \frac{\partial T}{\partial v_y} c_\psi. \quad (2.5)$$

By insertion of (2.4) and (2.5) into (2.1) and premultiplying with  $R(\psi)$ , the following two modified Euler-Lagrange equations are obtained:

$$\frac{d}{dt} \frac{\partial T}{\partial v_x} - \dot{\psi} \frac{\partial T}{\partial v_y} = \tau_x, \quad (2.6)$$

$$\frac{d}{dt} \frac{\partial T}{\partial v_y} + \dot{\psi} \frac{\partial T}{\partial v_x} = \tau_y. \quad (2.7)$$

Using the partial derivatives of the kinetic energy  $T$  (which can be expressed in the velocities in the vehicle frame) and (2.6)–(2.7), the following set of dynamic equations are derived:

$$m\dot{v}_x - m\dot{\psi}v_y = \tau_x, \quad m\dot{v}_y + m\dot{\psi}v_x = \tau_y, \quad (2.8)$$

$$I_r \ddot{\psi} = \tau_\psi, \quad (2.9)$$

$$I_{\phi,i} \ddot{\phi}_i = \tau_{\phi,i}, \quad I_{\delta,i} \ddot{\delta}_i = \tau_{\delta,i}, \quad i = 1, \dots, 4, \quad (2.10)$$

where  $\tau$  are the forces and torques for the respective generalized coordinate. We model the motor torques acting on wheel  $i$  as two independent torques for driving and steering,  $M_{\phi,i}$  and  $M_{\delta,i}$ , respectively. Moreover, we model friction forces and torques acting on wheel  $i$ , and denote them by  $F_{x,i}^f$ ,  $F_{y,i}^f$ , and  $M_i^f$ . These are defined in Figure 2.2. Finally, the drive torque generates

a resultant force  $F_{x,i}$  between the tire and road, see Figure 2.2. To compute the right-hand sides in (2.8)–(2.10), note that the longitudinal and lateral vehicle forces are

$$\tau_x = \sum_{i=1}^4 \left( c_{\delta_i} (F_{x,i} - F_{x,i}^f) - s_{\delta_i} (F_{y,i} - F_{y,i}^f) \right), \quad (2.11)$$

$$\tau_y = \sum_{i=1}^4 \left( s_{\delta_i} (F_{x,i} - F_{x,i}^f) + c_{\delta_i} (F_{y,i} - F_{y,i}^f) \right), \quad (2.12)$$

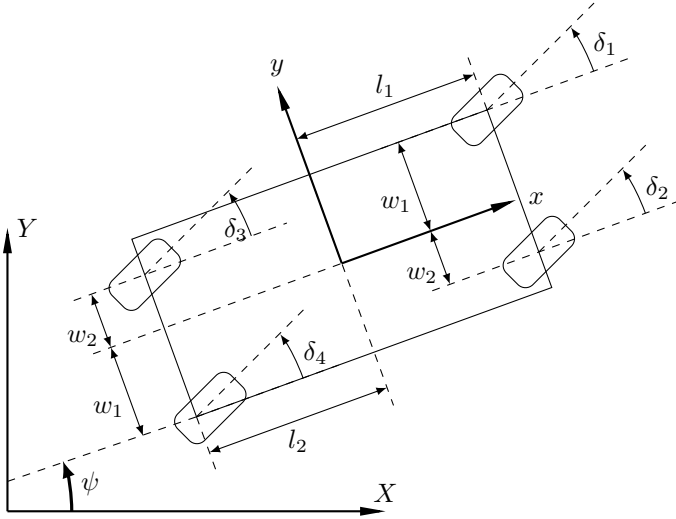
where  $c_\delta = \cos \delta$ ,  $s_\delta = \sin \delta$ , and  $F_{y,i}$  is the lateral force on wheel  $i$ , whereas the resulting torque acting on the vehicle is

$$\begin{aligned} \tau_\psi = & (l_1 s_{\delta_1} - w_1 c_{\delta_1})(F_{x,1} - F_{x,1}^f) + (l_1 s_{\delta_2} + w_2 c_{\delta_2})(F_{x,2} - F_{x,2}^f) \\ & - (l_2 s_{\delta_3} + w_2 c_{\delta_3})(F_{x,3} - F_{x,3}^f) - (l_2 s_{\delta_4} - w_1 c_{\delta_4})(F_{x,4} - F_{x,4}^f) \\ & + (l_1 c_{\delta_1} + w_1 s_{\delta_1})(F_{y,1} - F_{y,1}^f) + (l_1 c_{\delta_2} - w_2 s_{\delta_2})(F_{y,2} - F_{y,2}^f) \\ & - (l_2 c_{\delta_3} - w_2 s_{\delta_3})(F_{y,3} - F_{y,3}^f) - (l_2 c_{\delta_4} + w_1 s_{\delta_4})(F_{y,4} - F_{y,4}^f). \end{aligned} \quad (2.13)$$

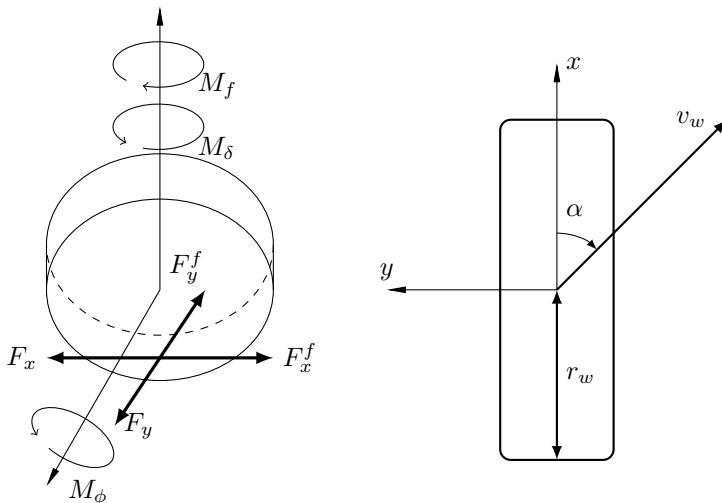
Moreover, the torques in the wheels' steer and drive directions are

$$\tau_{\delta,i} = M_{\delta,i} - M_i^f, \quad \tau_{\phi,i} = M_{\phi,i} - r_w (F_{x,i} - F_{x,i}^f), \quad (2.14)$$

for wheel  $i$ ,  $i = 1, \dots, 4$ , where  $r_w$  is the wheel radius.



**Figure 2.1** The vehicle and the coordinate systems used for modeling.



**Figure 2.2** An illustration of the forces acting on each wheel (left), and the wheel together with its coordinate system seen from above (right).

## 2.2 Wheel-Force Modeling

The friction forces for each wheel  $i$ ,  $i = 1, \dots, 4$ , are modeled using Coulomb friction. The index  $i$  is omitted in the presentation in this subsection for notational convenience. Hence, the friction forces and torque are given by

$$F_x^f = F_C^x \text{sign}(v_{w,x}), \quad (2.15)$$

$$F_y^f = F_C^y \text{sign}(v_{w,y}), \quad (2.16)$$

$$M^f = M_C \text{sign}(\dot{\delta}) \quad (2.17)$$

where  $F_C^x$ ,  $F_C^y$ , and  $M_C$  are the Coulomb-friction constants,  $\text{sign}(\cdot)$  is the signum function, and  $v_{w,x}$ ,  $v_{w,y}$  are the velocities of each wheel in the longitudinal and lateral direction, respectively (see Figure 2.2). The reason for only considering Coulomb friction is that we assume that velocities necessary for, for example, viscous friction to be significant will typically not be reached under normal operating conditions for the considered type of mobile platforms.<sup>1</sup>

When a vehicle accelerates or decelerates, longitudinal slip develops

<sup>1</sup>For very large velocities air drag becomes a factor, and will typically dominate over Coulomb and viscous friction. This can be modeled within this framework as pointed out in [Lipp and Boyd, 2014].

[Schindler, 2007]. Here, the slip  $\lambda$  is defined according to [Schindler, 2007] as

$$\lambda = \begin{cases} 1 - \frac{r_w \dot{\phi}}{v_{w,x}}, & \dot{v}_{w,x} < 0, \\ \frac{v_{w,x}}{r_w \dot{\phi}} - 1, & \dot{v}_{w,x} \geq 0. \end{cases} \quad (2.18)$$

The wheel velocities  $v_{w,x}$ ,  $v_{w,y}$  are straightforward to determine by using the geometric dimensions of the robot (see Figure 2.1), trigonometry and utilizing the velocity of the CoG, obtained from the dynamic model previously derived in this section. The lateral slip angle  $\alpha$  is defined according to convention (see, for example, [Pacejka, 2006]) by

$$\tan \alpha = -\frac{v_{w,y}}{v_{w,x}}. \quad (2.19)$$

For the types of maneuvers the considered class of four-wheeled mobile robots perform it is in general safe to assume that the longitudinal and lateral tire forces are proportional to the respective slip quantity.<sup>2</sup> The wheel forces caused by wheel slip are then

$$F_x = C_\lambda \lambda, \quad F_y = C_\alpha \alpha, \quad (2.20)$$

where  $C_\lambda$ ,  $C_\alpha$  are constants found from experiments, dependent on the robot mass, wheel material, and surface conditions. For small slip values, (2.20) is justified by experimental verification [Pacejka, 2006]. With this wheel-force modeling, the dynamic equations of the vehicle are constituted by (2.8)–(2.20). Note that (2.17)–(2.20) hold for each wheel individually.

## 2.3 Model for Optimization

Two different strategies can be adopted when formulating the trajectory-generation problem for time-optimal path tracking. The first strategy is to formulate it using the generalized forces and torques acting on the vehicle platform as inputs in the dynamic model instead of the wheel torques. This approach is considered for speed optimization for cars in [Lipp and Boyd, 2014; Castro et al., 2014], where the friction ellipse is used as a constraint for the maximum longitudinal and lateral forces in the optimization. The latter is necessary to consider when turning at high velocities, because then it is the maximum friction forces between wheel and road that are limiting the path traversal rather than the wheel torques. In [Lipp and Boyd, 2014]

---

<sup>2</sup>Here, vehicles performing under extreme conditions are excluded since they require detailed tire-force modeling and consideration of load-transfer effects in the case of aggressive maneuvers [Berntorp et al., 2014b].



it is further assumed that the motors or engines driving the vehicle have sufficient power to realize the desired wheel forces. However, for the class of mobile platforms targeted with the architecture proposed in this report, it is rather the constraints on the motors and not the maximum friction between the wheels and the floor that limit the path traversal. In addition, establishing the required parameters for the friction-ellipse modeling for mobile-robot systems is not straightforward without appropriate measurement equipment. In contrast, there are well-established models for rubber tires used for automobiles, see, for example, [Pacejka, 2006]. Considering these two aspects, it is advantageous to formulate the trajectory-generation problem based on the wheel dynamics and this is consequently the choice in this report. For simplification, we invoke the no-slip assumption, which implies that the torques applied to the wheels directly influence the vehicle movement. Hence, the wheel dynamics, derived from (2.10) and (2.14) by assuming that the longitudinal forces  $F_x$  are proportional to the angular acceleration of the wheels, can be written as

$$\begin{pmatrix} M_\phi \\ M_\delta \end{pmatrix} = M_{\phi,\delta} = I_e(\xi)\ddot{\xi} + F_C^\xi \text{sign}(\dot{\xi}), \quad (2.21)$$

where  $\xi = (\phi_1 \ \dots \ \phi_4 \ \delta_1 \ \dots \ \delta_4)^\top$  and  $F_C^\xi$  is the Coulomb-friction parameter vector. We refer to  $I_e(\xi)$  as the effective inertia matrix, because it accounts for both the wheel inertia and the inertia required for acceleration of the mobile platform.

#### REMARK 1

The no-slip assumption only holds during (arbitrarily fast) constant-velocity motions. However, the assumption was evaluated in simulations for a representative mobile robot platform in [Berntorp et al., 2014a], and was shown to be approximately true for the considered path traversals. This assumption is also valid for automotive systems in steady-state driving. The no-slip assumption also implies that the relations  $v_{w,x} = r_w \dot{\phi}$ ,  $v_{w,y} = 0$  hold. Moreover, we implicitly assume that the lateral friction required when turning is negligible for the considered types of vehicles and paths.  $\square$

## 2.4 Kinematics

The geometric vehicle path is in general determined in Cartesian space by a high-level path planner. Hence, for the optimization approach on wheel level presented in Chapter 3, a method is needed in order to transfer the Cartesian path coordinates  $\{p_X, p_Y, \psi\}$  to wheel-space coordinates  $\{\phi_i, \delta_i\}_{i=1}^4$ . Thus we want to find a transformation according to  $\Omega : \{p_X, p_Y, \psi\} \rightarrow \{\phi_i, \delta_i\}_{i=1}^4$ . Since wheels exhibit slip, finding a closed-form transformation is in general

not possible. To derive analytic expressions, we again impose the no-slip assumption. As pointed out in Remark 1 this assumption does not hold during acceleration and deceleration, but the resulting deviations are handled online using high-level feedback, see Chapter 4, and the low-level wheel control loops. Conceptually, the transformation for the drive angles can be derived as follows: Assume a path and orientation for the CoG of the robot for  $K$  grid points as  $\{p_X(k), p_Y(k), \psi(k)\}_{k=1}^K$ . The inverse kinematics derivation further assumes that the global robot velocity in the  $XY$  coordinate system is positive and that  $-\pi/2 \leq \psi \leq \pi/2$ . The other cases are straightforward to derive but omitted here.

Given the geometry of the vehicle, the specified path and orientation imply knowledge of the path at the wheel center point, for all wheels at time step  $k$ . With the no-slip assumption, the drive angle for each wheel at each grid point  $k$  is found as

$$\phi(k) = \phi(k-1) + \|\Delta p_w(k)\|_2 / r_w, \quad (2.22)$$

for small enough  $\Delta p_w(k) = p_w(k) - p_w(k-1)$  and  $\Delta\phi = \phi(k) - \phi(k-1)$ . To find the steer angles we apply trigonometry, yielding

$$\delta(k) = \arctan2(\Delta p_{w,y}(k), \Delta p_{w,x}(k)) - \psi(k), \quad (2.23)$$

where  $\arctan2(\cdot, \cdot)$  is the four-quadrant inverse tangent function. Again, the inequality holds for small enough differences. Note that  $\psi$  is subtracted since we want to know  $\delta$  with respect to the vehicle frame. To summarize, a valid approximation of the inverse kinematics is given by (2.22) and (2.23) for  $K$  large enough (that is, the sampling is dense enough with respect to the path).



# 3

## Optimal Trajectory Generation

In this section the approach for generating time-optimal trajectories given a nominal geometric path is described. In addition, the discretization of the continuous-time optimization problem for enabling online numerical solutions is considered.

### 3.1 Time-Optimal Trajectory Generation

For the trajectory generation, a convex optimization problem for time-optimal tracking of a given geometric path  $f$  for the wheel coordinates  $\xi$  is formulated. The formulation considers the constraints on the actuators in terms of maximum and minimum realizable torques. The slip is neglected, as discussed in Section 2.3, which means that the considered dynamic equations are expressed in (2.21). The path to be tracked is parametrized in a path coordinate  $s(t)$ , where the dependency on time  $t$  will be implicit in the rest of the report, according to<sup>1</sup>

$$f(s) = (f_1(s) \ \cdots \ f_n(s))^T, \quad s \in [s_0, s_f], \quad (3.1)$$

where  $n$  is the number of wheel coordinates in  $\xi$ . Also,  $s_0$  and  $s_f$  are the path coordinates at the start and end points of the path, respectively. To the purpose of tracking, the relation  $\xi = f(s)$  holds when the vehicle is on the desired path. From this requirement, the following relations can be established by using the chain rule:

$$\dot{\xi} = f'(s)\dot{s}, \quad \ddot{\xi} = f'(s)\ddot{s} + f''(s)\dot{s}^2, \quad (3.2)$$

---

<sup>1</sup>The derivation in this section is performed using the more general assumption on  $n$  wheel coordinates of the vehicle. In the case of the dynamic model considered in Chapter 2, it holds that  $n = 8$ .

where  $(\cdot)'$  denotes  $\frac{d}{ds}$ ,  $\dot{s}$  is the path velocity and  $\ddot{s}$  is the path acceleration. Utilizing the derivatives in (3.2), the dynamic equations in (2.21) can be reformulated in the path coordinate, [Bobrow et al., 1985; Pfeiffer and Johanni, 1987; Shin and McKay, 1985; Dahl, 1993], according to

$$M_{\phi,\delta} = \Gamma_1(s)\ddot{s} + \Gamma_2(s)\dot{s}^2 + \Gamma_3(s), \quad (3.3)$$

where

$$\Gamma_1(s) = I_e(\xi(s))f'(s), \quad \Gamma_2(s) = I_e(\xi(s))f''(s), \quad \Gamma_3(s) = F_C^\xi \text{sign}(f'(s)). \quad (3.4)$$

Since the aim of the trajectory generation is to minimize the execution time of the path tracking, the optimal control problem is formulated over the time horizon  $t \in [0, t_f]$ , with the cost function chosen as the end time  $t_f$ . Utilizing the path coordinate and its time-derivatives, the cost function is reformulated (see, for example, [Bobrow et al., 1985]) as follows

$$t_f = \int_0^{t_f} 1 dt = \int_{s_0}^{s_f} \frac{dt}{ds} ds = \int_{s_0}^{s_f} \frac{1}{\dot{s}} ds. \quad (3.5)$$

With the state variable  $\beta(s)$  and the algebraic variable  $\alpha(s)$  introduced as suggested in [Verscheure et al., 2009c] according to

$$\beta(s) = \dot{s}^2, \quad \alpha(s) = \ddot{s}, \quad (3.6)$$

the continuous-time optimal control problem to be solved is stated as in [Verscheure et al., 2009c] according to:

$$\begin{aligned} & \underset{\alpha(s), \beta(s)}{\text{minimize}} && \int_{s_0}^{s_f} \frac{1}{\sqrt{\beta(s)}} ds \\ & \text{subject to} && M_{\phi,\delta}(s) = \Gamma_1(s)\alpha(s) + \Gamma_2(s)\beta(s) + \Gamma_3(s), \\ & && \beta(s_0) = \beta(s_f) = 0, \quad \beta'(s) = 2\alpha(s), \\ & && \beta(s) \geq 0, \quad M_{\phi,\delta,\min} \leq M_{\phi,\delta}(s) \leq M_{\phi,\delta,\max}, \end{aligned} \quad (3.7)$$

where the assumptions that the vehicle has zero initial and terminal velocity were made. An explicit time dependency is recovered from the trajectories computed in the solution of (3.7) by using the relation

$$t(s) = \int_{s_0}^s \frac{1}{\sqrt{\beta(\zeta)}} d\zeta, \quad s_0 \leq s \leq s_f, \quad (3.8)$$

which can be utilized for determining the input trajectories as functions of time. The optimal control problem is convex as shown in [Verscheure et al., 2009c], since the cost function is a convex function of the state variable

and the input torques, and the model dynamics is affine in the optimization variables and inputs. It is to be noted that only one state (that is,  $\beta(s)$ ) and one algebraic variable  $\alpha(s)$  are required for formulation of the optimal control problem, compared to originally  $2n$  states required for the dynamics in (2.21).

## 3.2 Discretization and Numerical Solution

For numerical solution of (3.7) using convex optimization tools, the continuous-time optimization problem is discretized using direct transcription according to the procedure in [Verscheure et al., 2009c; Verscheure et al., 2008], where it is assumed that  $\alpha(s)$  is piecewise constant, which implies that  $\beta(s)$  is piecewise linear. Using a grid with  $L$  elements in the interval  $[s_0, s_f]$  and eliminating the torques  $M_{\phi, \delta}(s)$  and the algebraic variable  $\alpha(s)$  from (3.7), result in

$$\begin{aligned} & \underset{\beta_1, \dots, \beta_{L-1}}{\text{minimize}} && \sum_{k=0}^{L-1} \frac{2\Delta s_{k+1}}{\sqrt{\beta_{k+1}} + \sqrt{\beta_k}} \\ & \text{subject to} && \beta_k \geq 0, \quad k = 1, \dots, L-1, \\ & && M_{\phi, \delta, \min} \leq g(s_{k+1/2}) \leq M_{\phi, \delta, \max}, \\ & && k = 0, \dots, L-1, \end{aligned} \quad (3.9)$$

where  $\beta_0 = \beta_L = 0$  and

$$g(s_{k+1/2}) = \Gamma_1(s_{k+1/2}) \frac{\beta_{k+1} - \beta_k}{2\Delta s_{k+1}} + \Gamma_2(s_{k+1/2}) \frac{\beta_{k+1} + \beta_k}{2} + \Gamma_3(s_{k+1/2}), \quad (3.10)$$

with  $\Delta s_{k+1} = (s_{k+1} - s_k)$  and  $s_{k+1/2} = (s_{k+1} + s_k)/2$ . It is straightforward to solve (3.9) for the values of  $\beta(s)$  at the discretization points  $s_k$ ,  $k = 1, \dots, L-1$ , using general-purpose convex optimization tools such as CVX [CVX Research Inc., 2015; Grant and Boyd, 2008]. However, to find the solution online, we compute an approximate solution to (3.9) using a method suggested in [Verscheure et al., 2009b; Verscheure et al., 2009a; Wang and Boyd, 2010], where an unconstrained approximate version of (3.9) is formulated by employing logarithmic barrier-functions [Boyd and Vandenberghe, 2004] for the constraints. This gives that (3.9) approximates to

$$\underset{\beta_1, \dots, \beta_{L-1}}{\text{minimize}} \quad \sum_{k=0}^{L-1} \bar{g}(\beta_k, \beta_{k+1}), \quad (3.11)$$

where

$$\bar{g}(\beta_k, \beta_{k+1}) = \frac{2\Delta s_k}{\sqrt{\beta_{k+1}} + \sqrt{\beta_k}} - \mu \sum_{i=1}^n \left( \log(M_{\phi, \delta, \max} - g_i(s_{k+1/2})) + \log(g_i(s_{k+1/2}) - M_{\phi, \delta, \min}) \right).$$

In (3.11),  $\mu$  denotes the log-barrier parameter,  $n$  is the number of wheel coordinates, and  $g_i$  is element  $i$  in the vector  $g$ . With the unconstrained optimization problem (3.11), an approximately globally optimal solution is computed using Newton's method. This requires computation of the 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 element  $k$  in the Jacobian only depends on  $\beta_{k-1}$ ,  $\beta_k$ , and  $\beta_{k+1}$ . Hence, the time complexity for solving the inherent linear equation system is linear in the number of discretization elements as noted in [Golub and Van Loan, 1996; Verscheure et al., 2009b], which enables fast solutions also in the case of high grid density.

# 4

## High-Level Feedback Controller

The trajectory generator described in Chapter 3 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 robots operating in the same area, 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 completely new path once new sensor data are available, 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, hindering smooth vehicle movement and most certainly increasing the path-traversal time. Another approach is to decrease the velocity along the desired path to avoid the obstacle; however, this method is limited to moving obstacles and cannot handle new static obstacles intersecting the nominal path. Thus, to accommodate dynamic obstacles, not known in the map  $\mathbb{M}$  a priori and possibly moving during runtime, an obstacle-avoidance scheme leading to a local replanning of the path and trajectory is required. This scheme is here integrated with MPC. Consequently, the considered approach provides feedback from global coordinates for robustness to model uncertainties, present when determining the time-optimal trajectories in the optimization, and disturbances in the online task execution.

### 4.1 Obstacle-Avoidance Scheme

We assume that  $m$  range measurements  $\{d_i\}_{i=1}^m$ ,  $d_i \in \mathbb{R}_+$ , of  $l$  objects  $\{o_i\}_{i=1}^l$ ,  $o_i \in \mathbb{O} \subset \mathbb{R}^2$  are available in each time step. The obstacle-avoidance scheme is activated when  $\min_i (\{d_i\}_{i=1}^m) \leq \epsilon_1$ . To allow for a convex problem



formulation, the objects are modeled as hyperplanes  $\pi_o$ , computed from the  $N$  closest points on the object edge. Then the normal vector  $v_\perp$  orthogonal to the vector  $v_o$  between the plane  $\pi_o$  of the closest object  $o_{\min}$ , defined as  $o_{\min} \triangleq \operatorname{argmin}_{o_i}(d_j)$ ,  $i \in \{1, l\}$ ,  $j \in \{1, m\}$ , and the closest point on the hull of the vehicle is determined. The sign of  $v_\perp$  is based on the relative position between the vehicle and obstacle, aiming to minimize the deviation from the nominal path.<sup>1</sup> The norm of  $v_\perp$  is chosen as the corresponding norm of the time-optimal velocity trajectory,  $\|v_{\text{opt}}\|_2$ , for the point on the nominal path that is closest to the vehicle CoG. 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 current point in the  $XY$ -plane is used for computing the path coordinate. The modified velocity vector  $v_{\text{mod}}$ , which replaces  $v_{\text{opt}}$ , is computed as a linear combination of  $v_\perp$  and  $v_{\text{opt}}$ . The motivation for this choice is to allow for a smooth transition between the different velocity vectors. The scheme is illustrated in Figure 4.1 and formally defined in Algorithm 1. For a static obstacle, Algorithm 1 ensures obstacle avoidance since  $v_\perp$  is always directed parallel to the hyperplane approximating the shape of the obstacle at the closest point on the obstacle. If the distance  $d$  between  $o_{\min}$  and the vehicle is smaller than a predefined threshold  $\epsilon_2$ ,  $v_{\text{mod}}$  is increased proportional to the inverse of the distance  $d$ , in order to increase the likelihood of avoiding the obstacle. The scheme is deactivated when the robot is considered to have escaped the obstacle. Here, hysteresis is implemented so as to avoid chattering when close to the predefined threshold  $\epsilon_1$ .

---

**Algorithm 1** Obstacle-Avoidance Scheme
 

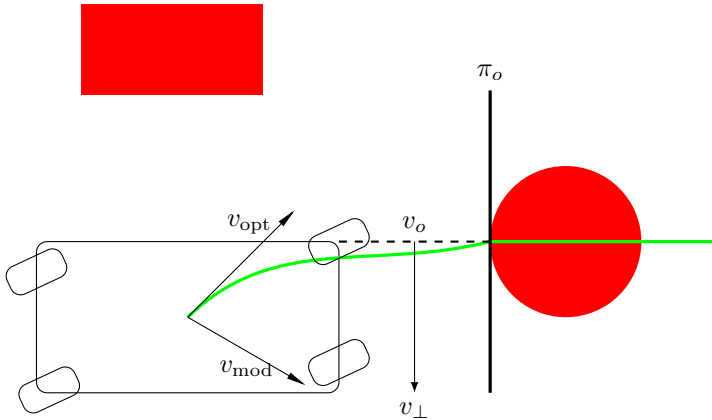
---

```

if  $\exists d_i \leq \epsilon_1$  &  $o_i \notin \mathbb{M}$  then ▷  $\epsilon_1 > 0$ 
     $d \leftarrow$  least element in  $\{d_i\}_{i=1}^m$ 
     $\bar{v}_\perp \leftarrow v_o \times \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^\top$ 
     $v_\perp \leftarrow \|v_{\text{opt}}\|_2 / \|\bar{v}_\perp\|_2 \bar{v}_\perp$ 
    if  $d \leq \epsilon_2$  then ▷  $\epsilon_1 > \epsilon_2 > 0$ 
         $v_{\text{mod}} \leftarrow (1 + c(\epsilon_2/d - 1))v_\perp$  ▷  $c \geq 0$ 
    else
         $v_{\text{mod}} \leftarrow (\epsilon_1 - d)v_\perp + (d - \epsilon_2)v_{\text{opt}}$ 
         $v_{\text{mod}} \leftarrow \|v_{\text{opt}}\|_2 / \|v_{\text{mod}}\|_2 v_{\text{mod}}$ 
    end if
end if ▷ Go back to beginning
    
```

---

<sup>1</sup>Note that the shape of the object is not critical, because the velocity vector is recomputed in each sample based on the new sensor data.



**Figure 4.1** A sketch of how the modified velocity reference is generated in each time instant in the case of dynamic (moving) obstacles. The green line is the nominal path consistent with  $v_{\text{opt}}$ . The sketch shows the case when the modified velocity reference  $v_{\text{mod}}$  is based on the minimal distance to one point. In the implementation, we use the  $N$  closest points on the object seen from the vehicle for robustness.

#### REMARK 2

The idea for approximating the obstacle shape 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.  $\square$

## 4.2 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 torque and velocity trajectories, computed by the time-optimal trajectory generation and collision-avoidance schemes, if sent directly to the internal low-level wheel torque controllers. In addition, wheel slip will lead to a discrepancy between the wheel coordinates and the corresponding global Cartesian vehicle pose. We use MPC to introduce feedback from the estimated global pose of the vehicle on a high level in a hierarchical control architecture, with the aim of suppressing the effects of model uncertainties and disturbances. The precomputed trajectories, computed as in Chapter 3, can here be seen as time-optimal feedforward control inputs. The control input to the vehicle is the desired global velocity. In practice,

this velocity is realized with torque-resolved wheel-rotational velocity controllers. In the case of no obstacles, the MPC computes references for the global velocity based on the time-optimal wheel velocity trajectories. Without disturbances and model errors, and with appropriate MPC tuning, this results in a path traversal according to the computed time-optimal control as computed in Chapter 3.

The MPC approach has the benefit that it naturally admits obstacle avoidance with a local replanning of the path and trajectory according to Algorithm 1. The reference values for the global velocity determined by the MPC are transformed to wheel velocities using the inverse differential kinematics of the vehicle and subsequently applied on each wheel. A kinematic model derived in discrete time is considered in the MPC design on the form

$$x_{k+1} = Ax_k + Bu_k, \quad (4.1)$$

with

$$x_k^T = (p_k^T \quad v_k^T \quad \psi_k \quad \dot{\psi}_k), \quad u_k^T = (v_{k,\text{ref}}^T \quad \dot{\psi}_{k,\text{ref}})$$

where  $p_k \in \mathbb{R}^2$  is the position in the  $XY$ -plane,  $v_k \in \mathbb{R}^2$  is the velocity in the  $XY$ -plane,  $\psi_k, \dot{\psi}_k \in \mathbb{R}$  are the yaw angle and yaw rate, respectively, and  $u_k \in \mathbb{R}^3$  contains the corresponding control inputs (reference values to the internal wheel controllers in the vehicle). All variables are expressed with respect to an earth-fixed, inertial frame (see Figure 2.1). Moreover,

$$A = \begin{pmatrix} 1 & 0 & \eta_1 & 0 & 0 & 0 \\ 0 & 1 & 0 & \eta_1 & 0 & 0 \\ 0 & 0 & \eta_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \eta_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \eta_1 \\ 0 & 0 & 0 & 0 & 0 & \eta_2 \end{pmatrix}, \quad B = \begin{pmatrix} \eta_3 & 0 & 0 \\ 0 & \eta_3 & 0 \\ \eta_4 & 0 & 0 \\ 0 & \eta_4 & 0 \\ 0 & 0 & \eta_3 \\ 0 & 0 & \eta_4 \end{pmatrix}$$

with

$$\eta_1 = T(1 - T\sigma), \quad \eta_2 = 1 - T\sigma, \quad \eta_3 = \frac{T^2\sigma}{2}, \quad \eta_4 = T\sigma,$$

where  $T$  is the sample time and  $\sigma$  represents the time constant of the low-level wheel control loops. The rationale behind (4.1) is that the wheel control loops ensure accurate velocity-reference tracking. The assumptions on mechanical properties of the vehicle imply that the translational and rotational dynamics are almost independent from each other. Hence a decoupled, kinematic model is sufficient. This is motivated here, since it is combined with low-level feedback in an hierarchical control architecture, where less complex models are natural in the high-level layer. With the weighting matrices  $Q$  and  $R$ , the

quadratic cost function in the MPC is<sup>2</sup>

$$\mathcal{J}_k = \sum_{m=1}^{H_p} \|x_{k+m} - r_{k+m}\|_Q^2 + \sum_{m=0}^{H_c-1} \|u_{k+m}\|_R^2, \quad (4.2)$$

where  $\|x\|_Q = x^T Q x$ , correspondingly for  $\|u\|_R$ , and  $r_m$  are the position and velocity references at time step  $m$  computed by the time-optimal trajectory generator or the local trajectory replanner in Algorithm 1. The optimization problem in the MPC at time step  $k$  is

$$\underset{\mathcal{U}_k}{\text{minimize}} \quad \mathcal{J}_k \quad (4.3a)$$

$$\text{subject to} \quad x_{k+m+1} = Ax_{k+m} + Bu_{k+m} \quad (4.3b)$$

$$x_{k+m+1, \min} \leq x_{k+m+1} \leq x_{k+m+1, \max} \quad (4.3c)$$

$$u_{\min} \leq u_{k+i} \leq u_{\max}, \quad (4.3d)$$

$$\Delta u_{\min} \leq (u_{k+i} - u_{k+i-1})/T \leq \Delta u_{\max} \quad (4.3e)$$

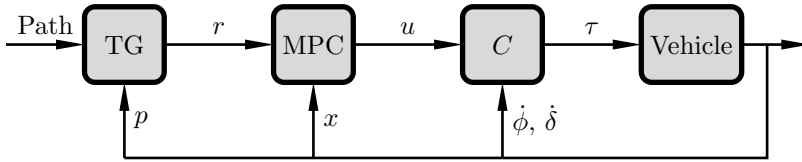
$$x_k = \bar{x} \quad (4.3f)$$

$$m = 0, \dots, H_p - 1, \quad i = 0, \dots, H_c - 1$$

where  $\mathcal{U}_k = \{u_k, \dots, u_{k+H_c-1}\}$  is the set of control inputs to be determined,  $\bar{x}$  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  $u_i$  is assumed to be constant and equal to  $u_{k+H_c-1}$  for all  $i \geq k + H_c$ . In (4.3), the position constraints included in (4.3c) are given by rectangular approximations of the surrounding a priori known obstacles in the map, and (4.3d)–(4.3e) are implied by the physical properties of the robot. As a summary of the high-level feedback controller, the complete control architecture is visualized in Figure 4.2.

---

<sup>2</sup>We exclude weights on the final state due to the nature of the a priori unknown dynamic obstacles. Given the vehicle dynamics, it is plausible that this does not imply stability violations. Further, it is easily added if desired.



**Figure 4.2** The control architecture. 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 internal wheel controllers,  $C$ , transform the Cartesian velocities to the respective wheel. Based on these references, wheel torques are computed. The trajectory generator only computes new trajectories when a new path is available.

# 5

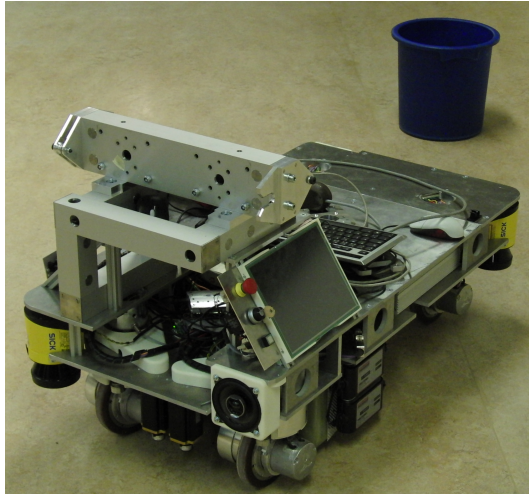
## Experimental Results

To validate the proposed approach to trajectory generation, path tracking, and obstacle avoidance for four-wheeled vehicles, the methods discussed in Secs. 3–4 were implemented and subsequently experimentally verified in a relevant scenario on a pseudo-omnidirectional mobile robot platform. The particular scenario was chosen to illustrate the capabilities of the algorithms in factory-type setups.

### 5.1 Experimental Setup

The robot employed for the experimental validation was a four-wheeled pseudo-omnidirectional mobile robot equipped with eight motors, two for each wheel realizing the steering and driving, see Figure 5.1. The mobile robot [Weisshardt and Garcia, 2014], which was built and designed at Fraunhofer IPA in Stuttgart, is the successor to the Care-O-Bot 3 mobile base [Connette et al., 2009; Weisshardt and Garcia, 2014]. It was equipped with two SICK S300 laser scanners, which delivered laser-range measurements from the front and rear corners of the robot. The robot was controlled and sensor data were acquired using the ROS software package [WillowGarage, 2015]. The wheel-encoder position and velocity measurements were extracted at a rate of 100 Hz. The individual wheels were controlled with torque-resolved cascaded position and velocity controllers running at 100 Hz. The controllers were implemented in C++ and executed internally in ROS. To estimate the parameters in the effective inertia matrix  $I_e(\xi)$  and Coulomb-friction parameter vector  $F_C^\xi$  required for the robot model in (2.21), experimental data were collected. Under the assumption that the translational motion of the robot is significantly larger than the rotational motion, the matrix is approximated to be diagonal with inertia elements

$$I_e(\xi) = \text{diag} \{ I_{\phi,1}^e, I_{\phi,2}^e, I_{\phi,3}^e, I_{\phi,4}^e, I_{\delta,1}^e, I_{\delta,2}^e, I_{\delta,3}^e, I_{\delta,4}^e \}, \quad (5.1)$$



**Figure 5.1** The mobile platform 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 corners of the robot were used for obstacle detection, localization, and map building.

where  $\text{diag}\{\cdot\}$  is a diagonal matrix with the specified elements along the diagonal. 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 torque measurements were accessible within the mobile robot platform via ROS. For details regarding estimating  $F_C^\xi$ , see [Berntorp et al., 2014a].

## 5.2 Implementation and Software Architecture

Figure 5.2 shows a schematic representation of the implementation structure. The path planner implemented in ROS generated a feasible geometric path, given a predefined static map of the environment, using Dijkstra's algorithm [LaValle, 2006]. The map was determined prior to the experiment based on data from the two laser scanners, attached to the corners of the mobile robot-platform, combined with the wheel odometry, using the Simultaneous Localization and Mapping (SLAM) algorithm described in [Grisetti et al., 2007; Grisetti et al., 2005]. The generated path was sent to the time-optimal trajectory generator, which provided wheel position and velocity trajectories

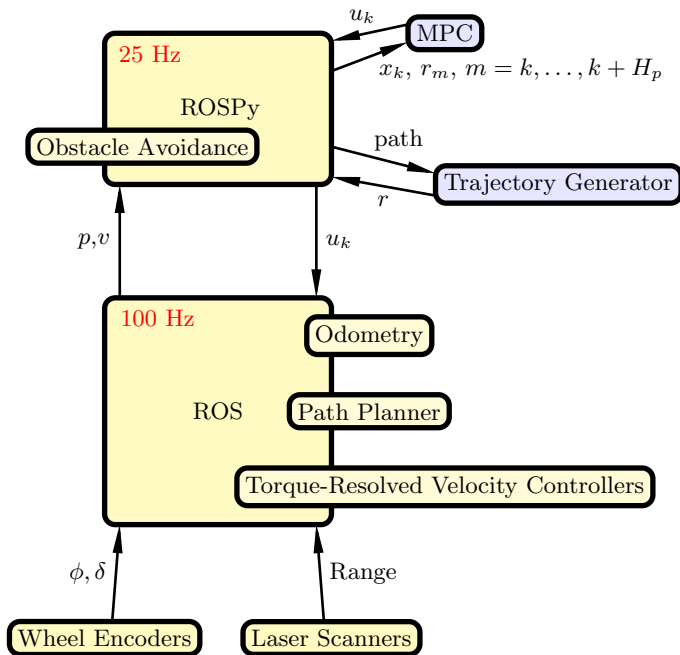
corresponding to the time-optimal input torques. The wheel-velocity references given by the trajectory generator were subsequently transformed to Cartesian velocity references using the forward differential kinematics. These velocity trajectories and the corresponding position trajectories were then 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 with the laser-scanner data and the wheel odometry.

The log-barrier solver in Section 3.2 was implemented in MATLAB and then transformed to C code using the Coder toolbox in MATLAB and compiled. The MPC was implemented in C using CVXGEN [Mattingley and Boyd, 2012], which resulted in an average solution time of 1 ms for the model at hand and the prediction horizons considered ( $H_p = H_c = 10$ , corresponding to a horizon of 0.4 s, was found to result in desired tracking behavior). Considering the fast solution times for the MPC, longer prediction horizons would therefore be possible if necessary. Further, it is not the MPC computations that are limiting the choice of sample time, but rather the navigation and computation algorithms in ROS running on the same computer. To link the developed controllers with the robot operating system, we employed a Python abstraction of ROS, denoted ROSPy, which executed at a rate of 25 Hz. To invoke the trajectory generator and the MPC, both implemented in C, from Python the `ctypes` library [Python Software Foundation, 2015] was employed.

### 5.3 Path-Tracking Scenario

The experiments were performed in a room with an area of approximately 75 m<sup>2</sup>. The map of the room, resulting from application of the SLAM algorithm, is shown in Figure 5.3. Based on this map, a geometric path was planned from the coordinate (0, 1.5) m to (7, -1.5) m; the orientation of the robot platform was computed such that the robot heading  $\psi$  was constant along the path. The time-optimal trajectory generation was performed, resulting in the velocity trajectories shown in Figure 5.4 (expressed in Cartesian  $XY$ -coordinates). The constraints on the steering actuators were introduced as  $M_{\delta, \max} = 0.27$  and on the driving actuators as  $M_{\phi, \max} = 0.31$  and symmetrically for the lower bound. These constraints were chosen based on the physical properties of the wheel motors in the mobile robot. Note that the wheel torques have been normalized. These constraints were chosen with a slight conservatism to provide actuation capability in the MPC for the obstacle avoidance. The corresponding time-optimal torques for the steering and driving motors are shown in Figure 5.5. Note that at least one input torque is at the limit at each time point, indicating the desired time-optimality [Chen and Desrochers, 1989]. In addition, it is clear that the driving motors are





**Figure 5.2** A sketch of the implementation structure. The odometry, navigation, and torque-resolved velocity controllers can be interacted with through ROS. They consist of compiled C++ code. We use a Python abstraction of ROS, called ROSPy, and connect the MPC and log-barrier solver to ROSPy using compiled C code via the `ctypes` library.

the limiting factors, which is expected from the geometry of the path with only limited orientation changes of the wheels. Still, there is significant motion perpendicular to the orientation of the platform, resulting in the steer torques observed in the upper plot in Figure 5.5.

Initially, in the absence of dynamic obstacles, the mobile robot tracked the nominal path and the time-optimal trajectories using the MPC. The controller parameters (that is, the weight matrices in (4.2)) were chosen as

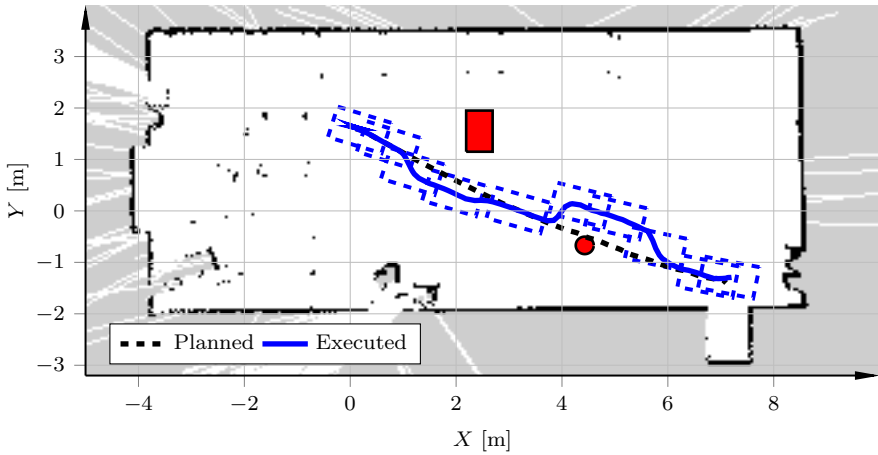
$$Q = \text{diag}\{1, 1, 0.1, 0.1, 0.1, 0.1\}, \quad R = \text{diag}\{0.1, 0.1, 1\}.$$

The constraints on the control signal were

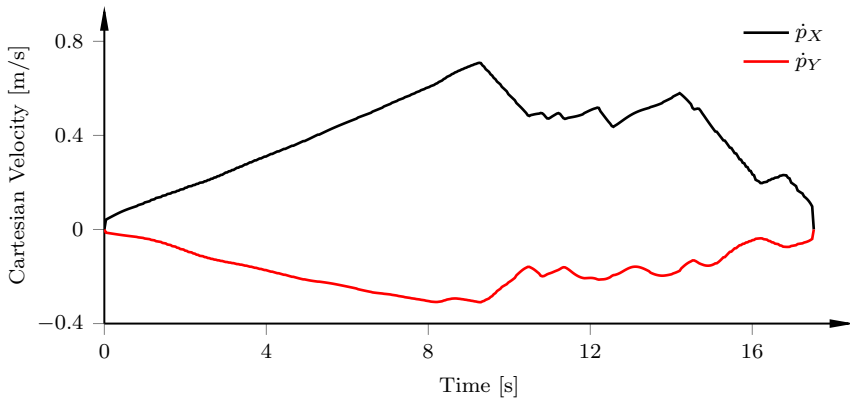
$$u_{\max} = (1 \quad 1 \quad 0.1)^T,$$

and  $u_{\min} = -u_{\max}$  with the units m/s and rad/s, whereas the slew-rate limit was chosen as

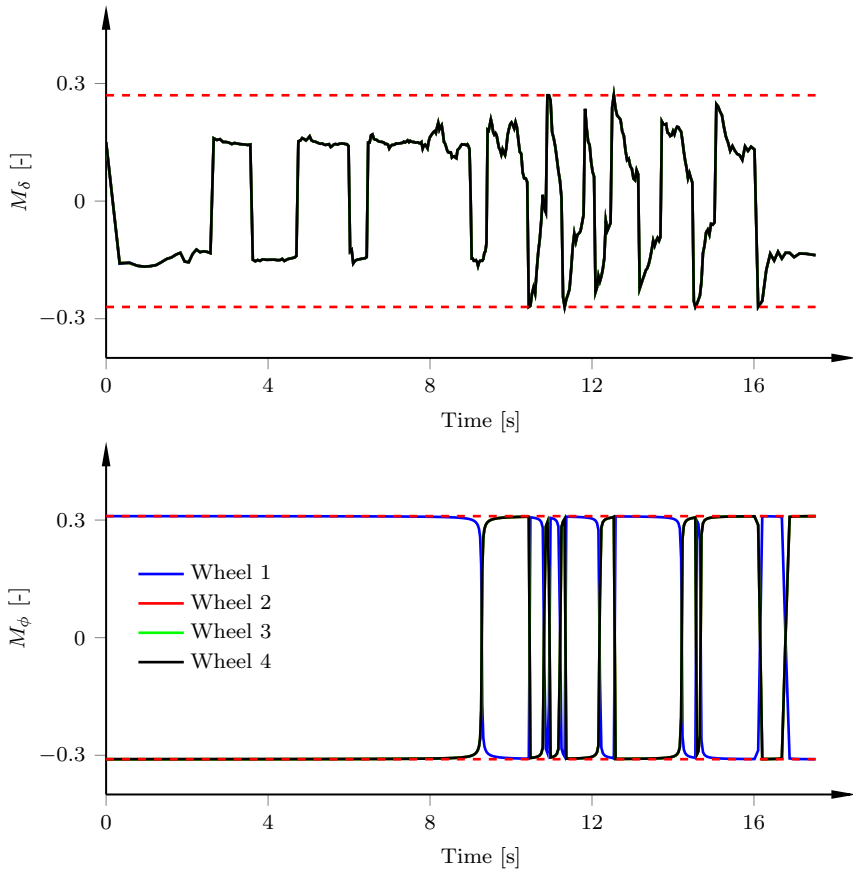
$$\Delta u_{\max} = (0.5 \quad 0.5 \quad 0.05)^T,$$



**Figure 5.3** The scenario considered for evaluation of the proposed approach to trajectory generation for four-wheeled vehicles. The nominal planned path is shown 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.

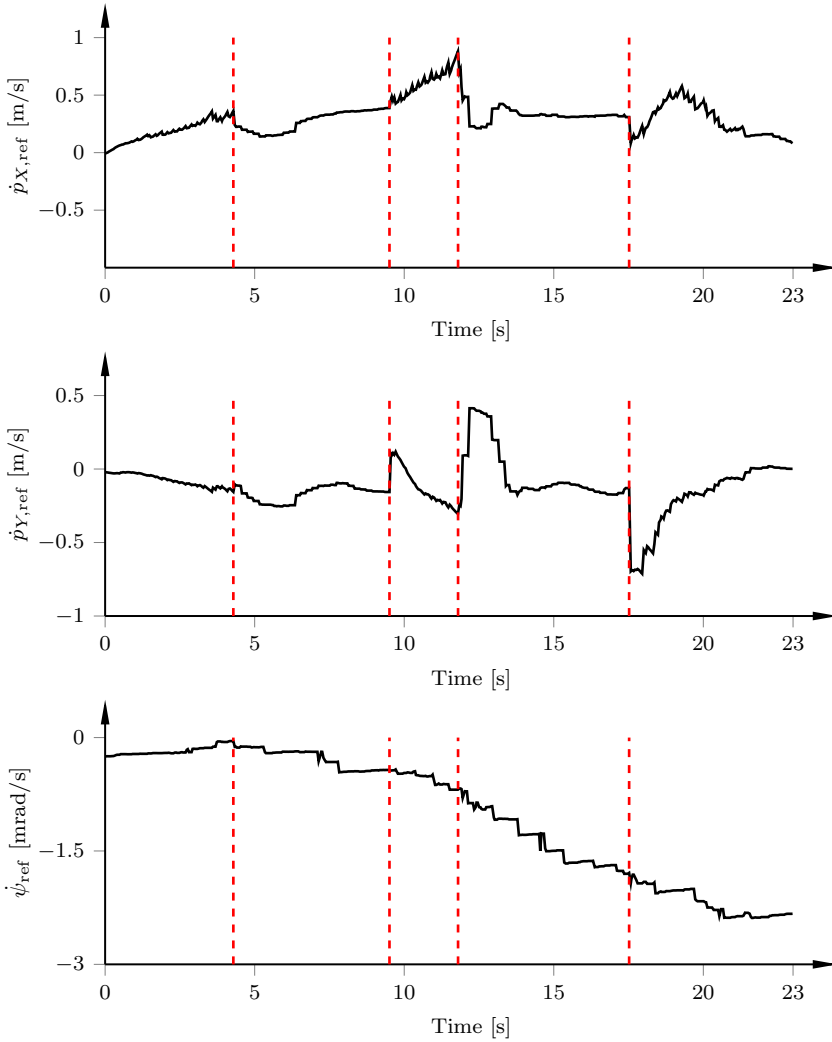


**Figure 5.4** Time-optimal velocity-trajectory references in the Cartesian  $XY$ -space obtained from the trajectory generator, measured in global coordinates.



**Figure 5.5** Time-optimal normalized motor-torque trajectories obtained from the trajectory generator. The torque constraints are indicated by the horizontal dashed red lines. Notice that due to the mechanical design of the robot, the wheel driving torques are pairwise equal and that the wheels are mounted in opposite directions to each other. Moreover, the steering torques are equal for all wheels since the orientation of the robot is fixed along the path.

and  $\Delta u_{\min} = -\Delta u_{\max}$  with the units  $\text{m/s}^2$  and  $\text{rad/s}^2$ . After the robot started to move along the nominal path, two new obstacles were placed such that they intersected the nominal geometric path (where the obstacle locations are not encoded in the static map defined previously). These obstacles were instead detected during runtime using the laser scanner sensors. To minimize the path deviation because of the obstacles, the threshold value  $\epsilon_1$  was chosen as short as possible; in this case to 400 mm. Further, the parameters  $c$  and  $\epsilon_2$  in Algorithm 1 were zero in the considered experiment, because the obstacles were assumed to have slowly varying or static positions, rendering the additional term negligible. The executed path is shown in Figure 5.3 together with the map of the environment. The corresponding Cartesian velocity references computed by the MPC are shown in Figure 5.6, where also the time instants at which the obstacle avoidance scheme was activated are displayed. From Figure 5.4 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. Further, the downmost plot in Figure 5.6 indicates that the coupling between translational and rotational movement is small, implying that the decoupled model (4.1) is indeed a valid approximation in this experiment.



**Figure 5.6** Cartesian velocity commands, measured in the global  $XY$ -coordinate system, computed by the MPC based on the time-optimal trajectories and the obstacle avoidance scheme. The instants at which switches between path tracking and obstacle avoidance occur are indicated by the vertical dashed red lines. Note that the time-optimal trajectories in Figure 5.4 are closely tracked when not avoiding the new obstacles.

# 6

## Discussion

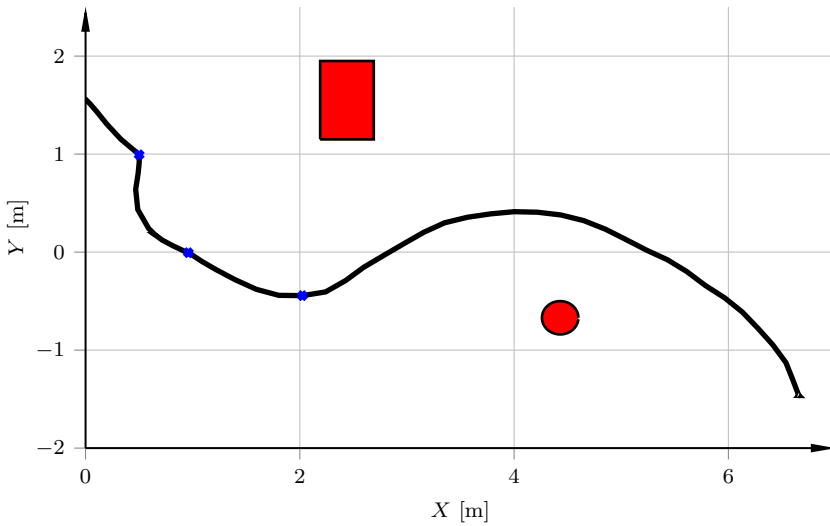
A key feature of the considered system architecture is that it combines generation of time-optimal reference trajectories (based on a nonlinear model of the vehicle incorporating friction) with a feedback controller for path tracking, where the problems of finding the reference trajectories and control signals are posed as convex optimization problems. This implies that an optimal solution will be found quickly, 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 (resulting in  $L = 500$  discretization elements in the optimization problem (3.9)), and the solution time for the MPC is almost always within 1 ms (corresponding to 5–10 iterations) for the scenarios we have tested. In our control architecture, the trajectory generator is only executing when a completely new path is required. Thus, a major part of the time in the motion-planning phase is spent on path planning, since this typically requires significantly longer time than the proposed trajectory-generation algorithm. Further, considering different optimization criteria than time, the trajectory generator can be modified to minimize combinations of path-traversal time and, for example, energy consumption. In [Grundelius, 2001], it is shown that solution of the minimum-energy optimal control problem over a fixed time-horizon, where the final time is chosen slightly longer than the corresponding time-optimal, is beneficial for robustness of the control.

Another desired property inherent in the architecture is that the MPC suppresses the effect of model errors. We showed in [Berntorp et al., 2014a] that the model errors caused by the no-slip assumption are small, but combining the proposed approach to time-optimal trajectory generation using a model without slip decreases the influence of this model limitation even further. Moreover, there are other imperfections present as well, such as uncertainties in the geometry. Since the MPC uses the global pose estimate for feedback, it effectively decreases the effects of model uncertainties when combined with the low-level wheel controllers. This is a difference to many existing reference implementations in mobile robots, where a reference tra-

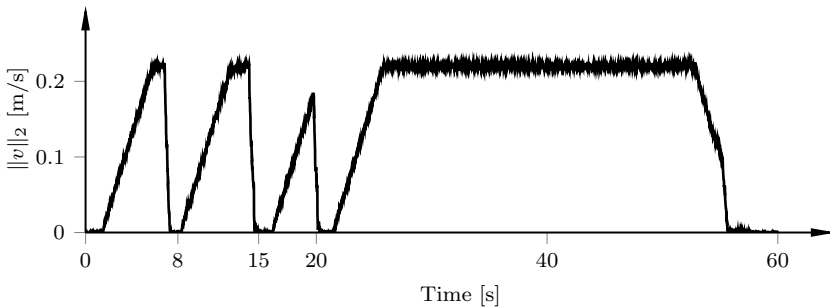
jectory is generated and then typically fed to the low-level loops without global feedback from workspace estimates.

As mentioned before, a motivation for using online trajectory and local path regeneration rather than replanning the complete path when an obstacle is encountered, is that it is sometimes desirable to stay close to the original (nominal) path. Another motivation is that successive replanning and trajectory generation might prevent task effectiveness. To demonstrate this, we used the robot's internal proprietary navigation module and applied it to the same scenario as considered in Chapter 5. 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 considers constraints on a kinematic level. Thus, it is not using the full potential of the wheel motors. Figure 6.1 shows the path traversed by the robot and Figure 6.2 shows the Euclidean norm of the velocity vector along the path. The same scenario that takes approximately 25 s to complete with our architecture now demands roughly 60 s. The longer execution time for the reference method is expected since the objective is not time-optimality. Moreover, there are more restrictive velocity constraints in the internal navigation module than in the approach presented in this report. More interesting, however, is that the robot stops and finds new feasible paths three times in total, with each replanning lasting about 1 s. Hence, it is clear that an approach that performs online collision avoidance based on local regeneration of the trajectory is advantageous when task effectiveness is desired. By inspection of Figure 6.1 it is also obvious that the resulting geometric path differs significantly from the nominal path, shown in Figure 5.3. In addition, note that if it really is desired to initialize a replanning of the path and trajectories when an obstacle is encountered, this is easily achievable with the trajectory generator in this report, which enables fast solution times for the reoptimization of the trajectory. Hence, it can be used together with the low-level wheel controllers to improve the current implementation of the navigation module.

The considered approach was verified and evaluated on a mobile-robot setup, employing relatively low velocities, which are typical for mobile robots in industrial production and manufacturing shop floors. However, the architecture could be valid for more scenarios where four-wheeled vehicles with independent steering and driving are employed. When considering vehicle platooning the routes are predetermined using maps of the available paths, the global positions are received from a global positioning system, and obstacles are typically detected using vision and/or sonar measurements. In these cases, it is not time-optimality alone that is the objective. Rather, it should be combined with other criteria, such as energy consumption, and with constraints on the acceleration and the jerk of the vehicle to allow for smooth and stable driving.



**Figure 6.1** Resulting geometric path for the CoG when instead using the internal navigation module in the considered mobile robot platform in a comparative study. The locations at which replanning of the path occurs are marked with blue +. The motion is performed from the coordinate (0, 1.5) m to (7, -1.5) m.



**Figure 6.2** Euclidean norm of the velocity vector when instead using the considered robot's internal navigation module in a comparative study for the scenario in Figure 5.3. In contrast to the architecture in this report, the robot stops for replanning purposes at  $t = 8, 15,$  and  $20$  seconds.





# 7

## Conclusions

We have considered an approach to time-optimal trajectory generation and online path tracking with obstacle avoidance for four-wheeled vehicles with independent steering and driving. The approach is based on convex optimization, allowing fast computations both for trajectory generation and online control. The obstacle-avoidance scheme was integrated in a high-level feedback controller based on MPC. The proposed architecture was fully implemented on a pseudo-omnidirectional mobile platform and evaluated in experiments in a demanding path-tracking scenario. The method was shown to perform well, and exhibited several advantages in comparison to a reference method, especially in terms of traversal time and velocity smoothness.



# Acknowledgments

Björn Olofsson and Anders Robertsson are members of the LCCC Linnaeus Center and the ELLIIT Excellence Center at Lund University. This research was supported by the Swedish Foundation for Strategic Research through the project ENGROSS and the European Commission's Seventh Framework Program under grant agreement SMERobotics (ref. #287787). This research was not sponsored by Mitsubishi Electric or any of its subsidiaries.



# Bibliography

- 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: *Proc. IFAC World Congress*. Milano, Italy, pp. 14648–14653.
- Berntorp, K, B Olofsson, and A Robertsson (2014a). “Path tracking with obstacle avoidance for pseudo-omnidirectional mobile robots using convex optimization”. In: *Proc. Am. Control Conf. (ACC)*. Portland, OR, pp. 517–524.
- Berntorp, K. and F. Magnusson (2015). “Hierarchical predictive control for ground-vehicle maneuvering”. In: *Proc. American Control Conf.* Chicago, IL, pp. 2771–2776.
- Berntorp, K., B. Olofsson, K. Lundahl, and L. Nielsen (2014b). “Models and methodology for optimal trajectory generation in safety-critical road-vehicle manoeuvres”. *Vehicle System Dynamics* **52**:10, pp. 1304–1332.
- Bobrow, J. E., S Dubowsky, and J. S. Gibson (1985). “Time-optimal control of robotic manipulators along specified paths”. *Int. J. Robotics Research* **4**:3, pp. 3–17.
- Boyd, S. and L. Vandenberghe (2004). *Convex Optimization*. 6th ed. Cambridge Univ. Press, Cambridge, UK.
- Castro, R. de, M. Tanelli, R. E. Araújo, and S. M. Savaresi (2014). “Minimum-time path following in highly redundant electric vehicles”. In: *Proc. IFAC World Congress*. Cape Town, South Africa, pp. 3918–3923.
- Chen, Y. and A. A. Desrochers (1989). “Structure of minimum-time control law for robotic manipulators with constrained paths”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. Scottsdale, AZ, pp. 971–976.
- Choi, J.-W., R. E. Curry, and G. H. Elkaim (2009). “Obstacle avoiding real-time trajectory generation and control of omnidirectional vehicles”. In: *Proc. Am. Control Conf. (ACC)*. St. Louis, MI, pp. 5510–5515.

- Connette, C. P., C. Parlitz, M. Hägele, and A. Verl (2009). “Singularity avoidance for over-actuated, pseudo-omnidirectional, wheeled mobile robots”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. Kobe, Japan, pp. 1706–1712.
- Connette, C. P., S. Hofmeister, A. Bübeck, M. Hägele, and A. Verl (2010). “Model-predictive undercarriage control for a pseudo-omnidirectional, wheeled mobile robot”. In: *Proc. 41st Int. Symp. Robotics (ISR) and 6th German Conf. Robotics (ROBOTIK)*. Munich, Germany, pp. 1–6.
- CVX Research Inc. (2015). *CVX: matlab software for disciplined convex programming, version 2.0 beta*. <http://cvxr.com/cvx>, Accessed: 2015-01-12.
- Dahl, O. (1992). *Path Constrained Robot Control*. ISRN LUTFD2/TFRT-1038--SE. PhD thesis. Department of Automatic Control, Lund University, Sweden.
- Dahl, O. (1993). “Path constrained motion optimization for rigid and flexible joint robots”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. Atlanta, GA, pp. 223–229.
- Dahl, O. and L. Nielsen (1990). “Torque limited path following by on-line trajectory time scaling”. *IEEE Trans. Robot. and Autom.* **6**:5, pp. 554–561.
- 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 Trans. Robot.* **29**:6, pp. 1485–1495.
- Fox, D., W. Burgard, and S. Thrun (1997). “The dynamic window approach to collision avoidance”. *IEEE Robot. Autom. Mag.* **4**:1, pp. 23–33.
- Golub, G. H. and C. F. Van Loan (1996). *Matrix Computations*. 3rd ed. The Johns Hopkins Univ. Press, Baltimore, MD.
- Grant, M. and S. Boyd (2008). “Graph implementations for nonsmooth convex programs”. In: Blondel, V. et al. (Eds.). *Recent Advances in Learning and Control*. Springer-Verlag, Berlin, Heidelberg, Germany, pp. 95–110.
- Grisetti, G., C. Stachniss, and W. Burgard (2005). “Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. Barcelona, Spain, pp. 2432–2437.
- Grisetti, G., C. Stachniss, and W. Burgard (2007). “Improved techniques for grid mapping with Rao-Blackwellized particle filters”. *IEEE Trans. Robot.* **23**, pp. 34–46.
- Grundelius, M. (2001). *Methods for Control of Liquid Slosh*. ISRN LUTFD2/TFRT--1062--SE. PhD thesis. Department of Automatic Control, Lund University, Sweden.

- Howard, T., C. Green, and A. Kelly (2009). “Receding horizon model-predictive control for mobile robot navigation of intricate paths”. In: *Proc. 7th Int. Conf. Field and Service Robotics*. Cambridge, MA.
- Kanjanawanishkul, K. and A. Zell (2009). “Path following for an omnidirectional mobile robot based on model predictive control”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. Kobe, Japan, pp. 3341–3346.
- Kant, K. and S. W. Zucker (1986). “Toward efficient trajectory planning: the path-velocity decomposition”. *Int. J. Robotics Research* **5**:3, pp. 72–89.
- Khatib, O. (1986). “Real-time obstacle avoidance for manipulators and mobile robots”. *Int. J. Robotics Research* **5**:1, pp. 90–98.
- 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.
- LaValle, S. M. (2006). *Planning Algorithms*. Cambridge Univ. Press, Cambridge, UK.
- Lipp, T. and S. Boyd (2014). “Minimum-time speed optimization over a fixed path”. *Int. J. Control* **87**:6, pp. 1297–1311.
- Maciejowski, J. M. (1999). *Predictive Control with Constraints*. Addison-Wesley, Boston, MA.
- Mattingley, J. and S. Boyd (2012). “CVXGEN: A Code Generator for Embedded Convex Optimization”. *Optimization and Engineering* **13**:1, pp. 1–27.
- Mayne, D. Q., J. B. Rawlings, C. V. Rao, and P. O. Scokaert (2000). “Constrained model predictive control: stability and optimality”. *Automatica* **36**:6, pp. 789–814.
- Norén, C. (2013). *Path Planning for Autonomous Heavy Duty Vehicles using Nonlinear Model Predictive Control*. LiTH-ISY-EX-13/4707-SE. Linköping Univ., Linköping, Sweden.
- Oftadeh, R., R. Ghabcheloo, and J. Mattila (2014). “Time optimal path following with bounded velocities and accelerations for mobile robots with independently steerable wheels”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. Hong Kong, China, pp. 2925–2931.
- Pacejka, H. B. (2006). *Tire and Vehicle Dynamics*. 2nd ed. Butterworth-Heinemann, Oxford, United Kingdom.
- Pfeiffer, F. and R. Johanni (1987). “A concept for manipulator trajectory planning”. *IEEE J. Robot. Autom.* **3**:2, pp. 115–123.
- Python Software Foundation (2015). *Ctypes — A foreign function library for Python*. <http://docs.python.org/2/library/ctypes.html>, Accessed: 2015-01-12.



- 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 Trans. Rob.* **20**:6, pp. 978–993.
- Quinlan, S. and O. Khatib (1993). “Elastic bands: connecting path planning and control”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. Atlanta, GA, pp. 802–807.
- Schindler, E. (2007). *Fahrdynamik: Grundlagen Des Lenkverhaltens Und Ihre Anwendung Für Fahrzeugregelsysteme*. Expert-Verlag, Renningen, Germany.
- Shin, K. G. and N. D. McKay (1985). “Minimum-time control of robotic manipulators with geometric path constraints”. *IEEE Trans. Autom. Control* **30**:6, pp. 531–541.
- Spong, M. W., S. Hutchinson, and M. Vidyasagar (2006). *Robot Modeling and Control*. John Wiley and Sons, Hoboken, NJ.
- The Orocos Project (2015). *Orocos—Open robot control software*. Accessed: 2015-01-12. URL: <http://www.orocos.org>.
- Van Loock, W., G. Pipeleers, and J. Swevers (2013). “Time-optimal path planning for flat systems with application to a wheeled mobile robot”. In: *Proc. Workshop Robot Motion and Control (RoMoCo)*. Wasowo, Poland, pp. 192–196.
- 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: *Proc. 10th Int. Workshop Advanced Motion Control*. Trento, Italy, pp. 727–732.
- Verscheure, D., M. Diehl, J. De Schutter, and J. Swevers (2009a). “On-line time-optimal path tracking for robots”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. Kobe, Japan, pp. 599–605.
- Verscheure, D., M. Diehl, J. De Schutter, and J. Swevers (2009b). “Recursive log-barrier method for on-line time-optimal robot path tracking”. In: *Proc. Am. Control Conf. (ACC)*. St. Louis, MI, pp. 4134–4140.
- Verscheure, D., B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl (2009c). “Time-optimal path tracking for robots: a convex optimization approach”. *IEEE Trans. Autom. Control* **54**:10, pp. 2318–2327.
- Wang, Y. and S. Boyd (2010). “Fast model predictive control using online optimization”. *IEEE Trans. Control Syst. Technol.* **18**:2, pp. 267–278.
- Weisshardt, F. and N. H. Garcia (2014). *Care-O-bot Manual: Manual for Care-O-bot users and administrators*. Fraunhofer IPA, Institute for Manufacturing Engineering and Automation, Stuttgart, Germany.
- WillowGarage (2015). *Robot Operating System*. Accessed: 2015-01-12. URL: <http://www.ros.org>.

Yuille, A. L. and A. Rangarajan (2003). *The Concave-Convex Procedure*.  
Vol. 15. 4. Neural Computation, MIT Press, Cambridge, MA, pp. 915–  
936.