An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles

Conference Paper · July 2013

DOI: 10.23919/ECC.2013.6669836

CITATIONS
177

READS
2,315

7 authors, including:

Andrew Gray
University of California, Berkeley
10 PUBLICATIONS **1,081** CITATIONS
SEE PROFILE

Mario Zanon
IMT School for Advanced Studies Lucca
146 PUBLICATIONS **2,977** CITATIONS
SEE PROFILE

Joachim Ferreau
Embotech AG
61 PUBLICATIONS **5,421** CITATIONS
SEE PROFILE

Sebastian Sager
Otto-von-Guericke-Universität Magdeburg
112 PUBLICATIONS **2,411** CITATIONS
SEE PROFILE

An Auto-generated Nonlinear MPC Algorithm for Real-Time Obstacle Avoidance of Ground Vehicles*

Janick V. Frasch[1,2,5], Andrew Gray[2], Mario Zanon[3],
Hans Joachim Ferreau[3,4], Sebastian Sager[5], Francesco Borrelli[2], and Moritz Diehl[3]

***Abstract*— We address the problem of real-time obstacle avoidance on low-friction road surfaces using spatial Nonlinear Model Predictive Control (NMPC). We use a nonlinear four-wheel vehicle dynamics model that includes load transfer. To overcome the computational difficulties we propose to use the ACADO Code Generation tool which generates NMPC algorithms based on the real-time iteration scheme for dynamic optimization. The exported plain C code is tailored to the model dynamics, resulting in faster run-times in effort for real-time feasibility. The advantages of the proposed method are shown through simulation.**

I. Introduction

Recent contributions to theory and algorithms have enlarged the application spectrum of real-time Model Predictive Control (MPC), cf. [3], [5], [10], [18], [26]. In the area of semi-autonomous vehicle control, MPC has become an attractive method for the reliable tracking of feasible trajectories by ground vehicles [7]. The real-time trajectory generation however, particularly in the presence of obstacles, remains very challenging. Trajectories generated by using linearized or oversimplified models can ignore important nonlinear dynamics that play a major role when the vehicle is operated close to the limits of its handling capability. This often demands a compromise in the MPC design between limited capabilities and violation of system constraints, particularly in the presence of external disturbances and model uncertainties. On the other hand, due to the computational demand of nonlinear optimization methods, nonlinear MPC (NMPC) is still not generally applicable to agile autonomous vehicles, thus limiting the usability of detailed nonlinear vehicle models.

* This research was supported by Research Council KUL: PFV/10/002 Optimization in Engineering Center OPTEC, GOA/10/09 MaNet and GOA/10/11 Global real- time optimal control of autonomous robots and mechatronic systems. Flemish Government: IOF/KP/SCORES4CHEM, FWO: PhD/postdoc grants and projects: G.0320.08 (convex MPC), G.0377.09 (Mechatronics MPC); IWT: PhD Grants, projects: SBO LeCoPro; Belgian Federal Science Policy Office: IUAP P7 (DYSCO, Dynamical systems, control and optimization, 2012-2017); EU: FP7-EMBOCON (ICT-248940), FP7-SADCO (MC ITN-264735), ERC ST HIGHWIND (259 166), Eurostars SMART, ACCM.

[1]Interdisciplinary Center for Scientific Computing (IWR), University of Heidelberg. Im Neuenheimer Feld 368, D-69120 Heidelberg, Germany janick.frasch at iwr.uni-heidelberg.de

[2]Department of Mechanical Engineering, UC Berkeley. 2169 Etcheverry Hall, Berkeley, CA, 94720, USA

[3]Department of Electrical Engineering, KU Leuven. Kasteelpark Arenberg 10, B-3001 Leuven, Belgium

[4]ABB Corporate Research. Segelhofstrasse 1K, CH-5405 Baden-Dättwil, Switzerland

[5]Institute for Mathematical Optimization, Faculty of Mathematics, Otto von Guericke University Magdeburg, Magdeburg, Germany

A simplification approach frequently chosen in previous work to reduce the computational effort is the decomposition of the problem into a two-level NMPC problem, featuring a high level path planner utilizing a simple model on a long prediction horizon, and a low level path follower utilizing a more detailed model on a short prediction horizon.

In [13] a motion primitive path planner is used at the high level, while a 6-state nonlinear bicycle model similar to [7] is used at the lower level. This approach however limits the vehicle maneuverability to only a subset of motions since the path planner selects a sequence of primitives from an offline precomputed look-up table.

The authors of [12] use an NMPC path planner based on a point-mass vehicle model, while basing the path follower on the more detailed model from [7]. The decomposition allowed for real-time implementation, but the trajectories generated by the point-mass NMPC path planner were reported to not always be feasible for the actual vehicle due to oversimplification and unmodeled dynamics (cf. [13], [11]).

An additional limiting factor to the vehicle maneuverability common to the work mentioned above is the assumption of a constant longitudinal vehicle speed. In an attempt to overcome this, the authors used a transformation of time-dependent vehicle dynamics into position-dependent vehicle dynamics for the high level path planner in [11]. Obstacle constraints can thus be modeled only depending on the free variable of the ODE system, independent of the vehicle velocity. The vehicle model includes dynamics for its position and velocity states and features a nonlinear Pacejka-type tire model [22]. This significantly improved the behavior of the controller proposed in [11] but the simplified bicycle model utilized for the planner still generated paths that could not be accurately followed by the low level controller.

In this paper we propose an auto-generated tailored NMPC algorithm for the path planning problem. It is based on the Real-Time Iteration (RTI) scheme [5] for Bock's multiple shooting method [2]. The generation of customized source code for optimization problems originates in [21], [20]. Recently this idea has been extended in [16] to the generation of fast NMPC algorithms, resulting in the open source ACADO Code Generation tool.

We extend the model used in [11] to a four-wheel vehicle dynamics model that models wheel dynamics and load transfer. Using the presented algorithmic approach we obtain faster computation and significantly faster feedback times, while increasing the accuracy of the model prediction. A simulation using the same parameters as in [11] is shown to

run in computation times of only a few milliseconds with a feedback delay of far less than a millisecond. A more computationally complex simulation, utilizing the higher-fidelity vehicle model, is shown to run well within real-time requirements. Since the ACADO Code Generation tool exports self-contained static memory C code it is particularly suited for an application on vehicle embedded hardware.

The article is structured as follows. Section II describes the vehicle model and details the spatial reformulation of the resulting optimal control problem. Section III presents the proposed algorithm for the real-time solution of the NMPC problem. Section IV shows simulation results for a passenger car in an obstacle avoidance scenario on an icy road similar to the scenario experimentally validated in [11]. Section V summarizes and concludes the paper.

II. System Model

In [11] a 6 degrees of freedom (DoF) vehicle model was used for the obstacle avoidance problem. We extend this model in the following to account for some of the unmodeled dynamics, namely wheel dynamics and load transfer. This extended vehicle model is also presented in [25], where we give slightly more details. The time-dependent dynamic system is eventually transformed into a track progress-dependent one in a similar fashion as presented in [11].

The vehicle chassis is modeled as a rigid body, described by its global position in the X-Y plane, its global orientation, and the corresponding velocities in a local x-y-z frame. The four wheels are modeled as independent bodies with only spinning inertia. Roll, pitch and heave (vertical displacement) motions of the car are neglected, but the effect of these motions on the vehicle load change, assuming a rigid suspension model, is modeled. We consider a car with front steering and rear wheel drive. The control inputs are the steering rate $\dot{\delta}$, the accelerating engine torque T^{a} and four braking torques $T^{\mathrm{b}}_{\mathrm{fl}}$, $T^{\mathrm{b}}_{\mathrm{fr}}$, $T^{\mathrm{b}}_{\mathrm{rl}}$, $T^{\mathrm{b}}_{\mathrm{rr}}$. Throughout this paper we use subscripts $\mathrm{fl}, \mathrm{fr}, \mathrm{rl}, \mathrm{rr}$ to denote quantities corresponding to the front left, front right, rear left and rear right wheel, respectively. For clarity of notation we define $\mathcal{F} := \{\mathrm{f}, \mathrm{r}\}$ and $\mathcal{S} := \{\mathrm{l}, \mathrm{r}\}$ and use $\mathcal{F} \times \mathcal{S} = \{\mathrm{fl}, \mathrm{fr}, \mathrm{rl}, \mathrm{rr}\}$.

A. Chassis Dynamics

We use an orthonormal reference frame in the vehicle's center of gravity (CoG) with the z-direction pointing upwards. The chassis dynamic equations used in this paper are

$$m\dot{v}^x = mv^y\dot{\psi} + F^x_{\mathrm{fr}} + F^x_{\mathrm{fl}} + F^x_{\mathrm{rr}} + F^x_{\mathrm{rl}} + F_D, \tag{1a}$$

$$m\dot{v}^y = -mv^x\dot{\psi} + F^y_{\mathrm{fr}} + F^y_{\mathrm{fl}} + F^y_{\mathrm{rr}} + F^y_{\mathrm{rl}}, \tag{1b}$$

$$I^z\ddot{\psi} = a(F^y_{\mathrm{fl}} + F^y_{\mathrm{fr}}) - b(F^y_{\mathrm{rl}} + F^y_{\mathrm{rr}}) + c(F^x_{\mathrm{fr}} - F^x_{\mathrm{fl}} + F^x_{\mathrm{rr}} - F^x_{\mathrm{rl}}), \tag{1c}$$

$$\dot{X} = v^x\cos\psi - v^y\sin\psi, \tag{1d}$$

$$\dot{Y} = v^x\sin\psi + v^y\cos\psi, \tag{1e}$$

where m denotes the mass and I^z the moment of inertia of the car. The distances of the tires from the vehicle's CoG

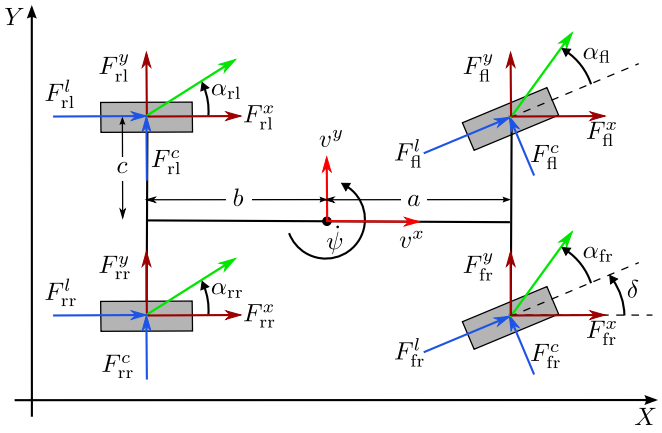

Fig. 1. Tire forces and slip angles of the 4-wheel vehicle model in inertial coordinates (cf. [25]). The tires' directions of movement are indicated by green vectors.

are characterized by a, b and c, cf. Figure 1. The vehicle's yaw angle ψ is obtained by direct integration of $\dot{\psi}$ as is the steering angle δ from input $\dot{\delta}$. The drag force due to air resistance is denoted by F_D, while $F^x_{\cdot\cdot}$ and $F^y_{\cdot\cdot}$ denote the components of the tire contact forces. These forces are computed from a combined longitudinal and lateral Pacejka-type tire slip model. For each tire $\diamond\star \in \mathcal{F} \times \mathcal{S}$, the inputs to this model are the side slip angles $\alpha_{\diamond\star}$ (cf. Figure 1), the slip ratio $\kappa_{\diamond\star} = \frac{\omega_{\diamond\star}R_{\mathrm{e}} - v_{\diamond\star}}{v_{\diamond\star}}$ representing the longitudinal slip during acceleration and deceleration, and the normal load $F^z_{\diamond\star}$. Here, $v_{\diamond\star}$ denotes the wheel speed with respect to the ground, while R_{e} denotes the tire radius. The rotational speed of the wheel denoted by $\omega_{\diamond\star}$. The influence of road friction is modeled by a parameter $\mu \in [0, 1]$ that also enters the tire model. We refer to [25] and [22] for more details on the tire model; the precise model implementation used for this paper, including all parameters, can be found at [14].

B. Wheel Dynamics

The wheels' rotational velocities $\omega_{\cdot\cdot}$ are computed from a first order model in the accelerating torques $T^{\mathrm{a}}_{\cdot\cdot}$ and the braking torques $T^{\mathrm{b}}_{\cdot\cdot}$, taking the wheel moment of inertia I^{w} into account. Note that the consideration of wheel dynamics enhances the predictive quality of the model but also renders the underlying ODE system stiff, making its solution computationally significantly more challenging. The dynamic equations are given by

$$\dot{\omega}_{\diamond\star} = \frac{1}{I^{\mathrm{w}}}(T^{\mathrm{a}}_{\diamond\star} + T^{\mathrm{b}}_{\diamond\star} - R_{\mathrm{e}}F^l_{\diamond\star}), \quad \forall\ \diamond\star \in \mathcal{F} \times \mathcal{S} \tag{2}$$

where R_{e} is taken to be the tire radius. We assume individual wheel braking and include a differential model for the total acceleration torque T^{a} (w.l.o.g. assuming a rear-wheel driven car), yielding

$$T^{\mathrm{a}}_{\mathrm{f}\star} = 0 \qquad \forall \star \in \mathcal{S},$$

$$T^{\mathrm{a}}_{\mathrm{r}\star} = T^{\mathrm{a}}\left(1 - \frac{\omega_{\mathrm{r}\star}}{\omega_{\mathrm{rl}} + \omega_{\mathrm{rr}}}\right) \qquad \forall \star \in \mathcal{S}.$$

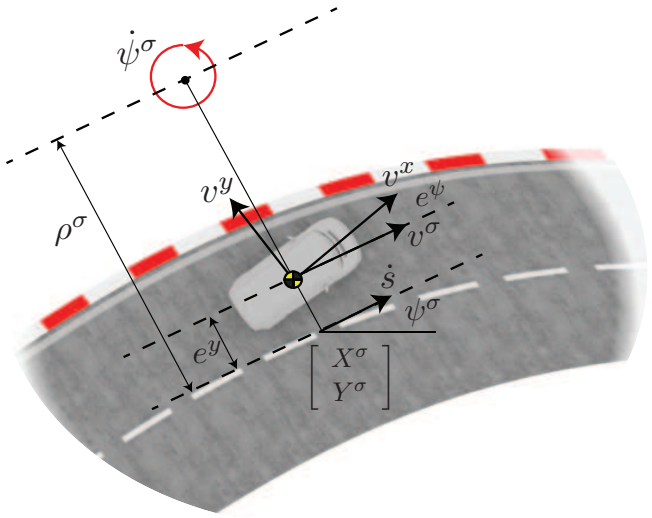

Fig. 2. The curvilinear coordinate system. The dynamics are derived about a curve defining the center-line of a track. The coordinate s defines the arc-length along the track. The relative spatial coordinates e^y and e^ψ are shown.

State	Unit	Description
v^x	$\frac{\text{m}}{\text{s}}$	Longitudinal velocity of vehicle
v^y	$\frac{\text{m}}{\text{s}}$	Lateral velocity of vehicle
$\dot{\psi}$	$\frac{\text{rad}}{\text{s}}$	Vehicle's yaw rate
e^ψ	rad	Yaw angle relative to path
e^y	m	Deviation from center-line
δ	rad	Steering angle
$\omega_{..}$	$\frac{\text{rad}}{\text{s}}$	Rotational wheel speeds
$\Delta^z_{..}$	N	Vertical load transfer

Control	Range	Unit	Description
$\dot{\delta}$	$[-1, 1]$	–	Normalized front steering rate
T_a	$[0, 1]$	–	Normalized engine torque
$T_{b_{..}}$	$[-1, 0]$	–	Normalized brake torques

TABLE I
STATES AND CONTROL INPUTS OF THE SPATIAL VEHICLE MODEL.

C. Vertical Forces

We obtain the vertical loads $F^z_{..}$ acting on each wheel as the equilibrium of the main body w.r.t. the vertical forces, and roll and pitch torques. For a height h between the vehicle's CoG and its projection onto the wheelbase, equilibrium longitudinal and lateral load transfer, $\tilde{\Delta}^z_\text{lon}$ and $\tilde{\Delta}^z_\text{lat}$, are given by

$$2\,\tilde{\Delta}^z_\text{lon} = (F^x_\text{fl} + F^x_\text{fr} + F^x_\text{rl} + F^x_\text{rr})\,\frac{h}{a+b},$$
$$2\,\tilde{\Delta}^z_\text{lat} = (F^y_\text{fl} + F^y_\text{fr} + F^y_\text{rl} + F^y_\text{rr})\,\frac{h}{2\,c}.$$

Note that this formulation of the load transfer implies an algebraic loop through the tire model, which can be relaxed by introducing first order models with time constant τ_LT for the load transfer states Δ^z_lon and Δ^z_lat,

$$\dot{\Delta}^z_\bullet = \frac{1}{\tau_\text{LT}}(\tilde{\Delta}^z_\bullet - \Delta^z_\bullet) \qquad \forall\, \bullet \in \{\text{lon}, \text{lat}\}. \tag{3}$$

Denoting the rest normal loads by $\bar{F}^z_{..}$, the vertical forces $F^z_{..}$ are then given by

$$F^z_\text{fl} = \bar{F}^z_\text{fl} - \Delta^z_\text{lon} - \Delta^z_\text{lat},$$
$$F^z_\text{fr} = \bar{F}^z_\text{fr} - \Delta^z_\text{lon} + \Delta^z_\text{lat},$$
$$F^z_\text{rl} = \bar{F}^z_\text{rl} + \Delta^z_\text{lon} - \Delta^z_\text{lat},$$
$$F^z_\text{rr} = \bar{F}^z_\text{rr} + \Delta^z_\text{lon} + \Delta^z_\text{lat}.$$

Future work aims at including suspension effects to improve predictive quality of the model. Note that a suspension model intrinsically avoids the aforementioned algebraic loop.

D. Spatial Reformulation of Dynamics

We propose a model transformation from time-dependent vehicle dynamics to track-dependent (spatial) dynamics, similar to [11], [17]. This allows a natural formulation of obstacles and general road bounds under varying vehicle speed. Note that similar ideas have been developed in area of robotics before, see, e.g., [23].

We project the X-Y coordinates on a curve σ which we take to be the center-line of a road. The ODE model is then stated w.r.t. the independent variable s, the parametrization of σ by its arc-length. The states X, Y, ψ are replaced by $e^y := \left\| [X, Y]^T - [X^\sigma, Y^\sigma]^T \right\|_2$ and $e^\psi := \psi - \psi^\sigma$, where $[X^\sigma, Y^\sigma]^T$ and ψ^σ denote the position and orientation of the current reference point on the path given by s. Figure 2 details the states in the new curvilinear coordinate system. The full system of states ξ and control inputs ν for this vehicle model are listed in Table I.

The spatial dynamics of the state vector ξ in relation to the time dependent dynamics are

$$\xi' := \frac{d\xi}{ds} = \frac{d\xi}{dt}\,\frac{dt}{ds}.$$

If $\dot{s} \neq 0$ is assumed at any time, by the inverse function theorem we have $\frac{dt}{ds} = \frac{1}{\dot{s}}$. It therefore holds

$$\xi' = \frac{1}{\dot{s}}\,\dot{\xi},$$

where $\dot{\xi}$ is defined in Equations (1), (2), and (3). For the computation of $\dot{s}$ we observe in Figure 2 that

$$v^\sigma = (\rho^\sigma - e^y)\,\dot{\psi}^\sigma \qquad \text{and}$$
$$v^\sigma = v^x\,\cos(e^\psi) - v^y\,\sin(e^\psi)$$

holds, where $\dot{\psi}^\sigma$ is the rate of change of the path orientation ψ^σ and ρ^σ is the radius of local curvature of σ. The vehicle's velocity along σ, $\dot{s} = \frac{ds}{dt}$, is then given by

$$\dot{s} = \rho^\sigma\,\dot{\psi}^s = \frac{\rho^\sigma}{\rho^\sigma - e^y}\,(v^x\,\cos(e^\psi) - v^y\,\sin(e^\psi)).$$

Note that ρ^σ only depends on the parametrization s through $\rho^\sigma(s) = \left(\frac{d^2}{ds^2}\sigma(s)\right)^{-1}$ but is independent of system inputs.

If necessary, time information may be recovered by integrating $\frac{dt}{ds}$ along σ:

$$t(s) = \int_{s_0}^{s} \frac{1}{\dot{s}(\tau)} \, \mathrm{d}\tau \, .$$

Inertial coordinates may be recovered by a transformation from the spatial coordinates to the global coordinates:

$$\begin{aligned} X &= X^\sigma - e^y \, \sin(\psi^\sigma) \\ Y &= Y^\sigma + e^y \, \cos(\psi^\sigma) \\ \psi &= \psi^\sigma + e^\psi \, . \end{aligned}$$

E. Optimal Control Problem

Using the spatial dynamics formulation, obstacle and road boundary constraints are modeled as simple bounds on the state vector. When considering a variable vehicle speed, this avoids the speed dependent (and thus implicitly input-dependent) non-convex path constraints from the generic time-dependent formulation in the inertial coordinate system. The decision in navigating to the left or right of an obstacle is assumed to be made by the obstacle recognition algorithm.

Denoting the right-hand side function of the ODE system by f and the vectors of state and control input by by ξ and ν, we yield the following optimal control problem to be solved repeatedly online for a receding horizon of finite length in space:

$$\begin{aligned} \min_{\xi(\cdot),\,\nu(\cdot)} \quad & \int_{s_0}^{s_\mathrm{f}} \|\xi(\tau) - \xi_{\text{ref}}(\tau)\|_Q^2 + \|\nu(\tau)\|_R^2 \, \mathrm{d}\tau \\ & + \|\xi(s_\mathrm{f}) - \xi_{\text{ref}}(s_\mathrm{f})\|_{P_{\mathrm{LQR}}}^2 && \text{(5a)} \\ \text{s.t.} \quad & \xi'(s) = f\left(s, \xi(s), \nu(s)\right) && \text{(5b)} \\ & e^y(s) \in [e_L^y(s), e_U^y(s)] && \text{(5c)} \\ & \nu(s) \in [-1, 1] \times [0, 1] \times [-1, 0]^4 && \text{(5d)} \\ & \xi(s_0) = \xi_0 \, , && \text{(5e)} \end{aligned}$$

where Equations (5b) through (5d) hold for all points of the current prediction horizon $s \in [s_0, s_f]$. Here, $\|\cdot\|_{\{Q,R,P_{\mathrm{LQR}}\}}$ denote the Euclidean norm with weighting matrices Q, R and P_{LQR}, respectively, while $\xi_0 \in \mathbb{R}^{12}$ is the current state measurement and $\xi_{\text{ref}} : \mathbb{R} \to \mathbb{R}^{12}$ denotes the parametric reference vector. The terminal weighting matrix P_{LQR} is obtained as the solution of the Riccati equation, computed with the chosen weighting matrices Q and R. Lower and upper road bounds, taking obstacles into account, are denoted by $e_L^y(\cdot)$ and $e_U^y(\cdot)$, respectively.

III. Numerical Solution Method

The underlying class of optimal control problems (OCP) can be generalized to the following form:

$$\begin{aligned} \min_{\xi(\cdot),\nu(\cdot)} \quad & \int_{s_0}^{s_\mathrm{f}} \|\xi(\tau) - \xi_{\text{ref}}(\tau)\|_Q^2 + \|\nu(\tau) - \nu_{\text{ref}}(\tau)\|_R^2 \, \mathrm{d}\tau \\ & + E(\xi(s_\mathrm{f})) && \text{(6a)} \\ \text{s.t.} \quad & \xi'(s) = f\left(s, \xi(s), \nu(s)\right) && \text{(6b)} \\ & \xi(s_0) = \xi_0 && \text{(6c)} \\ & \underline{\xi}(s) \le \xi(s) \le \overline{\xi}(s) \qquad \forall s \in [s_0, s_\mathrm{f}] && \text{(6d)} \\ & \underline{\nu} \le \nu(s) \le \overline{\nu} \qquad \forall s \in [s_0, s_\mathrm{f}] \, . && \text{(6e)} \end{aligned}$$

As above, the nonlinear right-hand side function f contains the model equations given w.r.t. the independent variable s, where $E(\cdot)$ is a Mayer term. The initial condition is denoted by $\xi_0 \in \mathbb{R}^n$. $\underline{\xi}, \overline{\xi} : \mathbb{R} \to \mathbb{R}^n$ and $\underline{\nu}, \overline{\nu} \in \mathbb{R}^m$ denote bounds for the n-dimensional state vector and the m-dimensional control vector. The least-squares objective function aims at tracking a state reference trajectory $\xi_{ref} : \mathbb{R} \to \mathbb{R}^n$, taking a control regularization term into account. Weighting matrices are given by $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$.

A. The Direct Multiple Shooting Method

OCPs of this kind can efficiently be treated using Bock's multiple shooting method [2], which transforms the optimal control problem into a finite dimensional optimization problem. The space of feasible control inputs is reduced to a finite-dimensional one by using basis functions with local support (see [2], [19]). State and control bounds are relaxed to a finite time grid. The resulting least-squares nonlinear programming (NLP) problem is solved by a tailored generalized Gauss-Newton method. This includes an extensive exploitation of the arising structures. In particular a condensing technique is applied for a problem size reduction of the quadratic programming problems (QP), cf. [19].

B. The Real-Time Iteration Scheme for Nonlinear MPC

For the receding horizon nonlinear MPC framework OCP (6) needs to be solved in each iteration for the current state measurement $\xi_0 \in \mathbb{R}^n$, shifting state bounds and reference functions by the corresponding s-progress of the system, Δs.

This repeated solution of the OCP can be performed efficiently by the real-time iteration (RTI) scheme which builds on the direct multiple shooting method and has originally been proposed in [5]. In contrast to a regular SQP scheme, the RTI scheme performs only one iteration per sampling time. Initializations are taken from the previous sampling time and shifted, if appropriate. Particularly, a large share of expensive operations like sensitivity generation and matrix-condensing can be performed in a so called *preparation phase* prior to observing the current state measurement ξ_0. The *feedback phase* between observing ξ_0 and providing process feedback $\nu(s_0)$ then reduces to a single solution of the parametric reduced-size QP (see [5] for details). Due to the possibility of active-set changes in the repeatedly solved QP, performance is guaranteed to be not worse than that of a linear least-squares tracking controller, see [5]. Contractivity as well as nominal stability of the RTI scheme have been shown [4], [6].

C. Code Generation

In order to provide sufficiently fast feedback for the real-time implementation of the proposed MPC problem, we make use of automatic source code generation, which recently became popular for convex optimization problems [20]. These ideas have been extended to nonlinear dynamic optimization problems in form of the ACADO Code Generation tool [1], [16]. This open-source software exports a tailored RTI algorithm that contains only

the absolutely essential algorithmic components. Problem-specific structure such as fixed dimensions or sparsity patterns is exploited during the code generation process to avoid irrelevant or redundant computations.

The ACADO Code Generation tool makes use of symbolic differentiation for generating plain C-code for all function and derivative evaluations. All problem dimensions are hard coded allowing static memory allocation. Moreover, inner loops of linear algebra operations are partially unrolled for increased performance without a significant increase in memory consumption. We make use of a tailored constant step-size Gauss-Legendre integration method of order 2 (as introduced in [24]) for the solution of the ODE system and its associated variational differential equations. For solving the condensed QP problem, an adapted variant of the online QP solver qpOASES [8], [9] is used. Unlike its original implementation, the adapted variant has been implemented in plain C and also avoids any dynamic memory allocation for an easy deployment on embedded hardware.

More details on the ACADO Code Generation tool can be found on the ACADO Toolkit Homepage [1].

IV. Simulation Results

The performance of the exported NMPC algorithm is demonstrated in a scenario similar to the experimental setup considered in [11]. Obstacles of each $6\,\mathrm{m}$ length are positioned at $s = 43\,\mathrm{m}$ and $s = 123\,\mathrm{m}$ (cf. Figure 3) on a $200\,\mathrm{m}$ long straight track with a slippery (e.g. snow-covered or icy) surface of friction coefficient $\mu = 0.3$. The first obstacle has a width of $2\,\mathrm{m}$ and needs to be avoided on the left, while the second obstacle of width $0.8\,\mathrm{m}$ needs to be avoided on the right. The parameters used for the vehicle model have been chosen to match those of a Jaguar X-type passenger car. It has a mass of $2050\,\mathrm{kg}$ and a moment of inertia of $3344\,\mathrm{kg/m^2}$. More details on the vehicle model parameters can be found in [11], [14].

The vehicle is traveling at $10\,\mathrm{m/s}$ tracking the initial speed and the road reference while avoiding the obstacles. A prediction horizon of 20 intervals with a total length $20\,\mathrm{m}$ is used. We chose diagonal weighting matrices $Q = \mathrm{diag}[1, 1, 10, 10, 10, 10^{-8}, 10^{-8}, 0.1, 0.1, 0.1, 0.1, 1, 1]$ and $R = \mathrm{diag}[10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 1]$. Integration of the dynamic system and the corresponding variational differential equations is based on a grid of 40 integrator steps. We assume full state observation, as well as knowledge of the road friction coefficient μ in this paper. For a corresponding moving horizon estimation (MHE) based observation scheme we refer to [25]. Computational results were obtained on a standard PC featuring an Intel i7 mobile CPU at 2.7GHz under Ubuntu 12.04. The generated C code has been compiled in a MEX function and all simulations have been run in Matlab R2011b.

Figure 3 shows the obtained results for the considered scenario. Dashed vertical lines indicate when the obstacle becomes visible to the controller. The proposed control scheme avoids both obstacles and regains its initial speed after passing the second obstacle.

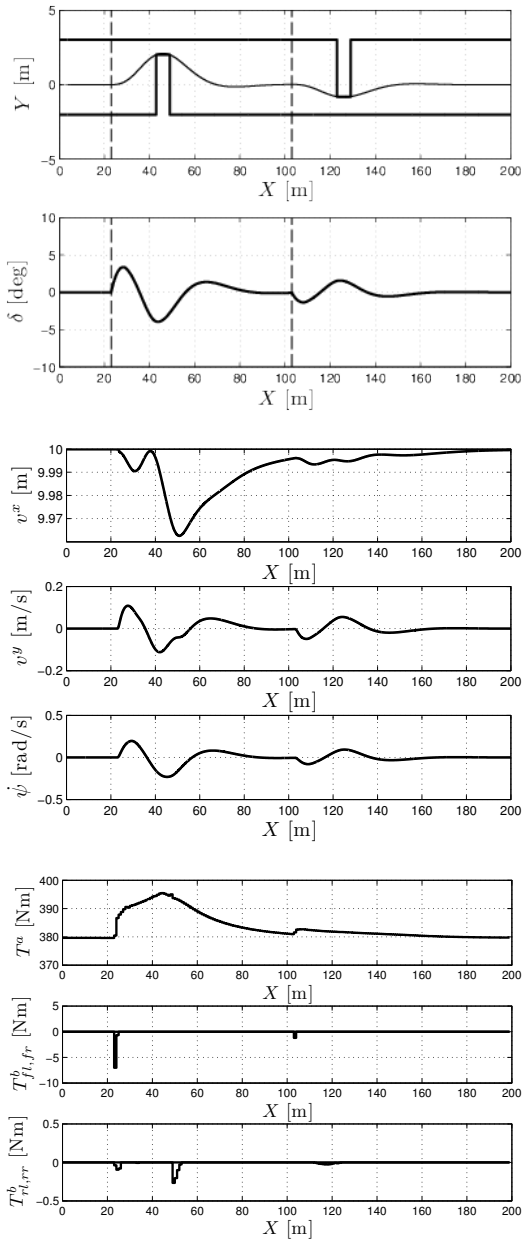

Fig. 3. Simulation results for an obstacle avoidance scenario using the model proposed in Section II.

Table II shows average and worst-case computation times from this scenario. We denote the model presented in Section II by $\mathcal{M}_{12}$. Note that the obtained worst-case computation times are still real-time feasible for a $50\,\mathrm{ms}$ actuation system as it was used in [11], while the feedback delay is even one order of magnitude faster. For comparison, we also include average computation times obtained using the model used for path planning in [11], here denoted by $\mathcal{M}_6$. Using the proposed NMPC scheme these are significantly faster as the model is non-stiff. We also include computation times using the path planner in the setup from [11] to give the reader a rough idea of the potential performance gain by using tailored solution algorithms, even though an

	Feedback time	Full Iteration
Setup [11] + $\mathcal{M}_6$ average	156.9 ms	156.9 ms
ACADO + $\mathcal{M}_6$ average	0.06 ms	5.03 ms
ACADO + $\mathcal{M}_{12}$ average	3.1 ms	27.1 ms
ACADO + $\mathcal{M}_{12}$ maximum	6.5 ms	33.4 ms

TABLE II
COMPUTATION TIMES OF ONE MPC ITERATION.

exact comparison is virtually impossible, as the proposed approaches differ in central algorithmic components like the used integration method and the number of SQP iterations performed per MPC step.

As the proposed control scheme is tracking the center-line reference, it will try to circumnavigate obstacles as tightly as possible. In the presence of perturbations, hard obstacle bounds might therefore need to be relaxed using a slack variable reformulation to avoid infeasible QP subproblems.

V. CONCLUSIONS

In this paper, we presented an approach for autonomous vehicle guidance using auto-generated NMPC algorithms tailored for the specific model dynamics. Even for a detailed vehicle model including wheel dynamics, a state-of-the-art tire model, and load transfer computation times in the range of milliseconds and even significantly lower feedback times were achieved for the obstacle avoidance problem, thus advancing precise autonomous guidance of vehicles in extreme conditions. For an efficient treatment of obstacles even in flexible vehicle speed models we proposed a spatial transformation of the dynamic system, which maintains the numerically favorable nature of the problem as a steady-state reference tracking problem.

Ongoing research includes the addition of a suspension model to the system dynamics and the utilization of Huber-type tracking penalties (see [15]) for a smooth controller performance also for large obstacles. Experimental validation of the proposed algorithmic scheme is envisaged for the future.

ACKNOWLEDGMENT

The authors thank Yiqi Gao for his active support in the research that lead to this paper as well as Florian Kehrle and Christian Kirches for fruitful discussions on the spatial model transformation.

REFERENCES

[1] ACADO Toolkit. http://www.acadotoolkit.org, 2009–2013.

[2] H.G. Bock and K.J. Plitt. A Multiple Shooting algorithm for direct solution of optimal control problems. In *Proc. 9th IFAC World Congress*, pages 242–247, 1984. Available at http://www.iwr.uni-heidelberg.de/groups/agbock/FILES/Bock1984.pdf.

[3] F. Borrelli, A. Bemporad, M. Fodor, and D. Hrovat. An MPC/hybrid system approach to traction control. *IEEE Trans. Control Systems Technology*, 14(3):541–552, May 2006.

[4] M. Diehl, H.G. Bock, and J.P. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5):1714–1736, 2005.

[5] M. Diehl, H.G. Bock, J.P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *J. Proc. Contr.*, 12(4):577–585, 2002.

[6] M. Diehl, R. Findeisen, and F. Allgöwer. A stabilizing real-time implementation of nonlinear model predictive control. In L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, editors, *Real-Time and Online PDE-Constrained Optimization*. SIAM, 2006.

[7] P. Falcone, F. Borrelli, J. Asgari, H.E. Tseng, and D. Hrovat. Low complexity MPC schemes for integrated vehicle dynamics control problems. *9^{th} International Symposium on Advanced Vehicle Control*, 2008.

[8] H.J. Ferreau, H.G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, 2008.

[9] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 2013. (under review).

[10] H.J. Ferreau, P. Ortner, P. Langthaler, L. del Re, and M. Diehl. Predictive control of a real-world diesel engine using an extended online active set strategy. *Annual Reviews in Control*, 31(2):293–301, 2007.

[11] Y. Gao, A. Gray, J.V. Frasch, T. Lin, E. Tseng, J.K. Hedrick, and F. Borrelli. Spatial predictive control for agile semi-autonomous ground vehicles. In *Proceedings of the 11th International Symposium on Advanced Vehicle Control*, 2012.

[12] Y. Gao, T. Lin, F. Borrelli, E. Tseng, and D. Hrovat. Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads. *Dynamic Systems and Control Conference, 2010*, 2010.

[13] A. Gray, Y. Gao, T. Lin, J.K. Hedrick, E. Tseng, and F. Borrelli. Predictive control for agile semi-autonomous ground vehicles using motion primitves. *Proceedings American Control Conference*, 2012.

[14] A. Gray, M. Zanon, and J. Frasch. Parameters for a Jaguar X-Type. http://www.mathopt.de/RESEARCH/obstacleAvoidance.php, 2012.

[15] S. Gros and M. Diehl. NMPC based on Huber Penalty Functions to Handle Large Deviations of Quadrature States. In *Proc. American Control Conference*, 2013.

[16] B. Houska, H.J. Ferreau, and M. Diehl. An Auto-Generated Real-Time Iteration Algorithm for Nonlinear MPC in the Microsecond Range. *Automatica*, 47(10):2279–2285, 2011.

[17] F. Kehrle, Frasch J.V., C. Kirches, and S. Sager. Optimal control of formula 1 race cars in a VDrift based virtual environment. In S. Bittanti, A. Cenedese, and S. Zampieri, editors, *Proceedings of the 18th IFAC World Congress*, pages 11907–11912, 2011.

[18] T. Keviczky and G.J. Balas. Flight Test of a Receding Horizon Controller for Autonomous UAV Guidance. In *Proceedings of the American Control Conference 2005*, pages 3518–3523, 2005.

[19] D.B. Leineweber, I. Bauer, A.A.S. Schäfer, H.G. Bock, and J.P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization (Parts I and II). *Computers and Chemical Engineering*, 27:157–174, 2003.

[20] J. Mattingley and S. Boyd. Real-time convex optimization in signal processing. *IEEE Signal Processing Magazine*, 27(3):50–62, 2010.

[21] T. Ohtsuka and A. Kodama. Automatic code generation system for nonlinear receding horizon control. *Transactions of the Society of Instrument and Control Engineers*, 38(7):617–623, 2002.

[22] H.B. Pacejka. *Tyre and Vehicle Dynamics*. Elsevier Ltd., Oxford Burlington, 2nd edition, 2006.

[23] F. Pfeiffer and R. Johanni. A concept for manipulator trajectory planning. *IEEE Journal Robotics & Automation*, 3(2):115 –123, 1987.

[24] R. Quirynen, M. Vukov, and M. Diehl. Auto generation of implicit integrators for embedded nmpc with microsecond sampling times. In M. Lazar and F. Allgöwer, editors, *Proceedings of the 4th IFAC Nonlinear Model Predictive Control Conference*, 2012.

[25] M. Zanon, J.V. Frasch, and M. Diehl. Nonlinear Moving Horizon Estimation for Combined State and Friction Coefficient Estimation in Autonomous Driving. In *Proceedings of the European Control Conference*, 2013.

[26] V.M. Zavala, C.D. Laird, and L.T. Biegler. Fast implementations and rigorous models: Can both be accommodated in NMPC? *International Journal of Robust and Nonlinear Control*, 18(8):800–815, 2008.