Efficient Real-Time Obstacle Avoidance Using Multi-Objective Nonlinear Model Predictive Control and Semi-Smooth Newton Method

Mostafa Emam[a], Thomas Rottmann and Matthias Gerdts[b]

Department of Aerospace Engineering, Institute of Applied Mathematics and Scientific Computing, University of the Bundeswehr Munich, 85579, Neubiberg, Germany

Keywords: Automated Driving, Obstacle Avoidance, Path Planning, Mutli-Objective NMPC, Semi-Smooth Newton Method.

Abstract: This work discusses the theory and methodology of applying Nonlinear Model Predictive Control (NMPC) in an efficient manner to achieve real-time path planning and obstacle avoidance for autonomous vehicles. First, we explain the optimization problem formulation and the numerical solution approach using a semi-smooth Newton method adapted for nonlinear problems. Then, an MPC path planning problem is described in terms of the vehicle model, the controller design, and the mathematical representation of obstacles as proper system constraints. Afterwards, the developed controller is numerically evaluated for different vehicle models in a simulated environment to dynamically assess its flexibility and real-time performance, which serves as a prerequisite to deferred real-life testing.

1 INTRODUCTION

Optimal Control has evolved as the natural successor to the Calculus of Variations to address optimization problems with path (state/control) constraints, contrary to the classic unconstrained problems proposed by early mathematicians and philosophers (Sussmann and Willems, 1997). The incorporation of these inequality constraints added another layer of complexity that was first solved in the late 1950s with the introduction of Pontryagin's Maximum Principle (PMP), yet the developed methodologies can be traced back to the work of earlier renowned mathematicians like Euler, Lagrange, and Legendre (Chachuat, 2007). A particular approach to handle constrained optimization problems is Model Predictive Control (MPC), also known as Receding Horizon Control (RHC), which was initially developed to tackle problems in the oil and gas industry in the 1980s (Morari and H. Lee, 1999), but has progressed since then to represent an effective and flexible framework that can be employed to optimally solve multivariable Optimal Control Problems (OCPs) with mixed state-control constraints (Grüne and Pannek, 2011).

Owing to its versatility, MPC has gained a noticeable popularity in autonomous driving applications, especially in path planning, trajectory following and multi-objective vehicle control (Vu et al., 2021; Qin et al., 2023; Musa et al., 2021; Emam and Gerdts, 2022). However, MPC comes with the caveat that the determined solution is only optimal for the exact problem parameters, e.g., the initial states, thus model deviations and system uncertainties will quickly invalidate the solution optimality. In other words, only a subset of the calculated control trajectory u^* can be applied before the need arises to re-solve the OCP to yield an optimal u^* (Rawlings and Mayne, 2009). In traditional MPC, only the first control sequence is applied and the OCP is re-solved every time step, which represents a computational bottleneck for real-time applications and is the main drive for research effort towards efficient solution techniques (Diehl et al., 2005).

In this work, we tackle the MPC real-time restriction by adapting a previously implemented efficient semi-smooth Newton method, originally designed for linear systems (Emam and Gerdts, 2023), so that it handles nonlinear Discrete Optimal Control Problems (DOCPs). We validate our approach by developing a generic controller with mixed state-control constraints for autonomous path planning, and propose a mathematically efficient method to incorporate obstacle avoidance using smooth, continuously differentiable envelopes, which enables faster convergence of the

[a] https://orcid.org/0000-0003-4942-1183
[b] https://orcid.org/0000-0001-8674-5764

Emam, M., Rottmann, T. and Gerdts, M.
Efficient Real-Time Obstacle Avoidance Using Multi-Objective Nonlinear Model Predictive Control and Semi-Smooth Newton Method.
DOI: 10.5220/0012738400003702
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 10th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2024)*, pages 178-189
ISBN: 978-989-758-703-0; ISSN: 2184-495X
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

problem solution. The controller is implemented as part of an integrated virtual testing environment and is evaluated for different vehicle models to ensure its flexibility and effectiveness. Thorough explanation and numerical results are discussed in the sequel.

2 CONTROL PROBLEM AND NUMERICAL SOLUTION

The basic MPC scheme entails solving instances of a parametric DOCP with specific objectives, system dynamics, and path constraints and, provided that a solution exists, employs this solution to derive the optimal feedback control law (Morari and H. Lee, 1999). This incorporates some implicit assumptions, e.g., the instantaneous computation of the DOCP's numerical solution and the absence of delays when applying the control inputs (Grüne and Pannek, 2011). Since this does not adequately reflect real applications, some adaptations are required before implementing the MPC approach, such as prediction, re-optimization, and sensitivity updates. For a comprehensive overview on formulating, discretizing, and solving OCPs, the reader is directed to (Gerdts, 2023), and for specific MPC considerations to (Palma, 2015). With brevity in mind, we focus in this section on the basic MPC scheme and begin discussing the strategy for calculating the numerical solution of the DOCP.

2.1 Newton Method for DOCPs

Newton's algorithm is a fundamental approach to reach an existing solution $z^* \in \mathbb{R}^{n_z}$ that satisfies the nonlinear and continuously differentiable function $\phi(z) = 0$ with $\phi : \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$. It starts with an initial estimate $\tilde{z}$ and if $\phi(\tilde{z}) \neq 0$, computes a direction d to update $\tilde{z}$ using a linear approximation of ϕ around $\tilde{z}$ from the first two terms in its Taylor series expansion. d is the solution of the linear system:

$$\begin{aligned} \nabla\phi(\tilde{z}^{(\ell)})d^{(\ell)} &= -\phi(\tilde{z}^{(\ell)}), \\ \tilde{z}^{(\ell+1)} &= \tilde{z}^{(\ell)} + d^{(\ell)}, \quad \ell = 0,1,2,\dots \end{aligned} \tag{1}$$

where $\nabla\phi(\tilde{z})$ is the Jacobian of ϕ around $\tilde{z}$, $\tilde{z}^{(\ell+1)}$ is the new estimate, and ℓ is the iteration index. Provided that $\nabla\phi(\tilde{z}^{(\ell)})$ is nonsingular, the process repeats and iteratively approaches $\phi(\tilde{z}^{(\ell)}) \approx 0$, such that $\tilde{z}^{(\ell)} \approx z^*$ (within a specified error margin) (Chachuat, 2007; Gerdts, 2023). To employ this method, we first need to formulate the DOCP with its system dynamics, objective function, constraints, and boundary conditions, as a differentiable function, which we will now discuss.

Consider the implicit structured nonlinear OCP in discrete time $k \in \mathbb{N}_0$ with general boundary conditions:

Minimize

$$J(z,p) := \sum_{k=0}^{N} F_0(k, z_k, p) \tag{2}$$

w.r.t. $z = (z_0, \dots, z_N)^\top \in \mathbb{R}^{(N+1)n_z}$, $p \in \mathbb{R}^{n_p}$, *and s.t. the constraints*

$$\begin{aligned} F(k, z_{k+1}, z_k, p) &= \ 0, \quad k = 0,\dots,N-1, && (3)\\ G(k, z_k, p) &\leq \ 0, \quad k = 0,\dots,N, && (4)\\ \Psi(z_0, z_N, p) &= \ 0. && (5) \end{aligned}$$

Herein, $z_k \in \mathbb{R}^{n_x \times n_u}$ denotes the state and control vector at time $k \in \mathbb{N}_0$, $p \in \mathbb{R}^{n_x}$ is the initial state vector. (3) represents the nonlinear system dynamics, (4) are the mixed state and control constraints, and (5) denote the general boundary conditions. Linearization of the constraints at some $\hat{z}$ and $\hat{p}$ yields:

$$\begin{aligned} G_k z_{k+1} + F_k z_k + P_k p &= -F[k], \ \ k = 0,\dots,N-1, && (6)\\ E_k z_k + D_k p &\leq -G[k], \ \ k = 0,\dots,N, && (7)\\ \Psi_0 z_0 + \Psi_N z_N + P_N p &= -\Psi(\hat{z_0}, \hat{z_N}, \hat{p}) && (8) \end{aligned}$$

where $F[k]$ indicates the evaluation of F at $k, \hat{z}, \hat{p}$ and likewise for other occurrences. Alternatively, the problem can be formulated with separated boundary conditions using:

$$\Psi(z_0, z_N, p) = \begin{pmatrix} \Psi_0(z_0, p) \\ \Psi_N(z_N, p) \end{pmatrix} = 0 \tag{9}$$

and linearization at $\hat{z}$ and $\hat{p}$ yields:

$$\begin{pmatrix} \Psi_0 \\ 0 \end{pmatrix} z_0 + \begin{pmatrix} 0 \\ \Psi_N \end{pmatrix} z_N + \begin{pmatrix} P^0 \\ P^N \end{pmatrix} p = -\begin{pmatrix} \Psi_0(\hat{z_0}, \hat{p}) \\ \Psi_N(\hat{z_N}, \hat{p}) \end{pmatrix} \tag{10}$$

Herein

$$\begin{gathered} G_k = F'_{z_{k+1}}[k], \quad F_k = F'_{z_k}[k], \quad P_k = F'_p[k], \\ E_k = G'_z[k], \quad D_k = G'_p[k], \\ \Psi_0 = \Psi'_{z_0}(\hat{z_0}, \hat{z_N}, \hat{p}), \quad \Psi_N = \Psi'_{z_N}(\hat{z_0}, \hat{z_N}, \hat{p}), \\ P_N = \Psi'_p(\hat{z_0}, \hat{z_N}, \hat{p}) \end{gathered}$$

and respectively for separated boundary conditions:

$$\begin{gathered} \Psi_0 = \Psi'_{0,z}(\hat{z_0}, \hat{p}), \quad P^0 = \Psi'_{0,p}(\hat{z_0}, \hat{p}) \\ \Psi_N = \Psi'_{N,z}(\hat{z_N}, \hat{p}), \quad P_N = \Psi'_{N,p}(\hat{z_N}, \hat{p}) \end{gathered}$$

Let $\lambda = (\lambda_0^\top, \dots, \lambda_{N-1}^\top)^\top$, $\mu = (\mu_0^\top, \dots, \mu_N^\top)^\top$, and σ denote the vectors of Lagrange multipliers for the constraints (3), (4), and (5) respectively, such that

$\sigma = (\sigma_0, \sigma_N)^\top$ for the separated boundary conditions. The DOCP Lagrange function is defined as:

$$\begin{aligned} L(z,p,\lambda,\mu,\sigma) :=& \kappa(z_0,z_N,p,\mu_N,\sigma) \\ &+ \sum_{k=0}^{N-1} H(k,z_{k+1},z_k,p,\lambda_k,\mu_k) \end{aligned} \tag{11}$$

with:

$$\begin{aligned} \kappa(z_0,z_N,p,\mu_N,\sigma) = F_0(N,z_N,p) + \mu_N^\top G(N,z_n,p) \\ + \sigma^\top \Psi(z_0,z_N,p) \end{aligned}$$

and the Hamilton function is defined as:

$$\begin{aligned} H(k,z_{k+1},z_k,p,\lambda_k,\mu_k) :=& F_0(k,z_k,p) \\ &+ \lambda_k^\top F(k,z_{k+1},z_k,p) \\ &+ \mu_k^\top G(k,z_k,p) \end{aligned} \tag{12}$$

Provided that some constraint qualification is satisfied and given a local minimum z^*, there exists Lagrangian multipliers that satisfy the upcoming first order necessary Karush-Kuhn-Tucker (KKT) conditions (Chachuat, 2007; Gerdts and Kunkel, 2008). These conditions can be solved by utilizing a semi-smooth Newton method, and we highlight that they shall be necessary and sufficient for a global minimum of the DOCP, on condition that the matrix H_k is positive semi-definite and symmetric $\forall k \in \mathbb{N}_0$ (Gerdts, 2023). The KKT conditions read:

$$\begin{aligned} 0 &= \nabla_{z_0} L = \nabla_{z_0}\kappa + \nabla_{z_k} H[0] \\ 0 &= \nabla_{z_k} L = \nabla_{z_k} H[k] + \nabla_{z_{k+1}} H[k-1] \\ 0 &= \nabla_{z_N} L = \nabla_{z_N}\kappa + \nabla_{z_{k+1}} H[N-1] \\ 0 &= \nabla_p L = \nabla_p \kappa + \sum_{k=0}^{N-1} \nabla_p H[k] \end{aligned}$$

with the second derivatives (for $k = 1,\dots,N-1$):

$$\begin{aligned} H_0 &= \nabla^2_{z_0,z_0} L = \nabla^2_{z_0,z_0}\kappa + \nabla^2_{z_k,z_k} H[0] \\ H_k &= \nabla^2_{z_k,z_k} L = \nabla^2_{z_k,z_k} H[k] + \nabla^2_{z_{k+1},z_{k+1}} H[k-1] \\ H_N &= \nabla^2_{z_N,z_N} L = \nabla^2_{z_N,z_N}\kappa + \nabla^2_{z_{k+1},z_{k+1}} H[N-1] \\ Q &= \nabla^2_{p,p} L = \nabla^2_{p,p}\kappa + \sum_{k=0}^{N-1} \nabla_{p,p} H[k] \\ Q_0^\top &= \nabla^2_{p,z_0} L = \nabla^2_{p,z_0}\kappa + \nabla^2_{p,z_k} H[0] \\ Q_k^\top &= \nabla^2_{p,z_k} L = \nabla^2_{p,z_k} H[k] + \nabla^2_{p,z_{k+1}} H[k-1] \\ Q_N^\top &= \nabla^2_{p,z_N} L = \nabla^2_{p,z_N}\kappa + \nabla^2_{p,z_{k+1}} H[N-1] \\ \Xi_{0,N} &= \Xi_{N,0}^\top = \nabla^2_{z_0,z_N} L = \nabla^2_{z_0,z_N}\kappa \\ J_k &= \nabla^2_{z_k,z_{k+1}} L = \nabla^2_{z_k,z_{k+1}} H[k] \\ J_{k-1}^\top &= \nabla^2_{z_k,z_{k-1}} L = \nabla^2_{z_{k+1},z_k} H[k-1] \end{aligned}$$

To incorporate the status of inequality constraints (active/inactive), we additionally enforce the complementarity conditions:

$$0 \le \mu_k \perp -G(k,z_k,p) \ge 0, \quad k = 0,\dots,N$$

by utilizing the convex and Lipschitz continuous Fischer-Burmeister function $\varphi : \mathcal{R}^2 \to \mathcal{R}$ (Jiang, 1999; Jiang and Qi, 1997) defined by (Fischer, 1992):

$$\varphi(a,b) := \sqrt{a^2+b^2} - a - b,$$

which has the appealing property that $\varphi(a,b) = 0$ holds either when $a,b \ge 0$ or $ab = 0$ (Gerdts and Kunkel, 2009). Hence, the complementarity conditions can be equivalently reformulated as the non-smooth equation:

$$\varphi(-G(k,z_k,p),\mu_k) = 0, \quad k = 0,\dots,N$$

and since $\partial\varphi$ does not exist at the origin, we instead utilize Clarke's generalized Jacobian of φ to determine its value (Gerdts and Kunkel, 2008).

Combining the necessary and sufficient conditions with the constraints produces the nonlinear equation:

$$\begin{gathered} \mathcal{F}(\eta) = 0, \\ \eta = (\sigma^\top, z_0^\top, \mu_0^\top, \lambda_0^\top, \dots, z_{N-1}^\top, \mu_{N-1}^\top, \lambda_{N-1}^\top, z_N^\top, \mu_N^\top)^\top \end{gathered} \tag{13}$$

with:

$$\mathcal{F}(\eta) = \begin{pmatrix} \Psi(z_0,z_N,p) \\ \beta_0(\eta) \\ \vdots \\ \beta_N(\eta) \end{pmatrix},$$

where for $k = 0,\dots,N-1$:

$$\begin{aligned} \beta_k &= \begin{pmatrix} \nabla_{z_k} L(z^{(\ell)}, p^{(\ell)}, \lambda^{(\ell)}, \mu^{(\ell)}, \sigma^{(\ell)}) \\ \varphi(-G(k, z_k^{(\ell)}, p^{(\ell)}), \mu_k^{(\ell)}) \\ F(k, z_{k+1}^{(\ell)}, z_k^{(\ell)}, p^{(\ell)}) \end{pmatrix}, \\ \beta_N &= \begin{pmatrix} \nabla_{z_N} L(z^{(\ell)}, p^{(\ell)}, \lambda^{(\ell)}, \mu^{(\ell)}, \sigma^{(\ell)}) \\ \varphi(-G(N, z_N^{(\ell)}, p^{(\ell)}), \mu_N^{(\ell)}) \end{pmatrix}. \end{aligned}$$

Applying a semi-smooth Newton method to this nonlinear and non-smooth equation (13) results in the iteration:

$$\begin{aligned} V_\ell d^{(\ell)} &= -\mathcal{F}(\eta^{(\ell)}), \\ \eta^{(\ell+1)} &= \eta^{(\ell)} + d^{(\ell)}, \quad \ell = 0,1,2,\dots \end{aligned} \tag{14}$$

which follows (1), where $V_\ell \in \partial\mathcal{F}(\eta^{(\ell)})$ and $\partial\mathcal{F}(\eta^{(\ell)})$ is Clarke's generalized Jacobian of $\mathcal{F}$ (Clarke, 1990). For simplification, we assume no mixed derivatives exist, i.e., $\nabla^2_{z_k,z_{k+1}} H[k] = \nabla^2_{z_{k+1},z_k} H[k] = 0$, and $\Xi_{0,N} = \Xi_{N,0} = 0$. This presents us with the following linear Newton system (14) that comprises a sparse, block-banded matrix structure:

$$[!th] \begin{pmatrix} & \Omega_0 & & & \\ \Omega_0^\top & \Gamma_0 & \Omega_1 & & \\ & \Omega_1^\top & \Gamma_1 & \ddots & \\ & & \ddots & \ddots & \Omega_N \\ & & & \Omega_N^\top & \Gamma_N \end{pmatrix} \begin{pmatrix} d_\sigma \\ d_0 \\ d_1 \\ \vdots \\ d_N \end{pmatrix} = - \begin{pmatrix} \Psi(z_0^{(\ell)}, z_N^{(\ell)}, p^{(\ell)}) \\ \beta_0 \\ \beta_1 \\ \vdots \\ \beta_N \end{pmatrix}$$

that can be exploited and efficiently factorized using, for example, the LAPACK routines for LU decomposition (Britzelmeier and Gerdts, 2020). In addition, it has been long established that structural exploitation can significantly increase the efficiency and reliability of solving sparse banded matrices by orders of magnitude (Morari and H. Lee, 1999; Gerdts, 2023).

For $k = 0, \ldots, N-1$, we have:

$$d_k = (d_{z_k}^\top, d_{\mu_k}^\top, d_{\lambda_k}^\top)^\top, \quad d_N = (d_{z_N}^\top, d_{\mu_N}^\top)^\top$$

$$\Gamma_k = \begin{pmatrix} H_k & E_k^\top & F_k^\top \\ -S_k E_K & T_k & \\ F_k & & \end{pmatrix}, \qquad \Gamma_N = \begin{pmatrix} H_N & E_N^\top \\ -S_N E_N & T_N \end{pmatrix},$$

where:

$$(S_k, T_k) \in \partial\varphi(-G(k, z_k^{(\ell)}, p^{(\ell)}), \mu_k^{(\ell)}),$$

and for $k = 1, \ldots, N-1$, we have:

$$\Omega_0 = \begin{pmatrix} \Psi & 0 & 0 \end{pmatrix}, \quad \Omega_k = \begin{pmatrix} J_k & 0 & 0 \\ 0 & 0 & 0 \\ G_k & 0 & 0 \end{pmatrix}, \qquad \Omega_N = \begin{pmatrix} J_N & 0 \\ 0 & 0 \\ G_N & 0 \end{pmatrix}.$$

Under certain conditions, the semi-smooth Newton method offers rapid local convergence (superlinear (Gerdts and Kunkel, 2008) or even q-quadratic (Jiang and Qi, 1997)). It can be naturally globalized without any hybrid strategy, e.g., using Armijo-type line-search (Jiang and Ralph, 1998; Jiang, 1999), and regularization strategies can be employed to ensure the existence of solutions (Gerdts and Kunkel, 2009). Moreover, it can be extended to infinite spaces, cf. (Ulbrich, 2002; Chen et al., 2000).

In summary, a DOCP solver based on the semi-smooth Newton method, introduced in (Emam and Gerdts, 2023) and adapted here, offers a fast and robust approach for numerically solving nonlinear DOCPs with boundary conditions and mixed control-state constraints. It can be implemented in any programming language of choice, and is computationally efficient as long as the foregoing structural exploitation is properly applied.

3 MPC FORMULATION

To evaluate the developed DOCP solver, we consider a typical problem in Automated Driving (AD) applications, i.e., path planning and obstacle avoidance. We achieve this by first introducing a system controller based on MPC, then proposing an efficient mathematical formulation of the obstacles, which guarantees safe driving while offering fast solution convergence.

3.1 Path Planning Controller

Given that the effectiveness of the MPC strategy is directly linked to the system representation, it is essential to adequately model the dynamical behavior of the ego-vehicle. Therefore, selecting a realistic motion model is imperative to produce smooth and feasible travel paths. Moreover, selecting a relatively simple model yields more compact matrices in (14) and simplifies calculating the Jacobian and Hessian matrices, which improves the approach's efficiency. In other words, the choice of the dynamical model is a compromise between a realistic (but slow), and an approximate (yet fast) system behavior (Grüne and Pannek, 2011). Herein, we use a linearized version of the point-mass kinematic vehicle model in Frenet frame (Emam and Gerdts, 2023).

$$\begin{aligned} s'(t) &= v(t) \\ d'(t) &= v(t)\chi(t) \\ \chi'(t) &= v(t)(\kappa(t) - \kappa_r(s(t))) \\ \kappa'(t) &= u_1(t) \\ v'(t) &= u_2(t) \end{aligned} \tag{15}$$

and we ensure the feasibility of the generated travel paths by introducing operating constraints. This model describes the movement of a given point ρ_{ego} (e.g., the middle point of the ego-vehicle's rear axle) across a reference curve $\gamma_r : [0, L] \to \mathbb{R}^2$. The model has the state vector $x = (s, d, \chi, \kappa, v)^T$, where s is the arclength of the projection of ρ_{ego} unto γ_r and d is its lateral offset. χ is the difference between the ego-vehicle's and the reference curve's heading at ρ_{ego}, κ is the curvature of the ego-vehicle, and v is its velocity. The curvature of a parameterized curve w.r.t. its arclength $\gamma_r(s)$ is defined as:

$$\kappa_r(s) = x'(s)y''(s) - x''(s)y'(s)$$

while κ may be determined from the ego-vehicle's yaw rate $\varphi'(t)$ with $\kappa(t) = \frac{\varphi'(t)}{v(t)}$, or, in case of (15), directly from the system input u_1. The model has the generic inputs $u = (u_1, u_2)^T$, which are the derivative of the curvature and the acceleration, that can be mapped to the controls of different vehicle models through a proper mapping $(x,u,X) \mapsto U = \mu(x,u,X)$ (Britzelmeier and Gerdts, 2020). Note that the model still imposes some complexities owing to the coupled system dynamics in χ.

The model is subject to the constraints:

$$\begin{aligned} \underline{d}(s(t)) \leq d(t) \leq \overline{d}(s(t)) \\ \underline{v} \leq v(t) \leq \overline{v}(s(t)) \\ -\overline{a_n} \leq \kappa(t)v(t)^2 \leq \overline{a_n} \\ \underline{u_1} \leq u_1(t) \leq \overline{u_1} \\ \underline{u_2} \leq u_2(t) \leq \overline{u_2} \end{aligned} \tag{16}$$

where obstacle avoidance is incorporated by restricting the lateral deviation to be within $d \in [\underline{d}(s), \overline{d}(s)]$, as well as keeping the ego-vehicle's speed under $v \leq \overline{v}(s)$. Determining these limits will be explained in detail in the sequel. $\underline{v} \leq v$ ensures the ego-vehicle moves forward. In addition, operational limits on the controls and lateral acceleration a_n are used to achieve feasible travel paths and human-like driving behavior. Finally, the OCP on the time horizon $[0, t_f]$ reads.

Minimize

$$\frac{1}{2}\int_0^{t_f} \omega_d d(t)^2 + \omega_\chi \chi(t)^2 + \omega_v (v(t) - \overline{v}(s(t)))^2 + \omega_{\eta_v}\eta_v^2 + \omega_{u_1}u_1(t)^2 + \omega_{u_2}u_2(t)^2 \quad dt$$

s.t. (15), (16), and the initial values $x(0) = \hat{p}$.

where the cost function encompasses minimizing the lateral deviation d and the heading error χ to reference path γ_r, as well as traveling with the highest possible velocity while adhering to the control limits. Note that working with *well-scaled quantities* greatly improves the convergence rate and overall performance of the MPC; it is hereby recommended to scale and normalize the problem variables, objective function, and system constraints. Finally, the problem can be discretized using Euler's method or the trapezoidal rule to employ the previously discussed approach (Chachuat, 2007).

3.2 Adaptations for Obstacle Avoidance

In scope of AD applications, a world model engulfs all relevant information about the ego-vehicle and its surrounding environment that are required to accomplish a Dynamic Driving Task (DDT), e.g., an obstacle avoidance maneuver. For example, this model may include the position, heading, and velocity of the ego-vehicle, similarly for other road users, data on the road infrastructure, the admissible driving area, and so on (Hoss et al., 2022). Since this information may suffer from noise, uncertainty, latency, interference, and failures, different hardware and software solutions exist to rectify this, e.g., using sensor redundancy and data fusion algorithms (Zhang et al., 2023). We briefly list the required data for our path planning and obstacle avoidance task, and assume the presence of sufficient hardware and software components to accurately and reliably deliver this data to the MPC (Wang and Liu, 2022). First, we need the system states x (e.g., using GNSS, IMU, SLAM modules). Second, location of the driving lane(s) (e.g., using cameras and localization modules). Finally, the obstacles' locations as defined by their 3-D bounding boxes (e.g., using LiDAR and camera sensor fusion modules).

For simplicity, we assume that the reference path γ_r lies at the center of a single driving lane of width w_{LN}. Since no control follows in the vertical direction, we only consider an obstacle's 2-D footprint from a bird's-eye view, which is represented by a set of unordered data points O_{xy} (that typically form a trapezoidal shape). Furthermore, the ego-vehicle's control point is located at the center of its rear axle, so we introduce the parameters w for the vehicle's width and (l_f, l_b) for the distances between the control point and the vehicle's front and rear bumpers respectively. To construct the arclength-dependent lateral deviation constraint functions $\underline{d}(s), \overline{d}(s)$, we start with the intuitive heuristic $\underline{d} = -d_w, \overline{d} = d_w$, *s.t.* $d_w = \frac{w_{LN}}{2} - \frac{w}{2}$. Afterwards, we incorporate obstacles by determining their in-lane protrusions against the nearest lane boundary in Frenet frame. This is realized by means of an efficient approximate projection with only two steps of Sutherland-Hodgman (SH) clipping (Sutherland and Hodgman, 1974) as demonstrated in Algorithm 1. Consequently, we build smooth envelopes $g_i(s)$ to engulf the protrusions of projected obstacles, resulting in the lane boundary functions:

$$\underline{d}(s) := -d_w + \sum_{i=1}^{n_r} g_{r,i}(s), \quad \overline{d}(s) := d_w - \sum_{i=1}^{n_l} g_{l,i}(s)$$

where $n_{r/l}$ are the number of obstacles aligned to the right and left lane boundaries respectively, and $g_{r/l,i}(s)$ denote their corresponding envelope functions. Note that this formulation can cater to varying lane boundaries by replacing d_w with smooth functions, like cubic splines, computed from the lane markings. We thus reach the continuously differentiable $\underline{d}(s), \overline{d}(s)$.

3.2.1 Detecting Obstacles' Protrusions

Algorithm 1: Project obstacle O_{xy} unto the nearest lane boundary and return its protrusions in Frenet frame.

```
Data: $\gamma_r, d_w, O_{xy} := \{(x_i, y_i) : i = 1, \ldots, 4\}$
Result: $\zeta \in \{-1, 1\}, O_{sd} := \{(s_i, d_i) : i \in \mathbb{N}_0\}$
/* 0:  Create convex hull                 */
$O_{xy} \leftarrow GrahamScan(O_{xy})$;
/* 1:  Add points at apex                 */
$O_{sd} \leftarrow TransformXYtoSD(\gamma_r, O_{xy})$;
$s_b \leftarrow min\{s_i : (s_i, d_i) \in O_{sd}\}$;          /* Back */
$s_f \leftarrow max\{s_i : (s_i, d_i) \in O_{sd}\}$;          /* Front */
$s_m \leftarrow \frac{s_b + s_f}{2}$;                          /* Middle/Apex */
$C \leftarrow \{(s_m, -d_w), (s_m, d_w)\}$;
$C \leftarrow TransformSDtoXY(\gamma_r, C)$;
$M \leftarrow SutherlandHodgmanAlg(O_{xy}, C)$;
$M \leftarrow M \setminus O_{xy}$;                              /* New points */
$O_{sd} \leftarrow O_{sd} \cup TransformXYtoSD(\gamma_r, M)$;
/* 2:  Get nearest lane boundary          */
$d_r \leftarrow min\{d_i : (s_i, d_i) \in O_{sd}\}$;
$d_l \leftarrow max\{d_i : (s_i, d_i) \in O_{sd}\}$;
if $((d_r <= 0) \wedge (d_l <= 0)) \vee$
   $((d_w + d_r) \leq (d_w - d_l))$ then
    $\zeta \leftarrow -1$;                                     /* Align right */
else
    $\zeta \leftarrow 1$;                                      /* Align left */
end
/* 3:  Clip to get protrusions            */
if $\zeta \leq 0$ then
    $C \leftarrow \{(s_b, -d_w), (s_f, -d_w)\}$;
else
    $C \leftarrow \{(s_f, d_w), (s_b, d_w)\}$;
end
$O_{sd} \leftarrow SutherlandHodgmanAlg(O_{sd}, C)$;
$e_1 \leftarrow [(s_b, 0), (s_m, 0)]^T$;
$e_2 \leftarrow [(s_m, 0), (s_f, 0)]^T$;
$(e_1, e_2) \leftarrow TransformSDtoXY(\gamma_r, (e_1, e_2))$;
if $((\zeta \leq 0) \wedge (\angle(e_1, e_2) \leq \pi)) \vee$
   $((\zeta > 0) \wedge (\angle(e_1, e_2) \geq \pi))$ then
    /* Clip:  Convex result               */
else
    /* Reverse clip:  Non-convex          */
end
```

Generally speaking, projecting an object unto a surface is a straightforward albeit complex process, which, with knowledge of both object's and surface's shapes and their corresponding transformation, can always be solved analytically (Hughes et al., 2009). Nonetheless, we forego a full object projection and use instead an approximate, yet more efficient approach in Algorithm 1 with the following assumptions: first, the road curvature's rate of change across the length of an obstacle is negligible $\Delta\kappa_r \approx 0$, otherwise the road cannot be safely traversed. Second, the road curvature itself is relatively small $\kappa_r \ll 1$ and its distortion effect on the obstacle's shape can be approximated using straight lines (Harwood and Mason, 1994).

We start our projection with creating a convex hull from the obstacle data points using Graham's scan (Graham, 1972) to ensure that O_{xy} is ordered correctly and remove redundant data. Graham's scan yields a satisfactory performance here due to the limited number of input data points. Next, we identify the apex of object distortion as its midpoints in Frenet frame, and use their respective Cartesian projection to augment O_{xy} with additional points that preserve the obstacle's shape in O_{sd}. Herein, we use the helper methods $TransformXYtoSD, TransformSDtoXY$ to convert between the Cartesian and Frenet coordinate systems. We also use $SutherlandHodgmanAlg$, which is one iteration of the standard SH algorithm with a single clipping edge. This step can be repeated with more points and clips to suit larger curvatures κ or varying lane boundaries d_w, yet one iteration proved ample for the problem at hand.

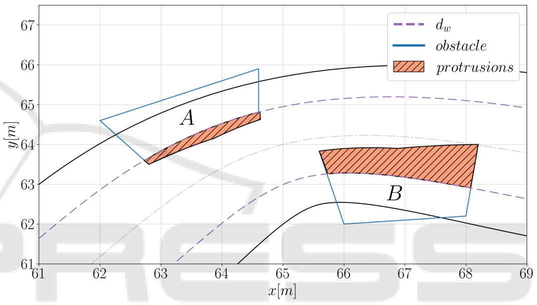

Figure 1: Obstacles A, B (blue) and their protrusions (orange) in Cartesian coordinates using Algorithm 1.

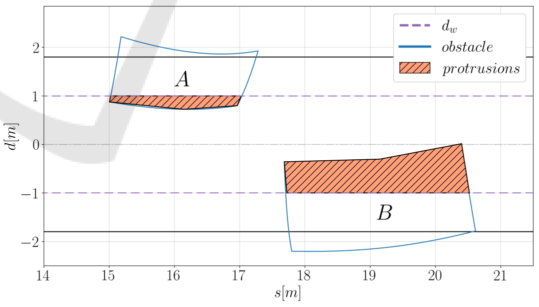

Figure 2: Obstacles A, B (blue) and their protrusions (orange) in Frenet frame using Algorithm 1.

Subsequently, we detect the nearest lane boundary ζ using the minimum d-to-boundary difference in Frenet frame. ζ and the road's orientation designate the clipping direction, with which we carry out a second iteration of SH at the lane boundary to yield O_{sd}, i.e., the set of protruded obstacle data points. Note that O_{sd} may contain concave or separate polygons in case of reverse clipping, and that $O_{sd} = \emptyset$ if the obstacle completely lies outside the lane. We initially handled the case of non-convex polygons by converting them to convex hulls (again with Graham's scan), yet this was abandoned in favor of employing more conservative safety margins in the envelope functions to achieve better performance. Overall, Algorithm 1 is quite efficient, in which determining the protrusions of the two obstacles A, B illustrated in Figure (1, 2) for $d_w = 1[m]$ took only $7.4 \times 10^{-4}[s]$, thus it can handle a multitude of obstacles in a few milliseconds.

3.2.2 Constructing Envelope Functions

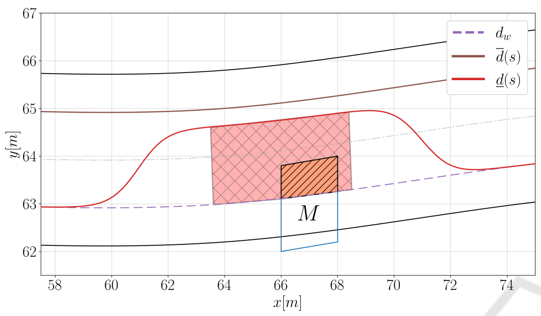

Figure 3: An obstacle with its envelope function in Cartesian coordinates. The red shaded area is the extended footprint on account of w, l_b, l_f and obstacle-specific safety margins.

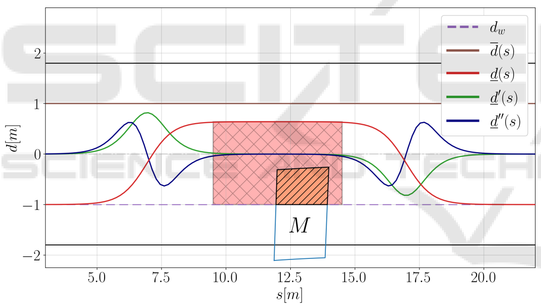

Figure 4: An obstacle with its envelope function in Frenet frame. The function is easily and sufficiently differentiable.

Finally, we compute s_b, s_f, d_r, d_l from the set of protrusions O_{sd} (similar to Algorithm 1 and derive a smooth envelope function that entirely captures the obstacle using logistic functions with:

$$g(s) := \alpha g_b(s) g_f(s), \quad g_{b/f}(s) = \frac{1}{1 + \exp^{-k(s-s_o)}}$$

where k, s_o are derived from the obstacle's location $s_{b/f}$ and the ego-vehicle's length $l_{f/b}$, and α is derived from d_r or d_l (based on ζ) and the ego-vehicle's w. Depending on the obstacle's type, parameters α, k, s_o may also be padded with additional safety margins. A complete envelope function with its derivatives are available in Figure (3, 4). Remarkably, the envelope function $g(s)$ is sufficiently differentiable and has the nice property that:

$$g' = kg(1-g), \quad g'' = kg'(1-2g)$$

i.e., its derivatives can be analytically reached, thus reducing the computational burden and enabling faster convergence. Incidentally, a more accurate (yet equally more demanding) approach to create sufficiently differentiable $g(s)$ would be to apply curve fitting methods directly to the protrusions O_{sd}, e.g., using polynomial or spline curves (Zhang and Chen, 2022). Unfortunately, this is not only slower, but also mandates having non-concave protrusions and also -preferably- convex hulls for obstacles that are too close to each other so as to avoid overaggressive maneuvering. Implementing and evaluating a curve fitting approach is hereby deferred for future work.

3.2.3 Handling Obstructed Travel Paths

After computing $\underline{d}, \overline{d}$, we evaluate these functions across the MPC prediction horizon and inspect if the travel path is blocked due to one or more obstacles by:

$$s_h := \{min(s) \in s(t) : \underline{d}(s) \geq \overline{d}(s), t \in [0, t_f]\}$$

where $s_h = \emptyset$ if no obstacles obstruct the travel path. Afterwards, we employ a single logistic function to construct the smooth bound $\overline{v}(s)$ $s.t.$ $\overline{v}(s_h) \approx 0$, which reduces the maximum allowed velocity to 0 at s_h. This can also be utilized to arbitrarily stop the ego-vehicle, e.g., at traffic lights or at a temporary destination, by means of a virtual blocking obstacle at a desired $\tilde{s_h}$. Otherwise, $\overline{v}(s)$ remains constant across the prediction horizon and is equal to the road speed limit.

To promote passenger comfort and avoid excessive maneuvering, we utilized different sets of MPC parameters ω_* for normal driving and stopping. We added the rigid activation condition $max(\overline{v}(s)) \leq 2.0[ms^-1]$ and evaluated it at the start of each prediction iteration to identify the driving mode and accordingly update ω_* for this iteration. This concludes our MPC design process for path planning and obstacle avoidance.

4 SIMULATION ENVIRONMENT

To properly assess our MPC, it is essential to understand how it fits as a component in a standard architecture for developing AD systems. Several references explain the different hardware and software elements comprising a functional architecture (Zong et al., 2018; Parekh et al., 2022; Ziegler et al., 2014), which we employed as guidelines to identify the key features required to effectively simulate the developed controller. These components alongside their communication channels are summarized in Figure (5) and will be briefly described in the subsequent sections.

In addition, a prevalent approach in developing AD features involves preparing the system components as

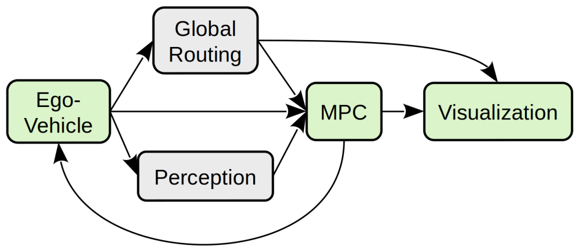

Figure 5: Simplified components of the simulation environment and their communication channels.

Robot Operating System (ROS) packages. ROS is an open-source middleware framework for robotics development and is extensively utilized in diverse fields, e.g., aviation, automotive, and agriculture (Quigley et al., 2009). Accordingly, we developed the simulation environment components as separate software packages in order to facilitate their migration to ROS in the future.

4.1 Ego-Vehicle

The first component is the motion model that suitably simulates the dynamical behavior of the ego-vehicle based on the received control action from the MPC. Herein, we offer two models defined in different coordinate systems to ensure the flexibility of our approach: first, a nonlinear version of the point-mass vehicle model in Frenet frame relative to a reference path (Emam and Gerdts, 2022):

$$\begin{aligned} s'(t) &= \frac{v(t)\cos\chi(t)}{1-d(t)\kappa_r(s(t))} \\ d'(t) &= v(t)\sin\chi(t) \\ \chi'(t) &= v(t)\kappa(t) - s'(t)\kappa_r(s(t)) \\ \kappa'(t) &= u_1(t) \\ v'(t) &= u_2(t) \end{aligned} \tag{17}$$

with the same state and input vectors as (15). Second, a simplified single-track (kinematic) model in Cartesian coordinates (Burger and Gerdts, 2019):

$$\begin{aligned} x'(t) &= v(t)\cos\varphi(t) \\ y'(t) &= v(t)\sin\varphi(t) \\ \varphi'(t) &= \frac{v(t)}{l_f}\tan\delta(t) \\ \delta'(t) &= l_f u_1(t)\cos^2\delta(t) \\ v'(t) &= u_2(t) \end{aligned} \tag{18}$$

with the states: (x,y) the position of the ego-vehicle, φ its yaw angle, δ its steering angle, and v its velocity. The model has the inputs: steering rate w_δ and acceleration, in which w_δ is determined from the MPC input $\kappa' = u_1$ with the mapping $w_\delta(t) = \ell u(t)\cos^2\delta(t)$ (Britzelmeier and Gerdts, 2020).

After each MPC solution iteration, we use the computed controls and employ a Runge-Kutta (RK4) method on the selected system model to accurately calculate the updated vehicle position, heading, and velocity at any desired time. The nonlinearities in both models ensure that the vehicle states are different from the MPC predictions each iteration, which mimics the process of deploying the controller on real hardware.

This component will be substituted with the actual vehicle equipped with sensors and actuator modules that can apply the MPC controls and update the vehicle states accordingly.

4.2 Global Routing

This package is responsible for traversing the road network to identify a feasible travel path γ_{ref} between the current ego-vehicle position and the desired destination as selected by the passenger. This can be achieved by means of, for example, a combination of A* or Dijkstra algorithms (Qin et al., 2023) and OpenStreetMap (OSM) data (OpenStreetMap contributors, 2017). This is a high-level task that a navigation system traditionally performs, and is stubbed in our simulation environment with a pre-defined reference path data, which is transmitted to the controller at the beginning of the simulation.

4.3 Perception

Perception is an umbrella term for environment analysis objectives, like: traffic signs recognition, object detection and tracking, sensor data processing and fusion, and many more (Parekh et al., 2022). Generally, it comprises several modules, which are responsible for building task-oriented world models to cater to the different system DDTs (Zhang et al., 2023). Due to its complexity, we implement a mock version, which simply transmits pre-defined positions and orientations of static obstacles located across the travel path to emulate an obstacle avoidance scenario.

4.4 MPC

We prepared the developed controller as a software package that receives the global path data, positions and orientations of obstacles, and relevant ego-vehicle states (e.g., position and velocity). Subsequently, the MPC solves the DOCP and determines the optimal control action, which is then transferred back to the ego-vehicle component to initiate its movement. Furthermore, the control action and other important data, such as the local planned travel path and the DOCP solution time, are transmitted to the visualization component for data collection and monitoring.

4.5 Visualization

Lastly, this helper package was implemented to facilitate gathering and displaying all relevant problem information. This includes the global reference path, the vehicle states, the iterative DOCP travel path and solution time, and so on. The package is used in the numerical simulation and in performance assessment of the developed controller.

5 NUMERICAL TESTING AND RESULTS

After laying out the components of the OCP solver and simulation environment, we implemented them entirely in C++ and recorded the simulation results on a Linux system with the processor i5-5200U of 2.20GHz and 8GB of RAM. For problem discretization, we utilized the trapezoidal rule, and for the MPC parameters, we selected a prediction horizon of $\Delta T = 4.5[s]$ with $N = 30$ control points and a time step of $T_s = 0.15[s]$. The objective function weights were initially selected based on the scaled contribution of system states and controls, then slightly modified using trial and error.

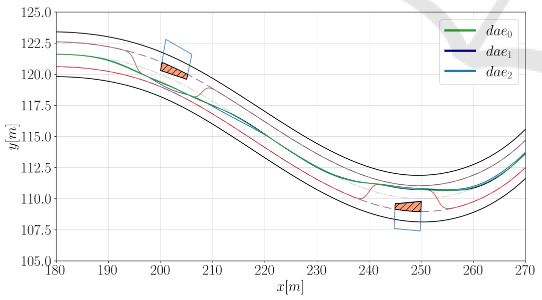

Figure 6: Path following with obstacle avoidance is successfully completed on a highly dynamic road.

We developed and executed two scenarios to highlight our DDTs, i.e., path tracking with obstacle avoidance and a braking maneuver in case of full road obstruction. Each scenario was simulated three times with the linear (15) and nonlinear (17), (18) motion models, which are denoted in the upcoming figures as dae_0, dae_1, dae_2 respectively. Also, we proceed with the same color palette for marking the reference path, driving lane, and admissible driving area on account of $(\underline{d}(s), \overline{d}(s))$.

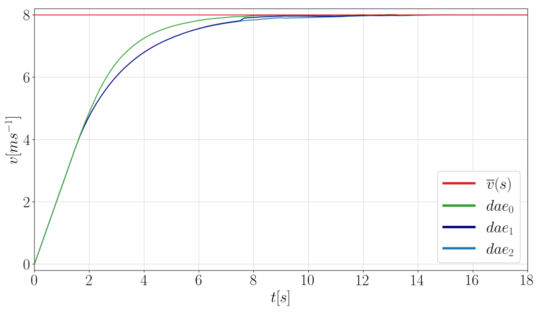

Figure 7: Velocity trajectories of the motion models in the first test scenario.

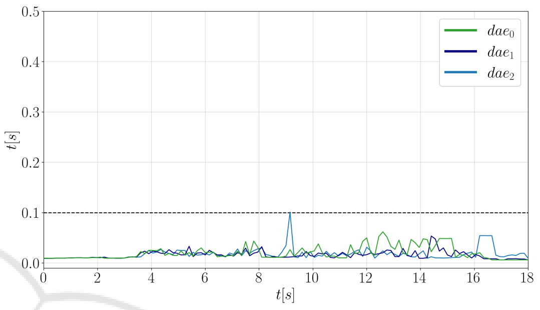

Figure 8: Solution time of the DOCPs in the first scenario.

In the first test scenario, the ego-vehicle starts moving from standstill and is able to successfully maneuver between two stationary obstacles on either side of its travel path as demonstrated in Figure (6). Slight variations between the motion models are noticeable, which is expected due to their different dynamical behaviors. The vehicle controls are properly mapped and minimal execution time is required to reach on optimal solution, cf. Figure (8).

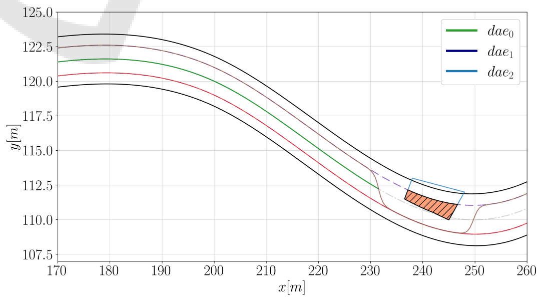

Figure 9: The ego-vehicle brakes properly in case of a fully obstructed path.

In the second scenario, the ego-vehicle starts with $v = 6[ms^{-1}]$ and accelerates at first to reach $\overline{v} = 8[ms^{-1}]$ before it recognizes that its travel path is obstructed. Accordingly, the smooth function $\overline{v}(s)$ is constructed and the vehicle decelerates and adequately stops behind the blocking obstacle as illustrated in Figure (9). Notice that the execution time marginally

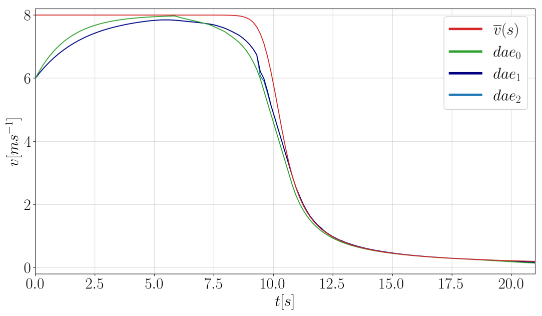

Figure 10: Velocity trajectories of the motion models in the second test scenario.

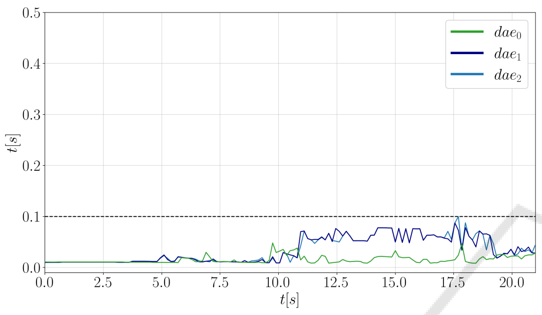

Figure 11: Solution time of the DOCPs in the second scenario.

increases in the second segment of the test scenario, i.e., after $t = 11.0[s]$, which is due to attempting to travel at a considerably low speed. This leads to ill-conditioning of the DOCP, which unnecessarily increases the computational time and can be remedied by refining the braking/stopping MPC parameters ω_*. Alternatively, a simpler controller (e.g., PID) can be developed and configured to halt the ego-vehicle, and a hybrid strategy can be introduced to switch between both controllers based on the driving situation.

Table 1: Average and maximum solution times (in seconds) for both test scenarios.

	Scenario 1		Scenario 2	
	mean	*max*	*mean*	*max*
dae_0	0.01982	0.06214	0.01582	0.04817
dae_1	0.01577	0.05378	0.03425	0.08739
dae_2	0.01796	0.10132	0.03424	0.09987

We highlight that in both test scenarios the MPC controller is almost always able to compute a feasible solution under $100[ms]$, cf. Table (1). This, coupled with its high update rate and appropriate prediction horizon, is sufficient to satisfy the real-time requirement for a typical autonomous driving application.

Lastly, an important point to discuss is that MPC controllers are problem-specific, so it is quite challenging to compare them to each other, even for the same DDT. For example, (Gutjahr et al., 2017; Britzelmeier and Gerdts, 2020) use MPC for obstacle avoidance, where they utilize a linear MPC for lateral vehicle control with independent longitudinal control. We, however, tackle a DOCP with coupled lateral and longitudinal controls. Moreover, (Zhang and Chen, 2022; Wang and Liu, 2022) generate collision-free travel paths using RRT* and A* algorithms, then employ MPC controllers for path following. Yet, our approach embeds the path generation aspect in the controller itself, in addition to handling fully obstructed travel paths. For more examples on MPC in AD applications, we refer the reader to (Musa et al., 2021).

6 CONCLUSION AND FUTURE WORK

In this work, we discussed the development process and successful deployment of an efficient NMPC controller to achieve real-time path planning and obstacle avoidance for autonomous vehicles. First, we extended a previously proposed semi-smooth Newton method to accommodate nonlinear DOCPs. Second, we elaborated on MPC development for path planning, while introducing a numerically efficient object projection algorithm that allows for obstacle incorporation and simplifies the problem constraints to guarantee faster solution convergence. Finally, we implemented a simulation environment and assessed the performance of the developed controller for different motion models, and in driving situations that mimic reality.

Ideas for future work include developing a high-fidelity vehicle model to more accurately simulate the vehicle dynamics and further expedite the controller development and tuning by minimizing the reliance on actual test vehicles. As for obstacle avoidance, adapting our proposed algorithm to handle dynamic objects and/or multiple driving lanes, as well as investigating a curve-fitting approach for more realistic safety margins, are intriguing prospects. Finally, we conclude our suggestions with a full migration of the controller to ROS and its integration and deployment on actual hardware. Nonetheless, this will require thorough testing and optimization to ensure that the real-time requirement still upholds.

ACKNOWLEDGEMENTS

This research paper is part of [project MORE – Munich Mobility Research Campus] and funded by dtec.bw – Digitalization and Technology Research Center of the Bundeswehr. dtec.bw is funded by the European Union – NextGenerationEU.

REFERENCES

Britzelmeier, A. and Gerdts, M. (2020). A nonsmooth newton method for linear model-predictive control in tracking tasks for a mobile robot with obstacle avoidance. *IEEE Control Systems Letters*, PP:1–1.

Burger, M. and Gerdts, M. (2019). *DAE Aspects in Vehicle Dynamics and Mobile Robotics*, pages 37–80. Springer International Publishing, Cham.

Chachuat, B. (2007). Nonlinear and dynamic optimization: From theory to practice.

Chen, X., Nashed, Z., and Qi, L. (2000). Smoothing methods and semismooth methods for nondifferentiable operator equations. *SIAM Journal on Numerical Analysis*, 38(4):1200–1216.

Clarke, F. H. (1990). *Optimization and Nonsmooth Analysis*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, USA.

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

Emam, M. and Gerdts, M. (2022). Deterministic operating strategy for multi-objective nmpc for safe autonomous driving in urban traffic. In *Proceedings of the 8th International Conference on Vehicle Technology and Intelligent Transport Systems - VEHITS*, pages 152–161.

Emam, M. and Gerdts, M. (2023). Sensitivity updates for linear-quadratic optimization problems in multi-step model predictive control. *Journal of Physics: Conference Series*, 2514(1):012008.

Fischer, A. (1992). A special newton-type optimization method. *Optimization*, 24(3-4):269–284.

Gerdts, M. (2023). *Optimal control of ODEs and DAEs*. De Gruyter Oldenbourg, Munich, Germany, 2 edition.

Gerdts, M. and Kunkel, M. (2008). A nonsmooth newton's method for discretized optimal control problems with state and control constraints. *Journal of Industrial & Management Optimization*, 4(2):247–270.

Gerdts, M. and Kunkel, M. (2009). A globally convergent semi-smooth newton method for control-state constrained dae optimal control problems. *Computational Optimization and Applications*, 48(3):601–633.

Graham, R. (1972). An efficient algorith for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132–133.

Grüne, L. and Pannek, J. (2011). *Nonlinear Model Predictive Control: Theory and Algorithms*. Communications and Control Engineering. Springer, London, England.

Gutjahr, B., Gröll, L., and Werling, M. (2017). Lateral vehicle trajectory optimization using constrained linear time-varying mpc. *IEEE Transactions on Intelligent Transportation Systems*, 18(6):1586–1595.

Harwood, D. W. and Mason, J. M. (1994). Horizontal curve design for passenger cars and trucks. *Transportation Research Record*.

Hoss, M., Scholtes, M., and Eckstein, L. (2022). A review of testing object-based environment perception for safe automated driving. *Automotive Innovation*, 5(3):223–250.

Hughes, J. F., McGuire, M. S., Foley, J., Sklar, D. F., Feiner, S. K., Akeley, K., Van Dam, A., and Foley, J. D. (2009). *Computer Graphics: Principles and Practice*. Addison-Wesley Educational, Boston, MA, 3 edition.

Jiang, H. (1999). Global convergence analysis of the generalized newton and gauss-newton methods of the fischer-burmeister equation for the complementarity problem. *Mathematics of Operations Research*, 24(3):529–543.

Jiang, H. and Qi, L. (1997). A new nonsmooth equations approach to nonlinear complementarity problems. *SIAM Journal on Control and Optimization*, 35(1):178–193.

Jiang, H. and Ralph, D. (1998). *Global and Local Superlinear Convergence Analysis of Newton-Type Methods for Semismooth Equations with Smooth Least Squares*, pages 181–209. Springer US.

Morari, M. and H. Lee, J. (1999). Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4–5):667–682.

Musa, A., Pipicelli, M., Spano, M., Tufano, F., De Nola, F., Di Blasio, G., Gimelli, A., Misul, D. A., and Toscano, G. (2021). A review of model predictive controls applied to advanced driver-assistance systems. *Energies*, 14(23).

OpenStreetMap contributors (2017). Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org.

Palma, V. G. (2015). *Robust Updated MPC Schemes*. PhD thesis, Universität Bayreuth, Bayreuth.

Parekh, D., Poddar, N., Rajpurkar, A., Chahal, M., Kumar, N., Joshi, G. P., and Cho, W. (2022). A review on autonomous vehicles: Progress, methods and challenges. *Electronics*, 11(14):2162.

Qin, H., Shao, S., Wang, T., Yu, X., Jiang, Y., and Cao, Z. (2023). Review of autonomous path planning algorithms for mobile robots. *Drones*, 7(3):211.

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

Rawlings, J. B. and Mayne, D. Q. (2009). *Model Predictive Control: Theory and Design*. Nob Hill Pub, LLC, California, USA.

Sussmann, H. and Willems, J. (1997). 300 years of optimal control: From the brachystochrone to the maximum principle. *IEEE Control Systems*, 17(3):32–44.

Sutherland, I. E. and Hodgman, G. W. (1974). Reentrant polygon clipping. *Communications of the ACM*, 17(1):32–42.

Ulbrich, M. (2002). Semismooth newton methods for operator equations in function spaces. *SIAM Journal on Optimization*, 13(3):805–841.

Vu, T. M., Moezzi, R., Cyrus, J., and Hlava, J. (2021). Model predictive control for autonomous driving vehicles. *Electronics*, 10(21):2593.

Wang, H. and Liu, B. (2022). Path planning and path tracking for collision avoidance of autonomous ground vehicles. *IEEE Systems Journal*, 16(3):3658–3667.

Zhang, D. and Chen, B. (2022). Path planning and predictive control of autonomous vehicles for obstacle avoidance. In *2022 18th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*. IEEE.

Zhang, Y., Carballo, A., Yang, H., and Takeda, K. (2023). Perception and sensing for autonomous vehicles under adverse weather conditions: A survey. *ISPRS Journal of Photogrammetry and Remote Sensing*, 196:146–177.

Ziegler, J., Bender, P., Dang, T., and Stiller, C. (2014). Trajectory planning for bertha - a local, continuous method. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE.

Zong, W., Zhang, C., Wang, Z., Zhu, J., and Chen, Q. (2018). Architecture design and implementation of an autonomous vehicle. *IEEE Access*, 6:21956–21970.