Sequential Convex Programming Methods for Real-Time Optimal Trajectory Planning in Autonomous Vehicle Racing

Patrick Scheffe, Theodor Mario Henneken, Maximilian Kloock, and Bassam Alrifaee

Abstract—**Optimization problems for trajectory planning in autonomous vehicle racing are characterized by their nonlinearity and nonconvexity. Instead of solving these optimization problems, usually a convex approximation is solved instead to achieve a high update rate. We present a real-time-capable model predictive control (MPC) trajectory planner based on a nonlinear single-track vehicle model and Pacejka's magic tire formula for autonomous vehicle racing. After formulating the general nonconvex trajectory optimization problem, we form a convex approximation using sequential convex programming (SCP). The state of the art convexifies track constraints using sequential linearization (SL), which is a method of relaxing the constraints. Solutions to the relaxed optimization problem are not guaranteed to be feasible in the nonconvex optimization problem. We propose sequential convex restriction (SCR) as a method to convexify track constraints. SCR guarantees that resulting solutions are feasible in the nonconvex optimization problem. We show recursive feasibility of solutions to the restricted optimization problem. The MPC is evaluated on a scaled version of the Hockenheimring racing track in simulation. The results show that MPC using SCR yields faster lap times than MPC using SL, while still being real-time capable.**

Index Terms—**Autonomous vehicles, control and optimization, motion planning, vehicle racing, vehicle control.**

I. Introduction

A. Motivation

Autonomous vehicle racing is challenging, as the vehicle is moving at its handling limits. The goal in autonomous vehicle racing is to finish a race in minimum time. One of the building blocks to reach that goal is planning a time-optimal trajectory that adheres to the vehicle's nonlinear dynamics and to generally nonconvex track constraints. In optimization-based trajectory planning, this represents solving a nonconvex optimization problem. Computing the global solution of a nonconvex optimization problem is computationally inefficient, i.e., the computation time grows exponentially with the size of the optimization problem. A method for nonconvex optimization is, e.g., mixed integer programming [1]. Driving at the vehicle's handling limits requires a trajectory planner with a high update rate and therefore short computation time. Finding close to globally optimal solutions requires a large enough horizon to cover subsequent curves. Only a computationally efficient method can fulfill both requirements.

The high computational cost of finding the global solution of a nonconvex optimization problem can be accelerated by algorithms that instead find a good local solution. Sequential convex programming (SCP) [2]–[5] is a local optimization method that can find such good local solutions to nonconvex optimization problems. SCP is an iterative algorithm which solves a sequence of convex optimization problems. This allows the use of convex optimization methods, which always compute the global solution up to a numerical accuracy and are computationally efficient. Their computation time grows only polynomially with the size of the optimization problem and the numerical accuracy [1].

In each iteration of SCP, the algorithm forms a convex approximation of the original optimization problem. The approximation of nonconvex constraints can either be a restriction, which tightens the constraints, or a relaxation, which loosens the constraints. A restriction leads to a feasible set of the optimization problem that is a subset of the original one, meaning that the solution to a restricted optimization problem is always feasible in the original one. We call SCP-algorithm which tightens the constraints SCR in the following.

The following section presents various approaches besides SCP that efficiently find good local solutions for nonconvex optimization problems. All of these approaches simplify the original problem at some point in the optimization algorithm, compromising between global optimality and computational efficiency [1].

B. Related Work

There are many works on trajectory planning for autonomous vehicle racing, and the research interest increased over the last years, as the recent article [6] shows. The authors of [6] present

Manuscript received 4 March 2022; revised 30 March 2022; accepted 31 March 2022. Date of publication 19 April 2022; date of current version 23 January 2023. This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) within the Priority Program SPP 1835 Cooperative Interacting Automobiles and the Post Graduate Program GRK 1856 Integrated Energy Supply Modules for Roadbound E-Mobility. *(Corresponding author: Patrick Scheffe.)*

The authors are with the Chair of Embedded Software, RWTH Aachen University, 52074 Aachen, NRW, Germany (e-mail: scheffe@embedded.rwth-aachen.de; theodor.henneken@rwth-aachen.de; kloock@embedded.rwth-aachen.de; bassam.alrifaee@rwth-aachen.de).

A video showing the results is available at https://youtu.be/7Iwh980JMCs. The software is published on Github: https://github.com/embedded-software-laboratory/sequential-convex-programming and ready-to-use on Code Ocean: https://codeocean.com/capsule/6818033/tree.

Digital Object Identifier 10.1109/TIV.2022.3168130

recent developments in trajectory planning for vehicle racing. Trajectory planning algorithms can mainly be categorized as graph-based and optimization-based [7]. These methods deal with the nonconvexity introduced by the track constraints differently.

Graph-based trajectory planning aims at finding the global solution of the nonconvex optimization problem. The discretization of the configuration space accelerates the optimization. However, their ability to find the global solution strongly depends on this discretization. Graph-based trajectory planning methods need to check if a trajectory is collision-free, which includes the adherence to track constraints, e.g., [8]–[10].

Optimization-based trajectory planning follows different strategies to deal with the generally nonconvex track constraints. We present four of them in the following.

One strategy is to add the track constraint to the objective function, which is known as a barrier function in optimization. This resolves the nonconvexity due to track constraints. model predictive control (MPC) with GPU computing is proposed in [11] for a rally car on loose ground operating within friction limits of the tires. The barrier function for the track constraints results in an undesirable high cost at the track boundaries. Consequently, the trajectories tend to be close to the track centerline, which is suboptimal regarding lap time. A different barrier function with varying weight is used in [12] with a two-track vehicle model. The optimization problem is solved iteratively, which leads to better constraint approximations at the cost of higher computation time.

A second strategy is to formulate the optimization problem with a spatial discretization [13], [14]. In combination with a coordinate transformation, this resolves both the nonlinearity and nonconvexity of the track constraints. The approaches approximate the minimization of lap time with the minimization of the path curvature. zThe speed profile is optimized afterwards such that the trajectories adhere to given vehicle dynamic constraints.

A third strategy is to combine the coordinate transformation with the usual temporal discretization of MPC [15]–[19]. In contrast to the spatial discretization, the vehicle dynamics can be incorporated in the trajectory planning problem by expressing the vehicle model based on the track coordinate system. The nonlinear model and nonlinear track constraints result in a nonlinear optimization problem, which can be solved with nonlinear programming (NLP). Common software, such as FORCES [20] or IPOPT [21], solve such nonlinear optimization problems using sequential quadratic programming (SQP). SQP is an iterative algorithm which approximates the objective with a quadratic function and linearizes constraints.

A fourth strategy is to explicitly form convex approximations of the nonconvex optimization problem with SCP. There are several works in the state of the art using this strategy, which form the convex approximation by linearization of nonlinear and nonconvex constraints. We call this method SL in the following. Similar to the approaches above, [22], [23] transform coordinates to the track coordinate system to form a nonlinear optimization problem, which is subsequently solved by SL. Experiments with a full-scale vehicle validate the approach in [23]. A linearization of the track constraints around the solution of the previous timestep to obtain a convex quadratic programming, (QP) is presented in [24]. This corresponds to SL with only one iteration.

In [25], model predictive contouring control for autonomous racing with obstacles and opponents is proposed. The work presents the first implementation of an autonomous racing trajectory planner and tracker for a model-scale vehicle testbed. The optimization problem is solved with SL in one iteration. In [26], a trajectory planner which linearizes race track constraints performs approximately as well as a professional driver in a full-scale experiment. We convexified track boundaries with SL for vehicle racing in [27]. Additionally, we extended the approach for priority-based MPC for multiple vehicles in [28].

The error from approximating the race track using linearization may lead to performance degradation. Since the linearization of track constraints represent a relaxation, the resulting solutions are not guaranteed to be feasible in the original optimization problem.

C. Contribution of This Article

We formulate a general nonconvex optimization problem for trajectory planning for autonomous vehicle racing. An MPC based on a single-track model and Pacejka's Magic Tire Formula for the vehicle dynamics simultaneously computes both the trajectory and control inputs. We convexify this original, nonconvex optimization problem using SCP. The key contribution of this paper lies in the application-specific approximation of the original problem to a convex quadratic program by SCR. We approximate the track constraints as a set of polygons. A polygon is a set of linear inequality constraints in the optimization problem, sustaining its convexity. Since each polygon represents a restriction of the original nonconvex track constraint, our algorithm guarantees the feasibility of a solution to the restricted problem in the original problem. Additionally, we show that trajectories resulting from the optimization of the restricted optimization problem keep the property of recursive feasibility.

We compare the proposed SCR with the widely-known SL [22]–[28] in simulation. We show that besides guaranteeing feasible trajectories in the nonconvex optimization problem, the trajectories resulting from SCR are superior to those from SL in terms of time-optimality.

D. Structure of This Article

The remainder of this article consists of three main parts. First, we formulate the general nonconvex optimization problem for trajectory planning for autonomous vehicle racing in Section II. Second, we present the concept of SCP in Section III, and summarize its variant SL to convexify the original problem in Section IV. Section V details a convex restriction of the original problem for trajectory planning with SCR. Third, we compare the presented SCR with the state-of-the-art SL in Section VI.

II. General Formulation of Trajectory Optimization Problem

We denote the temporal dependency by a parenthesized index in a superscript, e.g., $\boldsymbol{x}^{(j)}$. Operating points are denoted by a tilde, e.g., $\tilde{\boldsymbol{x}}$.

A. Vehicle Models

We implement a single-track model of an RC car suited for vehicle racing applications and compare it to the linear vehicle model introduced in our previous paper [27]. We summarize both models below for completeness, the detailed modeling process is presented in the referenced literature. Both models share the same assumptions of having no aerodynamic lift or downforce, no pitch, no wear, and no weight changes.

The generalized discretized model is formulated as

$$\boldsymbol{x}^{(j)} = \boldsymbol{M}\left(\boldsymbol{x}^{(j-1)}, \boldsymbol{u}^{(j-1)}\right), \quad \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{u} \in \mathbb{R}^m, \tag{1}$$

where the model $\boldsymbol{M}$ computes the state $\boldsymbol{x}^j$ at timestep j based on the state $\boldsymbol{x}^{(j-1)}$ and the input $\boldsymbol{u}^{(j-1)}$ of the previous timestep.

1) Single-Track Model: A kinetic single-track model is combined with Pacejka's Magic Tire Formula [29]. The resulting model considers rolling and air resistance, as well as the fixed-gear electric motor's characteristic curve of maximal torque and limited power. The model can further reflect vehicle racing conditions such as oversteering, understeering and drifting. Individual front and rear lateral tire slips are used for the computation of tire forces. Braking and acceleration torques are applied at the rear wheels only, representing the setup of the RC car from [30]. Thus, only the rear wheel needs consideration of combined tire forces, which is done implicitly via the lateral tire parameters [25], [31]. The detailed construction including parameters of the entire model was given in [25]. On a side note, the experimentally estimated tire parameters combined with the input constraints result in the tire forces remaining at the convex parts of Pacejka's Magic Tire Formula.

The states of the model are the position p from the center of gravity to the reference frame, the velocity v in directions x and y, the heading angle ϕ and the accompanying yaw rate ω. The input consists of the steering angle δ and the electric motor's duty cycle τ. The differential equations which we use in discretized form are

$$\dot{p}_x = v_x \cos(\phi) - v_y \sin(\phi), \tag{2a}$$
$$\dot{p}_y = v_x \sin(\phi) + v_y \cos(\phi), \tag{2b}$$
$$\dot{v}_x = \frac{1}{m}\left(F_{r,x} - F_{f,y}\sin(\delta) + m v_y \omega\right), \tag{2c}$$
$$\dot{v}_y = \frac{1}{m}\left(F_{r,y} + F_{f,y}\cos(\delta) - m v_x \omega\right), \tag{2d}$$
$$\dot{\phi} = \omega, \tag{2e}$$
$$\dot{\omega} = \frac{1}{I_z}\left(F_{f,y} l_f \cos(\delta) - F_{f,y} l_r\right), \tag{2f}$$

with the mass m, the moment of inertia I_z and the distances from the center of gravity to front and rear axles l_f and l_r, respectively. The rear longitudinal tire force $F_{r,x}$, rear lateral tire force $F_{r,y}$ and front lateral tire force $F_{f,y}$ are computed as

$$F_{r,x} = (C_{m1} - C_{m2} v_x)\,\tau - C_{r0} - C_{r2} v_x^2, \tag{3a}$$
$$F_{f,y} = D_f \sin(C_f \arctan(B_f \alpha_f)), \tag{3b}$$
$$\alpha_f = -\arctan\left(\frac{\omega l_f + v_y}{v_x}\right) + \delta, \tag{3c}$$
$$F_{r,y} = D_r \sin(C_r \arctan(B_r \alpha_r)), \tag{3d}$$
$$\alpha_r = \arctan\left(\frac{\omega l_r - v_y}{v_x}\right), \tag{3e}$$

where B_f, B_r, C_f, C_r, D_f and D_r are empiric tire formula coefficients and α_f and α_r the slip angles for front and rear tires, respectively. The electric motor is modeled with the parameters C_{m1} and C_{m2}, while the resistance parameters C_{r0} and C_{r2} are empirically determined. Singularities induced by the slip angle definitions are mitigated by replacing v_x with a small number in case of $v_x = 0$. Considering signed function arguments, we implement $\arctan$ with $\arctan2$.

The input constraint set is

$$\mathcal{U}_{\text{ST}} = \left\{\boldsymbol{u} \in \mathbb{R}^2 \mid \delta \in (-0.35\,\text{rad}, 0.35\,\text{rad}) \wedge \tau \in (10\%, 100\%)\right\} \tag{4}$$

2) Linear Model: The linear model was introduced in our previous paper [27]. The states of the model are the vehicle's position $\boldsymbol{p} = (p_x\, p_y)^T$ and velocity $\boldsymbol{v} = (v_x\, v_y)^T$, and the inputs $\boldsymbol{u} = (a_x\, a_y)^T$ are the accelerations in both x- and y-direction. No yaw is considered, the vehicle is always facing in the same direction as the longitudinal velocity. The model with the discretization time Δt is

$$\begin{bmatrix} p_x^{(j)} \\ p_y^{(j)} \\ v_x^{(j)} \\ v_y^{(j)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x^{(j-1)} \\ p_y^{(j-1)} \\ v_x^{(j-1)} \\ v_y^{(j-1)} \end{bmatrix} + \begin{bmatrix} \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{\Delta t^2}{2} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} a_x^{(j-1)} \\ a_y^{(j-1)} \end{bmatrix}. \tag{5}$$

The tires' acceleration limits for lateral and longitudinal slip are described by two velocity-dependent half ellipses [32]. The tire limits are symmetric in lateral direction but asymmetric in longitudinal direction due to driving resistances. The constraint set for the input can be efficiently represented in the vehicle coordinate system as

$$\mathcal{U}_{\text{L}} = \left\{\boldsymbol{u} \in \mathbb{R}^2 \mid \left(\frac{a_{\text{long}}}{a_{\text{long,max}}(\boldsymbol{v})}\right)^2 + \left(\frac{a_{\text{lat}}}{a_{\text{lat,max}}(\boldsymbol{v})}\right)^2 - 1 \leq 0\right\} \tag{6a}$$

$$a_{\text{long,max}}(\boldsymbol{v}) = \begin{cases} a_{\text{forward,max}}(\boldsymbol{v}) & \text{if } a_{\text{long}} > 0 \\ a_{\text{backward,max}}(\boldsymbol{v}) & \text{if } a_{\text{long}} \leq 0 \end{cases} \tag{6b}$$

The transformation to global acceleration inputs is presented in our previous paper [5]. The constraint set is derived with the

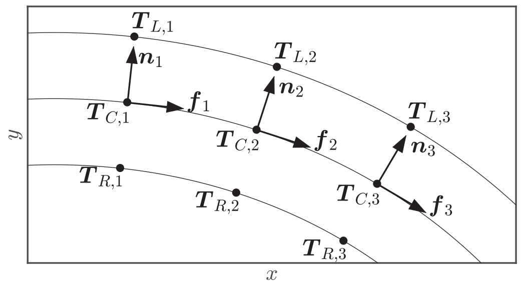


Fig. 1. Example of discrete race track model, a nonconvex constraint [27].

single-track model's input constraints to assure comparability between models.

B. Track Model

The race track is the allowed driving area of a vehicle. We assume the entire track has a constant width and its curve radii to be larger than the track width. A confinement to planar models is sufficient for most tracks, in the other cases it yields only limited error compared to non-planar models used with MPC [33]. In MPC, a numerical description of the track constraints is necessary. In the following, the track's discrete model and a function to measure a vehicle's progress on the track is introduced.

1) Discretized Track Model: Fig. 1 depicts the following discrete track definition. We define a center line $\boldsymbol{C} : [0, S] \to T$ in the track area $T \subseteq \mathbb{R}^2$ with the track length $S \in \mathbb{R}$ and the track progress $s \in [0, S]$. The discretized track points s_i are constructed with the discretization size Δs_i across the indices $i \in \mathbb{N}$. The track's normal $\boldsymbol{n}_i$ is defined to be orthogonal to the gradient of the center-line $\boldsymbol{C}$, called the forward vector $\boldsymbol{f}_i$, as

$$s_i = i \cdot \Delta s_i, \tag{7a}$$

$$\boldsymbol{n}_i = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \tag{7b}$$

$$\boldsymbol{f}_i = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \nabla \boldsymbol{C}(s_i). \tag{7c}$$

The left and right track boundary points $\boldsymbol{T}_{L,i}$ and $\boldsymbol{T}_{R,i}$ are constructed by translating the center point $\boldsymbol{T}_{C,i}$ along $\boldsymbol{n}_i$ with the track width $w \in \mathbb{R}_{>0}$ as

$$\boldsymbol{T}_{C,i} = \boldsymbol{C}(s_i), \qquad \boldsymbol{T}_{C,i} \in T, \tag{7d}$$

$$\boldsymbol{T}_{L,i} = \boldsymbol{T}_{C,i} + \tfrac{w}{2}\boldsymbol{n}_i, \qquad \boldsymbol{T}_{L,i} \in T, \tag{7e}$$

$$\boldsymbol{T}_{R,i} = \boldsymbol{T}_{C,i} - \tfrac{w}{2}\boldsymbol{n}_i, \qquad \boldsymbol{T}_{R,i} \in T. \tag{7f}$$

2) Progress Function: For an objective function, we require to link the position of the vehicle on the track to a progress measurement. We choose the center-line as the reference. This results in the usage of equivalent-valued lines orthogonal to the center-line $\boldsymbol{C}$, modeled by the progress function

$$s(\boldsymbol{p}) = \arg\min_{c\in[0,S]} \left\|\boldsymbol{C}(c) - \boldsymbol{p}\right\|_2. \tag{8}$$

For usage in the objective function, we linearize and discretize the progress function, starting with the linear interpolation of the center-line

$$\boldsymbol{C}(s) \approx \boldsymbol{C}(s_i) + \nabla \boldsymbol{C}(s_i)(s - s_i) \tag{9a}$$

$$= \boldsymbol{T}_{C,i} + \boldsymbol{f}_i(s - s_i). \tag{9b}$$

Inserting (9) into (8), we explicitly solve for s and neglect any constant parts, yielding the objective progress function

$$s(\boldsymbol{p}) \approx s_i + \boldsymbol{f}_j^T(\boldsymbol{p} - \boldsymbol{T}_{C,i}) \approx \boldsymbol{f}_i^T \boldsymbol{p}. \tag{10}$$

C. Model Predictive Control

We use the concept of MPC as the control structure [34]. In MPC, a model of the system is used to predict the states

$$\boldsymbol{X} = (\boldsymbol{x}^{(1)}\ \boldsymbol{x}^{(2)}\ \dots\ \boldsymbol{x}^{(H_p)}), \quad \boldsymbol{X} \in \mathbb{R}^{n\times H_p} \tag{11}$$

under the optimization of the inputs

$$\boldsymbol{U} = (\boldsymbol{u}^{(0)}\ \boldsymbol{u}^{(1)}\ \dots\ \boldsymbol{u}^{(H_u-1)}), \quad \boldsymbol{U} \in \mathbb{R}^{m\times H_u} \tag{12}$$

in the discrete timesteps $1, \dots, H_u. H_u \in \mathbb{N}$ is the finite control horizon, equaling the finite prediction horizon H_p in this article. With that, a state vector $\boldsymbol{x}^{(j)}$ can be assigned to a particular, discrete timestep $j \in \mathbb{N}$. An input vector $\boldsymbol{u}^{(j)}$ is applied starting at timestep j just until timestep $j + 1$.

To achieve recursive feasibility despite using a finite prediction horizon, we add the terminal constraint

$$\boldsymbol{v}^{(H_p)} = \boldsymbol{0}. \tag{13}$$

This requires the standstill of the vehicle at the end of the prediction horizon, thus ensuring to fulfill the track constraints in the next timesteps.

D. Complete General Trajectory Optimization Problem

This section combines the parts from Section II-A to II-C to form a nonlinear nonconvex optimization problem for trajectory planning on a race track without obstacles.

The objective is to maximize a vehicle's track progress (10) at the prediction horizon H_p, or minimize its negation respectively. It has been proven to be an adequate substitution for minimizing time on the race track [25], [27].

To reduce aggressive input changes, we add the damping penalty $R_u \in \mathbb{R}_{\geq 0}$ on the input changes.

The complete program – termed General Trajectory Program – is then

$$\min_{\boldsymbol{X},\boldsymbol{U}} -s\left(\boldsymbol{p}^{(H_p)}\right) + R_u \sum_{j=1}^{H_p-1} \left\|\boldsymbol{u}^{(j)} - \boldsymbol{u}^{(j-1)}\right\|_2^2 \tag{14a}$$

subject to

$$\boldsymbol{x}^{(j)} = \boldsymbol{M}\left(\boldsymbol{x}^{(j-1)}, \boldsymbol{u}^{(j-1)}\right), \quad \forall j=1,\dots,H_p, \tag{14b}$$

$$\boldsymbol{p}^{(j)} \in T, \qquad \forall j=1,\dots,H_p, \tag{14c}$$

$$\boldsymbol{u}^{(j)} \in \mathcal{U}_{\boldsymbol{M}}, \qquad \forall j=0,\dots,H_p-1, \tag{14d}$$

$$\boldsymbol{v}^{(H_p)} = \boldsymbol{0}. \tag{14e}$$

with both the state prediction $\boldsymbol{X}$ and the input prediction $\boldsymbol{U}$ matrices as the decision variables, and the input constraint set $\mathcal{U}_{\boldsymbol{M}}$ for the vehicle model $\boldsymbol{M}$. A sparse formulation is chosen

Algorithm 1: Pseudo-Code for SCP.

```
1: $\boldsymbol{X} \leftarrow \boldsymbol{X}_{\text{init}}$
2: repeat forever                                   ▷control loop
3:   $\boldsymbol{x}^{(0)} \leftarrow$ receiveCurrentState()
4:   for $i = 1$ to $N_{\text{RTI}}$ do
5:     QP ← createConvexQP($\boldsymbol{X}, \boldsymbol{x}^{(0)}$)
6:     $\boldsymbol{X}, \boldsymbol{U} \leftarrow$ solveQP(QP)
7:   end for
8:   applyInput($\boldsymbol{u}^{(1)}$)
9: until
```

as it fits the problem size and parameter selection of $H_u = H_p$ better than a dense one, yielding lower computational cost [35]. If started with an initially feasible trajectory along the track, the General Trajectory Program stays feasible.

III. Sequential Convex Programming Methods

The previous section introduced a nonconvex and thus NP-hard optimization problem [2], [36]. In this section, we convexify the problem using SCP, allowing us to find a solution iteratively in real-time applications [2], [4]. Two variants of SCP are presented: SL, being a relaxation, and SCR, being a restriction. Both methods handle the nonconvex parts of the optimization problem by approximating the track and input constraints.

Algorithm 1 shows the pseudo-code of the real-time iteration (RTI) SCP method. In line 1 the state prediction matrix is initialized with a feasible solution. The control loop starts from line 2. After receiving the current state vector in line 3, $N_{\text{RTI}} \in \mathbb{N}$ SCP iterations run. N_{RTI} is set constant for predictable computation times, turning the SCP into an RTI algorithm, and chosen such that the execution time of the control loop is smaller than the sampling time. In each SCP iteration, first a QP is formulated in line 5 with the current state vector and a warm-start of the previous state prediction matrix. Either the SL or SCR method can convexify the nonconvex parts to create this QP. Then, the QP is solved, yielding a new state prediction matrix and input prediction matrix. After N_{RTI} iterations, the first control input is applied in line 8.

IV. Sequential Linearization

The idea of SL is to linearize nonconvex constraints of the optimization problem at operating points, based on initial guesses or previous solutions, to obtain a QP. This linearization acts as a relaxation of the track and input constraints. The following paragraphs summarize SL, our previous paper [27] presents more details.

A. Input Constraint Linearization

The input constraints for the single-track model are already in linear form, so $\mathcal{U}_{\text{ST,SL}} = \mathcal{U}_{\text{ST}}$. For the linear model, we approximate the velocity-dependent tire acceleration limit ellipses of (6) with $n_{\text{acc}} \in \mathbb{N}$ evenly distributed tangents. This results in a relaxation, vanishing with $n \to \infty$ at higher computational effort when solving the optimization problem. The resulting set of linear inequalities is

$$\mathcal{U}^{(j)}_{\text{L,SL}} = \{\boldsymbol{u} \in \mathbb{R}^2 \mid \boldsymbol{A}_{\text{acc}}\left(\tilde{\boldsymbol{v}}^{(j)}, \tfrac{2\pi k}{n_{\text{acc}}}\right)\boldsymbol{u} \leq \boldsymbol{b}_{\text{acc}}\left(\tilde{\boldsymbol{v}}^{(j)}\right) \quad \forall k = 1, \ldots, n_{\text{acc}}, \}, \tag{15}$$

with the operating point $\tilde{\boldsymbol{v}}$. The matrix $\boldsymbol{A}_{\text{acc}} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2$ and vector $\boldsymbol{b}_{\text{acc}} : \mathbb{R}^2 \to \mathbb{R}$ are computed such that $\boldsymbol{u}$ is constrained to the ellipses.

B. Track Model Linearization

Based on the track discretization from Section II-B1, the track linearization is acquired by tangents to both the left track boundary $\boldsymbol{T}_L$ and the right track boundary $\boldsymbol{T}_R$ at each timestep. To formulate the linear track constraints, the closest center-line point's index i to a position $\boldsymbol{p}$ is acquired similar to (8) by the index function

$$i(\boldsymbol{p}) = \arg\min_{k\in\{0,\ldots,N_{\text{points}}\}} \left\|\boldsymbol{T}_{C,k} - \boldsymbol{p}\right\|_2 . \tag{16}$$

The progress function is used as defined in Section II-B2.

To mitigate approximation issues and subsequent infeasibility of the problem, the track constraints are softened by a slack variable $\xi \in \mathbb{R}_{\geq 0}$ [34]. It is added to the problem objective with a high weighting factor $S \in \mathbb{R}_{>0}$. Due to the minimization of the problem objective, ξ will be close to 0 except when the track constraints cannot be satisfied. Geometrically interpreted, ξ widens the track boundary to both sides by the track's unit. The track area constraint for SL is

$$T^{(j)}_{\text{SL}} = \left\{ \boldsymbol{p} \in \mathbb{R}^2 \;\middle|\; \begin{array}{l} \left(\boldsymbol{p} - \boldsymbol{T}_{L,\tilde{i}}\right)^T \quad \boldsymbol{n}_{\tilde{i}} \leq \xi, \\ \left(\boldsymbol{p} - \boldsymbol{T}_{R,\tilde{i}}\right)^T \quad (-\boldsymbol{n}_{\tilde{i}}) \leq \xi \end{array} \right\}, \tag{17}$$

with the index $\tilde{i} = i(\tilde{\boldsymbol{p}}^{(j)})$ from (16). The deviation of the position to the track boundary is computed via a dot product with the boundary's normal vector. It remains negative while being inside the track boundaries.

The track is approximated at the operating point $\tilde{\boldsymbol{p}}^{(j)}$. This approximation is good for small distances between the actual position and the operating point $\|\boldsymbol{p}^{(j)} - \tilde{\boldsymbol{p}}^{(j)}\|$. This distance is limited to maintain the quality of the track approximation by a trust region with size $L \in \mathbb{R}_{>0}$.

C. Complete Trajectory Optimization Problem

With the generalized formulation from Section II, this section's SL approximations yield the optimization problem

$$\min_{\boldsymbol{X},\boldsymbol{U},\xi} -\boldsymbol{f}^T_{i(\tilde{\boldsymbol{p}}^{(H_p)})}\boldsymbol{p}^{(H_p)} + S\xi + R_{\text{u}} \sum_{j=1}^{H_p-1} \left\|\boldsymbol{u}^{(j)} - \boldsymbol{u}^{(j-1)}\right\|_2^2 \tag{18a}$$

subject to

$$\boldsymbol{x}^{(j)} = \boldsymbol{M}\left(\boldsymbol{x}^{(j-1)}, \boldsymbol{u}^{(j-1)}\right), \qquad \forall j = 1, \ldots, H_p, \tag{18b}$$

$$\boldsymbol{p}^{(j)} \in T_{\text{SL}}^{(j)} \qquad \forall j = 1, \ldots, H_p, \tag{18c}$$

$$\boldsymbol{u}^{(j)} \in \mathcal{U}_{\text{M,SL}}^{(j)} \qquad \forall j = 0, \ldots, H_p - 1, \tag{18d}$$

$$\tilde{p}_x^{(j)} - L \leq p_x^{(j)} \leq \tilde{p}_x^{(j)} + L, \qquad \forall j = 1, \ldots, H_p, \tag{18e}$$

$$\tilde{p}_y^{(j)} - L \leq p_y^{(j)} \leq \tilde{p}_y^{(j)} + L, \qquad \forall j = 1, \ldots, H_p, \tag{18f}$$

$$\boldsymbol{v}^{(H_p)} = \boldsymbol{0}, \tag{18g}$$

$$\xi \geq 0. \tag{18h}$$

V. Sequential Convex Restriction

The method of SCR follows the idea of restricting a nonconvex feasible solution set to a strict convex subset. This section introduces the concept of SCR, followed by the construction and proof of feasible warm-start trajectories. It then applies SCR to our specific application, focusing on the track restriction function.

A. Concept of SCR

We present the core idea of SCR of allowing a new feasible trajectory to be computed based on an existing feasible trajectory. To simplify the argument, we use a generalized form of the General Trajectory Program (14), termed Generalized Unrestricted Program:

$$\min_{z} \quad q(z) \tag{19a}$$

$$\text{subject to} \quad Ez = b, \tag{19b}$$

$$z \in D, \tag{19c}$$

where $z \in \mathbb{R}^n,\ E \in \mathbb{R}^{m \times n},\ b \in \mathbb{R}^m,\ q : \mathbb{R}^n \to \mathbb{R},\ D \subseteq \mathbb{R}^n$. The Generalized Unrestricted Program is in general nonconvex. Let $\tilde{z}$ be a known feasible vector in this program. Based on $\tilde{z}$ we define a convex, restricted program – termed Generalized Restricted Program – as

$$\min_{z} \quad \hat{q}(z, \tilde{z}) \tag{20a}$$

$$\text{subject to} \quad Ez = b, \tag{20b}$$

$$z \in R(\tilde{z}), \tag{20c}$$

where $\hat{q} : \mathbb{R}^{n \times n} \to \mathbb{R}$ is a convex approximation of the objective function q and $R : D \to \mathcal{P}(D)$ is the restriction function converting a nonconvex area to a convex polygon representation, denoted by $\mathcal{P}$. R must satisfy the conditions

$$R(\tilde{z}) \subseteq D \qquad \forall \tilde{z} \in D, \tag{21a}$$

$$R(\tilde{z}) \text{ is convex} \qquad \forall \tilde{z} \in D, \tag{21b}$$

$$\tilde{z} \in R(\tilde{z}) \qquad \forall \tilde{z} \in D. \tag{21c}$$

Fig. 2 illustrates an example for the restriction function and the feasible sets of the Generalized Unrestricted Program and Generalized Restricted Program for a two-dimensional case. The Generalized Restricted Program has by definition at least one feasible point, $\tilde{z}$. The program is convex because it minimizes a convex function on a convex set. Thus, it has a unique optimum, denoted by $\mathcal{F}(\tilde{z})$.

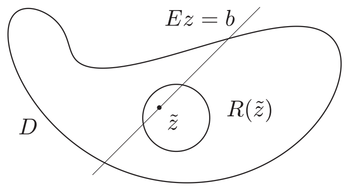


Fig. 2. Illustration of the restriction function and the linear constraints.

$\mathcal{F}(\tilde{z})$ is also a feasible point in the Generalized Unrestricted Program because $\tilde{z} \in R(\tilde{z}) \Rightarrow \tilde{z} \in D$. Thus, $\mathcal{F}(\tilde{z})$ can be used as a new starting point for the Generalized Restricted Program resulting in $\mathcal{F}(\mathcal{F}(\tilde{z}))$. By iteration, we can generate a sequence of points $\tilde{z}_{i+1} = \mathcal{F}(\tilde{z}_i)$ while maintaining the feasibility of each point in the Generalized Unrestricted Program. Whether this sequence approaches an optimal solution of the Generalized Unrestricted Program depends on the formulation of $\hat{q}$ and R.

B. Construction of Trajectories for Recursive Feasibility

Since SCR uses a known feasible trajectory to determine an improved feasible trajectory, we need to specify the former. If trajectories are recursively feasible, the trajectory of the previous timestep represents a starting point. A feasible initial trajectory is, e.g., $\boldsymbol{X} = [\boldsymbol{0}\ \boldsymbol{0}\ \ldots\ \boldsymbol{0}]$ and $\boldsymbol{U} = [\boldsymbol{0}\ \boldsymbol{0}\ \ldots\ \boldsymbol{0}]$.

Theorem 1: Trajectories with the terminal constraint (13) resulting from solving the restricted convex program are recursively feasible in the General Trajectory Program (14).

Proof: In each timestep, the solution of the previous timestep – denoted with a tilde, e.g., $\tilde{\boldsymbol{x}}$ – is shifted by one timestep to enable a warm-start as follows

$$\boldsymbol{x}^{(j-1)} = \tilde{\boldsymbol{x}}^{(j)}, \qquad \forall j = 2, \ldots, H_p, \tag{22a}$$

$$\boldsymbol{u}^{(j-1)} = \tilde{\boldsymbol{u}}^{(j)}, \qquad \forall j = 1, \ldots, H_p - 1. \tag{22b}$$

As $\tilde{\boldsymbol{x}}^{(H_p+1)}$ and $\tilde{\boldsymbol{u}}^{(H_p)}$ do not exist, the trajectory must be explicitly extended by one timestep as

$$\boldsymbol{x}^{(H_p)} = \tilde{\boldsymbol{x}}^{(H_p)}, \tag{22c}$$

$$\boldsymbol{u}^{(H_p-1)} = \boldsymbol{0}. \tag{22d}$$

Such a trajectory fulfills all constraints of the restricted convex program as we show in Appendix A. □

C. Application to Problem

To apply SCR to the General Trajectory Program (14) we start by rewriting it in the form of the Generalized Unrestricted Program (19), before applying the restriction and conforming to the Generalized Restricted Program (20) formulation. Again, we use the linear vehicle model for simplicity. The approach stays the same for the single-track model.

We combine the decision variables of the General Trajectory Program to $z \in \mathbb{R}^{(n+m)\times H_p}$

$$z = \begin{pmatrix} \boldsymbol{p}^{(1)} & \dots & \boldsymbol{p}^{(H_p)} \\ \boldsymbol{v}^{(1)} & \dots & \boldsymbol{v}^{(H_p)} \\ \boldsymbol{u}^{(0)} & \dots & \boldsymbol{u}^{(H_p-1)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{X} \\ \boldsymbol{U} \end{pmatrix}, \tag{23}$$

which is the decision variable in the Generalized Unrestricted Program.

Equation (14b) and (14e) are affine and therefore represented with $Ez = b$ (19b). The track constraints (14c) and acceleration constraints (14d) are nonconvex and nonlinear, respectively, and represent the constraint set D in (19c).

Combining the track and acceleration constraint sets yields

$$D = \Big\{ z \in \mathbb{R}^{n\times H_p} \mid \boldsymbol{p}^{(j)} \in T \wedge \boldsymbol{u}^{(j-1)} \in \mathcal{U}_{\boldsymbol{M}}, \forall j = 1, \dots, H_p \Big\}. \tag{24}$$

The objective function remains unchanged as

$$\hat{q}(z, \tilde{z}) = q(z) = -s\left(\left[z_1^{(H_p)}, z_2^{(H_p)}\right]\right) = -s\left(\boldsymbol{p}^{(H_p)}\right), \tag{25}$$

which concludes rewriting the General Trajectory Program as a Generalized Unrestricted Program.

For the transformation of the formulation to a Generalized Restricted Program, we need to construct a convex objective $\hat{q}$ and a restriction function R. The following Section V-D to V-E will formulate the restriction functions $R_{\mathcal{A}}$ and R_T for the acceleration and track constraints, respectively. They will fulfill the conditions for a restriction function (21) on their respective sets T and $\mathcal{A}_{\text{acc}}$. The overall restriction function $R(z)$ is then composed using the Cartesian product as follows:

$$R(z) = R_T\left(\boldsymbol{p}^{(1)}\right) \times \dots \times R_T\left(\boldsymbol{p}^{(H_p)}\right) \times \mathbb{R}^{2\times H_p} \times R_{\mathcal{A}}\left(\boldsymbol{u}^{(0)}\right) \times \dots \times R_{\mathcal{A}}\left(\boldsymbol{u}^{(H_p-1)}\right). \tag{26}$$

Note that this definition corresponds directly to the definition of the unrestricted constraint set D in (24) and the decision variable in (23), i.e., the order of the variables is matched.

We need to justify that the Cartesian product of two restriction functions also is a restriction function. Checking that the conditions (21) are preserved by the Cartesian product shows that this is the case.

D. Restriction Function for Acceleration Input

For the acceleration constraints, we restrict the tire acceleration limit ellipses set (14d) by shrinking the existing polygon representation with the restriction function $R_{\mathcal{A}} : \mathcal{U}_{\text{L}} \to \mathcal{P}(\mathcal{U}_{\text{L}})$

$$R_{\mathcal{A}}(\boldsymbol{u}) = \left\{ \boldsymbol{u} \in \mathcal{U}_{\text{L}} \mid \boldsymbol{A}_{\text{acc}}\left(\boldsymbol{v}^{(j)}\right) \boldsymbol{u} \le b_{\max} \boldsymbol{b}_{\text{acc}}\left(\boldsymbol{v}^{(j)}\right) \right\}, \tag{27}$$

with $b_{\max} \in (0, 1)$. The input constraint for the linear model are $\mathcal{U}_{\text{L,SCR}}^{(j)} = R_{\mathcal{A}}(\boldsymbol{u})$, and as the input constraints for the single-track model are linear, $\mathcal{U}_{\text{ST,SCR}}^{(j)} = \mathcal{U}_{\text{ST}}$.

Algorithm 2: Computation of the Track Polygons.

```
1: $\mathcal{T} \leftarrow$ tessellateTrack($\boldsymbol{T}_L, \boldsymbol{T}_R$)
2: $\mathcal{T} \leftarrow$ mergePolygons($\mathcal{T}$)
3: $\mathcal{T} \leftarrow$ addOverlaps($\mathcal{T}$)
4: **return** $\mathcal{T}$
```

E. Restriction Function for Track Model

In this section, we present the track's restriction function $R_T : T \to \mathcal{P}(T)$ in detail. After defining the function and proofing its validity in Section V-E1, the algorithm to create the restricted track is designed in Section V-E2. Lastly, adaptions to the progress functions are made in Section V-E3.

1) Definition & Proof of the Track Restriction Function: This section defines the structure of the track area $T \subseteq \mathbb{R}^2$ and the restriction function for the track area $R_T : T \to \mathcal{P}(T)$. Because we want to use R_T in the constraints of a QP, it must not just be convex but also consist of affine constraints. Thus,

$$R_T(\tilde{\boldsymbol{p}}) = P_{l(\tilde{\boldsymbol{p}})}, \tag{28a}$$

$$P_l = \{\boldsymbol{p} \in \mathbb{R}^2 \mid F_l \boldsymbol{p} \le \boldsymbol{g}_l\}, \tag{28b}$$

where $P_l \subseteq \mathbb{R}^2$ is a convex polygon, $F_l \in \mathbb{R}^{n_{\text{edges}}\times 2}$ and $\boldsymbol{g}_l \in \mathbb{R}^{n_{\text{edges}}}$ being the constraints defining the polygon P_l with $n_{\text{edges}} \in \mathcal{R}_{\ge 4}$ edges, and $l : \mathbb{R}^2 \to \mathbb{N}$ being an index function which selects the correct set of affine constraints, i.e., the polygon which includes $\boldsymbol{p}$ and is most forward on the track. Geometrically interpreted, the bounded set of linear constraints in two dimensions in (28b) forms a convex polygon. We model the nonconvex track area T with the union of N_{polygons} overlapping convex polygons P_l within the track area as

$$T = \bigcup_{l=1}^{N_{\text{polygons}}} P_l. \tag{29}$$

Theorem 2: The track area function (28a) is a restriction function. □

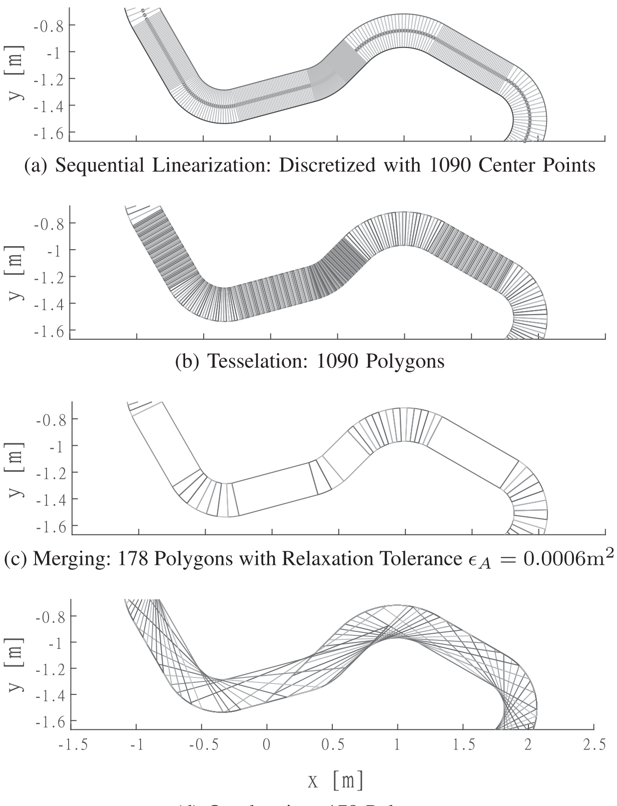
Proof: The track area function fulfills the conditions for a restriction function (21) as shown in Appendix B.

2) Design of the Track Restriction Function: As Section V-E1 demonstrated, a set of convex polygons can be used to define the track restriction function $R_T(\boldsymbol{p})$. The algorithm that determines the polygons should target to maximize

- polygon size, as larger polygons imply less restriction and
- polygon overlap, as the possibility to switch selected polygons for predicted trajectory points $\boldsymbol{p}^{(j)}$ across iterations is required.

Algorithm 2 shows the pseudo-code for the steps of tessellation, merging, and overlapping to compute the track polygons. These three steps are depicted in Fig. 3, starting from SL's discretized track as in Fig. 3(a).

a) Tessellation: With the general track model from Section II-B, let $hull : \mathcal{P}(\mathbb{R}^n) \to \mathcal{P}(\mathbb{R}^n)$ be the convex hull function, yielding the smallest convex super-set of the input set. Then we can define a list of track polygons $\mathcal{T}$ that tessellate the track



(a) Sequential Linearization: Discretized with 1090 Center Points

(b) Tesselation: 1090 Polygons

(c) Merging: 178 Polygons with Relaxation Tolerance $\epsilon_A = 0.0006\mathrm{m}^2$

(d) Overlapping: 178 Polygons

Fig. 3. Track section after each step of Algorithm 2.

as

$$P_i = hull\left(\{\boldsymbol{T}_{L,i}, \boldsymbol{T}_{R,i}, \boldsymbol{T}_{L,i+1}, \boldsymbol{T}_{R,i+1}\}\right), \quad \forall i = 1, \ldots, (N-1), \tag{30a}$$

$$P_N = hull(\{\boldsymbol{T}_{L,N}, \boldsymbol{T}_{R,N}, \boldsymbol{T}_{L,1}, \boldsymbol{T}_{R,1}\}), \tag{30b}$$

$$\mathcal{T} = (P_1, P_2, \ldots, P_N), \tag{30c}$$

where N is the number of track boundary points. Fig. 3(b) shows $\mathcal{T}$ for an example track section.

b) Merging: We iterate over the list of polygons $\mathcal{T}$ and merge polygons P_i and P_{i+1} if their union is convex. The two polygons are replaced with their union in the list and the polygons are renumbered from 1 to $N_{\text{merged}} - 1$. This process is repeated until no polygons can be merged anymore.

To reduce the number of polygons and thus the number of constraints in the program further, we merge polygons if the union $P_i \cup P_{i+1}$ is close to being convex. We define the function $\Delta A : \mathcal{P}(\mathbb{R}^n) \times \mathcal{P}(\mathbb{R}^n) \to \mathbb{R}_{\geq 0}$ which returns the required additional area to make the union of two polygons $P_A \cup P_B$ convex as

$$\Delta A(P_A, P_B) = |hull(P_A \cup P_B)| - |P_A \cup P_B|. \tag{31}$$

From the definition of $hull$ it follows that $\Delta A(P_A, P_B) \geq 0$. For $\Delta A(P_i, P_{i+1}) \leq \varepsilon_A$, where $\varepsilon_A > 0, \varepsilon_A \in \mathbb{R}$ is some small constant, we replace the polygons by their union as before. Note that this represents a minor relaxation.

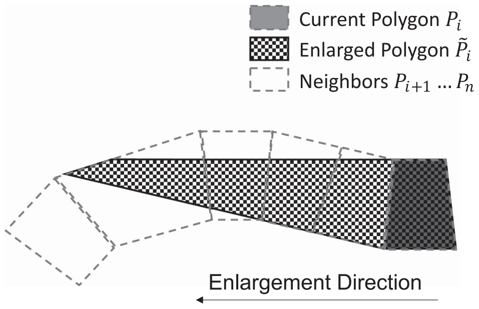

▨ Current Polygon P_i
▩ Enlarged Polygon $\tilde{P}_i$
⬚ Neighbors P_{i+1} ... P_n
Enlargement Direction

Fig. 4. Convex polygon expansion adhering to track limits.

Fig. 3(c) shows the example track's merged polygons, reducing the number of polygons from 1090 to 178 with $\varepsilon_A = 1.35e{-}05\,\mathrm{m}^2$, equaling $\varepsilon_A = 0.025\,\mathrm{m}^2$ on a 1:1-scale.

c) Overlapping: Pursuing our goal of maximizing polygon size and overlap, we want to find the largest convex polygon $\tilde{P}_i$ to replace P_i with

$$\tilde{P}_i \subset T, \tag{32a}$$

$$\tilde{P}_i \supseteq P_i \text{ and} \tag{32b}$$

$$\tilde{P}_i \text{ is convex.} \tag{32c}$$

Every polygon P_i is enlarged in both forward and backward direction. Fig. 4 illustrates the result of the enlargement algorithm in the forward direction. First, we enlarge P_i by moving the points of the edge shared with neighbor P_{i+1} infinitely far along the direction of the polygon edges at the track boundary. This enlarged polygon $\tilde{P}_{\text{forward},i}$ is then restricted by the following neighbors $P_{i+1}, \ldots, P_{\text{last}}$ where P_{last} is the last polygon overlapping with $\tilde{P}_{\text{forward},i}$. By enlarging P_i into the other direction, too, we can union both $\tilde{P}_{\text{forward},i}$ and $\tilde{P}_{\text{backward},i}$ to create $\tilde{P}_i$. This algorithm is detailed in the published code and video. With the slack variable ξ and the index function l we express the track area constraint for SCR as

$$T_{\text{SCR}}^{(j)} = \left\{ \boldsymbol{p} \in \mathbb{R}^2 \mid F_{l(\tilde{\boldsymbol{p}}^{(j)})} \boldsymbol{p} \leq \boldsymbol{g}_{l(\tilde{\boldsymbol{p}}^{(j)})} + \xi. \right\}. \tag{33}$$

3) Progress Function Adaptions: As the SCR track model doesn't allow the usage of the progress function defined in Section II-B2, we need to define $h_{\tilde{z}}(z)$ according to (20a) as

$$h_{\tilde{z}}(z) = -\boldsymbol{F}_{l(\tilde{\boldsymbol{p}}^{(H_p)})}^T \boldsymbol{p}^{(H_p)}, \tag{34}$$

with the discrete center-line forward direction vector $\boldsymbol{F}_i$ of a polygon and l yielding the most forward polygon index containing the point $\tilde{\boldsymbol{p}}^{(H_p)}$. We define $\boldsymbol{F}_i$ as the mean of the forward direction vectors of the polygon's center-line points analogous to (10) as

$$\boldsymbol{F}_i = \frac{1}{|I_i|} \sum_{k \in I_i} \boldsymbol{f}_k, \qquad \forall i = 1, \ldots, N_{\text{polygons}}, \tag{35a}$$

$$I_i = \{k = 1, \ldots, N_{\text{points}} \mid \boldsymbol{T}_{C,k} \in P_i\}, \forall i = 1, \ldots, N_{\text{polygons}}, \tag{35b}$$

where $I_i \subseteq \mathbb{N}$ is the index set of center-line points in the polygon P_i.

F. Complete Trajectory Optimization Problem

Combining the General Trajectory Program (14) with this section's formulations, namely

1) the objective function (34),
2) a slack variable $\xi \in \mathbb{R}_{\geq 0}$ to soften the track constraints (as in the formulation of SL in Section IV),
3) the track constraints (28) and
4) the acceleration constraints (27),

yields the quadratic optimization problem

$$\min_{\boldsymbol{X},\boldsymbol{U},\xi} \quad -\boldsymbol{F}^T_{i\left(\tilde{\boldsymbol{p}}^{(H_p)}\right)}\boldsymbol{p}^{(H_p)} + S\xi + R_\text{u}\sum_{j=2}^{H_p}\left\|\boldsymbol{u}^{(j)} - \boldsymbol{u}^{(j-1)}\right\|_2^2 \tag{36a}$$

subject to

$$\boldsymbol{x}^{(j)} = \boldsymbol{M}\left(\boldsymbol{x}^{(j-1)}, \boldsymbol{u}^{(j-1)}\right), \quad \forall j=1,\ldots,H_p, \tag{36b}$$

$$\boldsymbol{p}^{(j)} \in T^{(j)}_\text{SCR}, \quad \forall j=1,\ldots,H_p, \tag{36c}$$

$$\boldsymbol{u}^{(j)} \in \mathcal{U}^{(j)}_\text{M,SCR} \quad \forall j=0,\ldots,H_p-1, \tag{36d}$$

$$\boldsymbol{v}^{(H_p)} = \boldsymbol{0}, \tag{36e}$$

$$\xi \geq 0. \tag{36f}$$

VI. Numerical Results

This section compares the proposed convex approximation using SCR for autonomous vehicle racing with SL, a widely-spread approach in literature [22]–[28]. A video of the results and the code are linked in the beginning of the paper. The goal in vehicle racing is minimizing the lap time t_lap. Therefore, the following evaluation concentrates on this metric.

Our algorithms run in MATLAB R2021a with the solver `cplexqp` from IBM ILog CPLEX 12.10 [37] on a PC with an AMD Ryzen 5 3600 4.2 GHz hexacore CPU and 16 GB RAM on Windows 10 21H1 64-bit.

The simulation parameters vary between scenarios, but can be found in the code repository which reproduces the simulation results. We tuned the MPC weights Q and R and the trust region size L to achieve a robust planner with minimal lap time on a 1:43-scaled version of the Hockenheimring. In order to achieve real-time performance with the sampling time $\Delta t = 0.1\,\text{s}$, we set the number of iterations N_RTI to 1.

A. Linear and Single-Track Model

We equip a single vehicle with two trajectory planners, one using the linear model [27] and one using the single-track model. The constraints on the single-track model determine the acceleration bounds for the linear model.

Fig. 5 shows the difference in the predicted trajectories. When compared to the linear model, the single-track model achieves trajectories closer to racing lines, as it uses the entire track area to maintain speed. The prediction of the single-track model

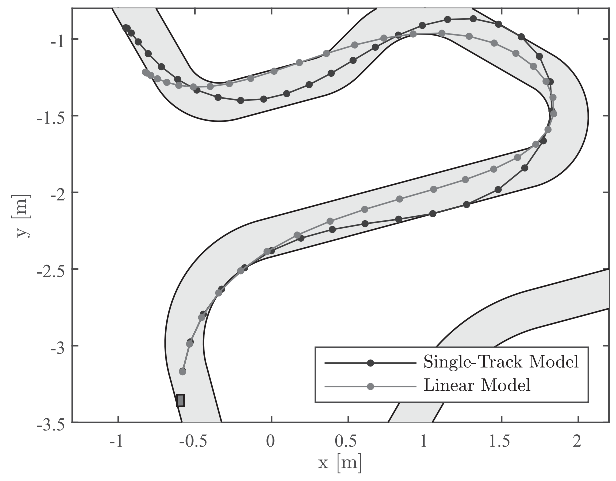


Fig. 5. Predicted MPC trajectories of linear and single-track vehicle model.

TABLE I
Lap Times of Track Discretization Methods

Measurement	$t_\text{lap,start}$ [s]	$t_\text{lap,flying}$ [s]
SL	11.0	10.0
SCR	10.1	9.3
Difference	0.9	0.7
Difference in [%]	8.91	7.53

progresses further on the track in the same time interval, which indicates a faster progress. In the following evaluation, we will use the single-track model only.

B. Track Convexification With SCR

Table I shows the lap times for both track convexification methods. The relative decrease for $t_\text{lap,start}$ is higher than for $t_\text{lap,flying}$. At the start of the race, the trajector planner is initialized with a trajectory prediction corresponding to a standstill situation. The trust region that restricts SL is particularly prohibitive at the first iterations after the start. If we increase the number of iterations N_RTI in SL to 50, the resulting lap times are still worse than with a single iteration of SCR, while the computation time is up to ten times higher.

SL approximates the track less accurately than SCR. Since SL relaxes the track constraints, the solutions obtained do not always satisfy the original nonconvex constraints. In contrast, SCR guarantees satisfying the track constraints, since the convexification is a restriction of the nonconvex constraints. Fig. 6 shows an example of such a situation. The dotted line depicts the solution of the previous timestep, which serves as the operating point. The convex constraints for SCR and SL are shown in dark and light shades, respectively.

Fig. 7 shows the trajectories a vehicle takes around the track with both the SL and the SCR convexification method. A marker signals every second on the trajectory to allow a comparison of the vehicle's progression. The vehicle starts with both the initial state and the prediction from a flying lap. The markers show

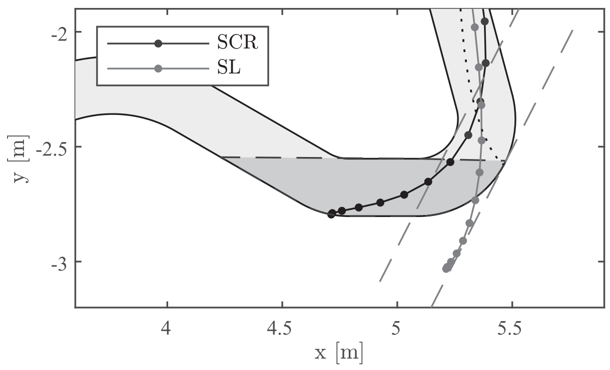


Fig. 6. Track convexification with SL and SCR and resulting planned trajectories. The trajectory resulting from the SL convexification violates the original nonconvex constraints. The dotted line shows the trajectory of the previous timestep around which the constraints are convexified.

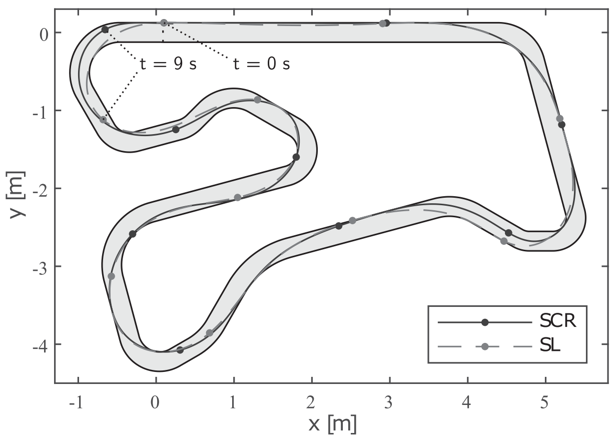


Fig. 7. Trajectories with SL and SCR; a marker is set every second. The race starts in the upper left and continues clockwise.

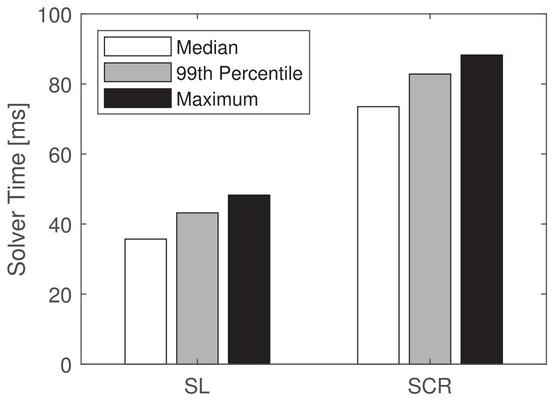


Fig. 8. Computation time of SL and SCR.

how the trajectory resulting from SCR progresses faster on the track than the one from SL.

Fig. 8 shows the median, 99th percentile, and maximum computation time for SL and SCR. In all three categories, SCR has approximately double the computation time than SL. The computation time is mainly determined by the optimization time, which grows with the number of constraints. The QP in SCR is subject to a higher number of constraints given by the polygonal representation of the track. The number of constraints can be reduced by a coarser representation of the track, i.e., using polygons with fewer sides. The times stay consistent, i.e., the maximum and 99th percentile execution times are close to the median, which is a desired property for real-time applications.

VII. Conclusion

This paper introduced a general problem formulation for trajectory planning of autonomous race cars. We showed an adaption of the general problem formulation for the convexification method SL, which is state of the art, and introduced an adaption for SCR. We showed that we achieve a valid restriction of the track with overlapping, convex polygons. The paper also proves recursive feasibility of trajectories resulting from the restricted optimization problem. Solutions found with SCR always satisfy the nonconvex constraints of the original problem in contrast to SL. We showed that SCR improves lap times compared to the state-of-the-art method SL, while still being real-time capable. Opponents and obstacles can be incorporated as additional nonconvex constraints in the optimization problem. SCR can be extended by linearizing these constraints as in [28] to form a convex restriction of the original problem.

Future work will investigate real experiments within the Cyber-Physical Mobility Lab, an open-source platform for networked and autonomous vehicles [38].

Appendix A
Proof of Theorem 1

Proof: <u>Constraints (14b) for $j = 1$:</u>

$$\boldsymbol{x}^{(1)} = \boldsymbol{A}\boldsymbol{x}^{(0)} + \boldsymbol{B}\boldsymbol{u}^{(0)} \tag{37a}$$

$$\tilde{\boldsymbol{x}}^{(2)} = \boldsymbol{A}\tilde{\boldsymbol{x}}^{(1)} + \boldsymbol{B}\tilde{\boldsymbol{u}}^{(1)} \tag{37b}$$

Subtracting (37a) and (37b) yields

$$\underbrace{\boldsymbol{x}^{(1)} - \tilde{\boldsymbol{x}}^{(2)}}_{=0\text{ (22a)}} = \boldsymbol{A}\Big(\boldsymbol{x}^{(0)} - \tilde{\boldsymbol{x}}^{(1)}\Big) + \boldsymbol{B}\underbrace{\Big(\boldsymbol{u}^{(0)} - \tilde{\boldsymbol{u}}^{(1)}\Big)}_{=0\text{ (22b)}} \tag{37c}$$

$$\iff \boldsymbol{x}^{(0)} = \tilde{\boldsymbol{x}}^{(1)}, \tag{37d}$$

where (37d) states that SCR assumes the vehicle following the predicted trajectory without disturbances.

<u>Constraints (14b) for $j = 2, \ldots, H_p - 1$:</u>

$$\boldsymbol{x}^{(j)} = \boldsymbol{A}\boldsymbol{x}^{(j-1)} + \boldsymbol{B}\boldsymbol{u}^{(j-1)} \tag{37e}$$

$$\tilde{\boldsymbol{x}}^{(j+1)} = \boldsymbol{A}\tilde{\boldsymbol{x}}^{(j)} + \boldsymbol{B}\tilde{\boldsymbol{u}}^{(j)} \tag{37f}$$

Subtracting (37e) and (37f) yields

$$\underbrace{\boldsymbol{x}^{(j)} - \tilde{\boldsymbol{x}}^{(j+1)}}_{=0\text{ (22a)}} = \boldsymbol{A}\underbrace{\Big(\boldsymbol{x}^{(j-1)} - \tilde{\boldsymbol{x}}^{(j)}\Big)}_{=0\text{ (22a)}} + \boldsymbol{B}\underbrace{\Big(\boldsymbol{u}^{(j-1)} - \tilde{\boldsymbol{u}}^{(j)}\Big)}_{=0\text{ (22b)}}. \tag{37g}$$

Constraints (14b) for $j = H_p$:

$$\boldsymbol{x}^{(H_p)} = \boldsymbol{A}\boldsymbol{x}^{(H_p-1)} + \boldsymbol{B}\boldsymbol{u}^{(H_p-1)} \tag{37h}$$

$$\overset{\text{(22a) and (22d)}}{\iff} \tilde{\boldsymbol{x}}^{(H_p)} = \boldsymbol{A}\tilde{\boldsymbol{x}}^{(H_p)} \tag{37i}$$

$$\iff \boldsymbol{0} = (\boldsymbol{A} - I)\tilde{\boldsymbol{x}}^{(H_p)} \tag{37j}$$

$$\overset{\text{(5) and (14e)}}{\iff} \boldsymbol{0} = \begin{bmatrix} 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{p}_x^{(H_p)} \\ \tilde{p}_y^{(H_p)} \\ 0 \\ 0 \end{bmatrix} \tag{37k}$$

Constraints (14c) for $j = 1, \ldots, H_p - 1$:

$$\boldsymbol{p}^{(j)} \in T \overset{\text{(22a)}}{\iff} \tilde{\boldsymbol{p}}^{(j+1)} \in T \tag{37l}$$

Constraints (14c) for $j = H_p$:

$$\boldsymbol{p}^{(H_p)} \in T \overset{\text{(22c)}}{\iff} \tilde{\boldsymbol{p}}^{(H_p)} \in T \tag{37m}$$

Constraints (14d) for $j = 0, \ldots, (H_p - 2)$:

$$\boldsymbol{A}_{\text{acc}}\left(\boldsymbol{v}^{(j)}\right)\boldsymbol{u}^{(j)} \le \boldsymbol{b}_{\text{acc}}\left(\boldsymbol{v}^{(j)}\right)$$

$$\overset{\text{(22b)}}{\iff} \boldsymbol{A}_{\text{acc}}\left(\tilde{\boldsymbol{v}}^{(j+1)}\right)\tilde{\boldsymbol{u}}^{(j+1)} \le \boldsymbol{b}_{\text{acc}}\left(\boldsymbol{v}^{(j)}\right) \tag{37n}$$

Constraints (14d) for $j = H_p - 1$:

$$\boldsymbol{A}_{\text{acc}}\left(\boldsymbol{v}^{(H_p-1)}\right)\boldsymbol{u}^{(H_p-1)} \le \boldsymbol{b}_{\text{acc}}\left(\boldsymbol{v}^{(H_p-1)}\right)$$

$$\overset{\text{(22d)}}{\iff} \boldsymbol{0} \le \boldsymbol{b}_{\text{acc}}\left(\boldsymbol{v}^{(H_p-1)}\right) \tag{37o}$$

Constraints (14e):

$$\boldsymbol{v}^{(H_p)} = \boldsymbol{0} \overset{\text{(22c)}}{\iff} \tilde{\boldsymbol{v}}^{(H_p)} = \boldsymbol{0} \tag{37p}$$

We have proven for every constraint and every timestep that the initial trajectory (22a) and (22d) is feasible in the General Trajectory Program (14) for the linear vehicle model (5). □

Appendix B
Proof of Theorem 2

Proof: Condition (21a):

$$R_T(\tilde{\boldsymbol{p}}) \subseteq T \qquad \forall \tilde{\boldsymbol{p}} \in T \tag{38a}$$

$$\overset{\text{(28a) and (29)}}{\iff} P_{l(\tilde{\boldsymbol{p}})} \subseteq \bigcup_{l=1}^{N_{\text{polygons}}} P_l \qquad \forall \tilde{\boldsymbol{p}} \in T \tag{38b}$$

$$\iff P_k \subseteq \bigcup_{l=1}^{N_{\text{polygons}}} P_l \quad \forall k = 1, \ldots, N_{\text{polygons}}. \tag{38c}$$

Condition (21b):

$$R_T(\tilde{\boldsymbol{p}}) \text{ is convex } \quad \forall \tilde{\boldsymbol{p}} \in T \tag{38d}$$

$$\overset{\text{(28a)}}{\iff} P_{l(\tilde{\boldsymbol{p}})} \text{ is convex } \quad \forall \tilde{\boldsymbol{p}} \in T \tag{38e}$$

$$\iff P_k \text{ is convex } \quad \forall k = 1, \ldots, N \tag{38f}$$

$$\overset{\text{(28b)}}{\iff} \{\boldsymbol{p} \in \mathbb{R}^2 \mid F_k \boldsymbol{p} \le \boldsymbol{g}_k\} \text{ is convex } \quad \forall k = 1, \ldots, N. \tag{38g}$$

Condition (21c):

$$\tilde{\boldsymbol{p}} \in R_T(\tilde{\boldsymbol{p}}) \quad \forall \tilde{\boldsymbol{p}} \in T \tag{38h}$$

$$\overset{\text{(28a)}}{\iff} \tilde{\boldsymbol{p}} \in P_{l(\tilde{\boldsymbol{p}})} \quad \forall \tilde{\boldsymbol{p}} \in T \tag{38i}$$

$$\overset{\text{(28b)}}{\iff} \tilde{\boldsymbol{p}} \in \{\boldsymbol{p} \in \mathbb{R}^2 \mid F_{l(\tilde{\boldsymbol{p}})}\boldsymbol{p} \le \boldsymbol{g}_{l(\tilde{\boldsymbol{p}})}\} \quad \forall \tilde{\boldsymbol{p}} \in T \tag{38j}$$

$$\iff F_{l(\tilde{\boldsymbol{p}})}\tilde{\boldsymbol{p}} \le \boldsymbol{g}_{l(\tilde{\boldsymbol{p}})} \quad \forall \tilde{\boldsymbol{p}} \in T \tag{38k}$$

$$\iff d_{l(\tilde{\boldsymbol{p}})}(\tilde{\boldsymbol{p}}) \le 0 \quad \forall \tilde{\boldsymbol{p}} \in T \tag{38l}$$

$$\iff \left(\min_{k=1,\ldots,N} d_k(\tilde{\boldsymbol{p}})\right) \le 0 \quad \forall \tilde{\boldsymbol{p}} \in T.$$

With $\tilde{\boldsymbol{p}} \in T \text{follows} \exists n : \tilde{\boldsymbol{p}} \in P_n \text{and} d_n(\tilde{\boldsymbol{p}}) \le 0$, thus

$$\left(\min_{k=1,\ldots,N} d_k(\tilde{\boldsymbol{p}})\right) \le d_n(\tilde{\boldsymbol{p}}) \le 0 \quad \forall \tilde{\boldsymbol{p}} \in T. \tag{38m}$$

We have proven that the track restriction function $R_T(\boldsymbol{p})$ is a valid restriction function by checking the conditions (21). □

References

[1] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.; New York, NY, USA: Cambridge Univ. Press, 2004.

[2] T. Lipp and S. Boyd, “Variations and extension of the convex concave procedure,” *Optim. Eng.*, vol. 17, no. 2, pp. 263–287, Jun. 2016.

[3] S. Boyd, “Sequential convex programming,” Stanford Univ. Press, Stanford, CA, USA, Tech. Rep., 2015. [Online]. Available: https://web.stanford.edu/class/ee364b/lectures/seq_notes.pdf

[4] J. Nocedal and S. J. Wright, *Numerical Optimization Ser. Springer Series in Operations Research*, 2nd ed. New York, NY, USA: Springer, 2006.

[5] B. Alrifaee, J. Maczijewski, and D. Abel, “Sequential convex programming MPC for dynamic vehicle collision avoidance,” in *Proc. IEEE Conf. Control Technol. Appl.*, 2017, pp. 2202–2207.

[6] J. Betz *et al.*, “Autonomous vehicles on the edge: A survey on autonomous vehicle racing,” Feb. 2022, *arXiv:2202.07008*.

[7] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles,” *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[8] L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma, and N. Zheng, “Efficient sampling-based motion planning for on-road autonomous driving,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 1961–1976, Aug. 2015.

[9] J. Betz *et al.*, “A software architecture for the dynamic path planning of an autonomous Racecar at the limits of handling,” in *Proc. IEEE Int. Conf. Connected Veh. Expo*, 2019, pp. 1–8.

[10] P. Scheffe, M. V. d. A. Pedrosa, K. Flaßkamp, and B. Alrifaee, “Receding horizon control using graph search for multi-agent trajectory planning,” TechRxiv, 2021, doi: 10.36227/techrxiv.16621963.v1.

[11] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, “Aggressive driving with model predictive path integral control,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1433–1440.

[12] A. Rucco, G. Notarstefano, and J. Hauser, “An efficient minimum-time trajectory generation strategy for two-track car vehicles,” *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 4, pp. 1505–1519, Jul. 2015.

[13] D. Caporale *et al.*, “Towards the design of robotic drivers for full-scale self-driving racing cars,” in *Proc. Int. Conf. Robot. Automat.*, Montreal, QC, Canada, 2019, pp. 5643–5649.

[14] A. Heilmeier, A. Wischnewski, L. Hermansdorfer, J. Betz, M. Lienkamp, and B. Lohmann, “Minimum curvature trajectory planning and control for an autonomous race car,” *Veh. Syst. Dyn.*, vol. 58, no. 10, pp. 1497–1527, Oct. 2020.

[15] D. Kloeser, T. Schoels, T. Sartor, A. Zanelli, G. Prison, and M. Diehl, “NMPC for racing using a singularity-free path-parametric model with obstacle avoidance,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14324–14329, Jan. 2020.

[16] R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl, “Time-optimal race car driving using an online exact hessian based nonlinear MPC algorithm,” in *Proc. Eur. Control Conf.*, 2016, pp. 141–147.

[17] E. Pagot, M. Piccinini, and F. Biral, “Real-time optimal control of an autonomous RC car with minimum-time maneuvers and a novel kineto-dynamical model,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 2390–2396.

[18] D. P. Kelly and R. S. Sharp, “Time-optimal control of the race car: A numerical method to emulate the ideal driver,” *Veh. Syst. Dyn.*, vol. 48, no. 12, pp. 1461–1474, Dec. 2010.

[19] R. Reiter, M. Kirchengast, D. Watzenig, and M. Diehl, “Mixed-integer optimization-based planning for autonomous racing with obstacles and rewards,” *IFAC-PapersOnLine*, vol. 54, no. 6, pp. 99–106, Jan. 2021.

[20] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, “FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs,” *Int. J. Control*, vol. 93, no. 1, pp. 13–29, Jan. 2020.

[21] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Math. Program.*, vol. 106, no. 1, pp. 25–57, Mar. 2006.

[22] D. Q. Tran and M. Diehl, “An application of sequential convex programming to time optimal trajectory planning for a car motion,” in *Proc. 48th IEEE Conf. Decis. Control, Held Jointly 28th Chin. Control Conf.*, 2009, pp. 4366–4371.

[23] N. R. Kapania, J. Subosits, and J. Christian Gerdes, “A sequential two-step algorithm for fast generation of vehicle racing trajectories,” *J. Dyn. Syst., Meas., Control*, vol. 138, no. 9, Jun. 2016, Art. no. 091005.

[24] J. P. Timings and D. J. Cole, “Vehicle trajectory linearisation to enable efficient optimisation of the constant speed racing line,” *Veh. Syst. Dyn.*, vol. 50, no. 6, pp. 883–901, Jun. 2012.

[25] A. Liniger, A. Domahidi, and M. Morari, “Optimization-based autonomous racing of 1:43 scale RC cars,” *Optimal Control Appl. Methods*, vol. 36, no. 5, pp. 628–647, Sep. 2015.

[26] S. Srinivasan, S. Nicolas Giles, and A. Liniger, “A holistic motion planning and control solution to challenge a professional racecar driver,” *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 7854–7860, Oct. 2021.

[27] B. Alrifaee and J. Maczijewski, “Real-time trajectory optimization for autonomous vehicle racing using sequential linearization,” in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2018, pp. 476–483.

[28] M. Kloock, P. Scheffe, L. Botz, J. Maczijewski, B. Alrifaee, and S. Kowalewski, “Networked model predictive vehicle race control,” in *Proc. IEEE Intell. Transp. Syst. Conf.*, 2019, pp. 1552–1557.

[29] H. B. Pacejka, *Tyre and Vehicle Dynamics*, 2nd ed. Oxford, U.K.: Butterworth-Heinemann, 2006.

[30] A. Liniger, “Path planning and control for autonomous racing,” Ph.D. dissertation, ETH Zurich, Zürich, Switzerland, 2018.

[31] C. Voser, R. Y. Hindiyeh, and J. C. Gerdes, “Analysis and control of high sideslip manoeuvres,” *Veh. Syst. Dyn.*, vol. 48, no. sup1, pp. 317–336, Dec. 2010.

[32] B. Heißing, M. Ersoy, and S. Gies Eds., *Fahrwerkhandbuch: Grundlagen, Fahrdynamik, Komponenten, Systeme, Mechatronik, Perspektiven; mit 84 Tabellenmit, Ser. ATZ-MTZ Fachbuch*, 3rd ed. Wiesbaden, Germany: Vieweg Teubner, 2011.

[33] T. Fork, H. E. Tseng, and F. Borrelli, “Models and predictive control for nonplanar vehicle navigation,” in *Proc. IEEE Int. Intell. Transp. Syst. Conf.*, 2021, pp. 749–754.

[34] J. M. Maciejowski, *Predictive Control: With Constraints*. Harlow, England; New York, NY, USA: Prentice-Hall, 2002.

[35] T. A. Rajasingham, *Nonlinear Model Predictive Control of Combustion Engines: From Fundamentals to Applications, Ser. Advances in Industrial Control*. Cham, Switzerland: Springer, 2021.

[36] C. A. Floudas, P. M. Pardalos, P. Pardalos, and R. Horst, Eds., *State of the Art in Global Optimization, Ser. Nonconvex Optimization and Its Applications*, vol. 7. Boston, MA, USA: Springer US, 1996.

[37] IBM, “IBM ILOG CPLEX 12.10 User Manual,” Dec. 2019.

[38] M. Kloock *et al.*, “Cyber-physical mobility lab: An open-source platform for networked and autonomous vehicles,” in *Proc. Eur. Control Conf.*, 2021, pp. 1937–1944.

Patrick Scheffe received the B.Sc. degree in mechanical engineering and the M.Sc. degree in automation engineering in 2016 and in 2019, respectively, from RWTH Aachen University, Aachen, Germany, where he is currently working toward the Ph.D. degree in computer science with the Chair of Embedded Software. His research interests include distributed decision-making for cyber-physical systems and its application to connected and autonomous vehicles.

Theodor Mario Henneken received the B.Sc. degree in mechanical engineering from RWTH Aachen University, Germany, in 2019, and the M.Sc. degrees in automation engineering and industrial engineering from RWTH Aachen University and Tsinghua University, Beijing, China, in 2021. He is currently working toward the dual Master of Business Administration degrees with the Collège des Ingénieurs, Paris, France.

Maximilian Kloock received the B.Sc. and M.Sc. degrees in computer science in 2016 in 2018, respectively, from RWTH Aachen University, Aachen, Germany, where he is currently working toward the Ph.D. degree in computer science with the Chair of Embedded Software. His research interests include distributed decision-making and verification for cyber-physical systems and their applications to connected and autonomous vehicles.

Bassam Alrifaee is currently a Junior Principal Investigator with the Chair of Embedded Software, RWTH Aachen University, Aachen, Germany. He is the Founder of the Cyber-Physical Mobility Group and Lab. His research interests include distributed decision-making and verification, service-oriented software architecture for control systems, and their applications to connected and autonomous vehicles. He is also the Principal Investigator of several projects.