Sequential Convex Programming Methods for Real-time Trajectory Optimization in Autonomous Vehicle Racing

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

Abstract—We present a real-time-capable *Model Predictive Controller* (MPC) based on a single-track vehicle model and Pacejka’s magic tire formula for autonomous racing applications. After formulating the general non-convex trajectory optimization problem, the model is linearized around estimated operating points and the constraints are convexified using the *Sequential Convex Programming* (SCP) method. We use two different methods to convexify the non-convex track constraints, namely *Sequential Linearization* (SL) and *Sequential Convex Restriction* (SCR). SL, a method of relaxing the constraints, was introduced in our previous paper. SCR, a method of restricting the constraints, is introduced in this paper. We show the application of SCR to autonomous racing and prove that it does not interfere with recursive feasibility. We compare the predicted trajectory quality for the nonlinear single-track model to the linear double integrator model from our previous paper. The MPC performance is evaluated on a scaled version of the Hockenheimring racing track. We show that an MPC with SCR yields faster lap times than an MPC with SL – for race starts as well as flying laps – while still being real-time capable. A video showing the results is available at <https://youtu.be/2iETsolCNQ>.

Index Terms—Autonomous Vehicles; Control and Optimization; Vehicle Control; Motion Planning

I. INTRODUCTION

FOR autonomous racing, trajectory planning and trajectory tracking is required. In this paper, we implement *Sequential Convex Programming* (SCP) methods to achieve fast lap times while adhering to the vehicle’s physical abilities and track boundaries. This task can be categorized as a kinodynamic planning problem, satisfying both kinematic and dynamic constraints.

In the greater task of automated driving of sensing, planning and acting, the presented approach has direct application in form of the planning component.

A. Related Work

In the field of online autonomous racing control, Model Predictive Control (MPC) with a non-linear vehicle and tire model is often used. This results in a non-linear optimal problem formulation, which can be solved by linearization.

In the past, problems were described as a steering optimization problem to follow a given trajectory, e.g., by [1]. In more recent years, the optimization task was extended to include path planning, e.g., [2]–[4]. The authors of [5] show the advantage of holistically co-designing the individual controllers,

claiming to reach equal performance to professional drivers under certain circumstances.

Many papers showed real-world implementations of their formulated problems, e.g., [3], [6]. The authors of [2] showed the implementation of stochastic optimal control on GPUs for a rally car on loose ground thus operating within friction limits of the tires. Publications around *Roborace*, an autonomous robotic race series with full-scale vehicles, used the same setup of non-linear MPC and vehicle models [7]. Others proposed to use a graph search instead of MPC due to their tendency to find globally optimal solutions [8].

In [9], a setup of non-linear vehicle models and MPC was used but with second-order derivatives instead of linearization. Additionally, the proposed online MPC of the authors of [9] was compared to a racing trajectory as derived from iterative optimal control. Another approach is to model racing as a periodically repetitive task and use a learning MPC by [10]. However, this constricts the usage to a race without opponents and requires prior offline computation.

Recently, [6] proposed an approach using a kineto-dynamic model, determining parameters by experiments. Due to the low computation time, a long horizon can be used, resulting in close to optimal trajectories. In contrast to other publications, the goal of optimization of [6] is the minimization of lap time instead of maximization of track progress.

The current literature is using approximations such as linearizations or graph searches to model the track. Instead, a more accurate restriction could be used.

B. Addressed Problem & Contribution

We focus on the goal of developing a general formulation of trajectory planning for racing. Offline trajectory computation is not possible due to changing environments. To adhere to safety standards, stability and recursive feasibility of the controller are required. The driving situations at the vehicle’s physical limits require a high controller update rate and real-time capable execution times. On the other hand, the planning horizon should be large to cover the entire length of subsequent curves, enabling us to find close to optimal solutions. Both demands in turn require a low computational cost.

Following these requirements, we propose a planning method using a single-track model and *Pacejka’s Magic Tire Formula* for the vehicle dynamics. We construct an optimization problem as an MPC. The modeling results in a non-convex optimization problem. To leverage readily available

solvers, we approximate the problem as a linearly constrained convex *Quadratic Optimization Problem* (QP). SCP describes the required iterative convex approximation of a non-convex problem. SCP is a good approximation but doesn't guarantee a solution [11], [12].

The key contribution of this paper lies in the conversion of the application-specific problem formulation to a QP using two different SCP schemes, namely *Sequential Linearization* (SL) and *Sequential Convex Restriction* (SCR). The conducted parameter study can be used to

- 1) validate the SL method with a more complex single-track vehicle and tire model in comparison to the author's previously proposed linear model [13].
- 2) compare both methods with regarding the computation time and quality of solution.

C. Structure of this Article

We structure the remainder of this article into three main parts. Firstly, we construct the general optimization problem in a non-convex way in Section II. Secondly, the problem is convexified using the SL and SCR methods in Section III, IV and V. This includes modification of the general problem to match the individual methods. Thirdly, the formulated optimization problems are evaluated in Section VI with a parameter study. Lastly, the paper's conclusion is drawn in Section VII.

II. GENERAL FORMULATION OF TRAJECTORY OPTIMIZATION PROBLEM

In this section, we formulate the trajectory optimization problem in a general manner, before adapting it in the later sections to the SCP methods SL and SCR. We assume stable environmental conditions such as weather and road surface. We assume the current states, e.g., pose, velocity, acceleration, and yaw rate, to be known through sensors or being estimated correctly.

In the following, we introduce the vehicle models including the input constraints in Section II-A, the model of the track on which is driven in Section II-B and the MPC in Section II-C each before combining them into a complete trajectory optimization problem in Section II-D.

We will denote the temporal dependency with a parenthesized index in a superscript, e.g., $\mathbf{x}^{(j)}$.

A. Vehicle Models

To compare the our previous method SL [14] with our new method SCR, we implement a vehicle model suited for racing applications. Thus, we use a single-track model of an RC car, from which we derive the parameters for the previous paper's linear model. We present both models below for completeness, the detailed modeling process was 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

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

where the model $\mathbf{M}$ computes the state $\mathbf{x}$ at time step j based on the previous time step's state $\mathbf{x}^{(j-1)}$ and current input $\mathbf{u}^{(j)}$.

1) *Single-Track Model*: A kinetic single-track model is combined with *Pacejka's Magic Tire Formula* [15]. Rolling and air resistance are considered, as well as the fixed-gear electric motor's characteristic curve of max torque and limited power. Racing conditions such as oversteer, understeer, and drifting can be represented by this model. 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 [16]. Thus, only the rear wheel needs consideration of combined tire forces, which is done implicitly via the lateral tire parameters [3], [17]. The detailed construction including parameters of the entire model was given in [3]. 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 input consists of the steering angle $\delta \in (-0.35\text{rad}, 0.35\text{rad})$ and the electric motor's duty cycle $\tau \in (10\%, 100\%)$.

The model from [3] contains the states of position p from the enter of gravity to the reference frame and velocity v in directions x and y . Additionally, the heading angle ϕ and the accompanying yaw rate ω are used. The differential equations which will be used in discretized form are

$$\dot{p}_x = v_x \cos(\phi) - v_y \sin(\phi) \quad (2a)$$

$$\dot{p}_y = v_x \sin(\phi) + v_y \cos(\phi) \quad (2b)$$

$$\dot{v}_x = \frac{1}{m} (F_{r,x} - F_{f,y} \sin(\delta) + m v_y \omega) \quad (2c)$$

$$\dot{v}_y = \frac{1}{m} (F_{r,y} + F_{f,y} \cos(\delta) - m v_x \omega) \quad (2d)$$

$$\dot{\phi} = \omega \quad (2e)$$

$$\dot{\omega} = \frac{1}{I_z} (F_{f,y} l_f \cos(\delta) - F_{f,y} l_r), \quad (2f)$$

where the vehicle parameters are 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

$$F_{r,x} = (C_{m1} - C_{m2} v_x) \tau - C_{r0} - C_{r2} v_x^2 \quad (3a)$$

$$F_{f,y} = D_f \sin(C_f \arctan(B_f \alpha_f)), \quad (3b)$$

$$\alpha_f = -\arctan\left(\frac{\omega l_f + v_y}{v_x}\right) + \delta \quad (3c)$$

$$F_{r,y} = D_r \sin(C_r \arctan(B_r \alpha_r)), \quad (3d)$$

$$\alpha_r = \arctan\left(\frac{\omega l_r - v_y}{v_x}\right), \quad (3e)$$

where B_f , B_r , C_f , C_r , D_f and D_r are empiric tire formula coefficients and α the tire slip angles for front and rear, respectively. Singularities induced by the slip angle definitions are mitigated by replacing v_x with a small number in case of $v_x = 0$. The electric motor is modeled with the parameters C_m , while the resistance parameters C_r are empirically determined.

2) *Linear Model*: The linear model was introduced in our previous paper [13]. It is a double-integrator model propagating the inputs of longitudinal and lateral accelerations a_x and

a_y to the vehicle's velocity and position. No yaw is considered, the vehicle is always facing in the same direction as the longitudinal velocity. As a simplification, the tires' acceleration limits for lateral and longitudinal slip are described by two half ellipses [18]. The limits are velocity-dependent, symmetrical for lateral direction but due to resistances asymmetric for longitudinal direction.

To assure shared parameters and comparability with the previous II-A1's single-track model, the input bounds are derived with the single-track model. The single-track model's input bounds are kept enforced, which results in implicit incorporation in the linear model limits. In the case of longitudinal maximum acceleration determination, the tire force only depends on the input duty cycle τ and velocity. Thus, applying maximal and minimal τ yields maximal forward and backward acceleration at any v_x , respectively. In the case of lateral maximum acceleration determination, the tire force depends on the slip angle. By using steady-state cornering at varying velocities the maximum lateral acceleration can be determined.

The model from [13] with the discretization time Δt is

$$\begin{bmatrix} p_x^{(j)} \\ p_y^{(j)} \\ v_x^{(j)} \\ v_y^{(j)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x^{(j-1)} \\ p_y^{(j-1)} \\ v_x^{(j-1)} \\ v_y^{(j-1)} \end{bmatrix} + \begin{bmatrix} \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{\Delta t^2}{2} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} a_x^{(j)} \\ a_y^{(j)} \end{bmatrix} \quad (4a)$$

$$\iff \mathbf{x}^{(j)} = \mathbf{A}\mathbf{x}^{(j-1)} + \mathbf{B}\mathbf{u}^{(j)} \quad (4b)$$

with $\mathbf{p} = (p_x \ p_y)^T$, $\mathbf{v} = (v_x \ v_y)^T$ and $\mathbf{a} = (a_x \ a_y)^T$ being the position, velocity and acceleration in directions x and y , respectively.

We use $n_{\text{acc}} \in \mathbb{N}$ evenly distributed tangents to approximate the velocity-dependent tire acceleration limit ellipses. This results in a relaxation, vanishing with $n \rightarrow \infty$ at higher computational costs. The resulting set of linear inequalities is

$$\mathbf{A}_{\text{acc}} \left(\mathbf{v}, \frac{2\pi k}{n_{\text{acc}}} \right) \mathbf{u} \leq \mathbf{b}_{\text{acc}}(\mathbf{v}) \quad \forall k = 1, \dots, n_{\text{acc}},$$

$$\mathbf{A}_{\text{acc}} : \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}^2, \mathbf{b}_{\text{acc}} : \mathbb{R}^2 \rightarrow \mathbb{R}, \quad (5)$$

with $\mathbf{v} = (v_x \ v_y)^T$ being the vehicle's velocity and $\mathbf{u} = (a_x \ a_y)^T$ the input. The matrix $\mathbf{A}_{\text{acc}}$ and vector $\mathbf{b}_{\text{acc}}$ are computed such that $\mathbf{u}$ is constrained to the ellipses.

B. Track Model

The track circuit represents the allowable driving area of the vehicle. We assume the entire track has a constant width. Also, we assume curve radii to be larger than the track width. A confinement to planar models is sufficient for most paths, in the other cases it yields only limited error compared to non-planar models used with a Model Predictive Controller [19]. In the following, the track's discrete model and a function to measure a vehicle's progression on the track is introduced.

Fig. 1. Example of race discrete track model, being a non-convex part [13]

1) *Discretized Track Model*: We express the condition of the vehicle's position being on the track as $\mathbf{p} \in T$, with $\mathbf{p}$ being the vehicle's position and $T \subseteq \mathbb{R}^2$ being the track area. To define T , we start with a center line $C : [0, S] \rightarrow T$ with $S \in \mathbb{R}$ being the track length and $s \in [0, S]$ being the track progress. The discretized track points are constructed with the discretization size Δs_i where the subscript $i \in \mathbb{N}$ denotes the discrete track point. The forward vector $\mathbf{f}_i$ is defined by the gradient of the center-line C . The left and right boundary points $\mathbf{T}_{L,i}$ and $\mathbf{T}_{R,i}$ are constructed by translating the center point $\mathbf{T}_{C,i}$ along the normal direction $\mathbf{n}_i$ with the track width $w \in \mathbb{R}_{>0}$. Fig. 1 depicts this discrete track definition, the accompanying formulas are

$$s_i = i \cdot \Delta s_i \quad (6a)$$

$$\mathbf{f}_i = \nabla C(s_i) \quad (6b)$$

$$\mathbf{n}_i = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \mathbf{f}_i \quad (6c)$$

$$\mathbf{T}_{C,i} = C(s_i), \quad \mathbf{T}_{C,i} \in T, \quad (6d)$$

$$\mathbf{T}_{L,i} = C(s_i) + \frac{w}{2} \mathbf{n}_i, \quad \mathbf{T}_{L,i} \in T, \quad (6e)$$

$$\mathbf{T}_{R,i} = C(s_i) - \frac{w}{2} \mathbf{n}_i, \quad \mathbf{T}_{R,i} \in T. \quad (6f)$$

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 C , modeled by the progress function

$$s(\mathbf{p}) = \arg \min_{c \in [0, S]} \|C(c) - \mathbf{p}\|_2, \quad (7)$$

with $s \in \mathbb{R}$.

To be usable as part of the objective function, we linearize and discretize the progress function, starting with the linear interpolation of the center-line

$$C(s) \approx C(s_i) + \nabla C(s_i)(s - s_i) \quad (8a)$$

$$= \mathbf{T}_{C,i} + \mathbf{f}_i(s - s_i). \quad (8b)$$

Inserting (8) into (7), we explicitly solve for

$$s(\mathbf{p}) \approx s_i + \mathbf{f}_i^T (\mathbf{p} - \mathbf{T}_{C,i}). \quad (9)$$

We neglect any constant part of this equation, resulting in the objective progress function

$$s(\mathbf{p}) \approx \mathbf{f}_i^T \mathbf{p}. \quad (10)$$

C. Model Predictive Control

We use the concept of MPC as the control structure [20]. In MPC, a model of the system is used to predict the states

$$\mathcal{X} = (\mathbf{x}^{(1)} \ \mathbf{x}^{(2)} \ \dots \ \mathbf{x}^{(H_p)}), \quad \mathcal{X} \in \mathbb{R}^{6 \times H_p} \quad (11)$$

under the optimization of the inputs

$$\mathcal{U} = (\mathbf{u}^{(1)} \ \mathbf{u}^{(2)} \ \dots \ \mathbf{u}^{(H_p)}), \quad \mathcal{U} \in \mathbb{R}^{2 \times H_u} \quad (12)$$

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

To achieve recursive feasibility despite using a finite prediction horizon, we add the *terminal constraint* $\mathbf{v}^{(H_p)} = \mathbf{0}$. This requires the standstill of the vehicle at the end of the prediction horizon, thus ensuring to fulfill the track constraints in the next time steps. In the case of the single-track model from Section II-A1, we have to add lower and upper bounds close to zero instead to avoid numerical issues.

D. Complete General Trajectory Optimization Problem

After we have constructed all necessary sub-models across the previous sections, we combine them to form a non-linear non-convex optimization problem in this section.

We choose the goal to maximize the track progression from Section II-B2 at the prediction horizon H_p . It has been proven to be an adequate substitution for minimizing time on the race track, e.g., in [3], [13]. For compliance with conventions, the negative track progression is minimized, resulting in the formulation $-s(\mathbf{p}^{(H_p)})$ for the objective function of the problem formulation.

The optimization problem's decision variables are $\mathcal{X}$ being the state prediction matrix and $\mathcal{U}$ being the input prediction matrix.

To reduce aggressive input changes, we add the damping penalty $R_u \in \mathbb{R}$ on the input changes. With the equations derived in the Sections II-A to II-C extended across the prediction horizon, the complete model – termed *General Trajectory Program* – is

$$\min_{\mathcal{X}, \mathcal{U}} -s(\mathbf{p}^{(H_p)}) + R_u \sum_{j=2}^{H_p} \left\| \mathbf{u}^{(j)} - \mathbf{u}^{(j-1)} \right\|_2^2 \quad (13a)$$

subject to

$$\mathbf{x}^{(j)} = \mathbf{M}(\mathbf{x}^{(j-1)}, \mathbf{u}^{(j)}) \quad \forall j = 1, \dots, H_p \quad (13b)$$

$$\mathbf{p}^{(j)} \in T \quad \forall j = 1, \dots, H_p \quad (13c)$$

$$\mathbf{A}_{\text{acc}}(\mathbf{v}^{(j)}) \ \mathbf{u}^{(j)} \leq \mathbf{b}_{\text{acc}}(\mathbf{v}^{(j)}) \quad \forall j = 1, \dots, H_p \quad (13d)$$

$$\mathbf{v}^{(H_p)} = \mathbf{0} \quad (13e)$$

A sparse formulation is chosen as it fits the problem parameters of Section VI better than a dense one, yielding lower computational cost [21]. If started with an initially feasible trajectory, the *General Trajectory Program* stays feasible.

Algorithm 1 Pseudo-Code for SCP Implementation

```

1:  $\mathcal{X} \leftarrow [\mathbf{0} \ \mathbf{0} \ \dots \ \mathbf{0}]$ 
2: repeat forever
3:    $\mathbf{x}^{(0)} \leftarrow \text{receiveCurrentState}()$ 
4:   for  $i = 1$  to  $N_{\text{RTI}}$  do
5:      $\text{QP} \leftarrow \text{createConvexifiedQP}(\mathcal{X}, \mathbf{x}^{(0)})$ 
6:      $\mathcal{X}, \mathcal{U} \leftarrow \text{solveQP}(\text{QP})$ 
7:   end for
8:    $\text{sendRequiredInput}(\mathbf{u}^{(1)})$ 

```

III. SEQUENTIAL CONVEX PROGRAMMING METHODS

The previous section introduced a non-convex and thus NP-hard optimization problem [11], [22]. In this section, we convexify the problem using SCP, allowing us to find a solution iteratively in real-time applications [11], [12]. Two variants of SCP are presented: SL, being a relaxation, and SCR, being a restriction. Both methods handle the non-convex parts of the optimization problem by approximating the track and input constraints.

Algorithm 1 shows the pseudo-code of the SCP method – more specifically, the special case of *Real-Time Iteration* (RTI) – and is described in the following. After initializing the state in line 1 to allow a feasible solution, the actual control loop runs concurrently to the vehicle driving starting at line 2. After receiving the current vehicle state in line 3, $N_{\text{RTI}} \in \mathbb{N}$ SCP iterations run. N is set constant for predictable computation times and chosen such that the execution time of loop 2 is smaller than the time step. First, the QP is formulated as a convex problem in line 5 with a warm-start of the previous state prediction matrix and the current state. Here, either the SL or SCR methods can be used. Then, the QP problem is solved, yielding the new state prediction matrix and input prediction matrix. The first input is used for control of the vehicle in line 8.

IV. SEQUENTIAL LINEARIZATION

The idea of SL is to linearize non-convex constraints of the optimization problem at the operating points, based on initial guesses or previous solutions. This linearization acts as a relaxation of the constraints and we obtain the form of a QP. We depict operating points' values with a tilde, e.g., $\tilde{\mathcal{X}}$. The following paragraphs summarize SL, a detailed presentation can be found in our paper [13].

The linear approximation of the general track model is described in Section IV-A. A trust-region is introduced in Section IV-B to support the track approximations. The complete SL formulation is given in Section IV-C.

A. Track Model: Linear Approximation

Based on the track discretization from Section II-B1, the track linearization is acquired by a tangent to the left track boundary T_L and right track boundary T_R each at each time step, respectively. To formulate the linear track constraints, the

closest center-line point's index i to a position $\mathbf{p}$ is acquired similar to (7) by the index function

$$i(\mathbf{p}) = \arg \min_{k \in \{0, \dots, N_{\text{points}}\}} \|\mathbf{T}_{C,k} - \mathbf{p}\|_2. \quad (14)$$

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 $\xi \geq 0$ [20]. It is added to the problem objective with a numerical big 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 either side by the track's unit. The track constraints can be formulated for the left and right sides respectively

$$(\mathbf{p} - \mathbf{T}_{L,i(\tilde{\mathbf{p}})})^T (\mathbf{n}_{i(\tilde{\mathbf{p}})}) \leq \xi \quad (15a)$$

$$(\mathbf{p} - \mathbf{T}_{R,i(\tilde{\mathbf{p}})})^T (-\mathbf{n}_{i(\tilde{\mathbf{p}})}) \leq \xi. \quad (15b)$$

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.

B. Trust Region

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

$$\tilde{\mathbf{p}}_k - L \leq \mathbf{p}_k \leq \tilde{\mathbf{p}}_k + L \quad \forall k \in \{x, y\}. \quad (16)$$

C. Complete Trajectory Optimization Problem

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

$$\min_{\mathbf{x}, \mathbf{u}, \xi} -\mathbf{f}_{i(\tilde{\mathbf{p}}^{(H_p)})}^T \mathbf{p}^{(H_p)} + S\xi + R_u \sum_{j=2}^{H_p} \left\| \mathbf{u}^{(j)} - \mathbf{u}^{(j-1)} \right\|_2^2 \quad (17a)$$

subject to

$$\mathbf{x}^{(j)} = \mathbf{M}(\mathbf{x}^{(j-1)}, \mathbf{u}^{(j)}), \quad \forall j=1, \dots, H_p \quad (17b)$$

$$(\mathbf{p}^{(j)} - \mathbf{T}_{L,j(\tilde{\mathbf{p}}^{(j)})})^T (\mathbf{n}_{j(\tilde{\mathbf{p}}^{(j)})}) \leq \xi \quad \forall j=1, \dots, H_p \quad (17c)$$

$$(\mathbf{p}^{(j)} - \mathbf{T}_{R,j(\tilde{\mathbf{p}}^{(j)})})^T (-\mathbf{n}_{j(\tilde{\mathbf{p}}^{(j)})}) \leq \xi \quad \forall j=1, \dots, H_p \quad (17d)$$

$$A_{\text{acc}}(\tilde{\mathbf{v}}^{(j)}, \frac{2\pi k}{n} \mathbf{u}^{(j)}) \leq b_{\text{acc}}(\tilde{\mathbf{v}}^{(j)}) \quad \forall k=1, \dots, n \quad (17e)$$

$$\tilde{\mathbf{p}}^{(j)} - L \leq \mathbf{p}^{(j)} \leq \tilde{\mathbf{p}}^{(j)} + L \quad \forall j=1, \dots, H_p \quad (17f)$$

$$\mathbf{v}^{(H_p)} = \mathbf{0} \quad (17g)$$

$$0 \leq \xi. \quad (17h)$$

V. SEQUENTIAL CONVEX RESTRICTION

The method of SCR follows the idea of restricting a non-convex feasible solution set to a strict convex subset. This section starts with the introduction to the concept of SCR in Section V-A, followed by the construction and proof of feasible warm-start trajectories in Section V-B and Section V-C, respectively. Sections V-D to V-G will apply SCR to our specific application where we will focus 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* (13a) to (13e), termed *Generalized Unrestricted Program*:

$$\min_z q(z) \quad (18a)$$

$$\text{subject to } Ez = b \quad (18b)$$

$$z \in R, \quad (18c)$$

where $z \in \mathbb{R}^n$, $E \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $q : \mathbb{R}^n \rightarrow \mathbb{R}$, $R \subseteq \mathbb{R}^n$. The *Generalized Unrestricted Program* is in general non-convex. 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

$$\min_z \hat{q}(z, \tilde{z}) \quad (19a)$$

$$\text{subject to } Ez = b \quad (19b)$$

$$z \in R(\tilde{z}), \quad (19c)$$

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

$$R(\tilde{z}) \subseteq D \quad \forall \tilde{z} \in D, \quad (20a)$$

$$R(\tilde{z}) \text{ is convex} \quad \forall \tilde{z} \in D, \quad (20b)$$

$$\tilde{z} \in R(\tilde{z}) \quad \forall \tilde{z} \in D. \quad (20c)$$

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 solution, denoted by $\mathcal{F}(\tilde{z})$.

$\mathcal{F}(\tilde{z})$ is also a feasible point in the *Generalized Unrestricted Program* because $\tilde{z} \in R(\tilde{z}) \implies \tilde{z} \in D$. 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 $h_{\tilde{z}}$ and $R(\tilde{z})$.

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

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. A feasible initial trajectory is, e.g., $\mathcal{X} = [\mathbf{0} \ \mathbf{0} \ \dots \ \mathbf{0}]$ and $\mathcal{U} = [\mathbf{0} \ \mathbf{0} \ \dots \ \mathbf{0}]$. Thus, a vehicle that is standing still is following a feasible trajectory.

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

$$\mathbf{x}^{(j-1)} = \tilde{\mathbf{x}}^{(j)} \quad \forall j = 2, \dots, H_p, \quad (21a)$$

$$\mathbf{u}^{(j-1)} = \tilde{\mathbf{u}}^{(j)} \quad \forall j = 2, \dots, H_p. \quad (21b)$$

As $\tilde{\mathbf{x}}^{(H_p+1)}$ and $\tilde{\mathbf{u}}^{(H_p+1)}$ do not exist, the trajectory must be explicitly extended by one time step using

$$\mathbf{x}^{(H_p)} = \tilde{\mathbf{x}}^{(H_p)}, \quad (21c)$$

$$\mathbf{u}^{(H_p)} = \mathbf{0}. \quad (21d)$$

The resulting extended trajectory is ensured to be feasible by the terminal constraint (13e).

C. Proof of Trajectories for Recursive Feasibility

We prove that the trajectory in (21) is feasible in the *General Trajectory Program* (13) for every constraint, if the previous time step's solution was feasible. This is called *Recursive Feasibility*. We use the linear vehicle model for simplicity, for the single-track model the proof can be performed analogously. Constraints (13b) for $j = 1$:

$$\mathbf{x}^{(1)} = \mathbf{A}\mathbf{x}^{(0)} + \mathbf{B}\mathbf{u}^{(1)} \quad (22a)$$

$$\text{subtracting } \tilde{\mathbf{x}}^{(2)} = \mathbf{A}\tilde{\mathbf{x}}^{(1)} + \mathbf{B}\tilde{\mathbf{u}}^{(2)} \text{ yields} \quad (22b)$$

$$\underbrace{\mathbf{x}^{(1)} - \tilde{\mathbf{x}}^{(2)}}_{=0 \text{ (21a)}} = \mathbf{A}(\underbrace{\mathbf{x}^{(0)} - \tilde{\mathbf{x}}^{(1)}}_{=0 \text{ (21b)}}) + \mathbf{B}(\underbrace{\mathbf{u}^{(1)} - \tilde{\mathbf{u}}^{(2)}}_{=0 \text{ (21b)}}) \quad (22c)$$

$$\iff \mathbf{x}^{(0)} = \tilde{\mathbf{x}}^{(1)}, \quad (22d)$$

where (22d) states that SCR assumes the vehicle following the predicted trajectory without disturbances.

Constraints (13b) for $j = 2, \dots, H_p - 1$:

$$\mathbf{x}^{(j)} = \mathbf{A}\mathbf{x}^{(j-1)} + \mathbf{B}\mathbf{u}^{(j)} \quad (22e)$$

$$\text{and } \tilde{\mathbf{x}}^{(j+1)} = \mathbf{A}\tilde{\mathbf{x}}^{(j)} + \mathbf{B}\tilde{\mathbf{u}}^{(j+1)} \text{ yields} \quad (22f)$$

$$\underbrace{\mathbf{x}^{(j)} - \tilde{\mathbf{x}}^{(j+1)}}_{=0 \text{ (21a)}} = \mathbf{A}(\underbrace{\mathbf{x}^{(j-1)} - \tilde{\mathbf{x}}^{(j)}}_{=0 \text{ (21a)}}) + \mathbf{B}(\underbrace{\mathbf{u}^{(j)} - \tilde{\mathbf{u}}^{(j+1)}}_{=0 \text{ (21b)}}). \quad (22g)$$

Constraints (13b) for $j = H_p$:

$$\mathbf{x}^{(H_p)} = \mathbf{A}\mathbf{x}^{(H_p-1)} + \mathbf{B}\mathbf{u}^{(H_p)} \quad (22h)$$

$$\stackrel{(21a) \text{ and } (21d)}{\iff} \tilde{\mathbf{x}}^{(H_p)} = \mathbf{A}\tilde{\mathbf{x}}^{(H_p)} \quad (22i)$$

$$\iff \mathbf{0} = (\mathbf{A} - \mathbf{I})\tilde{\mathbf{x}}^{(H_p)} \quad (22j)$$

$$\stackrel{(4) \text{ and } (13e)}{\iff} \mathbf{0} = \begin{bmatrix} 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{p}_x^{(H_p)} \\ \tilde{p}_y^{(H_p)} \\ 0 \\ 0 \end{bmatrix} \quad (22k)$$

Constraints (13c) for $j = 1, \dots, H_p - 1$:

$$\mathbf{p}^{(j)} \in T \stackrel{(21a)}{\iff} \tilde{\mathbf{p}}^{(j+1)} \in T \quad (22l)$$

Constraints (13c) for $j = H_p$:

$$\mathbf{p}^{(H_p)} \in T \stackrel{(21c)}{\iff} \tilde{\mathbf{p}}^{(H_p)} \in T \quad (22m)$$

Constraints (13d) for $j = 1, \dots, (H_p - 1)$:

$$\mathbf{A}_{\text{acc}}(\mathbf{v}^{(j)})\mathbf{u}^{(j)} \leq \mathbf{b}_{\text{acc}}(\mathbf{v}^{(j)}) \quad (22n)$$

$$\stackrel{(21b)}{\iff} \mathbf{A}_{\text{acc}}(\tilde{\mathbf{v}}^{(j+1)})\tilde{\mathbf{u}}^{(j+1)} \leq \mathbf{b}_{\text{acc}}(\mathbf{v}^{(j)})$$

Constraints (13d) for $j = H_p$:

$$\mathbf{A}_{\text{acc}}(\mathbf{v}^{(H_p)})\mathbf{u}^{(H_p)} \leq \mathbf{b}_{\text{acc}}(\mathbf{v}^{(H_p)}) \stackrel{(21d)}{\iff} \mathbf{0} \leq \mathbf{b}_{\text{acc}}(\mathbf{v}^{(H_p)}) \quad (22o)$$

Constraints (13e):

$$\mathbf{v}^{(H_p)} = \mathbf{0} \stackrel{(21c)}{\iff} \tilde{\mathbf{v}}^{(H_p)} = \mathbf{0} \quad (22p)$$

We have proven for every constraint and every time step that the initial trajectory (21a) and (21d) is feasible in the *General Trajectory Program* (13) for the linear vehicle model (4).

D. Application to Problem

To apply SCR to the *General Trajectory Program* (13) we start by rewriting it in the form of the *Generalized Unrestricted Program* (18), before applying the restriction and conforming to the *Generalized Restricted Program* (19) 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

$$\mathbf{z} = \begin{pmatrix} \mathbf{p}^{(1)} & \dots & \mathbf{p}^{(H_p)} \\ \mathbf{v}^{(1)} & \dots & \mathbf{v}^{(H_p)} \\ \mathbf{u}^{(1)} & \dots & \mathbf{u}^{(H_p)} \end{pmatrix} = \begin{pmatrix} \mathcal{X} \\ \mathcal{U} \end{pmatrix}, \quad (23)$$

where $\mathbf{z} \in \mathbb{R}^{6 \times H_p}$ is the decision variable in the *Generalized Unrestricted Program*.

(13b) and (13e) are affine and therefore represented with $Ez = b$ (18b). The track constraints (13c) and acceleration constraints (13d) are non-convex and non-linear, respectively, and therefore represented with $\mathbf{z} \in D$ (18c). For the acceleration constraints, we reformulate (13d) as a set

$$\mathcal{A}_{\text{acc}} = \left\{ \mathbf{u} \in \mathbb{R}^2 \mid \mathbf{A}_{\text{acc}}(\mathbf{v}^{(j)})\mathbf{u}^{(j)} \leq \mathbf{b}_{\text{acc}}(\mathbf{v}^{(j)}) \right\}. \quad (24)$$

Combining the track and acceleration constraint sets with the Cartesian product yields

$$(\mathbf{p}^{(j)} \in T) \wedge (\mathbf{u}^{(j)} \in \mathcal{A}_{\text{acc}}), \quad \forall j = 1, \dots, H_p \quad (25a)$$

$$\iff z \in \underbrace{(T^{H_p} \times \mathbb{R}^{2 \times H_p} \times \mathcal{A}_{\text{acc}}^{H_p})}_{=D}, \quad (25b)$$

where $D \subseteq \mathbb{R}^{6 \times H_p}$ is the constraint set in (18c).

The objective function remains unchanged as

$$\hat{q}(z, \tilde{z}) = q(z) = -s \left(\begin{bmatrix} z_1^{(H_p)} \\ z_2^{(H_p)} \end{bmatrix} \right) = -s \left(\mathbf{p}^{(H_p)} \right), \quad (26)$$

which concludes the rewrite of the *General Trajectory Program* as *Generalized Unrestricted Program*.

Now we need to transform the formulation to the *Generalized Restricted Program*. For this, we need to construct a convex objective $h_{\tilde{z}}$ and a restriction function $R(\tilde{z})$. The following Sections V-E to V-F will formulate the restriction functions $R_{\mathcal{A}}$ and R_T for the acceleration and track, respectively. They will fulfill the conditions for a restriction function (20) 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:

$$R(z) = R_T(\mathbf{p}^{(1)}) \times \dots \times R_T(\mathbf{p}^{(H_p)}) \times \mathbb{R}^{2 \times H_p} \\ \times R_{\mathcal{A}}(\mathbf{u}^{(1)}) \times \dots \times R_{\mathcal{A}}(\mathbf{u}^{(H_p)}). \quad (27)$$

Note that this definition corresponds directly to the definition of the unrestricted constraint set D in (25b) and to the definition of 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 (20) are preserved by the Cartesian product shows that this is the case.

E. Restriction Function for Acceleration Input

For the acceleration constraints, we restrict the tire acceleration limit ellipses set from (24) by shrinking the existing polygon representation with the restriction function $R_{\mathcal{A}} : \mathcal{A}_{\text{acc}} \rightarrow \mathcal{P}(\mathcal{A}_{\text{acc}})$

$$R_{\mathcal{A}}(\mathbf{u}) = \left\{ \mathbf{u} \in \mathcal{A}_{\text{acc}} \mid \mathbf{A}_{\text{acc}} \left(\mathbf{v}^{(j)} \right) \mathbf{u}^{(j)} \leq a_{\text{max}} \mathbf{b}_{\text{acc}} \left(\mathbf{v}^{(j)} \right) \right\}, \quad (28)$$

where a_{max} is chosen so that $R_{\mathcal{A}}$ becomes a restriction.

F. Restriction Function for Track Model

In this section, we present the track's restriction function $R_T : T \rightarrow \mathcal{P}(T)$ in detail. After defining the function and proving its validity in Section V-F1, the algorithm to create the restricted track is designed in Section V-F2. Lastly, adaptations to the progress functions are made in Section V-F3.

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 $R_T : T \rightarrow \mathcal{P}(T)$. 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,

$$R_T(\tilde{\mathbf{p}}) = P_{l(\tilde{\mathbf{p}})}, \quad (29a)$$

$$P_l = \{\mathbf{p} \in \mathbb{R}^2 \mid F_l \mathbf{p} \leq \mathbf{g}_l\}, \quad (29b)$$

where $P_l \subseteq \mathbb{R}^2$ is a convex polygon, $F_l \in \mathbb{R}^{n_{\text{edges}} \times 2}$ and $\mathbf{g}_l \in \mathbb{R}^{n_{\text{edges}}}$ being the constraints defining the polygon P_l with $n_{\text{edges}} \in \mathbb{R}_{\geq 4}$ edges, and $l : \mathbb{R}^2 \rightarrow \mathbb{Z}$ being an index function defined in the next paragraphs which selects the correct set of affine constraints, i.e., the polygon which includes $\mathbf{p}$. (29b) can be interpreted geometrically: A bounded set of linear constraints in two dimensions forms a convex polygon. We model the non-convex track area with the union of N_{polygons} overlapping, inside the track area lying, convex polygons P_l as

$$T = \bigcup_{l=1}^{N_{\text{polygons}}} P_l. \quad (30)$$

The polygon at a position $\mathbf{p}$ is given by an index function l . From the overlapping polygons that include the position $\mathbf{p}$, the index function selects the most forward polygon as

$$l_{\text{potential}}(\mathbf{p}) = \left\{ P_k \in T \mid \arg \max_{k=1, \dots, N_{\text{polygons}}} \mathbf{p} \in P_k \right\} \quad (31)$$

$$l(\mathbf{p}) = l_{\text{potential}}(\arg \max(\Delta l_{\text{potential}}(\mathbf{p}))). \quad (32)$$

Choosing the most forward overlapped polygon ensures using most progressive track constraints, agreeing with the objective function (26).

In the case of $\mathbf{p}$ not being inside the track area T due to numerical issues, we want to find the polygon being closest to $\mathbf{p}$ as a fallback. We assume that the affine constraints that construct the polygons are normalized, i.e., $\|\mathbf{f}_{k,l}\| = 1$ where $\mathbf{f}_{k,l} \in \mathbb{R}^2$ is a row vector in F_l . Thus, the signed distances to the polygon boundaries are $F_l \mathbf{p} - \mathbf{g}_l$. The fallback index function can be written as

$$l(\mathbf{p}) = \arg \min_{k=1, \dots, N_{\text{polygons}}} d_k(\mathbf{p}) \quad (33a)$$

$$d_k(\mathbf{p}) = \max_m (F_k \mathbf{p} - \mathbf{g}_k)_m, \quad (33b)$$

where $d_k(\mathbf{p}) : \mathbb{R}^2 \rightarrow \mathbb{R}$ is the maximum signed distance to an edge of P_k and the subscript m selects the m -th entry from the signed distances vector $F_k \mathbf{p} - \mathbf{g}_k$.

With the above definitions we can prove that the track restriction function $R_T(\tilde{\mathbf{p}})$ in (29a) fulfills the conditions (20) for a restriction function:

Condition (20a):

$$R_T(\tilde{\mathbf{p}}) \subseteq T \quad \forall \tilde{\mathbf{p}} \in T \quad (34a)$$

$$\stackrel{(29a) \text{ and } (30)}{\iff} P_{l(\tilde{\mathbf{p}})} \subseteq \bigcup_{l=1}^{N_{\text{polygons}}} P_l \quad \forall \tilde{\mathbf{p}} \in T \quad (34b)$$

$$\iff P_k \subseteq \bigcup_{l=1}^{N_{\text{polygons}}} P_l \quad \forall k = 1, \dots, N_{\text{polygons}}. \quad (34c)$$

Algorithm 2 Computation of the Track Polygons

```

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

```

Condition (20b):

$$R_T(\tilde{\mathbf{p}}) \text{ is convex } \forall \tilde{\mathbf{p}} \in T \quad (34d)$$

$$\stackrel{(29a)}{\iff} P_{l(\tilde{\mathbf{p}})} \text{ is convex } \forall \tilde{\mathbf{p}} \in T \quad (34e)$$

$$\iff P_k \text{ is convex } \forall k = 1, \dots, N \quad (34f)$$

$$\stackrel{(29b)}{\iff} \{\mathbf{p} \in \mathbb{R}^2 \mid F_k \mathbf{p} \leq \mathbf{g}_k\} \text{ is convex } \forall k = 1, \dots, N. \quad (34g)$$

Condition (20c):

$$\tilde{\mathbf{p}} \in R_T(\tilde{\mathbf{p}}) \quad \forall \tilde{\mathbf{p}} \in T \quad (34h)$$

$$\stackrel{(29a)}{\iff} \tilde{\mathbf{p}} \in P_{l(\tilde{\mathbf{p}})} \quad \forall \tilde{\mathbf{p}} \in T \quad (34i)$$

$$\stackrel{(29b)}{\iff} \tilde{\mathbf{p}} \in \{\mathbf{p} \in \mathbb{R}^2 \mid F_{l(\tilde{\mathbf{p}})} \mathbf{p} \leq \mathbf{g}_{l(\tilde{\mathbf{p}})}\} \quad \forall \tilde{\mathbf{p}} \in T \quad (34j)$$

$$\iff F_{l(\tilde{\mathbf{p}})} \tilde{\mathbf{p}} \leq \mathbf{g}_{l(\tilde{\mathbf{p}})} \quad \forall \tilde{\mathbf{p}} \in T \quad (34k)$$

$$\stackrel{(33b)}{\iff} d_{l(\tilde{\mathbf{p}})}(\tilde{\mathbf{p}}) \leq 0 \quad \forall \tilde{\mathbf{p}} \in T \quad (34l)$$

$$\stackrel{(33a)}{\iff} \left(\min_{k=1, \dots, N} d_k(\tilde{\mathbf{p}}) \right) \leq 0 \quad \forall \tilde{\mathbf{p}} \in T. \quad (34m)$$

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

$$\left(\min_{k=1, \dots, N} d_k(\tilde{\mathbf{p}}) \right) \leq d_n(\tilde{\mathbf{p}}) \leq 0 \quad \forall \tilde{\mathbf{p}} \in T. \quad (34n)$$

We have proven that the track restriction function $R_T(\mathbf{p})$ is a valid restriction function by checking the conditions (20).

2) *Design of the Track Restriction Function:* As Section V-F1 demonstrated that a set of convex polygons can be used to define the track restriction function $R_T(\mathbf{p})$, we design an algorithm for the polygons' computation. The algorithm 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 $\mathbf{p}^{(j)}$ across iterations is beneficial.

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. 3a.

a) *Tessellation:* With the general track model from Section II-B, let $\text{hull} : \mathcal{P}(\mathbb{R}^n) \rightarrow \mathcal{P}(\mathbb{R}^n)$ 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 as

$$P_i = \text{hull}(\{\mathbf{T}_{L,i}, \mathbf{T}_{R,i}, \mathbf{T}_{L,i+1}, \mathbf{T}_{R,i+1}\}) \quad \forall i = 1, \dots, (N-1), \quad (35a)$$

$$P_N = \text{hull}(\{\mathbf{T}_{L,N}, \mathbf{T}_{R,N}, \mathbf{T}_{L,1}, \mathbf{T}_{R,1}\}), \quad (35b)$$

$$\mathcal{T} = (P_1, P_2, \dots, P_N), \quad (35c)$$

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

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

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 $\Delta A : \mathcal{P}(\mathbb{R}^n) \times \mathcal{P}(\mathbb{R}^n) \rightarrow \mathbb{R}_{\geq 0}$ which returns the required additional area to make the union of two polygons $P_A \cup P_B$ convex as

$$\Delta A(P_A, P_B) = |\text{hull}(P_A \cup P_B)| - |P_A \cup P_B|. \quad (36)$$

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

Fig. 3c shows the example track's merged polygons, reducing the number of polygons from 1090 to 178 with $\epsilon_A = 1.35e-05 \text{ m}^2$, equaling $\epsilon_A = 0.025 \text{ m}^2$ 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

$$\tilde{P}_i \subset T, \quad (37a)$$

$$\tilde{P}_i \supseteq P_i \text{ and } \quad (37b)$$

$$\tilde{P}_i \text{ is convex.} \quad (37c)$$

Every polygon P_i is enlarged in both forward and backward direction. For the forward direction, the result of the

Fig. 4. Convex polygon expansion adhering to track limits

enlargement algorithm is illustrated in Fig. 4 and the process is described briefly in the following. 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 $P_{i+1}, \dots, P_{\text{last}}$ 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 described in more detail in the published code repository.

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 (19a) as

$$h_{\tilde{z}}(z) = -\mathbf{F}_{l(\tilde{\mathbf{p}}^{(H_p)})}^T \mathbf{p}^{(H_p)}, \quad (38)$$

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

$$\mathbf{F}_i = \frac{1}{|I_i|} \sum_{k \in I_i} \mathbf{f}_k \quad \forall i = 1, \dots, N_{\text{polygons}}, \quad (39a)$$

$$I_i = \{k = 1, \dots, N_{\text{points}} \mid \mathbf{T}_{C,k} \in P_i\} \quad \forall i = 1, \dots, N_{\text{polygons}}, \quad (39b)$$

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

G. Complete Trajectory Optimization Problem

Combining the *General Trajectory Program* (13) with this section's formulations, namely (i) the objective function (38), (ii) a slack variable $\xi \in \mathbb{R}$ to soften the track constraints (as in the formulation of SL in Section IV), (iii) the track constraints (29) and (iv) the acceleration constraints (28), yields the quadratic optimization problem

$$\min_{\mathbf{x}, \mathbf{u}, \xi} -\mathbf{F}_{l(\tilde{\mathbf{p}}^{(H_p)})}^T \mathbf{p}^{(H_p)} + S\xi + R_u \sum_{j=2}^{H_p} \left\| \mathbf{u}^{(j)} - \mathbf{u}^{(j-1)} \right\|_2^2 \quad (40a)$$

subject to

$$\mathbf{x}^{(j)} = \mathbf{M}(\mathbf{x}^{(j-1)}, \mathbf{u}^{(j)}) \quad \forall j = 1, \dots, H_p \quad (40b)$$

$$F_{l(\tilde{\mathbf{p}}^{(j)})} \mathbf{p}^{(j)} \leq \mathbf{g}_{l(\tilde{\mathbf{p}}^{(j)})} + \xi \quad \forall j = 1, \dots, H_p \quad (40c)$$

$$\left(\cos\left(\frac{2\pi k}{n}\right), \sin\left(\frac{2\pi k}{n}\right) \right)^T \mathbf{u}^{(j)} \leq a_{\max} \cos\left(\frac{\pi}{n}\right) \quad \forall k = 1, \dots, n, \quad \forall j = 1, \dots, H_p \quad (40d)$$

$$\mathbf{v}^{(H_p)} = \mathbf{0} \quad (40e)$$

$$0 \leq \xi. \quad (40f)$$

VI. IMPLEMENTATION & COMPUTATIONAL RESULTS

To validate and compare the presented methods, varying combinations of vehicle models and track discretization methods are evaluated in simulation. A video of the results is available at <https://youtu.be/21iETsolCNQ>. This video shows not only the paper contents but also that our implementation works with multiple vehicle racing, adhering to collision avoidance constraints. The complete software is published ready-to-run on Github and CodeOcean, <https://github.com/embedded-software-laboratory/sequential-convex-programming> and <https://codeocean.com/capsule/6818033/tree/v1>, respectively.

In the following, implementation details are given before the results of the evaluation are presented.

A. Implementation

The controller is implemented in software using *MATLAB R2021a*. To solve the quadratic programs *cplexqp* of *IBM ILog CPLEX 12.10* is used. A simulation is run using the same *MATLAB* instance. The complete software package is executed on a PC equipped with an *AMD Ryzen 5 3600* CPU, a hexacore with 12 logical cores @ 4.2GHz, and 16GB RAM at 3200MHz on the operating system *Windows 10 21H1 64-bit*.

As the parameters such as vehicle model for controller and simulation, MPC objective weights, and track discretization type vary between individual scenarios, no overview is given. The parameters were chosen with the goal of the maximum controller execution time to fall below the discretization time of 100ms and can be found in the code repository. The MPC-Weights Q and R as well as SL's trust region size were tuned with a semi-automated parameter study to achieve a robust controller with minimal lap time on a 1:43-scaled version of the *Hockenheimring*. This track is a down-scaled version of the one used in our previous paper [13] and will be used for the following evaluation, too.

B. Computational Results

For the results, we have two objectives to examine in the following. Firstly, how the single-track vehicle model compares to the linear double integrator model used in our previous paper. Secondly, how our new track discretization method SCR compares to SL.

Fig. 5. Showcase of predicted trajectories of linear versus single-track vehicle model used in the controller

TABLE I
LAP TIMES OF TRACK DISCRETIZATION METHODS

Measurement	$t_{\text{lap,start}} [\text{s}]$	$t_{\text{lap,flying}} [\text{s}]$
SL	11.0	10.0
SCR	10.0	9.5
Δ	1.0	0.5
Δ in [%]	10%	5%

a) *Comparison of Linear vs. Single-Track Model:* We equip one vehicle with two controllers, one using the linear model and one using the single-track model. This showcases the difference between their predictions. The controllers use SL, start from the same state and run with the same parameters. The acceleration bounds for the linear model were derived by steady-state tests with the single-track model, as described in Section II-A2.

Fig. 5 shows that the predicted trajectories differ qualitatively. When compared to the linear model, the single-track model achieves trajectories closer to typical racing lines found in reality, using the entire track area to maintain speed. In the following evaluation, we will use the single-track model only.

b) *Comparison of Track Discretization Methods SL and SCR:* To compare the track discretization methods of SL and SCR, we choose the evaluation criteria of 1) the first lap time from standstill $t_{\text{lap,start}}$ and 2) the flying lap time $t_{\text{lap,flying}}$, which we define as lap number two. For evaluation, we keep all parameters other than the track discretization method fixed. The trajectories of SL and SCR are qualitatively close. A trajectory of the single-track model was already displayed in gray using SL in Fig. 5.

The lap time results show that SCR yields better lap times in both evaluation criteria, as displayed in Table I. The relative decrease for $t_{\text{lap,start}}$ is higher than for $t_{\text{lap,flying}}$, as the trust-region restricts SL at the very first iterations after start.

To get an understanding of the significance of the decrease in lap time $t_{\text{lap,flying}}$ we use *Formula 1*'s 2019 Hockenheimring qualifying results [23]. The results are down-scaled to match Hockenheimring's shorter track layout, which we use

Fig. 6. Statistics for computation time of SL vs SCR controller over two laps from standstill

in simulation [24]. The lap time difference of 0.5s equals the difference between 1st position and 6th, 14th, and 7th for qualifying Q3, Q2, and Q1, respectively.

Regarding lap times, SL performs worse than SCR because it uses a higher level of approximation and relaxes the track area. Due to the approximations, a trust-region is required to restrict the deviation of future states from the current states. Moreover, the approximations manifest in the solver throwing warnings about numerical issues. SCR is presumably producing more accurate predictions, i.e., being closer to the optimal non-convex solution.

SCR not always obtains the optimal racing line due to numerical issues. Instead, the vehicle tends to drift outwards due to the aggressive choice of selecting the most forward polygon for track representation. The polygon selection function could be optimized for better lap times in the future. Nevertheless, the selection method proved functional and consistently achieving better lap times than SL.

Regarding computation times, we examine the median, 99th percentile, and maximum computation time for SL and SCR, each. For all three time categories, we see a doubling of times for SCR under the chosen parameters, as displayed in Fig. 6. The times stay consistent, i.e., maximum and 99th percentile execution times are close to the median.

VII. CONCLUSION

A general model and problem formulation for the trajectory planning of autonomous race cars were introduced. The general problem formulation was adapted to the specific solution methods SL and SCR, the latter being introduced in detail in this paper. By comparison, it was found that SCR provides improved lap times, especially in dynamic situations, while being real-time capable.

Future work will investigate real experiments within the *Cyber-Physical Mobility Lab*, an open source platform for networked and autonomous vehicles [25], [26]. Furthermore, SCR could be optimized for faster execution times and SCR's polygon index function could be improved for even better lap times.

REFERENCES

- [1] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat, "MPC-based approach to active steering for autonomous vehicle systems," *International Journal of Vehicle Autonomous Systems*, vol. 3, no. 2/3/4, p. 265, 2005. [Online]. Available: <http://www.inderscience.com/link.php?id=8237>
- [2] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm: IEEE, May 2016, pp. 1433–1440. [Online]. Available: <https://ieeexplore.ieee.org/document/7487277/>
- [3] A. Liniger, A. Domahidi, and M. Morari, "Optimization-Based Autonomous Racing of 1:43 Scale RC Cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, Sep. 2015, arXiv: 1711.07300. [Online]. Available: <http://arxiv.org/abs/1711.07300>
- [4] M. Kloock, P. Scheffe, L. Botz, J. Maczjewski, B. Alrifaae, and S. Kowalewski, "Networked Model Predictive Vehicle Race Control," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Auckland, New Zealand: IEEE, Oct. 2019, pp. 1552–1557. [Online]. Available: <https://ieeexplore.ieee.org/document/8917222/>
- [5] S. Srinivasan, S. N. Giles, and A. Liniger, "A Holistic Motion Planning and Control Solution to Challenge a Professional Racecar Driver," arXiv:2103.00358 [cs, eess], Feb. 2021, arXiv: 2103.00358. [Online]. Available: <http://arxiv.org/abs/2103.00358>
- [6] 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," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, p. 7, 2020.
- [7] D. Caporale, A. Settimi, F. Massa, F. Amerotti, A. Corti, A. Fagiolini, M. Guiggiani, A. Bicchì, and L. Pallottino, "Towards the Design of Robotic Drivers for Full-Scale Self-Driving Racing Cars," in *2019 International Conference on Robotics and Automation (ICRA)*. Montreal, QC, Canada: IEEE, May 2019, pp. 5643–5649. [Online]. Available: <https://ieeexplore.ieee.org/document/8793882/>
- [8] J. Betz, A. Wischniewski, A. Heilmeier, F. Nobis, L. Hermansdorfer, T. Stahl, T. Herrmann, and M. Lienkamp, "A Software Architecture for the Dynamic Path Planning of an Autonomous Racecar at the Limits of Handling," in *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. Graz, Austria: IEEE, Nov. 2019, pp. 1–8. [Online]. Available: <https://ieeexplore.ieee.org/document/8965238/>
- [9] R. Verschuere, M. Zanon, R. Quirynen, and M. Diehl, "Time-optimal race car driving using an online exact hessian based nonlinear MPC algorithm," in *2016 European Control Conference (ECC)*. Aalborg, Denmark: IEEE, Jun. 2016, pp. 141–147. [Online]. Available: <http://ieeexplore.ieee.org/document/7810277/>
- [10] U. Rosolia and F. Borrelli, "Learning How to Autonomously Race a Car: A Predictive Control Approach," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2713–2719, Nov. 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/8896988/>
- [11] T. Lipp and S. Boyd, "Variations and extension of the convex-concave procedure," *Optimization and Engineering*, vol. 17, no. 2, pp. 263–287, Jun. 2016. [Online]. Available: <http://link.springer.com/10.1007/s11081-015-9294-x>
- [12] J. Nocedal and S. J. Wright, *Numerical optimization*, 2nd ed., ser. Springer series in operations research. New York: Springer, 2006, oCLC: ocm68629100.
- [13] B. Alrifaae and J. Maczjewski, "Real-time Trajectory optimization for Autonomous Vehicle Racing using Sequential Linearization," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. Changshu: IEEE, Jun. 2018, pp. 476–483. [Online]. Available: <https://ieeexplore.ieee.org/document/8500634/>
- [14] B. Alrifaae, J. Maczjewski, and D. Abel, "Sequential Convex Programming MPC for Dynamic Vehicle Collision Avoidance," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, Aug. 2017, pp. 2202–2207.
- [15] H. B. Pacejka, *Tyre and vehicle dynamics*, 2nd ed. Oxford: Butterworth-Heinemann, 2006, oCLC: ocm70252538.
- [16] A. Liniger, "Path Planning and Control for Autonomous Racing," Dissertation, ETH Zurich, 2018, artwork Size: 136 p. Medium: application/pdf Pages: 136 p. [Online]. Available: <http://hdl.handle.net/20.500.11850/302942>
- [17] C. Voser, R. Y. Hindiyeh, and J. C. Gerdes, "Analysis and control of high sideslip manoeuvres," *Vehicle System Dynamics*, vol. 48, no. sup1, pp. 317–336, Dec. 2010. [Online]. Available: <http://www.tandfonline.com/doi/abs/10.1080/00423111003746140>
- [18] B. Heißing, M. Ersoy, and S. Gies, Eds., *Fahrwerkhandbuch: Grundlagen, Fahrdynamik, Komponenten, Systeme, Mechatronik, Perspektiven ; mit 84 Tabellen*, 3rd ed., ser. ATZ-MTZ Fachbuch. Wiesbaden: Vieweg + Teubner, 2011, oCLC: 742341171.
- [19] T. Fork, H. E. Tseng, and F. Borrelli, "Models and Predictive Control for Nonplanar Vehicle Navigation," arXiv:2104.08427 [cs, eess], May 2021, arXiv: 2104.08427. [Online]. Available: <http://arxiv.org/abs/2104.08427>
- [20] J. M. Maciejowski, *Predictive control: with constraints*. Harlow, England ; New York: Prentice Hall, 2002.
- [21] T. Albin Rajasingham, *Nonlinear Model Predictive Control of Combustion Engines: From Fundamentals to Applications*, ser. Advances in Industrial Control. Cham: Springer International Publishing, 2021. [Online]. Available: <https://link.springer.com/10.1007/978-3-030-68010-7>
- [22] 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. Boston, MA: Springer US, 1996, vol. 7. [Online]. Available: <http://link.springer.com/10.1007/978-1-4613-3437-8>
- [23] A. Lewin, "2019 German Grand Prix - Qualifying results," DIGITAL MOTORSPORT MEDIA, Tech. Rep., Jul. 2019. [Online]. Available: <https://f1i.com/news/349236-2019-german-grand-prix-qualifying-results.html>
- [24] Wikipedia, "Hockenheimring," Jul. 2021. [Online]. Available: <https://en.wikipedia.org/wiki/Hockenheimring>
- [25] P. Scheffe, J. Maczjewski, M. Kloock, A. Kampmann, A. Derks, S. Kowalewski, and B. Alrifaae, "Networked and Autonomous Model-scale Vehicles for Experiments in Research and Education," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 17 332–17 337, 2020. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S2405896320324319>
- [26] M. Kloock, P. Scheffe, J. Maczjewski, A. Kampmann, A. Mokhtarian, S. Kowalewski, and B. Alrifaae, "Cyber-Physical Mobility Lab: An Open-Source Platform for Networked and Autonomous Vehicles," arXiv:2004.10063 [cs], Apr. 2021, arXiv: 2004.10063. [Online]. Available: <http://arxiv.org/abs/2004.10063>

Bassam Alrifaae is a Junior Principal Investigator at the Chair of Embedded Software, RWTH Aachen University, 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 the Principal Investigator of a project in the DFG Priority Program 1835: Cooperatively Interacting Automobiles. He is the domain coordinator of the Software Architecture

in the large project UNICARagil. From 2011 to 2017, he was a scientific associate with the Institute of Automatic Control at RWTH Aachen University, where he received his PhD.

Patrick Scheffe is a PhD candidate in Computer Science at the Chair of Embedded Software, RWTH Aachen University, Germany. He received the B.Sc. degree in Mechanical Engineering in 2016 and the M.Sc. degree in Automation Engineering in 2019 from RWTH Aachen University. His research interests include distributed decision-making and trajectory planning, especially in the context of networked and autonomous vehicles.

Maximilian Kloock is a PhD candidate in Computer Science at the Chair of Embedded Software, RWTH Aachen University, Germany. He received his B.Sc. in 2016 an M.Sc. in 2018, both in Computer Science at RWTH Aachen University. His research interests include distributed decision-making and verification for cyber-physical systems and their applications to networked and autonomous vehicles.

Theodor Mario Henneken received the B.Sc. degree in mechanical engineering from RWTH Aachen University, Germany, in 2019. Currently, he pursues his M.Sc. degrees in automation engineering and industrial engineering at RWTH Aachen University, Germany, and Tsinghua University, China, respectively.