Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 54-6 (2021) 99–106

Mixed-integer optimization-based planning for autonomous racing with obstacles and rewards ⋆

Rudolf Reiter * **Martin Kirchengast** ** **Daniel Watzenig** *** **Moritz Diehl** ****

* *Virtual Vehicle Research Center, Graz, Austria, (e-mail: rudolf.reiter@imtek.uni-freiburg.de).*
** *Institute of Automation and Control, Graz University of Technology, Graz, Austria, (e-mail: martin.kirchengast@tugraz.at)*
*** *Institute of Automation and Control, Graz University of Technology and Virtual Vehicle Research Center, Graz, Austria, (e-mail: daniel.watzenig@v2c2.at)*
**** *Department of Microsystems Engineering and the Department of Mathematics, University Freiburg, 79110 Freiburg, Germany, (e-mail: moritz.diehl@imtek.uni-freiburg.de)*

Abstract: Trajectory planning with the consideration of obstacles is a classical task in autonomous driving and robotics applications. This paper introduces a novel solution approach for the subclass of autonomous racing problems which is additionally capable of dealing with reward objects. This special type of objects is representing particular regions in state space, whose optional reaching is somehow beneficial (e.g. results in bonus points during a race). First, a homotopy class is selected which represents the left/right and catch/ignore decisions related to obstacle avoidance and reward collection, respectively. For this purpose, a linear mixed-integer problem is posed such that an approximated combinatorial problem is solved and repetitive switching decisions between solver calls are avoided. Secondly, an optimal control problem (OCP) based on a single-track vehicle model is solved within this homotopy class. In the corresponding nonlinear program, homotopy iterations are performed on the race track boundaries which correspond to the previously chosen homotopy class. This leads to an improved convergence of the solver compared to the direct approach. The mixed-integer method's effectiveness is demonstrated within a real-world test scenario during the autonomous racing competition *Roborace*. Furthermore, its combination with the OCP as well as the performance gain resulting from the homotopy iterations are shown in simulation.

Keywords: optimal trajectory, obstacle avoidance, autonomous mobile robots, autonomous vehicles, integer programming, planning

1. INTRODUCTION

Autonomous racing poses great challenges for the development of planning and control algorithms. As the vehicles move with high velocities close to their physical limits, nonlinear effects on their system dynamics arise and must already be considered during planning. Sudden obstacles or race opponents with uncertain behavior require evasion or overtaking maneuvers, planned in real-time on hardware with typically rather limited computational power. As part of the racing series *Roborace*, the participating teams develop software for the fully autonomous operation of electric race cars and are confronted with increasingly demanding objectives from one event to the other. This work addresses real-time trajectory planning for lap time minimization while avoiding obstacles and collecting rewards. Whereas the vehicle moves on a real race track, the purely virtual obstacles and rewards, which translate into lap time penalties and bonuses respectively, are provided to the car's software about 200 m in advance. The planning algorithm not only has to compute a trajectory which eludes obstacles and is feasible regarding vehicle kinematics and dynamics as well as race track geometry. It also must decide whether rewards are worth gathering, which distinguishes the considered task from related problems in literature. The proposed algorithm was successfully used in the *Roborace* competition at the Bedford race circuit in December 2020. Figure 1 shows the race car collecting a virtual reward object, which was live-streamed as augmented-reality animation during the race.

⋆ The work was financially supported by the Austrian funding program "Mobilität der Zukunft", the German Federal Ministry for Economic Affairs and Energy (BMWi) via DyConPV (0324166B), the project NewControl (GA Nr. 826653-2), supported by ECSEL JU, Horizon2020 and the Austrian funding authority FFG, and by the COMET K2 Competence Centers for Excellent Technologies Program.

10.1016/j.ifacol.2021.08.530

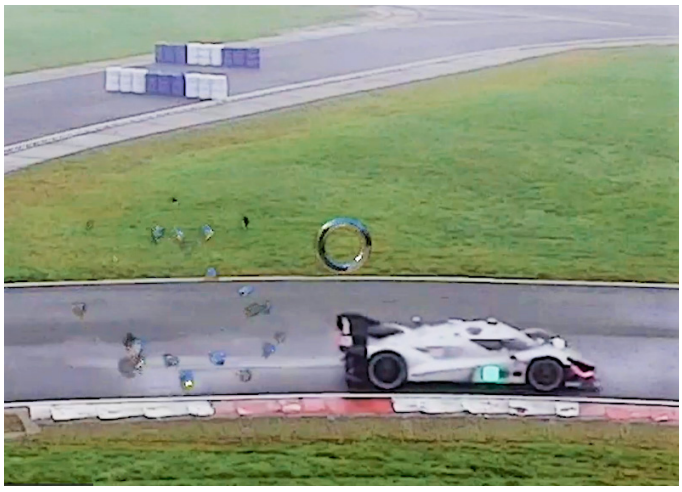

Fig. 1. Race car hitting an augmented reality reward at the competition in Bedford, England.

1.1 Related work

Trajectory planning with obstacle avoidance occurs in many domains and consequently, different approaches exist. Typically, in a certain situation there are infinitely many trajectories fulfilling the feasibility conditions. Therefore, cost functions are used to choose an optimal one depending on the application. Graph-based methods perform a global search on the time- and space-discretized configuration space to find the best trajectory. However, their ability to find the global optimum strongly depends on the chosen discretization and if that is fixed, even finding a feasible solution may fail, e.g. in narrow passages. Common graph search techniques include Dijkstra's algorithm, A*, and D* with their variants (Paden et al. (2016)). Exemplary, Rizano et al. (2013) describe the graph construction for racing applications and use a Dijkstra-like search algorithm. Depending on the vehicle model fidelity and the discretization grid size, the computation times of graph search quickly rise. Techniques based on nonlinear continuous optimization (aka variational methods) tend to perform better in these cases, although, they only provide locally optimal solutions unless the initial guess is sufficiently close to the global optimum (Paden et al. (2016)). An application within the racing domain is shown in Heilmeier et al. (2020a), where first a minimum curvature path is computed via solving of a quadratic program (QP). Afterwards, the velocity profile is generated such that the acceleration limits are not violated. Many variational methods have strong similarities to nonlinear model predictive control (NMPC, see e.g. Johansen (2011)), but instead of directly applying their predicted model inputs to the plant, their computed trajectories serve as references for underlying controllers. Alcalá et al. (2020) demonstrate how to express an NMPC formulation of the trajectory planning problem in a linear parameter varying form. In Liniger et al. (2015), a formulation as model predictive contouring control problem leads to a progress maximization on the race track's center line. Bergman and Axehill (2017) introduce a homotopy strategy to account for the strong nonlinearity of the obstacles. In both papers obstacles are considered via variations of the track boundaries. Thereby, the decision on which side an obstacle should be bypassed is delegated to a higher-level planner. Bergman et al. (2019) and Bergman et al. (2020) consider the combinatorial problem by combining optimal control with lattice-based path planning. Their work addresses the same underlying problem, but focuses on unstructured environment like parking lots. Schouwenaars et al. (2001) and Richards et al. (2002) illustrate how to integrate these binary decisions into a single mixed-integer linear program (MILP) and show its application to solve obstacle avoidance for vehicles and spacecrafts. In the context of controller design Janeček et al. (2017) use similar ideas in an MPC based on mixed-integer QP (MIQP) for avoiding obstacles. Park et al. (2015) first decompose the collision free space into convex cells which are connected to distinct homotopy classes and subsequently solve MIQPs for each of them. Thereby, the globally optimal solution is selected from the local optima of each homotopy class.

1.2 Contribution

This paper proposes a planning procedure considering obstacles and rewards which consists of both offline and online steps. Prior to the actual race, an optimal *racing line* for the given track geometry is computed, assuming that there are no obstacles. This is done with an approach similar to Heilmeier et al. (2020a), however, since it is not the focus of this paper, further details are omitted. The online part is twofold: First, a MILP is formulated whose solution selects a homotopy class, i.e. which rewards are collected and on which side obstacles are eluded. This homotopy class is represented by deformed boundaries of the original track. Secondly, a continuous optimization problem is stated where deflections from the racing line are minimized, considering the modified boundaries from the MILP and vehicle constraints. The NMPC-like optimization problem is not directly solved, instead homotopy iterations are performed which gradually put more weight on the modified boundaries' adherence and improve the convergence of the solver. The paper contributes with a motion planning approach which is capable of solving time-optimal motion planning problems with a combinatorial structure, without fully discretizing the state-space, as opposed to graph-based state-of-the-art algorithms.

2. VEHICLE AND OBJECT MODELS

Separating the trajectory optimization from the combinatorial homotopy class selection facilitates using different models. For solving the combinatorial obstacle and reward problem, an utterly simplified geometric model is utilized which represents a deviation from the racing line with limited maximum steepness. The continuous optimization problem is stated on basis of a single-track model.

2.1 Single-track model for continuous optimization

The utilized kinematic model is given for instance in Kloeser et al. (2020) and does not consider tire slip. In order to reduce the computation time of the optimizer later on, drifting motion is not considered during planning. Instead, the subsequent trajectory following controller is assumed to take lateral and longitudinal tire slip into

account. The movement direction of the center of gravity (CG) with the vehicle mass m is given by the angle $\psi + \beta$, where ψ is the vehicle orientation. The side slip angle

$$\beta = \arctan\left(\frac{l_\mathrm{r}}{l_\mathrm{r} + l_\mathrm{f}} \tan\delta\right) \tag{1}$$

is defined as depicted in Fig. 2 and gives the relative angle of motion related to the vehicle coordinate system (Kloeser et al. (2020)). The vehicle motion is governed by

$$\dot{p}_\mathrm{X} = v\cos(\psi + \beta) \tag{2a}$$
$$\dot{p}_\mathrm{Y} = v\sin(\psi + \beta) \tag{2b}$$
$$\dot{\psi} = \frac{v}{l_\mathrm{r}}\sin\beta \tag{2c}$$
$$\dot{v} = \frac{F_\mathrm{x}^\mathrm{d}}{m}\cos\beta \tag{2d}$$

in the Cartesian coordinate frame where v is the longitudinal velocity in movement direction of the CG. The vehicle's geometry is described by the longitudinal position of the CG with front distance l_f and rear distance l_r. The input force F_x^d only acts on the rear wheel, whereas the steering angle δ only deflects the front wheel. As second input the steering rate $r = \dot{\delta}$ is utilized to avoid discontinuities in the steering angle δ, which would arise with directly choosing δ as an input.

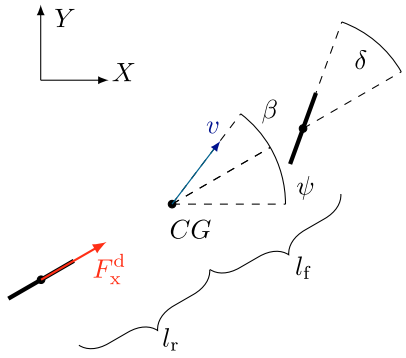

Fig. 2. Kinematic single-track model.

2.2 Frenet transformation

System (2) is transformed into Frenet coordinates as described in Kloeser et al. (2020) with the difference that this transformation is performed along the racing line instead of the center path. The resulting nonlinear dynamic system $\dot{x} = f(x, u)$ now depends on the curvature $\kappa(s)$ and reads as

$$\dot{s} = \frac{v\cos(\alpha + \beta)}{1 - n\kappa(s)} \tag{3a}$$
$$\dot{n} = v\sin(\alpha + \beta) \tag{3b}$$
$$\dot{\alpha} = \dot{\psi} - \kappa(s)\dot{s} \tag{3c}$$
$$\dot{v} = \frac{F_\mathrm{x}^\mathrm{d}}{m}\cos(\beta) \tag{3d}$$
$$\dot{\delta} = r \tag{3e}$$

with states $x = [s,\ n,\ \alpha,\ v,\ \delta]^T$ and controls $u = \left[F_\mathrm{x}^\mathrm{d},\ r\right]^T$. Path aligned states are used which describe the progress on the transformation path $s(t)$, the normal distance to the transformation path $n(t)$ and the heading angle mismatch $\alpha(s,t) = \psi(t) - \psi_\mathrm{r}(s)$. Note that $\kappa(s)$ together with bounds on the normal distance state n now fully describe the road geometry. Fig. 3 shows the transformation of a point p

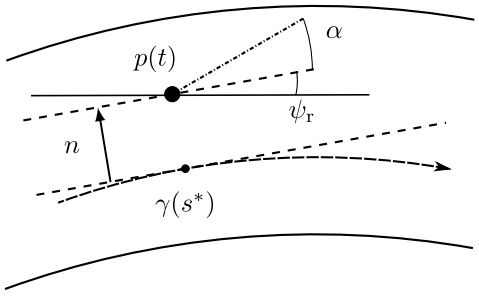

Fig. 3. Path-parametric model as in Kloeser et al. (2020).

with respect to the curve $\gamma(s)$, normal distance n, the error angle α and the heading angle ψ_r of $\gamma(s)$. The path parameter s^* corresponds to the parameter of the closest point $\gamma(s^*)$ to the given position p of the vehicle model. The lateral acceleration of this model can be stated as

$$a_\mathrm{lat} = \frac{1}{l_\mathrm{f} + l_\mathrm{r}} v^2\delta + \frac{F_\mathrm{x}^\mathrm{d}}{m}\sin\left(\frac{\delta l_\mathrm{r}}{l_\mathrm{f} + l_\mathrm{r}}\right). \tag{4}$$

2.3 Linear model for mixed-integer programming

For the integer problem an utterly simplified model is used, which basically just accounts for a geometric offset from the Frenet transformation line and is formulated in the Frenet frame as well. The model lacks time dependency, it rather depends on the path progress s, which is discretized on a grid related to the obstacles and written as s_k. The discretization is performed to represent the obstacle shape related to the racing line. The lateral distance n_k is limited based on the road constraints $n_{k,\mathrm{left}}$ and $n_{k,\mathrm{right}}$. The continuous control variable u relates the only state variables n_k to each other and is split into a negative and positive part, due to the integer formulation of the optimization objective. The control variable u expresses the steepness of the path deviation from the racing line between two nodes and is limited by u_max. The model can be written as

$$\begin{aligned} n_{k+1} &= f_k(u_k^\mathrm{pos}, u_k^\mathrm{neg}) \\ &= n_k + (u_k^\mathrm{pos} - u_k^\mathrm{neg})(s_{k+1} - s_k) \end{aligned} \tag{5a}$$
$$0 \le u_k^\mathrm{pos} \le u_\mathrm{max} \tag{5b}$$
$$0 \le u_k^\mathrm{neg} \le u_\mathrm{max} \tag{5c}$$
$$n_\mathrm{l,right} \le n_l \le n_\mathrm{l,left} \tag{5d}$$

for $k = 0...N_\mathrm{d} - 1$ and $l = 0...N_\mathrm{d}$, where N_d is the number of discretizations of the longitudinal path variable s. This model describes a piece-wise linear path in Frenet coordinates.

2.4 Object representation

Objects are modeled in the two dimensional Frenet frame by means of polygons as shown in Fig. 4. A particular obstacle i of totally N_O obstacles is represented by a polygon with $N_\mathrm{p}^{\mathrm{O}i}$ points and a reward d of N_R rewards with $N_\mathrm{p}^{\mathrm{R}d}$ points respectively. The object's vertices are aligned with the discretization of the longitudinal coordinate, which is chosen such that it resembles the shape of the objects "well enough". Therefore, a vertex always has an opposite side

point with the same longitudinal coordinate. Right or left sides are noted with $\{r, l\}$. The coordinate points in the Frenet frame of the polygon characterizing each reward or obstacle are written as

$$p_k^{\mathrm{R}d,\{\mathrm{r,l}\}} = \begin{bmatrix} s_k^{\mathrm{R}d} & n_k^{\mathrm{R}d,\{\mathrm{r,l}\}} \end{bmatrix}^T \tag{6a}$$

$$p_k^{\mathrm{O}i,\{\mathrm{r,l}\}} = \begin{bmatrix} s_k^{\mathrm{O}i} & n_k^{\mathrm{O}i,\{\mathrm{r,l}\}} \end{bmatrix}^T \tag{6b}$$

with k as the global spatial index of the longitudinal Frenet axis.

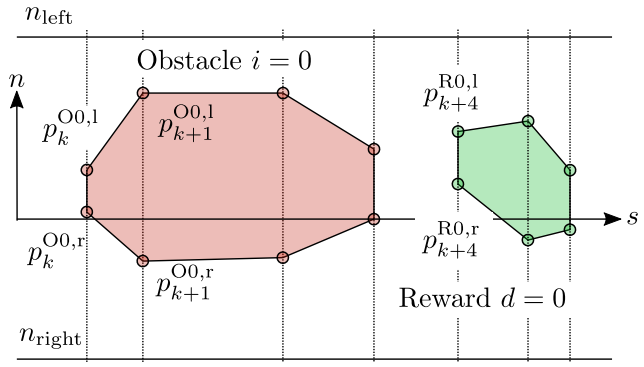

Fig. 4. Object configuration in Frenet frame.

3. COMBINATORIAL OPTIMIZATION

3.1 Binary object-boundary relation

In order to find an optimal path through the obstacle setting in the Frenet frame with the reduced model (5), binary integer variables are used to construct a linear mixed-integer problem. For each obstacle i, one binary variable ω^i indicates the passing side (left/right). Rewards are treated differently because they could be catched on different positions. For "short" rewards (where "short" is related to the longitudinal extention of the reward) one binary variable β^d sufficiently specifies whether this reward should be caught or ignored. For long rewards "gates" with several binary variables β_k^d are used. For each long reward d, totally $N_{\mathrm{p}}^{\mathrm{R}d}/2$ binary variables are used, which are indexed by $l = 0 \dots (N_{\mathrm{p}}^{\mathrm{R}d}/2 - 1)$. Each binary variable β_l^d indicates, if boundaries are set to "gate" the corresponding lateral state variable at the particular longitudinal position of the reward. A logical OR connective of the binary "gate" variables

$$\bar{\beta}^d = \beta_k^d \vee \dots \vee \beta_{k+N_{\mathrm{p}}^{\mathrm{R}d}-1}^d \tag{7}$$

indicates the final binary state for reward d, which is then used to specifiy the associated cost. That means if at least one gate is "closed" (meaning the binary variable sets the boundaries active), the final path is crossing the reward polygon at some point. This formulation leads to a high number of decision variables, which is necessary if a reward can be caught at multiple positions. For example a very long "reward zone" aligned with the road might be entered at very different positions. With known shapes of the rewards, particularly where the longitudinal length in Frenet frame is "small" (e.g. smaller than 5 meters in the setting used for the competition) (7) can be simplified by taking the left-most and the right-most polygon points to define a "gate", that is either switched active or inactive by just one integer variable β^d.

The binary variables are subsequently used to adjust boundaries forming a homotopy class for the gradient-based optimization. Detailed considerations about homotopy classes in this context are shown in Bergman and Axehill (2017). Obstacle boundary values $n_k^{\mathrm{O}i,\{\mathrm{r,l}\}}$ and its binary variables ω_i are related to state constraint on n_k with the inequalities

$$n_k \geq \omega^i n_k^{\mathrm{O}i,\mathrm{r}} + (1-\omega^i) n_{k,\mathrm{right}} = \mathcal{H}_{\mathrm{low}}(k,\omega) \tag{8}$$

$$n_k \leq (1-\omega^i) n_k^{\mathrm{O}i,\mathrm{l}} + \omega^i n_{k,\mathrm{left}} = \mathcal{H}_{\mathrm{upp}}(k,\omega), \tag{9}$$

for $k = 0 \dots N_{\mathrm{d}}$ where a binary state equal "0" is defined as "passing left". For relating the borders to the reward polygon, the equations

$$n_k \geq \beta_k^d n_k^{\mathrm{R}d,\mathrm{r}} - (1-\beta_k^d) n_{k,\mathrm{right}} = \mathcal{I}_{\mathrm{low}}(k,\beta) \tag{10}$$

$$n_k \leq \beta_k^d n_k^{\mathrm{R}d,\mathrm{l}} - (1-\beta_k^d) n_{k,\mathrm{left}} = \mathcal{I}_{\mathrm{upp}}(k,\beta), \tag{11}$$

are used, where the binary variable equal to "1" is defined as "catch". Fig. 5 shows the changed bounds due to the integer variables.

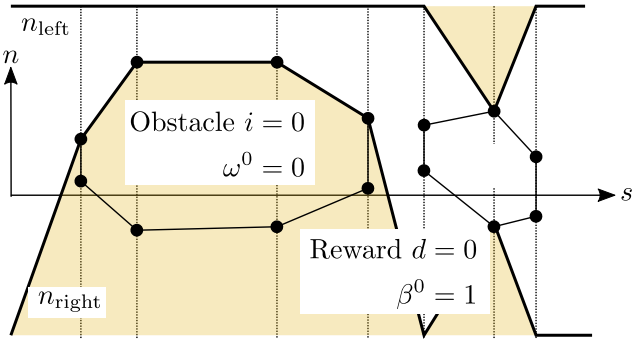

Fig. 5. Road bounds aligned to object integer variables.

3.2 Decision-flickering avoidance

During real-world conditions on vehicle hardware the perception system receives slightly shifted versions of the same problem, where just infrequently new objects appear. Very often, two solutions nearly have the same cost (e.g. an obstacle in the middle of the road) which could lead to jumping binary variables. For that reason, a penalty for changes to previous decisions is introduced that is increasing with the number this decision was computed previously by the algorithm and decreasing with the distance of the object to the vehicle. In other words, binary re-decisions for objects that are far from the vehicle and which have just appeared are "cheaper" than re-decisions for closer and longer present objects. To account for re-decisions, the binary values of c_i^{R} and c_i^{O} are used to form a logic XOR connective between the binary decision variables β_i or ω_i and the previous decisions $\hat{\beta}_i$ or $\hat{\omega}_i$.

$$c_i^{\mathrm{R}} = \beta_i(1-\hat{\beta}_i) + (1-\beta_i)\hat{\beta}_i \tag{12}$$

$$c_i^{\mathrm{O}} = \omega_i(1-\hat{\omega}_i) + (1-\omega_i)\hat{\omega}_i \tag{13}$$

A parameter p_i accounts for the weight that this decision should be kept. The weight is updated iteratively and

related to the mentioned criteria of distance and decision account by

$$p_i = w_{0,\text{bin}} \max\left(1 - \frac{\Delta s_i}{d_0}, 0\right) \frac{\exp(a_0 c_i)}{1 + \exp(a_0 c_i)}, \tag{14}$$

with $w_{0,\text{bin}}, d_0$ and a_0 as scaling constants, c_i as the counter of decision repetitions for each binary variable and Δs_i as the longitudinal distance of an obstacle corresponding to the binary variable i to the vehicle position. In (26) the related cost function for re-decisions is stated.

3.3 Cost functions

The final cost function consists of several parts. First, the L1-norm of the deviation from the Frenet-transformation line (i.e. the optimal racing line) is stated. It is proportional by a weight w_n to the area of absolute lateral error in the Frenet frame and computed as

$$C_\text{n} = w_\text{n} \sum_{k=0}^{N_d} |n_k|. \tag{15}$$

The L1-norm cannot be directly formulated as linear function. Therefore (5) is used, with the splitting of u into a positive and negative part according to a reformulation shown in Boyd and Vandenberghe (2004). With $\Delta s_k := s_{k+1} - s_k$, the variable $n_k = n_k^\text{pos} - n_k^\text{neg}$ and the cost C_n can also be written as

$$C_\text{n} = w_\text{n} \sum_{k=0}^{N_d} n_k^\text{pos} + n_k^\text{neg} \tag{16}$$

$$\text{with } n_k^\text{pos} \geq 0,\ n_k^\text{neg} \geq 0. \tag{17}$$

The positive and negative parts of n_k can be directly obtained by summing up the associated controls u_k. The inequalities (17) are therefore fulfilled with (5). The initial value n_0 is a constant and can also be "split" into a positive n_0^pos and a negative n_0^neg with $n_0 = n_0^\text{pos} - n_0^\text{neg}$. Consequently, n_{k+1} can be decomposed according to the following steps.

$$n_{k+1} = n_k + \Delta s_k u_k \tag{18a}$$

$$n_{k+1} = n_0 + \sum_{i=0}^{k} \Delta s_i u_i \tag{18b}$$

$$n_{k+1} = (n_0^\text{pos} + \sum_{i=0}^{k} \Delta s_i u_i^\text{pos}) - (n_0^\text{neg} + \sum_{i=0}^{k} \Delta s_i u_i^\text{neg}) \tag{18c}$$

$$n_{k+1} = n_{k+1}^\text{pos} - n_{k+1}^\text{neg} \tag{18d}$$

The result (18d) is used to state (16) and therefore the cost in (15).

Secondly, the steepness of the deviation from the racing line defines the cost term $C_{\Delta\text{n}}$ with its associated cost $w_{\Delta\text{n}}$ in a similar way by using the approach of Boyd and Vandenberghe (2004) for the "basis pursuit problem". Since the controls u_k represent the steepness of deviation, the decomposed parts of (5) can be added as costs for the steepness by

$$C_{\Delta\text{n}} = w_{\Delta\text{n}} \sum_{k=0}^{N_d} (u_k^\text{pos} + u_k^\text{neg}). \tag{19}$$

Thirdly, negative costs are added for catching a reward by

$$C_\text{rew} = \begin{cases} C_\text{rew}^\text{ext}, & \text{if } l_\text{R} \geq \bar{l}_\text{R} \\ C_\text{rew}^\text{short}, & \text{otherwise} \end{cases}. \tag{20}$$

These reward costs with their associated negative weight w_rew are split into extended costs C_rew^ext for long rewards (with "long" referring to the longitudinal dimension l_R in the Frenet frame) and simplified ones $C_\text{rew}^\text{short}$ for short rewards. A threshold $\bar{l}_\text{R}$ is used for their classification. The simpler version of the problem for longitudinally short rewards reads as

$$C_\text{rew}^\text{short} = w_\text{rew} \sum_{i=0}^{N_\text{R}-1} \beta^i. \tag{21}$$

In the extended reward formulation (25) the logic OR connective is needed. Auxiliary continuous optimization variables $x^{\text{R}i}$ for the cost reduction of reward i are introduced. These auxiliary variables realize the OR connective between the l reward "gates" associated with their binary "gating" variable β_l^i when being minimized in the overall optimization problem. The set $\mathcal{K}_\text{rew}$ restricts the auxiliary cost variable to the (negative) weight value w_rew as a general lower bound or to 0 if all "gates" are open, thus their binary values are zero. This set reads as

$$\begin{aligned} \mathcal{K}_\text{rew}(\beta) = \Big\{ & x^\text{R} \in \mathbb{R}^{N_\text{R}} | x^{\text{R}i} \geq w_\text{rew}, \\ & x^{\text{R}i} \geq w_\text{rew} \sum_{l=0}^{N_{\text{R}i}/2-1} \beta_l^i,\ i = 0 \ldots N_\text{R} - 1 \Big\} \end{aligned} \tag{22}$$

with cost summarized as

$$\beta = \left[\beta^0, \ \ldots, \ \beta^{(N_\text{R}-1)}\right], \tag{23}$$

$$\omega = \left[\omega^0 \ \ldots, \ \omega^{(N_\text{O}-1)}\right]. \tag{24}$$

The extended reward costs

$$C_\text{rew}^\text{ext} = \sum_{i=0}^{N_\text{R}-1} x^{\text{R}i} \text{ with } x^\text{R} \in \mathcal{K}_\text{rew}(\beta) \tag{25}$$

are simply the sum of auxiliary variables realizing the OR connective under their minimization.

Finally, the "re-decision costs" C_bin penalize toggling of binary variables. Here, the weights p_i are different for every binary variable, as described in (14). The variable $c_i^{\text{O,R}}$ for obstacles and rewards indicates if the binary variable for the object has changed, which is equal to an XOR logic connective in (12) and (13). Please note that for each reward there might be several binary variables, therefore $\bar{N}_R$ is used to account for all binary variables related to rewards.

$$C_\text{bin} = \sum_{i=0}^{\bar{N}_\text{R}-1} p_i c_i^\text{R} + \sum_{i=0}^{N_\text{O}-1} p_i c_i^\text{O} \tag{26}$$

3.4 Final problem formulation

For the ease of notation, the continuous optimization variables are combined in

$$u = \begin{bmatrix} u_0^\text{neg} & \ldots & u_{N_d-1}^\text{neg} \\ u_0^\text{pos} & \ldots & u_{N_d-1}^\text{pos}. \end{bmatrix} \tag{27}$$

With the set of binary numbers $\mathcal{B} = \{0, 1\}$, the final mixed-integer problem (28) is stated by combining the model (5)

with the cost model function (16) by using the result of (18c) splitting the race path deviation n_k.

$$\min_{\substack{u\in\mathbb{R}^{2\times N_d},\\ \beta\in\mathcal{B}^{N_\beta},\\ \omega\in\mathcal{B}^{N_\omega},\\ x^{\mathrm{R}}\in\mathbb{R}^{N_{\mathrm{R}}}}} \quad C_{\mathrm{n}}(u)+C_{\Delta\mathrm{n}}(u)+C_{\mathrm{rew}}(x^{\mathrm{R}})+C_{\mathrm{bin}}(\beta,\omega) \tag{28a}$$

$$\text{s.t.} \quad n_{k+1}=f_k(u_k^{\mathrm{pos}},u_k^{\mathrm{neg}}), \tag{28b}$$
$$0\le u_k^{\mathrm{pos}}\le u_{\max}, \tag{28c}$$
$$0\le u_k^{\mathrm{neg}}\le u_{\max}, \tag{28d}$$
$$\mathcal{I}_{\mathrm{low}}(k,\beta)\le n_k\le\mathcal{I}_{\mathrm{upp}}(k,\beta), \tag{28e}$$
$$\mathcal{H}_{\mathrm{low}}(k,\omega)\le n_k\le\mathcal{H}_{\mathrm{upp}}(k,\omega), \tag{28f}$$
$$\text{with } k=0,\dots,N_d-1,$$
$$x^{\mathrm{R}}\in\mathcal{K}_{\mathrm{rew}}(\beta) \tag{28g}$$

Also, the reduced costs for reward catching (21) as well as the costs for keeping binary variables in (26) are included. N_β and N_ω denote the total count of all binary decision variables for rewards and obstacles.

4. TRAJECTORY OPTIMIZATION

After solving (28), a homotopy class is computed from the object polygons and their associated binary states. The road bounds are described by $\underline{n}(s)$ and $\overline{n}(s)$ which linearly interpolate the original road boundary points or the associated object boundaries (see Fig. 5) according to the homotopy class. As described in Kloeser et al. (2020), the optimal control problem (OCP) of time-optimal racing can be described very generally by the following multiple shooting nonlinear program (NLP)

$$\min_{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1}\\ \zeta_0,\dots,\zeta_N}} \quad \sum_{k=0}^{N-1}\|x_k-x_{k,\mathrm{ref}}\|_Q^2+\|u_k\|_R^2+ \tag{29a}$$
$$\mu_i\|\zeta_k\|^2+\nu_i\|\zeta_k\|_1+\|x_N-x_{N,\mathrm{ref}}\|_{Q_{\mathrm{N}}}^2$$

$$\text{s.t.} \quad x_0=\bar{x}_0, \tag{29b}$$
$$x_{k+1}=F(x_k,u_k,\Delta t), \tag{29c}$$
$$\underline{u}\le u_k\le\overline{u}, \tag{29d}$$
$$\underline{x}\le x_k\le\overline{x}, \tag{29e}$$
$$\underline{n}(s_k)-\zeta_k\le n_k\le\overline{n}(s_k)+\zeta_k, \tag{29f}$$
$$-\overline{a}_{\mathrm{lat}}\le a_{\mathrm{lat}}(x_k)\le\overline{a}_{\mathrm{lat}}, \tag{29g}$$
$$\zeta_k\ge 0, \tag{29h}$$
$$\text{with } k=0,\dots,N-1.$$

It represents a tracking problem of a vehicle model, where the final reference point $x_{N,\mathrm{ref}}$ is set "out-of-reach" to obtain approximate time-optimal trajectories. Here, the 1.2-fold maximum achievable distance was chosen for the final reference point, where the maximum distance would correspond to the distance obtained by the maximum speed driven for the given time interval $(N+1)\Delta t$. Its structure of a tracking problem and the associated quadratic cost function allows the usage of a fast Gauss-Newton Hessian approximation. As opposed to Kloeser et al. (2020), the reference in (29) is set as a previously approximated optimal racing path, rather than the center line of the road. The final values for the cost function weightings were tuned by experiments and are shown in Table 2. The vehicle model (3) is discretized with an integration scheme $F(x_k,u_k,\Delta t)$ with fixed time intervals Δt and incorporated as (29c) into the NLP. Inequality (29e) puts box constraints on the states, which include maximum velocity and steering angle. The lateral state constraints (29f) represent the road boundary and dependent on the longitudinal state variable s. Slack variables ζ_k for violating (29f) are used. An iterative procedure increases their penalty weighting in consecutive optimization iterations, which we call homotopy iterations. The lateral acceleration (4) is limited via (29g).

After n_{SQP} sequential quadratic programming (SQP) iterations of the solver, the weighting parameters μ_i and ν_i are increased as shown in Algorithm (1). Strictly increasing scheduling functions $\alpha_\mu(i)$ and $\alpha_\nu(i)$ govern the weighting of the boundary slack variables. This lets the weights for the boundary slack variables "grow", which leads to a smooth transition of the boundary non-linearity. With this procedure, the convergence time is reduced, which is shown in the results Section 5.2. As a drawback the re-weighting might be unnecessary for smooth obstacle boundaries and takes additional time.

$i=0$;
while $i\le i_{\max}$ **do**
  $\mu\leftarrow\alpha_\mu(i)$;
  $\nu\leftarrow\alpha_\nu(i)$;
  solve NLP with n_{SQP} iterations;
  $i\leftarrow i+1$;
end

Algorithm 1: Homotopy iterations for the NLP.

5. REAL-WORLD AND SIMULATION RESULTS

The presented strategies for combinatorial optimization by mixed-integer programming (COMIP) and the trajectory optimization of Section 4 (TO) are two independent algorithms, which were tested in two settings. Both, COMIP and TO were tested in simulations together. The COMIP algorithm was furthermore used on embedded hardware in the third Roborace competition in its so called "season beta" (second series of competitions in 2020), where the presented TO was replaced by a simpler decoupled trajectory planning algorithm. The competition took place on the Bedford race circuit (England) in December 2020.

5.1 Field test on the Bedford race circuit (COMIP only)

The proposed COMIP algorithm was tested on the NVIDIA DRIVE PX2, with Ubuntu 16.04. This electronic control unit (ECU) provides two CPUs (4x ARM Denver, 8x ARM Cortex A57) and two GPUs (2x Tegra X2, 2x Pascal GPU). The open-source Coin-OR CBC solver was used in a mixed Python/C++ ROS-framework for solving problem (28) with a nominal rate of 0.5 Hz. A varying amount between 0 and 50 virtual objects was received in generally random sizes but rectangular shapes in Cartesian coordinates. The race car "devbot" is described in Roborace (2020). This algorithm was also tested in simulations with the second setup, namely a HP Elitebook with an Intel Core i7-8550 CPU (1.8 GHz), which turned out to be faster by a factor of 4-20. In this setup, the output of the COMIP was used together with a subsequent algorithm for curvature minimization (which is a

simplified and decoupled approach compared to the one described in Section 5.2). It obtained a minimum curvature path approximation as well as a final analytical speed maximization based on this minimum curvature path and road friction parameters as shown in Subosits and Gerdes (2015) and Kapania et al. (2016). According to Braghin et al. (2008) and Heilmeier et al. (2020b), the minimum curvature path is a good approximation for the optimal race path. The curvature approximation was computed with the algorithm described in Heilmeier et al. (2020b), but with only first order differences for race line deviations. Fig. 6 illustrates planning results from this event. Table 1 shows the maximum computation times for the combined algorithm (due to logging limitations by the embedded system), where the COMIP part accounts for 70% of the computation time on average.

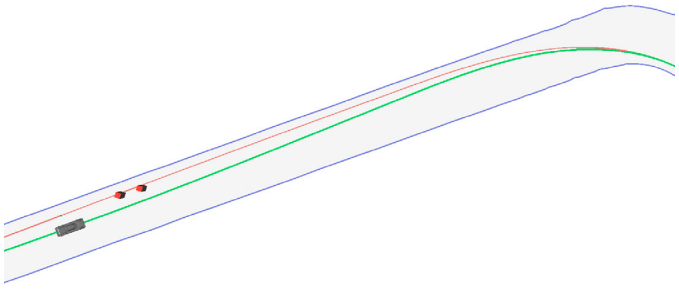

Fig. 6. Visualization of recorded data of an evasion maneuver on the Bedford race circuit. The track boundaries are shown in blue and the racing line (Frenet transformation line) in red. The car follows the computed trajectory that is shown in green.

Table 1. Maximum computation times of the overall optimization algorithm

	ECU	Simulation
max. comp. time	6.0s	0.4s
av. comp. time	1.9s	0.2s

5.2 Simulation in virtual environments (COMIP + TO)

The COMIP algorithm was also tested extensively on different race tracks with different object settings, including up to 40 differently shaped objects simultaneously within a prediction horizon of 300 meters. The measured times for the simulated algorithm on the Bedford race circuit are shown in Table 1.

The proposed subsequent TO was tested on the same simulation setup. After COMIP, the boundaries in (29f) were approximated by linear splines with a spatial discretization interval of the longitudinal coordinate s of 1 meter. For solving the NLP in (29), the kinematic vehicle model of Section 2.1 was used with a center of gravity at lengths $l_\mathrm{r} = 1.4\mathrm{m}$ and $l_\mathrm{f} = 1.6\mathrm{m}$ and parameters according to Table 2. The NLP was solved with acados (Verschueren et al. (2018)), where a Gauss-Newton Hessian approximation was used together with a two-stage implicit Runge-Kutta integration scheme. For solving the quadratic program (QP), the interior point solver HPIPM (Frison and Diehl (2020)) was used. Altogether, with Algorithm (1) 8 SQP

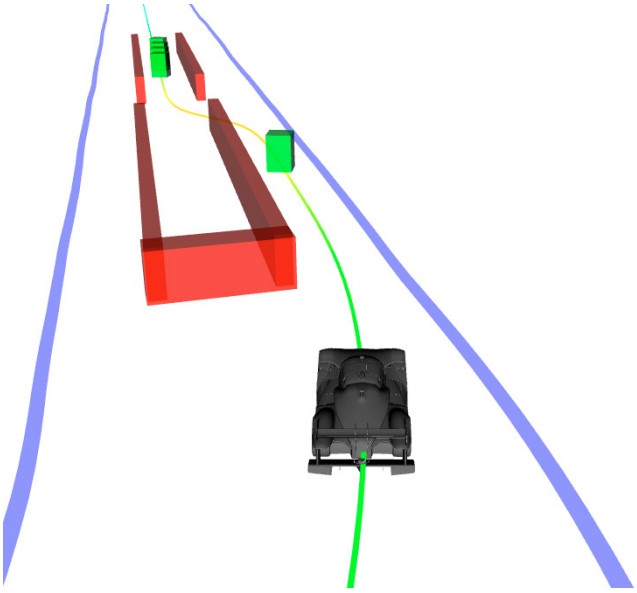

Fig. 7. Trajectory obtained by the presented algorithms in a ROS simulation framework with obstacles (red) and rewards (green).

Table 2. Parameters for Algorithm 1

Parameter Name	Value
Q	$\begin{bmatrix}10^{-6} & 10^{-3} & 1 & 10^{-4} & 10^{-1}\end{bmatrix}^T$
Q_N	$\begin{bmatrix}10^{-2} & 10^{-1} & 10^{-2} & 10^{-4} & 2\cdot 10^{-3}\end{bmatrix}^T$
R	$\begin{bmatrix}10^{-3} & 10^{-2}\end{bmatrix}^T$
$\alpha_\mu(i)$	10^i
$\alpha_\nu(i)$	$0.1\cdot 10^{0.7i}$
i_max	3
n_SQP	2
$\overline{a}_\mathrm{lat}$	$5\frac{m}{s}$
N	100
Δt	$0.05s$

iterations are performed, with different weights for the slack variables, accounting for the homotopy iterations. The algorithm was compared to a standard setting with 8 SQP iterations on the final slack weights according to $i = 3$ in Table 2. With the constant slack weight setting the NLP solver acados could not find solutions for several obstacle configurations, where either QP iterations failed or the solution trajectory got stuck in front of obstacles. With the presented homotopy iterations both problems were mitigated. The results are summarized in Table 3.

Table 3. Results for Algorithm 1

Name	Value
Single SQP iteration comp. time (mean)	0.009s
Single SQP iteration comp. time (min)	0.006s
Single SQP iteration comp. time (min)	0.02s
Total SQP iterations	8
Total NLP solution time (mean)	0.072s
QP iterations (mean)	8
QP iterations (max)	10
QP iterations (min)	6

6. CONCLUSION

This work presents two contributions to the sophisticated sub-problems of trajectory planning for autonomous racing. First, a novel approximation of the combinatorial problem as Frenet-frame based linear mixed-integer problem is derived, which allows a fast and robust computation of a distinct homotopy class. Secondly, a homotopy strategy is presented to obtain robust convergence of a consecutive NLP. By using these approaches on a real embedded setup, it is verified to achieve high performance on novel autonomous race car competitions and provide an alternative to the full state-space discretization of graph-search methods. Further considerations may include the combination of homotopy iterations with the integer problem and the extension to time varying objects.

ACKNOWLEDGEMENTS

The authors thank Adrian Bürger (University of Freiburg and Karlsruhe University of Applied Sciences) for his valuable input to integer optimization and the "Autonomous Racing Graz" team for developing the software stack that embeds the presented algorithms. Furthermore, the authors thank Jonathan Frey (University of Freiburg) for his assistance integrating the acados solver.

REFERENCES

Alcalá, E., Puig, V., and Quevedo, J. (2020). Lpv-mp planning for autonomous racing vehicles considering obstacles. *Robotics and Autonomous Systems*, 124, 103392. doi:https://doi.org/10.1016/j.robot.2019.103392.

Bergman, K. and Axehill, D. (2017). Combining Homotopy Methods and Numerical Optimal Control to Solve Motion Planning Problems. *arXiv:1703.07546 [math]*. 00015 arXiv: 1703.07546.

Bergman, K., Ljungqvist, O., and Axehill, D. (2020). Improved Path Planning by Tightly Combining Lattice-Based Path Planning and Optimal Control. *IEEE Transactions on Intelligent Vehicles*, 1–1. doi: 10.1109/TIV.2020.2991951. 00005.

Bergman, K., Ljungqvist, O., Glad, T., and Axehill, D. (2019). An Optimization-Based Receding Horizon Trajectory Planning Algorithm. *arXiv:1912.05259 [math]*. 00003 arXiv: 1912.05259.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, USA.

Braghin, F., Cheli, F., Melzi, S., and Sabbioni, E. (2008). Race driver model. *Computers & Structures*, 86(13-14), 1503–1516. doi:10.1016/j.compstruc.2007.04.028. 00094.

Frison, G. and Diehl, M. (2020). Hpipm: a high-performance quadratic programming framework for model predictive control.

Heilmeier, A., Wischnewski, A., Hermansdorfer, L., Betz, J., Lienkamp, M., and Lohmann, B. (2020a). Minimum curvature trajectory planning and control for an autonomous race car. *Vehicle System Dynamics*, 58(10), 1497–1527. doi:10.1080/00423114.2019.1631455.

Heilmeier, A., Wischnewski, A., Hermansdorfer, L., Betz, J., Lienkamp, M., and Lohmann, B. (2020b). Minimum curvature trajectory planning and control for an autonomous race car. *Vehicle System Dynamics*, 58(10), 1497–1527. doi:10.1080/00423114.2019.1631455. 00021.

Janeček, F., Klaučo, M., Kalúz, M., and Kvasnica, M. (2017). Optiplan: A matlab toolbox for model predictive control with obstacle avoidance. *IFAC-PapersOnLine*, 50(1), 531 – 536. doi: https://doi.org/10.1016/j.ifacol.2017.08.057. 20th IFAC World Congress.

Johansen, T.A. (2011). Introduction to nonlinear model predictive control and moving horizon estimation. In M. Huba, S. Skogestad, M. Fikar, M. Hovd, T.A. Johansen, and B. Rohal'-Ilkiv (eds.), *Selected Topics on Constrained and Nonlinear Control*, chapter 5.

Kapania, N.R., Subosits, J., and Christian Gerdes, J. (2016). A Sequential Two-Step Algorithm for Fast Generation of Vehicle Racing Trajectories. *Journal of Dynamic Systems, Measurement, and Control*, 138(9), 091005. doi:10.1115/1.4033311. 00043.

Kloeser, D., Schoels, T., Sartor, T., Zanelli, A., Frison, G., and Diehl, M. (2020). NMPC for racing using a singularity-free path-parametric model with obstacle avoidance. In *Proceedings of the IFAC World Congress*.

Liniger, A., Domahidi, A., and Morari, M. (2015). Optimization-based autonomous racing of 1:43 scale RC cars. *Optimal Control Applications and Methods*, 36(5), 628–647. doi:10.1002/oca.2123.

Paden, B., Čáp, M., Yong, S.Z., Yershov, D., and Frazzoli, E. (2016). A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1), 33–55. doi: 10.1109/TIV.2016.2578706.

Park, J., Karumanchi, S., and Iagnemma, K. (2015). Homotopy-Based Divide-and-Conquer Strategy for Optimal Trajectory Planning via Mixed-Integer Programming. *IEEE Transactions on Robotics*, 31(5), 1101–1115. doi:10.1109/TRO.2015.2459373. 00054 Conference Name: IEEE Transactions on Robotics.

Richards, A., Schouwenaars, T., How, J.P., and Feron, E. (2002). Spacecraft Trajectory Planning with Avoidance Constraints Using Mixed-Integer Linear Programming. *Journal of Guidance, Control, and Dynamics*, 25(4), 755–764. doi:10.2514/2.4943. 00492.

Rizano, T., Fontanelli, D., Palopoli, L., Pallottino, L., and Salaris, P. (2013). Global path planning for competitive robotic cars. In *52nd IEEE Conference on Decision and Control*, 4510–4516. doi:10.1109/CDC.2013.6760584.

Roborace (2020). Roborace season beta. URL `https://roborace.com/`.

Schouwenaars, T., De Moor, B., Feron, E., and How, J. (2001). Mixed integer programming for multi-vehicle path-planning. *European Control Conf.* 00580.

Subosits, J. and Gerdes, J.C. (2015). Autonomous vehicle control for emergency maneuvers: The effect of topography. In *2015 American Control Conference (ACC)*, 1405–1410. IEEE, Chicago, IL, USA. doi: 10.1109/ACC.2015.7170930. 00018.

Verschueren, R., Frison, G., Kouzoupis, D., van Duijkeren, N., Zanelli, A., Quirynen, R., and Diehl, M. (2018). Towards a modular software package for embedded optimization. In *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (NMPC)*, 374–380. Madison, Wisconsin (USA).