2023 European Control Conference (ECC)
June 13-16, 2023. Bucharest, Romania

A Hierarchical Approach for Strategic Motion Planning in Autonomous Racing

Rudolf Reiter[1], Jasper Hoffmann[2], Joschka Boedecker[2] and Moritz Diehl[1,3]

Abstract— **We present an approach for safe trajectory planning, where a strategic task related to autonomous racing is learned sample efficiently within a simulation environment. A high-level policy, represented as a neural network, outputs a reward specification that is used within the function of a parametric nonlinear model predictive controller. By including constraints and vehicle kinematics in the nonlinear program, we can guarantee safe and feasible trajectories related to the used model. Compared to classical reinforcement learning, our approach restricts the exploration to safe trajectories, starts with an excellent prior performance and yields complete trajectories that can be passed to a tracking lowest-level controller. We do not address the lowest-level controller in this work and assume perfect tracking of feasible trajectories. We show the superior performance of our algorithm on simulated racing tasks that include high-level decision-making. The vehicle learns to efficiently overtake slower vehicles and avoids getting overtaken by blocking faster ones.**

I. Introduction

Motion planning for autonomous racing is challenging due to the fact that vehicles operate at performance limits and planning requires interactive yet safe behavior. This work focuses on strategic planning for fixed opponent policies with safety guarantees. Current research usually is based on either graph-based, sampling-based, learning-based or optimization-based planners [1], [2]. We propose a combination of model-predictive control (MPC) and a neural network (NN) trained by a reinforcement learning (RL) algorithm within simulations. MPC is a powerful optimization-based technique, commonly used for solving trajectory planning and control problems. Using efficient numerical solvers and the possibility to incorporate constraints directly makes MPC attractive in terms of safety, explainability and performance [3]. Nevertheless, in problems like interactive driving, it is difficult to model the behavior of other vehicles. In contrast to MPC, RL is an exploration-driven approach for solving optimal control problems. Instead of an optimization-friendly model, RL only requires samples of the dynamics and can, in theory, optimize over arbitrary cost functions. The flexibility of RL comes at the cost of a high sample inefficiency that is often unfavorable for real-world applications, where data is expensive and rare. Furthermore, RL, in the general setting, lacks safety guarantees. However, once the amount and quality of data is sufficient, the learned policies can show impressive results [4]. In this paper, we combine MPC and RL by using an MPC-inspired low-level trajectory planner to yield kinematic feasible and safe trajectories and use the high-level RL policy for strategic decision-making. We use the expression MPP (Parameterized Model Predictive Planner) to refer to an MPC-based planner, which outputs feasible reference trajectories that we assume to be tracked by a *lowest*-level control systems. This hierarchical approach is common in automotive software stacks [2], [5]. We use the MPP to formulate safety-critical constraints and basic time-optimal behavior but let the cost function be subject to changes by the high-level RL policy. Particularly, we propose an interface where the high-level RL policy outputs a reference in the Frenet coordinate frame. With this approach, we start with an excellent prior strategy for known model parts. We can guarantee safe behavior concerning the chosen vehicle model and the prediction of opponents.

The structure of this paper is as follows. In Sec. II, we motivate our approach by a similar formulation named *safety filter* [6], in Sec. IV, we describe the MPC-based planner, and in Sec. V, we explain the implementation of the high-level RL policy and how we train it. Finally, in Sec. VI, we evaluate the algorithm, which we refer to as HILEPP (*Hierarchical Learning-based Predictive Planner*), in a multi-agent simulation which involves strategic decision-making.

Contribution: We contribute by deriving and evaluating a sample efficient and safe motion planning algorithm for autonomous race-cars. It includes a novel cost function formulation for the interaction of MPC and RL with a strong prior performance, real-time applicability and high interpretability.

Related work: Several works consider RL as a set-point generator for MPC for autonomous agents [7], [8]. As opposed to our approach, they focus on final target points. Another research branch focuses on safety verification with a so-called "safety filter" [9]. For instance, in [6], a rudimentary MPC variant is proposed that considers constraints using MPC as a verification module. Similarly, the authors of [10] use MPC to correct an RL policy if a collision check fails. RL is also used for MPC weight tuning, such as in [11] for UAVs and in [12] for adaptive control in autonomous driving. Related research for motion planning of autonomous racing was recently surveyed in [1]. Several works focus on local planning without strategic considerations [4], [5], thus can not directly be used in multi-agent settings. Other works use a game-theoretic framework [13] which often limits the applicability due to its complexity. An algorithm for

[1]Department of Microsystems Engineering, University Freiburg, 79110 Freiburg, Germany {rudolf.reiter, moritz.diehl}@imtek.uni-freiburg.de
[2]Neurorobotics Lab, University Freiburg, 79110 Freiburg, Germany {hoffmaja, jboedeck}@informatik.uni-freiburg.de
[3]Department of Mathematics, University Freiburg, 79110 Freiburg, Germany

obtaining Nash equilibria is iterated best response (IBR), as shown for drone racing in [14] or for vehicle racing in [15]. However, IBR has high computation times. An algorithm aiming at the necessary KKT conditions of the generalized Nash equilibrium problem is presented in [16]. However, the resulting optimization problem is hard to solve. In [17], long-term strategic behavior is learned through simulation without safety considerations.

II. Background and Motivation

A trained neural network (NN) used as a function approximator for the policy $\pi^{\theta}(s)$, where $\theta \in \mathbb{R}^{n_\theta}$ is the learned parameter vector and $s \in \mathbb{R}^{n_s}$ is the RL environment state, can generally not guarantee safety. Safety is related to constraints for states and controls that must be satisfied at all times. Therefore, the authors in [6] propose an MPC-based policy $\pi^{\mathrm{S}} : \mathbb{R}^{n_a} \to \mathbb{R}^{n_a}$ that projects the NN output $a \in \mathbb{R}^{n_a}$ to a safe control $u^{\mathrm{S}} = \pi^{\mathrm{S}}(x, a)$, where it is guaranteed that $u^{\mathrm{S}} \in \mathcal{U}^{\mathrm{S}} \subseteq \mathbb{R}^{n_a}$. The safe set $\mathcal{U}^{\mathrm{S}}$ is defined for a known (simple) system model $\dot{x} = f(x, u)$ with states x and controls u and corresponding, often tightened, constraints. In this formulation, the input u has the same interpretation as the action a and the state x relates to the model inside the filter. Constraint satisfaction for states is expressed via the set membership $x \in \mathcal{X}$ and for controls via $u \in \mathcal{U}$. The system model is usually transformed to discrete-time via an integration function $x_{i+1} = F(x_i, u_i)$ with step size Δt. When using direct multiple shooting [18] one obtains decision variables for the state $X = [x_0, \ldots, x_N] \in \mathbb{R}^{n_x \times (N+1)}$ and for the controls $u = [u_0, \ldots, u_{N-1}] \in \mathbb{R}^{n_u \times N}$. Since the optimization problem can only be formulated for a finite horizon, a control invariant terminal set S^t needs to be included. The safety filter solves the following optimization problem

$$\begin{aligned} \min_{X, U} \quad & \|u_0 - \bar{a}\|_R^2 \\ \text{s.t.} \quad & x_0 = \bar{x}_0, \quad x_N \in \mathcal{S}^{\mathrm{t}}, \\ & x_{i+1} = F(x_i, u_i), \quad i = 0, \ldots, N-1, \\ & x_i \in \mathcal{X}, u_i \in \mathcal{U}, \quad i = 0, \ldots, N-1 \end{aligned} \tag{1}$$

and takes the first control u_0^* of the solution (X^*, U^*) as output $u^{\mathrm{S}} := u_0^*$. The authors in [6] use the filter as a post-processing safety adaption. However, we propose to use this formulation as a basis for an online filter, even during learning, which makes it applicable for safety-relevant environments. We do not require the same physical inputs to our filter, rather modifications to a parametric optimization problem, similar to [19]. We propose a general interface between the high-level RL policy and MPC, namely a cost function $L(X, U, a)$, modified by action a. Our version of the MPP as a fundamental part of the algorithm solves the optimization problem

$$\begin{aligned} \min_{X, U} \quad & L(X, U, a) \\ \text{s.t.} \quad & x_0 = \hat{x}_0, \quad x_N \in \mathcal{S}^{\mathrm{t}}, \\ & x_{i+1} = F(x_i, u_i), \quad i = 0, \ldots, N-1, \\ & x_i \in \mathcal{X}, u_i \in \mathcal{U}, \quad i = 0, \ldots, N-1, \end{aligned} \tag{2}$$

and takes the optimal state trajectory of the solution (X^*, U^*) as output $X_{\mathrm{ref}} := X^*$ of the MPP algorithm. Due to the pruning of infeasible, i.e., unsafe, trajectories of the actual control, the algorithm becomes sample efficient.

III. General Method

We apply our algorithm to a multi-agent vehicle competition on a race track. We aim to obtain a sample efficient planner that performs time-optimal trajectory planning, avoids interactive opponents and learns strategic behavior, such as blocking other vehicles in simulation. We assume fixed policies of a fixed number of N_{ob} opponents and, therefore, do not consider the interaction as a game-theoretical problem [20]. We use an obstacle avoidance rule, according to the autonomous racing competitions *Roborace* [21] and *F1TENTH* [22], where in a dueling situation, the following vehicle (FV) is mainly responsible avoiding a crash. However, the leading vehicle (LV) must not provoke a crash. Unfortunately, to the best of the author's knowledge, there is no rigorous rule for determining the allowed actions of dueling vehicles. However, we formalize the competition rules of *F1TENTH* similar to [23], where the LV only avoids an inevitable crash, which we state detailed in Sec. IV-B.2. A block diagram of our proposed algorithm is shown in Fig. 1, where we assume a multi-agent environment with a measured state $z \in \mathbb{R}^{n_z}$, which concatenates the ego agent states x, the obstacle/opponent vehicle states x^{ob} and the road curvature $\kappa(\zeta_i)$ on evaluation points ζ_i. We include prior domain knowledge to get the high-level RL policy state $s \in \mathbb{R}^{n_s}$ with the pre-processing function $s = g_s(z)$. For instance, we use relative distances of the opponents to the ego vehicle instead of absolute values. An expansion function $P = G_P(a)$, with the high-level RL policy $a = \pi^{\theta}(s)$, is used to increase the dimension of the NN output in to obtain a parametric cost function. The expansion function is used to include prior knowledge and to obtain an optimization-friendly cost function in the MPP.

IV. Parameterized Model Predictive Planner

Our core component MPP constitutes an MPC formulation that accounts for safety and strong initial racing performance. It comprises a vehicle model, safety constraints and a parameterized cost function, which we will explain in the following section.

A. Vehicle Model

We use rear-wheel-centered kinematic single-track vehicle models in the Frenet coordinate frame, as motivated in previous work [24]. The models are governed by the longitudinal force F_{d} that accounts for accelerating and braking, and the

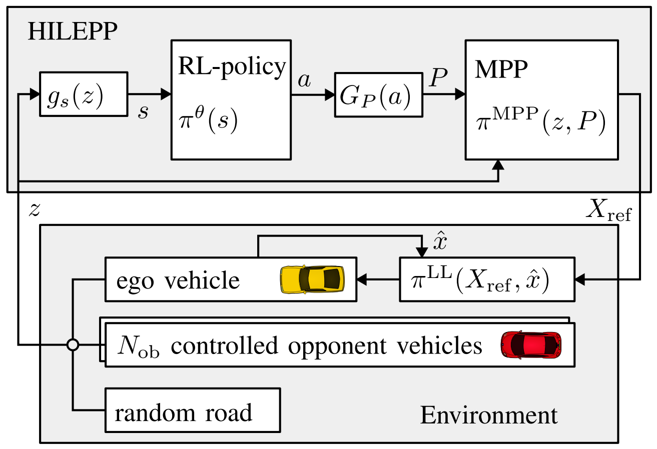

Fig. 1. Proposed control structure. The multi-vehicle environment constitutes a trajectory tracking ego agent (lowest-level controller $\pi^{\mathrm{LL}}(\cdot)$). A state z concatenates all $N_{\mathrm{ob}}+1$ vehicle states and road curvature information. A function $g_s(z)$ projects the state to a lower dimensional state space. A high-level RL policy $\pi^\theta(s)$ and an expanding function $G_P(a)$ modify the cost function of Parameterized Model Predictive Planner (MPP) $\pi^{\mathrm{MPC}}(z,P)$ by action a. The MPP outputs a feasible and safe trajectory X_{ref} to the ego lowest-level controller.

steering rate r, which is the first derivative of the steering angle δ. The most prominent resistance forces $F_{\mathrm{res}}(v) = c_{\mathrm{air}}v^2+c_{\mathrm{roll}}\mathrm{sign}(v)$ are included. The air drag depends on the vehicle speed v with the constant c_{air}. The rolling resistance is proportional to $\mathrm{sign}(v)$ by the constant c_{roll}. We drop the sign function since we only consider positive speed. As shown in previous work [24], [25], the Frenet transformation $\mathcal{F}(\cdot)$ relates Cartesian states $x^{\mathrm{C}} = [x_{\mathrm{e}} \quad y_{\mathrm{e}} \quad \varphi]^\top$, where x_{e} and y_{e} are Cartesian positions and φ is the heading angle, to the curvilinear states

$$x^{\mathrm{F}} = \mathcal{F}(x^{\mathrm{C}}) = [\zeta \quad n \quad \alpha]^\top. \tag{3}$$

The Frenet states are related to the center lane $\gamma(\zeta) = [\gamma_x(\zeta) \quad \gamma_y(\zeta)]$, with signed curvature $\kappa(\zeta)$ and tangent angle $\varphi^\gamma(\zeta)$, where ζ is the 1d-position of the closest point of the center lane, n is the lateral normal distance and α is the difference of the vehicle heading angle to the tangent of the reference curve. Under mild assumptions [24], the Frenet transformation and its inverse

$$x^{\mathrm{C}} = \mathcal{F}^{-1}(x^{\mathrm{F}}) = \begin{bmatrix} \gamma_x(\zeta) - n\sin(\varphi^\gamma(\zeta)) \\ \gamma_y(\zeta) + n\cos(\varphi^\gamma(\zeta)) \\ \varphi^\gamma(\zeta) - \alpha \end{bmatrix} \tag{4}$$

are well-defined. We summarize the states with $x = \begin{bmatrix}\zeta & n & \alpha & v & \delta\end{bmatrix}^\top$ and controls with $u = \begin{bmatrix}F_{\mathrm{d}} & r\end{bmatrix}^\top$. The Frenet frame vehicle model is parameterized by the mass m and length l and given as

$$\dot{x} = f(x,u) = \begin{bmatrix} \frac{v\cos(\alpha)}{1-n\kappa(\zeta)} \\ v\sin(\alpha) \\ \frac{v}{l}\tan(\delta) - \frac{\kappa(\zeta)v\cos(\alpha)}{1-n\kappa(\zeta)} \\ \frac{1}{m}(F_{\mathrm{d}} - F_{\mathrm{res}}(v)) \\ r \end{bmatrix}. \tag{5}$$

The discrete states x_k at sampling time $k\Delta t$ are obtained by an RK4 integration function $x_{k+1} = F(x_k, u_k, \Delta t)$.

B. Safety Constraints

As stated in Sec. II, the MPP formulation should restrict trajectories X_{ref} to be within model constraints. Since we assume known vehicle parameters and no measurement noise, this can be guaranteed for most limitations in a straightforward way. Nevertheless, the interactive behavior of the opponent vehicles poses a severe challenge to the formulation. On one extreme, we could model the other vehicles robustly, which means we account for all possible maneuvers, which yields quite conservative constraints. On the other extreme, with known parameters of all vehicles, one could model the opponent by "leaving space" for at least one possible motion of the opponent without a crash, thus not forcing a collision. The latter leads to a hard bi-level optimization problem since the feasibility problem, which is an optimization problem itself, is needed as a constraint of the MPP. In this work, we aim at a heuristic explained in Sec. IV-B.2.

1) Vehicle Limitations: Slack variables $\sigma = [\sigma_v, \sigma_\alpha, \sigma_n, \sigma_\delta, \sigma_a, \sigma_{\mathrm{o}}]^\top \in \mathbb{R}^6$, for state (6), acceleration (8) and obstacles constraints (10) are used to achieve numerically robust behavior. We use box constraints for states

$$B_x(\sigma) := \Big\{x \,\Big|\, -\sigma_n + \underline{n} \le n \le \overline{n} + \sigma_n, \tag{6a}$$
$$-\sigma_\alpha + \underline{\alpha} \le \alpha \le \overline{\alpha} + \sigma_\alpha, \tag{6b}$$
$$0 \le v \le \overline{v} + \sigma_v, \tag{6c}$$
$$-\sigma_\delta + \underline{\delta} \le \delta \le \overline{\delta} + \sigma_\delta\Big\}, \tag{6d}$$

and controls

$$B_u := \{u \mid \underline{F}_{\mathrm{d}} \le F_{\mathrm{d}} \le \overline{F}_{\mathrm{d}}, \quad \underline{r} \le r \le \overline{r}\}. \tag{7}$$

Further, we use a lateral acceleration constraints set

$$B_{\mathrm{lat}}(\sigma) := \left\{x \,\middle|\, \left|\frac{v^2\tan(\delta)}{l}\right| \le \overline{a}_{\mathrm{lat}} + \sigma_a\right\}, \tag{8}$$

to account for friction limits.

2) Obstacle Constraints: We approximate the rectangular shape of obstacles (referenced by "ob") in the Cartesian coordinate frame by an ellipse and the ego vehicle by a circle, which yields superior computational properties compared to other approaches, cf. [26]. We assume a predictor of an obstacle vehicle i that outputs the expected Cartesian positions of the vehicle center $p_k^{\mathrm{ob}i} = [x_{\mathrm{e},k}^{\mathrm{ob}i} \quad y_{\mathrm{e},k}^{\mathrm{ob}i}]^\top \in \mathbb{R}^2$ with a constraint ellipse shape matrix $\hat{\Sigma}_k^{\mathrm{ob}i}(x) \in \mathbb{R}^{2\times 2}$ at time step k that depends on the (Frenet) vehicle state in x. The ellipse area is increased by $\Sigma^{\mathrm{ob}i}(x) = \hat{\Sigma}^{\mathrm{ob}i}(x) + \mathbb{I}(r+\Delta r)^2$ with radii of the ego covering circle r and a safety distance Δr. Since the ego vehicle position $p^\top = [x_{\mathrm{e}} \quad y_{\mathrm{e}}]$ is measured at the rear axis and in order to have a centered covering circle, we project the rear position to the ego vehicle center p_{mid} by

$$p_{\mathrm{mid}} = \begin{bmatrix} x_{\mathrm{e,mid}} \\ y_{\mathrm{e,mid}} \end{bmatrix} = P(x^{\mathrm{C}}) = \begin{bmatrix} x_{\mathrm{e}} + \frac{l}{2}\cos\varphi \\ y_{\mathrm{e}} + \frac{l}{2}\sin\varphi \end{bmatrix} \tag{9}$$

For obstacle avoidance with respect to the ellipse matrix, we use the constraint set in compact notation

$$B_{\mathrm{O}}(x^{\mathrm{ob}}, \Sigma^{\mathrm{ob}}, \sigma) = \Big\{ x \in \mathbb{R}^2 \Big| \left\| P(\mathcal{F}^{-1}(x)) - p^{\mathrm{ob}} \right\|^2_{(\Sigma^{\mathrm{ob}}(x))^{-1}} \geq 1 - \sigma_{\mathrm{o}} \Big\}. \tag{10}$$

3) Obstacle Prediction: The opponent prediction uses a simplified model with states $x^{\mathrm{ob}} = [\zeta^{\mathrm{ob}}, n^{\mathrm{ob}}, v^{\mathrm{ob}}]^\top$ and assumes curvilinear motion depending on the initial estimated state $\hat{x}^{\mathrm{ob}}$. With the constant acceleration force $F_{\mathrm{d}}^{\mathrm{ob}}$, the ODE of the opponent estimator can be written as

$$\dot{\zeta}^{\mathrm{ob}} = \frac{v^{\mathrm{ob}}(t)\cos(\hat{\alpha}^{\mathrm{ob}})}{1 - n^{\mathrm{ob}}\kappa(\zeta^{\mathrm{ob}})} \tag{11a}$$

$$\dot{n}^{\mathrm{ob}} = v^{\mathrm{ob}}(t)\sin(\hat{\alpha}^{\mathrm{ob}}) \tag{11b}$$

$$\dot{v}^{\mathrm{ob}} = \frac{1}{m^{\mathrm{ob}}} F_{\mathrm{d}}^{\mathrm{ob}}. \tag{11c}$$

Since the FV is responsible for a crash, it *generously* predicts the LV by assuming constant velocity motion, where $F_{\mathrm{d}}^{\mathrm{ob}}$ is set to 0. The LV predicts the FV most *evasively*, which we realize by assuming a FV full stop with its maximum braking force $F_{\mathrm{d}}^{\mathrm{ob}} = \underline{F}_{\mathrm{d}}^{\mathrm{ob}}$. In any situation, this allows the FV to plan for at least one safe trajectory (i.e., a full stop) Thus the LV does not "provoke" a crash, as required in racing competition rules [21], [22]. Besides these minimum safety restrictions, interaction should be learned by the high-level RL policy. We simulate the system forward with a function $\Phi()$, using steps of the RK4 integration function to obtain the predicted states $[x_0^{\mathrm{ob}}, \ldots, x_N^{\mathrm{ob}}] = \Phi(\hat{x}^{\mathrm{ob}}, \hat{\alpha}^{\mathrm{ob}}, F_{\mathrm{d}}^{\mathrm{ob}})$.

4) Recursive feasibility: In order to guarantee safety for a finite horizon and constraints (6), (7) and (8), we refer to the concept of recursive feasibility and control invariant sets (CIS) [3]. A straight-forward CIS is the trivial set of zero velocity $\{x \mid v = 0\}$. An approximation to the CIS, which is theoretically not a CIS but which has shown good performance in practice, is the limited-velocity terminal set $\mathcal{S}^{\mathrm{t}} := \{x \mid \alpha = 0, v \leq \overline{v}_{\max}\}$. For long horizons, the influence of the terminal set vanishes.

C. Objective

For the parameterized cost function $L(X, U, a)$, we propose a formulation with the following properties:

1) Simple structure for reliable and fast NLP iterations
2) Expressive behavior related to strategic driving
3) Low dimensional action space
4) Good initial performance

The first property is achieved by restricting the cost function to a quadratic form. The second property is achieved by formulating the state reference in the Frenet coordinate frame. The final properties of a low dimensional action space and an excellent initial performance are achieved by interpreting the actions as reference lateral position n_{ref} and reference speed v_{ref}. By setting the reference speed, also the corresponding longitudinal state $\zeta_{\mathrm{ref},k}$ of a curvilinear trajectory is defined by $\zeta_{\mathrm{ref},k} = \hat{\zeta} + k\Delta t v_{\mathrm{ref}}$. The reference heading angle miss-match α_{ref} and the steering angle δ_{ref} are set to zero, with fixed weights w_α and w_δ, since these weights are tuned for smooth driving behavior. Setting the reference speed v_{ref} above maximum speed approximates time-optimal driving [27]. We compare the influence using references with their associated weights w_v, w_n (HILEPP-II with $a_{\mathrm{II}} = [v_{\mathrm{ref}} \quad n_{\mathrm{ref}} \quad w_v \quad w_n]^\top$) to fixed weights without using them in the action space (HILEPP-I with $a_{\mathrm{I}} = [v_{\mathrm{ref}} \quad n_{\mathrm{ref}}]^\top$).

D. NLP formulation

We use the action-dependent stage cost matrix $Q_{\mathrm{w}}(a)$ with $Q_{\mathrm{w}} : \mathbb{R}^{n_a} \to \mathbb{R}^{n_x \times n_x}$ and a cost independent terminal cost $Q^t \in \mathbb{R}^{n_x \times n_x}$. We set the values of R, Q_0 and Q^{t} to values corresponding to driving smoothly and time-optimally. With constant action inputs $\bar{a}$, this leads to a strong initial performance at the beginning of training the high-level RL policy. With the constant time action-dependent reference values $\xi_{\mathrm{ref},k}(a) = [0 \quad n \quad 0 \quad v_x \quad 0]^\top \in \mathbb{R}^{n_x}$ for HILEPP-I/II and constant time reference weights $Q_{\mathrm{w}}(a) = \mathrm{diag}([0 \quad w_n \quad 0 \quad w_v \quad 0])$ for HILEPP-II, we can write the expanding function as

$$G_P(a) : a \to \Big(\xi_{\mathrm{ref},0}(a), \ldots, \xi_{\mathrm{ref},N}(a), Q_{\mathrm{w}}(a) \Big), \tag{12}$$

which maps n_a to $n_x^2(N+1) + n_x(N+1)$ dimensions for cost matrices and reference values. We state the final NLP, using the vehicle model (5), the MPC path constraints for obstacle avoidance (10), vehicle constraints (6), (7) and (8) and the parametric cost functions of (2). The full objective, including slack variables $\Xi = [\sigma_0, \ldots, \sigma_N] \in \mathbb{R}^{6 \times N}$ for each stage, associated L2 weights $Q_{\sigma,2} = \mathrm{diag}(q_{\sigma,2}) \in \mathbb{R}^{6\times 6}$ and L1 weights $q_{\sigma,1} \in \mathbb{R}^6$, reads as

$$L(X, U, a, \Xi) = \sum_{k=0}^{N-1} \left\| x_k - \xi_{\mathrm{ref},k}(a) \right\|^2_{Q_{\mathrm{w}}(a)} + \left\| u_k \right\|^2_R + \left\| x_N - \xi_{\mathrm{ref},N}(a) \right\|^2_{Q^t} + \sum_{k=0}^{N} \left\| \sigma_k \right\|^2_{Q_{\sigma,2}} + |q_{\sigma,1}^\top \sigma_k|. \tag{13}$$

Together with the predictor for time step k of the j-th future opponent vehicle states, represented as bounding ellipses with the parameters $p_i^{\mathrm{ob},j}, \Sigma_i^{\mathrm{ob},j}$, the parametric NLP can be written as

$$\begin{aligned} \min_{X,U,\Xi} \quad & L(X, U, a, \Xi) \\ \text{s.t.} \quad & x_0 = \hat{x}, \quad \Xi \geq 0, \quad x_N \in \mathcal{S}^{\mathrm{t}}, \\ & x_{i+1} = F(x_i, u_i) && i = 0, \ldots, N-1, \\ & u_i \in B_u, && i = 0, \ldots, N-1, \\ & x_i \in B_x(\sigma_k) \cap B_{\mathrm{lat}}(\sigma_k) && i = 0, \ldots, N, \\ & x_i \in B_{\mathrm{ob}}(p_i^{\mathrm{ob},j}, \Sigma_i^{\mathrm{ob},j}, \sigma_k) && i = 0, \ldots, N, \\ & && j = 0, \ldots, N_{\mathrm{ob}}. \end{aligned} \tag{14}$$

The final MPP algorithm is stated in Alg. (1).

Algorithm 1: MPP

```
input : action $a$, ego states $\hat{x}$, $N_{\mathrm{ob}}$ obstacle
        states $\hat{x}^{\mathrm{ob}}$
output: planned trajectory $X_{\mathrm{ref}}$
1  for $j$ in range($N_{\mathrm{ob}}$) do
2      if $\hat{\zeta}^{\mathrm{ob}} \leq \hat{\zeta}$ then
3          Consider opp. as FV: $F_{\mathrm{d}}^{\mathrm{ob}} \leftarrow \underline{F}_{\mathrm{d}}^{\mathrm{ob}}$
4      end
5      else
6          Consider opp. as LV $F_{\mathrm{d}}^{\mathrm{ob}} \leftarrow 0$
7      end
8      Predict $[x_0^{\mathrm{ob}}, \dots, x_N^{\mathrm{ob}}] = \Phi(\hat{x}^{\mathrm{ob}}, \hat{\alpha}^{\mathrm{ob}}, F_{\mathrm{d}}^{\mathrm{ob}})$;
9      Compute constraint ellipses $\Sigma_k^{\mathrm{ob},j} = \Sigma_0(\varphi^{\mathrm{ob},j})$;
10 end
11 Compute weights $(\zeta_{\mathrm{ref},k}, Q_{\mathrm{w}}) \leftarrow G_P(a)$;
12 $X_{\mathrm{ref}} \leftarrow$Solve NLP (2) with $(\zeta_{\mathrm{ref},k}, Q_{\mathrm{w}})$;
```

V. *Hierarchical Learning-based Predictive Planner*

The MPP of Sec. IV plans safely and time-optimally, but not strategically. Therefore, we learn a policy π^θ with RL that decides how to parameterize the MPP to achieve a strategic goal at each time step. Since we assume stationary opponent policies, we can apply standard, i.e., single-agent RL algorithms [20] and solve for the best response. In the following, we give a brief theoretical background to policy gradient methods and then describe the training procedure in detail.

A. *Policy gradient*

RL requires a Markov Decision Process (MDP) framework. A MDP consists of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition kernel $P(s_{k+1} \mid s_k, a_k)$, a reward function $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ that describes how desirable a state is (equal to the negative cost) and a discount factor $\gamma \in [0, 1)$. The goal for a given MDP is finding a policy $\pi^\theta : \mathcal{S} \mapsto \mathcal{A}$ that maximizes the expected discounted return

$$\begin{gathered} J(\pi^\theta) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(s_k, a_k) \mid s_0 = s\right], \\ s_k \sim P(s_{k+1} \mid s_k, a_k), a_k \sim \pi^\theta(s_k). \end{gathered} \tag{15}$$

where s_k is the state and a_k the action taken by the policy π^θ at time step k. An important additional concept is the state-action value function

$$Q^{\pi^\theta}(s, a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(s_k, a_k) \mid s_0 = s, a_0 = a\right] \tag{16}$$

that is the expected value of the policy π^θ starting in state s and taking an action a. In general, finding an optimal policy π^θ by directly optimizing θ in (15) is impossible. The expectation in (15) might be computationally intractable, or the exact transition probabilities P may be unknown. Thus, the policy gradient $\nabla J(\pi^\theta)$ is approximated using only transition samples from the environment. We sample these transitions from a simulator. However, they could also come from real-world experiments. A particularly successful branch of policy gradient methods are *actor-critic* methods [28], where we train two neural networks, an actor π^θ and a critic Q^ϕ. The critic estimates the value of a chosen action and is trained by minimizing the temporal difference loss

$$\begin{aligned} &J_Q(\phi) = \\ &\quad \mathbb{E}_{s,a,r,s' \sim \mathcal{D}}\left[\left(r + \gamma Q_{\phi'}\left(s', \pi^\theta(s')\right) - Q_\phi(s, a)\right)^2\right]. \end{aligned} \tag{17}$$

The trained critic is used to train the actor with the objective

$$J_\pi(\theta) = \mathbb{E}_{s \sim \mathcal{D}}\left[Q^\phi(s, \pi^\theta(s))\right]. \tag{18}$$

To derive the gradient for the policy π^θ from (18) we can use the chain rule [29]

$$\nabla_\theta J_\pi(\theta) = \mathbb{E}_{s \sim \mathcal{D}}\left[\nabla_\theta \pi^\theta(s)\ \nabla_a Q^\phi(s, a)\big|_{a=\pi^\theta(x)}\right]. \tag{19}$$

The soft-actor critic method, introduced by [30] enhances the actor-critic method by adding an entropy term into (18) and using the *reparameterization trick* to calculate the gradient. For a complete description, we refer to [30]. Note that, with a slight abuse of notation, in the equations from above, we sample transitions tuple (s, a, r, s') and states s from the same distribution $\mathcal{D}$. In our context, $\mathcal{D}$ is a buffer storing all transitions and states that have occurred so far by interacting with the environment.

B. *Training Environment*

We reduce the RL state space, based on domain knowledge which we put into the function $g_s(z_k)$. The race track layout is approximated by finite curvature evaluations $\kappa(\zeta + d_i)$ at different longitudinal distances d_i relative to the ego vehicle position ζ, for $i = 1, \dots, N_\kappa$. For the RL ego state $s(z_k) = [n, v, \alpha]^\top$, we include the lateral position n, the velocity v and the heading angle miss-match α. For opponent i, we additionally add the opponent longitudinal distance $\zeta_{\mathrm{ob}} - \zeta$ to the ego vehicle to state $s_{\mathrm{ob}_i} = [\zeta_{\mathrm{ob}_i} - \zeta, n_{\mathrm{ob}_i}, v_{\mathrm{ob}_i}, \alpha_{\mathrm{ob}_i}]^\top$. Combined, we get the following state definition for the RL agent

$$\begin{aligned} &s_k = g_s(z_k) = \\ &\quad [\kappa(\zeta + d_i), \dots, \kappa(\zeta + d_N), s^\top, s_{\mathrm{ob}_1}^\top, \dots, s_{\mathrm{ob}_{N_{\mathrm{ob}}}}^\top]^\top. \end{aligned} \tag{20}$$

We propose a simple reward that encourages time-optimal and strategic driving: For driving time-optimally, we reward the progress on the race track by measuring the velocity of the ego vehicle projected point on the center line $\dot{s}_k$. For driving strategically, we reward ego vehicle overall rank by adding 1 for being in front of every opponent. Combined, we get the reward function

$$R(s, a) = \frac{\dot{s}}{200} + \sum_{i=1}^{N_{\mathrm{ob}}} 1_{\zeta_k > \zeta_k^{\mathrm{ob}_i}}. \tag{21}$$

At each time step, the high-level RL policy chooses a parameter for the MPP; thus the action space is the parameter space of the MPP. For training the high-level RL policy, an essential part is the simulation function of the environment $z_{\text{next}} = \text{sim}(z, \kappa(\cdot))$, which we simulate for n_{epi} episodes and a maximum of n_{scene} steps. The road layout defined by $\kappa(\zeta)$ is randomized within an interval $[-0.04, 0.04]\text{m}^{-1}$ before each training episode. The curvature is set together with initial random vehicle states z by a reset function $(z, \kappa(\zeta)) = Z()$. We use Alg. 2 for training and Alg. 3 for the final deployment of HILEPP.

Algorithm 2: HILEPP training

```
input : number of episodes $n_{\text{epi}}$, maximum scenario
        steps $n_{\text{scene}}$, reset function $(z, \kappa(\zeta)) = Z()$,
        reward function $r(z)$
output: learned policy $\pi^{\theta}(\zeta)$
for j in range($n_{\text{epi}}$) do
    reset+randomize environment $(z, \kappa(\zeta)) \leftarrow Z()$;
    for i in range($n_{\text{scene}}$) do
        get NN input state $s \leftarrow g_s(z)$;
        get high-level action $a \leftarrow \pi^{\theta}(s)$;
        evaluate planner $X_{\text{ref}} \leftarrow \text{MPP}(a, z)$;
        simulate environment $z_{\text{next}} = \text{sim}(X_{\text{ref}})$;
        get reward $(r, \text{done}) \leftarrow R(z_{\text{next}}, a)$;
        RL update $\theta \leftarrow \text{train}(z, z_{\text{next}}, r, a)$;
        if done then
            exit loop
        end
        z$\leftarrow z_{\text{next}}$
    end
end
return $\pi^{\theta}(s)$;
```

Algorithm 3: HILEPP deployment

```
input : environment state $z$, trained policy $\pi^{\theta}(s)$
output: reference trajectory $X_{\text{ref}}$
compute NN input state $s \leftarrow g_s(z)$;
compute high-level RL policy output $a \leftarrow \pi^{\theta}(s)$;
return MPP output $X_{\text{ref}} \leftarrow \text{MPP}(z, a)$;
```

VI. Simulated Experiments

We evaluate (Alg. 3) and train (Alg. 2) HILEPP on three different scenarios that resemble racing situations (cf. Fig. 2). The first scenario *overtaking* constitutes three "weaker", initially leading opponent agents, where "weaker" relates to the parameters of maximum accelerations, maximum torques and vehicle mass (cf. Tab. I). The second scenario *blocking* constitutes three "stronger", initially subsequent opponents. The ego agent starts between a stronger and a weaker opponent in a third *mixed* scenario. Each scenario is simulated for one minute, where the ego agent has to perform best related to the reward (21). We train the HILEPP agent with the different proposed action interfaces (I: $\mathcal{A} = \{n_{\text{ref}}, v_{\text{ref}}\}$, II: $\mathcal{A} = \{n_{\text{ref}}, v_{\text{ref}}, w_n, w_v\}$). Opponent agents, as well as

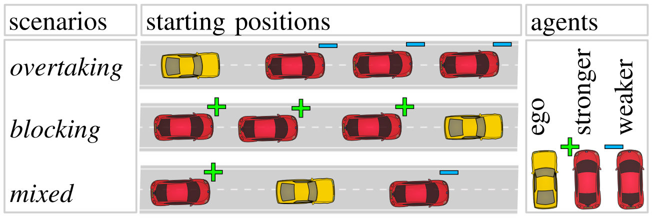

Fig. 2. Scenarios differ in the initial rank and performance of vehicles.

Name	Variable	Ego Agent	"Weak" Agent	"Strong" Agent
wheelbase	l_r, l_f	1.7	1.7	1.7
chassis lengths	$l_{r,\text{ch}}, l_{f,\text{ch}}$	2	2	2
chassis width	w_{ch}	1.9	1.9	1.9
mass	m	1160	2000	600
max. lateral acc.	$\underline{a}_{\text{lat}}, \overline{a}_{\text{lat}}$	±8	±5	±13
max. acc. force	$\overline{F}_{\text{d}}$	10kN	8kN	12kN
max. brake force	$\underline{F}_{\text{d}}$	20kN	20kN	20kN
max. steering rate	$\underline{r}, \overline{r}$	±0.39	±0.39	±0.39
velocity bound	$\overline{v}$	60	60	60
steering angle bound	$\underline{\delta}, \overline{\delta}$	±0.3	±0.3	±0.3
road bounds	$\underline{n}, \overline{n}$	±7	±7	±7

TABLE I
Vehicle model parameters. SI-units, if not stated explicitly.

the ego agent baseline, are simulated with the state-of-the-art MPP (Alg. 1) with a fixed action a that accounts for non-strategic time-optimal driving with obstacle avoidance. We perform a hyper-parameter (HP) search for the RL parameters with *Optuna* [31]. The search space was defined by $[10^{-5}, 10^{-3}]$ for the learning rate, $\tau \in [10^{-5}, 10^{-2}]$ for the polyak averaging of the target networks, $\{64, 128, 256\}$ for the width of the hidden layers, $\{1, 2, 3\}$ for the number of hidden layers and $\{128, 256\}$ for the batch size. We used the average return of 30 evaluation episodes after training for 10^5 steps as the search metric. We trained on randomized scenarios for $10 \cdot 10^5$ steps with 10 different seeds on each scenario. For estimating the performance of the final policy, we evaluated the episode return (sum of rewards) on 100 episodes. We further compare our trained HILEPP against a pure RL policy that directly outputs the controls u. The final experiments were run on a computing cluster were all 30 runs for one method where run on 8 GeForce RTX 2080 Ti with a AMD EPYC 7502 32-Core Processor with a training time of around 6 hours. We use the NLP solver `acados` [32] with `HPIPM` [33], RTI iterations and a partial condensing horizon of $\frac{N}{2}$.

A. Results

In Fig. 3, we compare the training performance related to the reward (21) and in Fig. 4, we show the final performance of the two HILEPP formulations. HILEPP quickly outperforms the base-line MPP as well as the pure RL formulation, showing its high sample efficiency. With a smaller action space, HILEPP-I seems to learns faster. However, with more

Name	Variable	Value
nodes / disc. time	$N/\ \Delta t$	50/ 0.1
terminal velocity	$\overline{v}_N$	15
state weights	q	$[1, 500, 10^3, 10^3, 10^4]\Delta t$
terminal state weights	q_N	$[10, 90, 100, 10, 10]$
L2 slack weights	$q_{\sigma,2}$	$[10^2, 10^3, 10^6, 10^3, 10^6, 10^6]$
L1 slack weights	$q_{\sigma,1}$	$[0, 0, 10^6, 10^4, 10^7, 10^6]$
control weights	R	$\mathrm{diag}([10^{-3}, 2\cdot 10^6])\Delta t$

TABLE II
PARAMETERS FOR MPP IN SI UNITS

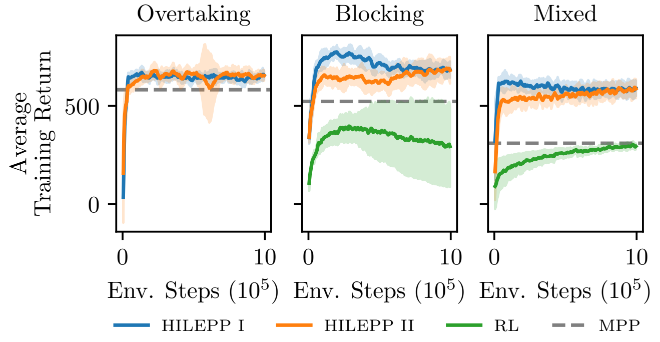

Fig. 3. Training performance of average episode return (sum of rewards) of HILEPP with different action interfaces (I: actions $v_{\mathrm{ref}}, n_{\mathrm{ref}}$, II: actions $v_{\mathrm{ref}}, n_{\mathrm{ref}}, w_v, w_n$), pure RL and the MPP baseline. We used a moving average over 1000 steps. Remarkably, we could not train a successful pure RL agent in the overtaking scenario.

samples, HILEPP-II outperforms the smaller action space in all three scenarios on the evaluation runs in terms of median performance, see Fig. 4. The training was stopped after 10^6 steps due to the already high training time and the slow return increase, as shown in the Fig. 3. Despite using state-of-the-art RL learning algorithms and an extensive HP search on GPU clusters, the pure RL agent could not in general outperform the MPP baseline. Furthermore, the pure RL policy could not prevent crashes, whereas MPP successfully filters the actions within HILEPP to safe actions that do not cause safety violations. Notably, due to the struggle of the pure RL agent with lateral acceleration constraints, it has learned a less efficient strategy to drive slowly and just focus on blocking subsequent opponents in scenarios *blocking* and *mixed*. Therefore, pure RL could not perform efficient overtaking maneuvers in the *overtaking* scenario and yields evasive returns (consequently excluded in Fig. 4). In Tab. III we show that HILEPP is capable of planning trajectories with approximately 100Hz, which is sufficient and competitive for automotive motion planning [1]. A rendered plot of learned blocking is shown in Fig. 5, where also the time signals are shown of how the high-level RL policy sets the references of HILEPP-I. A rendered simulation for all three scenarios can be found on the website `https://rudolfreiter.github.io/hilepp_vis/`.

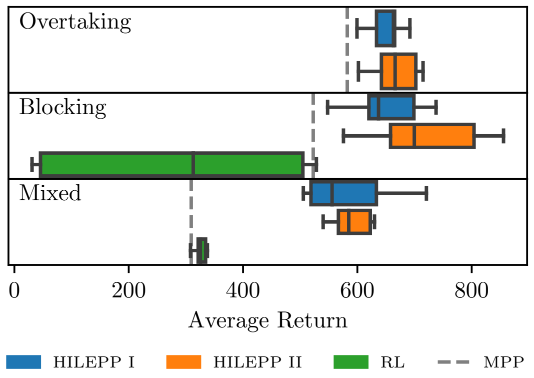

Fig. 4. Final episode return of 100 evaluation runs of the proposed interfaces for different scenarios (Fig. 2).

Module	Mean± Std.	Max	Module	Mean± Std.	Max
MPP	5.45 ± 2.73	8.62	HILEPP-I	6.90 ± 3.17	9.56
RL policy	0.13 ± 0.01	0.26	HILEPP-II	7.41 ± 2.28	9.21

TABLE III
COMPUTATION TIMES (MS) OF MODULES.

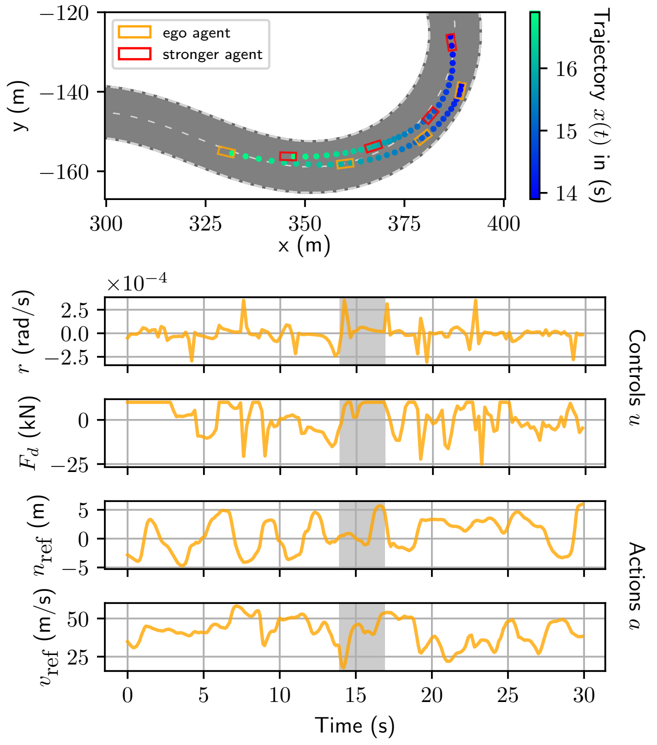

Fig. 5. Exemplary evaluation episode of the HILEPP-I planner in the mixed scenario. On the bottom, the controls of the MPP and the actions of the high-level RL policy are shown. The grey box indicates a time window where snapshots of a blocking maneuver are shown in the top plot. The vehicles move from right to left.

VII. Conclusions

We have shown a hierarchical planning algorithm for strategic racing. We use RL to train for strategies in simulated environments and have shown to outperform a basic time-optimal and obstacle-avoiding approach, as well as pure deep-learning-based RL in several scenarios. The major drawback of our approach is the restrictive prediction and the stationary policy of opponents. Further work could consider multi-agent RL (MARL) algorithms based on Markov games which, however, is still a challenging open research area [20].

Acknowledgment

This research was supported by DFG via Research Unit FOR 2401 and project 424107692 and by the EU via ELO-X 953348. The authors thank Christian Leininger, the SYSCOP and the Autonomous Racing Graz team for inspiring discussions.

References

[1] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 458–488, 2022.

[2] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[3] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Nob Hill, 2017.

[4] P. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, L. Gilpin, P. Khandelwal, V. Kompella, H. Lin, P. MacAlpine, D. Oller, T. Seno, C. Sherstan, M. Thomure, and H. Kitano, "Outracing champion gran turismo drivers with deep reinforcement learning," *Nature*, vol. 602, pp. 223–228, 02 2022.

[5] J. L. Vázquez, M. Brühlmeier, A. Liniger, A. Rupenyan, and J. Lygeros, "Optimization-based hierarchical motion planning for autonomous racing," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2397–2403, 2020.

[6] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, p. 109597, 2021.

[7] C. Greatwood and A. G. Richards, "Reinforcement learning and model predictive control for robust embedded quadrotor guidance and control," *Autonomous Robots*, vol. 43, no. 7, pp. 1681–1693, Oct. 2019. [Online]. Available: http://link.springer.com/10.1007/s10514-019-09829-4

[8] B. Brito, M. Everett, J. P. How, and J. Alonso-Mora, "Where to go next: Learning a subgoal recommendation policy for navigation in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 6, pp. 4616–4623, 2021.

[9] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022. [Online]. Available: https://doi.org/10.1146/annurev-control-042920-020211

[10] J. Lubars, H. Gupta, S. Chinchali, L. Li, A. Raja, R. Srikant, and X. Wu, "Combining reinforcement learning with model predictive control for on-ramp merging," 2021.

[11] Y. Song and D. Scaramuzza, "Learning high-level policies for model predictive control," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7629–7636, 2020.

[12] B. Zarrouki, V. Klös, N. Heppner, S. Schwan, R. Ritschel, and R. Voßwinkel, "Weights-varying mpc for autonomous vehicle guidance: a deep reinforcement learning approach," in *2021 European Control Conference (ECC)*, 2021, pp. 119–125.

[13] A. Liniger and J. Lygeros, "A noncooperative game approach to autonomous racing," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 884–897, 2020.

[14] R. Spica, E. Cristofalo, Z. Wang, E. Montijano, and M. Schwager, "A real-time game theoretic planner for autonomous two-player drone racing," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1389–1403, 2020.

[15] M. Wang, Z. Wang, J. Talbot, J. C. Gerdes, and M. Schwager, "Game-theoretic planning for self-driving cars in multivehicle competitive scenarios," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1313–1325, 2021.

[16] S. Le Cleac'h, M. Schwager, and Z. Manchester, "Algames: A fast augmented lagrangian solver for constrained dynamic games," *Auton. Robots*, vol. 46, no. 1, p. 201–215, jan 2022. [Online]. Available: https://doi.org/10.1007/s10514-021-10024-7

[17] W. Schwarting, T. Seyde, I. Gilitschenski, L. Liebenwein, R. Sander, S. Karaman, and D. Rus, "Deep latent competition: Learning to race using visual control policies in latent space," 2021. [Online]. Available: https://arxiv.org/abs/2102.09812

[18] H. G. Bock and K. J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," in *Proceedings of the IFAC World Congress*. Pergamon Press, 1984, pp. 242–247.

[19] S. Gros and M. Zanon, "Data-driven economic nmpc using reinforcement learning," *IEEE Transactions on Automatic Control*, vol. 65, no. 2, p. 636–648, Feb 2020. [Online]. Available: http://dx.doi.org/10.1109/TAC.2019.2913768

[20] K. Zhang, Z. Yang, and T. Başar, *Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms*. Cham: Springer International Publishing, 2021, pp. 321–384. [Online]. Available: https://doi.org/10.1007/978-3-030-60990-0_12

[21] Roborace. (2020) Roborace season beta. [Online]. Available: https://roborace.com/

[22] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," *Proceedings of Machine Learning Research*, vol. 123. [Online]. Available: https://par.nsf.gov/biblio/10221872

[23] N. Li, E. Goubault, L. Pautet, and S. Putot, "Autonomous racecar control in head-to-head competition using Mixed-Integer Quadratic Programming," in *Opportunities and challenges with autonomous racing, 2021 ICRA workshop*, Online, United States, May 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03749355

[24] R. Reiter and M. Diehl, "Parameterization approach of the frenet transformation for model predictive control of autonomous vehicles," *Proceedings of the European Control Conference (ECC)*, 2021.

[25] R. Reiter, M. Kirchengast, D. Watzenig, and M. Diehl, "Mixed-integer optimization-based planning for autonomous racing with obstacles and rewards," *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (NMPC)*, 2021.

[26] R. Reiter, A. Nurkanović, J. Frey, and M. Diehl, "Frenet-cartesian model representations for automotive obstacle avoidance within nonlinear mpc," 2023.

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

[28] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[29] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. PMLR, 2014, pp. 387–395.

[30] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[31] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

[32] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, Oct 2021.

[33] G. Frison and M. Diehl, "HPIPM: a high-performance quadratic programming framework for model predictive control," in *Proceedings of the IFAC World Congress*, Berlin, Germany, July 2020.