Hierarchical Reinforcement Learning and Model Predictive Control for Strategic Motion Planning in Autonomous Racing

Rudolf Reiter [1] **Jasper Hoffmann** [2] **Joschka Boedecker** [2] **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 objective of a parametric nonlinear model predictive controller. We can guarantee safe and feasible trajectories by including constraints and vehicle kinematics in the nonlinear program. Compared to classical reinforcement learning, our approach restricts exploration to safe trajectories, starts with good prior performance, and yields complete trajectories that can be passed to a tracking lowest-level controller. We validate the performance of our algorithm in simulation and show how it learns to overtake and block other vehicles efficiently.

1. Introduction

This work focuses on strategic planning for fixed opponent policies with safety guarantees. 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 to solve 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 (Rawlings et al., 2017). 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 sample inefficiency, which is often unfavorable for real-world applications, where data are expensive and rare. Furthermore, RL, in the general setting, lacks safety guarantees. However, once the amount and quality of data are sufficient, the learned policies can show superior results (Wurman et al., 2022). In this paper, we combine MPC and RL 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. The Algorithm is referred to as *Hierarchical Learning-based Predictive Planner* (HILEPP). 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 (Vázquez et al., 2020; Paden et al., 2016). 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.

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 interpretability.

Related work: Several works consider RL as a set-point generator for MPC for autonomous agents (Greatwood & Richards, 2019; Brito et al., 2021). As opposed to our approach, they focus on final target points. Another research branch focuses on safety verification with a so-called "safety filter" (Brunke et al., 2022). For instance, in (Wabersich & Zeilinger, 2021), a rudimentary MPC variant is proposed that considers constraints using MPC as a verification module. Similarly, the authors of (Lubars et al., 2021) use MPC

[1]Department of Microsystems Engineering, University of Freiburg, 79110 Freiburg, Germany [2]Neurorobotics Lab, University of Freiburg, 79110 Freiburg, Germany [3]Department of Mathematics, University of Freiburg, 79110 Freiburg, Germany. Correspondence to: Rudolf Reiter <rudolf.reiter@imtek.uni-freiburg.de>.

Workshop on Foundations of Reinforcement Learning and Control at the 41st International Conference on Machine Learning, Vienna, Austria. Copyright 2024 by the author(s).

to correct an RL policy if a collision check fails. RL is also used for MPC weight tuning, such as in (Song & Scaramuzza, 2020) for UAVs and in (Zarrouki et al., 2021) for adaptive control in autonomous driving. Related research for motion planning of autonomous racing was recently surveyed in (Betz et al., 2022). Several works focus on local planning without strategic considerations (Vázquez et al., 2020; Wurman et al., 2022), thus can not directly be used in multi-agent settings. Other works use a game-theoretic framework (Liniger & Lygeros, 2020), which often limits the applicability due to its complexity. An algorithm for obtaining Nash equilibria is iterated best response (IBR), as shown for drone racing in (Spica et al., 2020) or for vehicle racing in (Wang et al., 2021). However, IBR has high computation times. An algorithm aiming at the necessary KKT conditions of the generalized Nash equilibrium problem is presented in (Le Cleac'h et al., 2022). However, the resulting optimization problem is complicated to solve.

2. 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 (Wabersich & Zeilinger, 2021) 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 (Bock & Plitt, 1984) one obtains decision variables for the state $X = [x_0, \dots, x_N] \in \mathbb{R}^{n_x \times (N+1)}$ and for the controls $u = [u_0, \dots, u_{N-1}] \in \mathbb{R}^{n_u \times N}$. Since the optimization problem can only be formulated for a finite horizon, a control invariant terminal set S^t must be included. The safety filter solves the following optimization problem

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

and takes the first control u_0^* of the solution (X^*, U^*) as output $u^{\mathrm{S}} := u_0^*$. The authors in (Wabersich & Zeilinger, 2021) 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 to safety-relevant environments. We do not require the same physical inputs to our filter but modifications to a parametric optimization problem, similar to (Gros & Zanon, 2020). 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

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

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

3. 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 (Zhang et al., 2021). We use an obstacle avoidance rule, according to the autonomous racing competitions *Roborace* (Roborace, 2020) and *F1TENTH* (O'Kelly et al.), 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 (Li et al., 2021), where the LV only avoids an inevitable crash, which we state detailed in Sec. 4.2.2. 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 to obtain a parametric cost function. The expansion

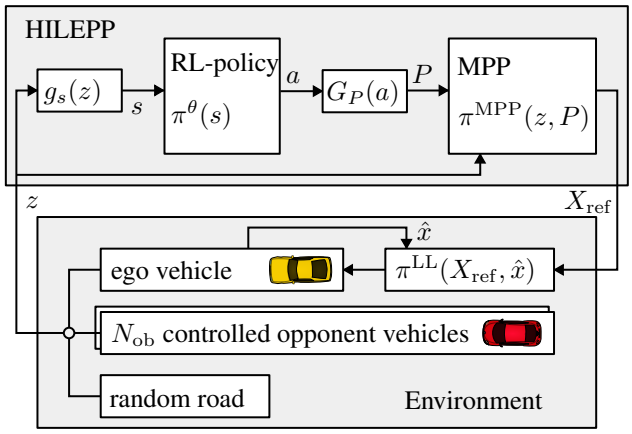

Figure 1. Proposed control structure. The multi-vehicle environment constitutes a trajectory tracking ego agent (lowest-level controller $\pi^{\mathrm{LL}}(\cdot)$). A state z concatenates all $N_{\mathrm{ob}} + 1$ vehicle states and road curvature information. A function $g_s(z)$ projects the state 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.

function is used to include prior knowledge and to obtain an optimization-friendly cost function in the MPP.

4. 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.

4.1. Vehicle Model

We use rear-wheel-centered kinematic single-track vehicle models in the Frenet coordinate frame, similar to (Kloeser et al., 2020). We use the states $x = \begin{bmatrix}\zeta & n & \alpha & v & \delta\end{bmatrix}^\top$ and controls $u = \begin{bmatrix}F_{\mathrm{d}} & r\end{bmatrix}^\top$, with the longitudinal force F_{d} and the steering rate r, which is the first derivative of the steering angle δ. Moreover, the states include the longitudinal position ζ, the lateral position n, the heading angle mismatch α, and the velocity v. The Frenet frame vehicle model is parameterized by the curvature $\kappa(\zeta)$, the mass m and length l and given as

$$\dot{x} = f(x, u) = \begin{bmatrix} \frac{v\cos(\alpha)}{1-n\kappa(\zeta)} \\ v\sin(\alpha) \\ \frac{v}{l}\tan(\delta) - \frac{\kappa(\zeta)v\cos(\alpha)}{1-n\kappa(\zeta)} \\ \frac{1}{m}(F_{\mathrm{d}} - F_{\mathrm{res}}(v)) \\ r \end{bmatrix}. \tag{3}$$

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

4.2. Safety Constraints

The MPP formulation should restrict trajectories X_{ref} to satisfy model constraints. Safety requires feasibility regarding vehicle limitations and obstacle constraints, which are explained in the following section. Recursive feasibility in (2) is given by the terminal safe set $\mathcal{S}^{\mathrm{t}} := \{x \mid \alpha = 0, v \leq 0\}$.

4.2.1. Vehicle Limitations

We use box constraints for states B_x and controls B_u as in (Kloeser et al., 2020). Further, we use a lateral acceleration constraints set

$$B_{\mathrm{lat}}(\sigma) := \left\{ x \,\middle|\, \left| \frac{v^2\tan(\delta)}{l} \right| \leq \overline{a}_{\mathrm{lat}} + \sigma_a \right\}, \tag{4}$$

to account for friction limits.

4.2.2. Obstacle Constraints

We approximate by an ellipse and 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. For obstacle avoidance concerning the ellipse matrix, we use the constraint set in compact notation

$$B_{\mathrm{O}}(x^{\mathrm{ob}}, \Sigma^{\mathrm{ob}}) = \left\{ x \in \mathbb{R}^2 \,\middle|\, \left\| P(\mathcal{F}^{-1}(x)) - p^{\mathrm{ob}} \right\|^2_{(\Sigma^{\mathrm{ob}}(x))^{-1}} \geq 1 \right\}. \tag{5}$$

4.2.3. 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

$$\dot{\zeta}^{\mathrm{ob}} = \frac{v^{\mathrm{ob}}(t)\cos(\hat{\alpha}^{\mathrm{ob}})}{1 - n^{\mathrm{ob}}\kappa(\zeta^{\mathrm{ob}})} \tag{6a}$$

$$\dot{n}^{\mathrm{ob}} = v^{\mathrm{ob}}(t)\sin(\hat{\alpha}^{\mathrm{ob}}) \tag{6b}$$

$$\dot{v}^{\mathrm{ob}} = \frac{1}{m^{\mathrm{ob}}} F_{\mathrm{d}}^{\mathrm{ob}}. \tag{6c}$$

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 an 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 (Roborace, 2020; O'Kelly et al.). Besides these minimum safety restrictions, interaction should be learned by the high-level RL policy. We simulate the system forward with a function $\Phi(\cdot)$, using steps of the RK4 integration function to obtain the predicted states $[x_0^{\mathrm{ob}}, \dots, x_N^{\mathrm{ob}}] = \Phi(\hat{x}^{\mathrm{ob}}, \hat{\alpha}^{\mathrm{ob}}, F_{\mathrm{d}}^{\mathrm{ob}})$.

4.3. Objective

For the parameterized cost function $L(X, U, a)$, we propose a formulation with the following properties: (i) simple structure for reliable and fast NLP iterations, (ii) expressive behavior related to strategic driving, (iii) low-dimensional action space, (iv) 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. 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 $a_{\mathrm{I}} = [v_{\mathrm{ref}} \quad n_{\mathrm{ref}}]^\top$).

4.4. Nonlinear programming 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 superior 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

$$G_P(a) : a \to \Big(\xi_{\mathrm{ref},0}(a), \dots, \xi_{\mathrm{ref},N}(a), Q_{\mathrm{w}}(a)\Big), \tag{7}$$

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 (3), the MPC path constraints for obstacle avoidance (5), vehicle constraints B_x, B_u and (4) and the parametric cost functions of (2). The full objective 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

$$L(X, U, a) = \sum_{k=0}^{N-1} \|x_k - \xi_{\mathrm{ref},k}(a)\|^2_{Q_{\mathrm{w}}(a)} + \|u_k\|^2_R + \|x_N - \xi_{\mathrm{ref},N}(a)\|^2_{Q^t}.$$

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

$$\begin{aligned}
\min_{X,U} \quad & L(X, U, a) \\
\text{s.t.} \quad & x_0 = \hat{x}, \quad x_N \in \mathcal{S}^{\mathrm{t}}, \\
& x_{i+1} = F(x_i, u_i) && i = 0, \dots, N-1, \\
& u_i \in B_u, && i = 0, \dots, N-1, \\
& x_i \in B_x \cap B_{\mathrm{lat}} && i = 0, \dots, N, \\
& x_i \in B_{\mathrm{ob}}(p_i^{\mathrm{ob},j}, \Sigma_i^{\mathrm{ob},j}) && i = 0, \dots, N, \\
& && j = 0, \dots, N_{\mathrm{ob}}.
\end{aligned} \tag{8}$$

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

Algorithm 1 MPP

Input: action a, ego states $\hat{x}$, N_{ob} obstacle states $\hat{x}^{\mathrm{ob}}$
Output: planned trajectory X_{ref}
for $j = 1$ **to** N_{ob} **do**
  if $\hat{\zeta}^{\mathrm{ob}} \leq \hat{\zeta}$ **then**
    Consider opp. as FV: $F_{\mathrm{d}}^{\mathrm{ob}} \leftarrow \underline{F}_{\mathrm{d}}^{\mathrm{ob}}$
  else
    Consider opp. as LV $F_{\mathrm{d}}^{\mathrm{ob}} \leftarrow 0$
  end if
  Predict $[x_0^{\mathrm{ob}}, \dots, x_N^{\mathrm{ob}}] = \Phi(\hat{x}^{\mathrm{ob}}, \hat{\alpha}^{\mathrm{ob}}, F_{\mathrm{d}}^{\mathrm{ob}})$
  Compute constraint ellipses $\Sigma_k^{\mathrm{ob},j} = \Sigma_0(\varphi^{\mathrm{ob},j})$
end for
Compute weights $\big(\zeta_{\mathrm{ref},k}, Q_{\mathrm{w}}\big) \leftarrow G_P(a)$
$X_{\mathrm{ref}} \leftarrow$Solve NLP (2) with $\big(\zeta_{\mathrm{ref},k}, Q_{\mathrm{w}}\big)$

5. *Hierarchical Learning-based Predictive Planner*

The MPP of Sec. 4 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 (Zhang et al., 2021) and solve for the best response. We use the soft actor-critic (SAC) (Haarnoja et al., 2018) to learn the parameterization of the MPP, where the exploration noise is added at the parameters a of the MPP, which is the sampled output of the policy. In the following, we describe the training procedure in detail.

5.1. Training Environment

In the following, we describe Alg. 2 which we use for training and Alg. 3, used for the final deployment of HILEPP.

Algorithm 2 HILEPP training

Input: num. of episodes n_{epi}, max. env. steps n_{scene}, reset function $(z, \kappa(\zeta)) = Z()$, reward function $R(s, a)$
Output: learned policy $\pi^\theta(\zeta)$
for $j = 1$ **to** n_{epi} **do**
  reset + randomize environment $(z, \kappa(\zeta)) \leftarrow Z()$
  for i in range(n_{scene}) **do**
    get policy input state $s \leftarrow g_s(z)$
    sample high-level action $a \sim \pi^\theta(\cdot|s)$
    evaluate planner $X_{\text{ref}} \leftarrow$MPP$(a, z)$
    simulate environment $z_{\text{next}} = \text{sim}(X_{\text{ref}})$
    get reward $r \leftarrow R(z_{\text{next}}, a)$
    store $(z, z_{\text{next}}, r, a)$ in buffer $\mathcal{D}$
    update z$\leftarrow z_{\text{next}}$
    if it's time to update **then**
      $\theta \leftarrow$SAC update sampling from $\mathcal{D}$
    end if
  end for
end for
return $\pi^\theta(s)$

Algorithm 3 HILEPP deployment

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

In the HILEPP training and deployment Alg. 2 and 3, we reduce the RL state space, based on domain knowledge which we put into the function $g_s(z_k)$. Therefore, the infinite-dimensional race track layout defined through its curvature $\kappa(\zeta)$ is approximated by finite 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 $\zeta_{\text{ob}} - \zeta$ to the ego vehicle to state $s_{\text{ob}_i} = [\zeta_{\text{ob}_i} - \zeta, n_{\text{ob}_i}, v_{\text{ob}_i}, \alpha_{\text{ob}_i}]^\top$. Combined, we get the following position-invariant RL-state

$$s_k = g_s(z_k) = \\ [\kappa(\zeta + d_i), \dots, \kappa(\zeta + d_N), s^\top, s_{\text{ob}_1}^\top, \dots, s_{\text{ob}_{N_{\text{ob}}}}^\top]^\top. \tag{9}$$

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

$$R(s, a) = \frac{\dot{s}}{200} + \sum_{i=1}^{N_{\text{ob}}} 1_{\zeta_k > \zeta_k^{\text{ob}_i}}. \tag{10}$$

At each time step, the high-level RL policy chooses a parameter for the MPP; thus, the RL action space is the parameter space of the MPP.

For training the high-level RL policy, the environment is simulated for n_{epi} episodes and a maximum of n_{scene} steps by a numerical integration of the dynamics with $z_{\text{next}} = \text{sim}(z, \kappa(\cdot))$. The road layout defined by $\kappa(\zeta)$ is randomized within an interval $[\text{-}0.04, 0.04]\text{m}^{-1}$ before each training episode. The curvature is set together with initial random vehicle states z by a reset function $(z, \kappa(\zeta)) = Z()$. Notice that Alg. 2 involves additional parameters related to the SAC algorithm (Haarnoja et al., 2018), e.g., weights of the critic networks, which are not explicitly stated here.

6. 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. 1). 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 (10). 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 the ego agent baseline, are simulated with the state-of-the-art MPP (Alg. 1) with a fixed action a that accounts

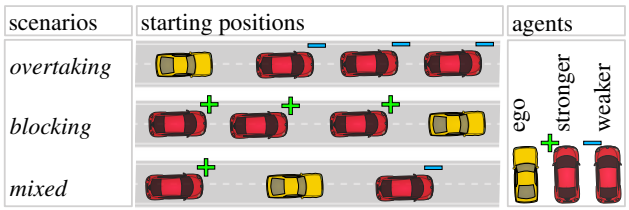

Figure 2. Scenarios differ in the initial rank and performance of vehicles.

Table 1. Vehicle model parameters. SI-units, if not stated explicitly.

Name	Variable	Ego Agent	"Weak" Agent	"Strong" Agent
wheelbase	$l_{\mathrm{r}}, l_{\mathrm{f}}$	1.7	1.7	1.7
chassis lengths	$l_{\mathrm{r,ch}}, l_{\mathrm{f,ch}}$	2	2	2
chassis width	w_{ch}	1.9	1.9	1.9
mass	m	1160	2000	600
max. lateral acc.	$\underline{a}_{\mathrm{lat}}, \overline{a}_{\mathrm{lat}}$	± 8	± 5	± 13
max. acc. force	$\overline{F}_{\mathrm{d}}$	10kN	8kN	12kN
max. brake force	$\underline{F}_{\mathrm{d}}$	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

for non-strategic time-optimal driving with obstacle avoidance. We perform a hyper-parameter (HP) search for the RL parameters with *Optuna* (Akiba et al., 2019). 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 in each scenario. To estimate the final policy's performance, 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 where all 30 runs for one method were run on 8 GeForce RTX 2080 Ti with an AMD EPYC 7502 32-Core Processor with a training time of around 6 hours. For solving the NLP, the solver `acados` (Verschueren et al., 2021) was used with parameters of Tab. 2 using slack variables to increase numerical stability.

6.1. Results

In Fig. 3, we compare the training performance related to the reward (10) and in Fig. 4, we show the final performance of the two HILEPP formulations. HILEPP quickly outperforms the baseline MPP as well as the pure RL formulation,

Table 2. Parameters for MPP in SI units

Name	Variable	Value
nodes/disc. time	$N/\,\Delta t$	$50/\,0.1$
state weights	q	$[1, 500, 10^3, 10^3, 10^4]\Delta t$
terminal state weights	q_N	$[10, 90, 100, 10, 10]$
control weights	R	$\mathrm{diag}([10^{-3}, 2 \cdot 10^6])\Delta t$

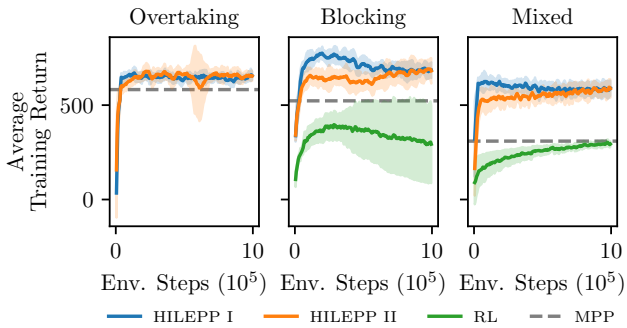

Figure 3. Training performance of average episode returns of HILEPP with different action interfaces, pure RL and the MPP baseline. Remarkably, we could not train a successful pure RL agent in the overtaking scenario.

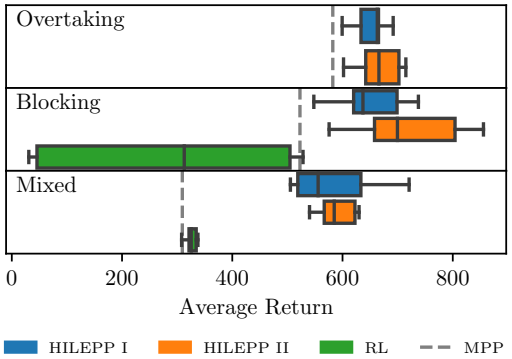

Figure 4. Final episode return of 100 evaluation runs of the proposed interfaces for different scenarios (Fig. 2).

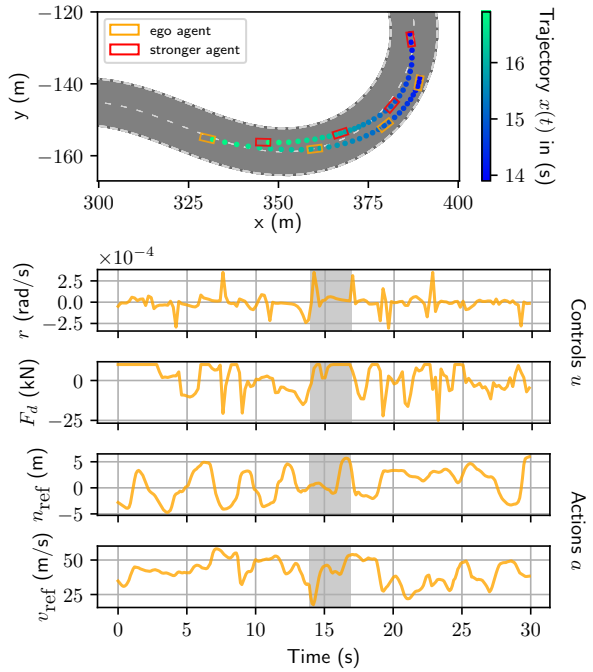

Figure 5. Exemplary evaluation episode of the HILEPP-I planner in the mixed scenario. On the bottom, the controls of the MPP and the actions of the high-level RL policy are shown. The grey box indicates a time window where snapshots of a blocking maneuver are shown in the top plot.

showing its high sample efficiency. With a smaller action space, HILEPP-I learns faster. However, with more samples, HILEPP-II outperforms the smaller action space in all three scenarios on the evaluation runs regarding median performance, see Fig. 4. The training was stopped after 10^6 steps due to the high training time and the slow return increase, as shown in 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 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 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). The HILEPP is capable of planning trajectories with approximately 100Hz, which is sufficient and competitive for automotive motion planning (Betz et al., 2022). 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/`.

7. Conclusions

We have shown how a hierarchical combination of RL and MPC can outperform a basic time-optimal and obstacle-avoiding approach and pure deep-learning-based RL in several scenarios. Since the MPP can be considered part of the environment, as seen from the RL policy, the policy can be updated by sampling from the replay buffer without solving the optimization problem again. If MPP was used as part of the policy, i.e., exploration noise and the SAC critic were evaluated at the controls u rather than at the parameters a, the parameter update would require differentiating the optimizer. This would require a converged solution of the MPP with an NLP solver which can hardly be parallelized and used within standard RL frameworks. The approach is restricted to stationary policies of opponents.

References

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

Betz, J., Zheng, H., Liniger, A., Rosolia, U., Karle, P., Behl, M., Krovi, V., and Mangharam, R. Autonomous vehicles on the edge: A survey on autonomous vehicle racing. *IEEE Open Journal of Intelligent Transportation Systems*, 3:458–488, 2022. doi: 10.1109/ojits.2022.3181510.

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

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

Brunke, L., Greeff, M., Hall, A. W., Yuan, Z., Zhou, S., Panerati, J., and Schoellig, A. P. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1):411–444, 2022. doi: 10.1146/annurev-control-042920-020211.

Greatwood, C. and Richards, A. G. Reinforcement learning and model predictive control for robust embedded quadrotor guidance and control. *Autonomous Robots*, 43 (7):1681–1693, October 2019. ISSN 0929-5593, 1573-7527. doi: 10.1007/s10514-019-09829-4.

Gros, S. and Zanon, M. Data-driven economic nmpc using reinforcement learning. *IEEE Transactions on Automatic Control*, 65(2):636–648, Feb 2020. ISSN 2334-3303. doi: 10.1109/tac.2019.2913768.

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

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

Le Cleac'h, S., Schwager, M., and Manchester, Z. Algames: A fast augmented lagrangian solver for constrained dynamic games. *Auton. Robots*, 46(1):201–215, jan 2022. ISSN 0929-5593. doi: 10.1007/s10514-021-10024-7.

Li, N., Goubault, E., Pautet, L., and Putot, S. 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.

Liniger, A. and Lygeros, J. A noncooperative game approach to autonomous racing. *IEEE Transactions on Control Systems Technology*, 28(3):884–897, 2020. doi: 10.1109/TCST.2019.2895282.

Lubars, J., Gupta, H., Chinchali, S., Li, L., Raja, A., Srikant, R., and Wu, X. Combining reinforcement learning with model predictive control for on-ramp merging. In *IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 942–947, 2021. doi: 10.1109/ITSC48978.2021.9564954.

O'Kelly, M., Zheng, H., Karthik, D., and Mangharam, R. F1tenth: An open-source evaluation environment for continuous control and reinforcement learning. *Proceedings of Machine Learning Research*, 123.

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

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

Roborace. Roborace season beta, 2020. URL https://roborace.com/.

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

Spica, R., Cristofalo, E., Wang, Z., Montijano, E., and Schwager, M. A real-time game theoretic planner for autonomous two-player drone racing. *IEEE Transactions on Robotics*, 36(5):1389–1403, 2020. doi: 10.1109/TRO.2020.2994881.

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

Verschueren, R., Frison, G., Kouzoupis, D., Frey, J., van Duijkeren, N., Zanelli, A., Novoselnik, B., Albin, T., Quirynen, R., and Diehl, M. acados – a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, Oct 2021. ISSN 1867-2957. doi: 10.1007/s12532-021-00208-8.

Wabersich, K. P. and Zeilinger, M. N. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica*, 129:109597, 2021. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2021.109597.

Wang, M., Wang, Z., Talbot, J., Gerdes, J. C., and Schwager, M. Game-theoretic planning for self-driving cars in multivehicle competitive scenarios. *IEEE Transactions on Robotics*, 37(4):1313–1325, 2021. doi: 10.1109/TRO.2020.3047521.

Wurman, P., Barrett, S., Kawamoto, K., MacGlashan, J., Subramanian, K., Walsh, T., Capobianco, R., Devlic, A., Eckert, F., Fuchs, F., Gilpin, L., Khandelwal, P., Kompella, V., Lin, H., MacAlpine, P., Oller, D., Seno, T., Sherstan, C., Thomure, M., and Kitano, H. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 602:223–228, 02 2022. doi: 10.1038/s41586-021-04357-7.

Zarrouki, B., Klös, V., Heppner, N., Schwan, S., Ritschel, R., and Voßwinkel, R. Weights-varying mpc for autonomous vehicle guidance: a deep reinforcement learning approach. In *2021 European Control Conference (ECC)*, pp. 119–125, 2021. doi: 10.23919/ECC54610.2021.9655042.

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