Preprints of the 23rd IFAC World Congress
Busan, Republic of Korea, August 23-28, 2026

Safe and Optimal Trajectory Learning for Autonomous Racing via Deep Reinforcement Learning and Control Barrier Functions

José P. Carvalho * **A. Pedro Aguiar** *

* *Research Center for Systems and Technologies, Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, 4200-465, Portugal (e-mail: jpcarvalho@fe.up.pt, pedro.aguiar@fe.up.pt).*

Abstract: This paper presents a safety-certified reinforcement learning framework for autonomous racing that minimizes lap times while strictly enforcing safety and velocity constraints. To address the brittleness and lack of safety guarantees in standard Deep Reinforcement Learning (DRL), we propose a multi-stage architecture. In this framework, a Proximal Policy Optimization (PPO) policy optimizes racing performance, while a structured safety filtering stack combining Control Barrier Functions (CBF), Control Lyapunov Functions (CLF), and a feasibility-enforcing mechanism, enforces track limits and velocity constraints while guaranteeing episode completion. This filtering stack restricts exploration to admissible regions, stabilizing training, and preventing constraint violations. Simulations on 1:10-scale tracks demonstrate near-optimal trajectories and successful zero-shot generalization to unseen environments, without requiring computationally expensive offline trajectory optimization.

Keywords: Autonomous racing, Deep Reinforcement Learning, Control Barrier Functions, Safe control, Path following, Motion planning

1. INTRODUCTION

As autonomous and cyber-physical systems become increasingly prevalent, motion planning remains a challenge across applications (Garrett et al., 2021). Driven by the push toward greater autonomy, the complexity of both tasks and motion-planning requirements is rapidly increasing. This growing complexity introduces higher risks, as failures become more unpredictable and severe (Brunke et al., 2022). Consequently, motion controllers must not only be performant, but also provably safe.

In the context of motion planning, one task of particular interest is Path-Following (PF), which consists of driving a vehicle along a pre-defined geometric path while maintaining a path-dependent velocity profile (Hung et al., 2023; Aguiar, 2017; Reis et al., 2019). This task is os interest in applications where the main concern is accurately traversing a path, while speed is a secondary objective.

Nonetheless, the classical PF task can be too restrictive in the paradigm of intelligent robots. A PF controller yields trajectories akin to a railway track, with the vehicles not deviating from the nominal path. However, an intelligent system should be able to deviate from the nominal path as long as it benefits its overall goal. Naturally, this deviation from the path must be controlled so that the vehicle still completes it while not violating safety constraints.

A classical case where an autonomous agent might benefit from deviating from its pre-defined path is in autonomous racing (Betz et al., 2022). In this case, one would wish to follow the path and speed profile that yields the minimum-time solution for the trajectory that completes it. However, finding such a path depends on the vehicle dynamics and environmental conditions, making it a non-trivial task. Therefore, most classical control algorithms track a potentially suboptimal path that is computed offline.

Deep Reinforcement Learning (DRL) has appeared as a potential solution for autonomous racing (Wang et al., 2025). The most famous success story is Swift, which beat the human world champions in drone racing (Kaufmann et al., 2023). In Swift, the learning policy does not track a nominal path, instead, it directly learns to find the best trajectories through trial and error. However, these policies lose desirable properties such as liveness and safety guarantees, which can potentially lead to critical failures.

In this paper, we present a novel learning-based framework that formulates the autonomous racing problem as an optimal control problem with safety constraints that stands in the middle between classical path-following algorithms and common DRL approaches. We provide a look-ahead nominal path that is simply the geometric center of the race track, allowing our framework to skip expensive computations of optimized trajectories. Instead, we use a time-varying Control Barrier Function-based safety filter

⋆ This work was conducted under the FCT grant 2024.01894.BD and partially supported by SYSTEC (UID/00147 — DOI 10.54499/UID/00147/2025) and ARISE (DOI 10.54499/LA/P/0112/2020), both funded by Fundação para a Ciência e a Tecnologia, I.P./ MECI through the national funds.

©2026 the authors. Accepted by IFAC for publication under a Creative Commons Licence CC-BY-NC-ND

that encodes the track boundaries to restrict our policy's learning to trajectories that are provably safe. Through a feasibility filter and a backstepping Control Lyapunov Function, we also guarantee that the vehicle is always capable of finishing the PF task. The main contributions of the paper are:

- A formulation of the PF task that can be used to find a novel class of control policies for autonomous racing.
- A Multi-Stage control architecture consisting of: 1) A DRL policy; 2) A smoothening filter for DRL Output; 3) A CBF-based safety filter that computes safe velocity commands; 4) A feasibility QP that ensures the existence of a valid velocity command and liveness of the task; 5) A Control Lyapunov Function that transforms the safe velocity commands into acceleration commands.
- A learning-based control policy that is safe and *live* by construction and yields optimized racing trajectories.

The remainder of the paper is organized as follows: Section 2 formulates the modified path-following task, Section 3 formulates the Markov Decision Process and Deep Reinforcement Learning framework for learning the control policies and introduces Multi-Stage architecture that guarantees safety and liveness of the Path-Following task. Finally, Section 4 presents the main simulation results, and Section 5 provides some brief conclusions and future work.

2. PROBLEM FORMULATION

In this paper, we address the problem of path following for a differential wheeled robot that can be modeled as a nonlinear control-affine system

$$\dot{x} = f(x) + g(x)u, \tag{1}$$

with state $x := [\mathrm{x}, \mathrm{y}, \theta, v, \omega]^T \in \mathcal{X} \subset \mathbb{R}^5$, the control input $u \in \mathbb{R}^2$. Here, $\mathrm{x}, \mathrm{y} \in \mathbb{R}$ denote the Cartesian position of the robot in the plane, $\theta \in \mathbb{S}^1$ represents its orientation with respect to an inertial frame, and $v, \omega \in \mathbb{R}$ are the linear and angular velocities, respectively. The control input $u \in \mathbb{R}^2$ corresponds to the commanded linear and angular accelerations. The robot is modeled as a second-order unicycle with $f(x) = [v\cos\theta - \omega L\sin\theta \;\; v\sin\theta + \omega L\cos\theta \;\; \omega \;\; 0 \;\; 0]^T$ and $g(x) = [\mathbf{0}_{2\times 3} \;\; I_2]^T$, where $\mathbf{0}_{2\times 3} \in \mathbb{R}^{2\times 3}$ is the zero matrix, $I_2 \in \mathbb{R}^{2\times 2}$ is the identity, and $L \in \mathbb{R}_{>0}$ is the distance from the robot's center of rotation to the front. This system can be cascaded into a two control layers with $q := [\mathrm{x}, \mathrm{y}, \theta]^T$ and $\xi := [v, \omega]^T$ as

$$\dot{q} = f_0(q) + g_0(q)\xi, \quad \dot{\xi} = f_1(q, \xi) + g_1(q, \xi)u, \tag{2}$$

where f_0, f_1, g_0 and g_1 can be derived from the original vector fields f, g. We further consider that the vehicle has velocity constraints such that $\xi \in \mathcal{V}$, where $-v_{\min} \leq v \leq v_{\max}$ and $-\omega_{\max} \leq \omega \leq \omega_{\max}$, with $\mathcal{V}$ defined accordingly.

2.1 Path Following

The robot is required to follow a path represented by a parameterized m-th continuous differentiable function $p_d(\gamma) : \mathbb{R} \to \mathbb{R}^q$, where $\gamma \in \Gamma$ denotes the path parameter and q corresponds to the dimension of the task space. Without loss of generality, we consider that $\Gamma := [\gamma_0, \gamma_1]$ is a compact set for any $\gamma_1 > \gamma_0$.

Consider a two-dimensional task space, $\mathbb{R}^q = \mathbb{R}^2$, in which the parameter γ evolves according to the dynamics

$$\dot{\gamma}(t) = v_d(t), \tag{3}$$

where $v_d(t) \in \mathbb{R}$ is the speed assignment of the path at time t, and is a free control variable. Thus, the position of the virtual point on the path is given by $p_d(\gamma(t)) \in \mathbb{R}^2$ and its speed in the task space is obtained as $v_p = v_d\|p_{d\gamma}\|$, where $p_{d\gamma} := \frac{\partial p_d}{\partial \gamma}$. Now, the time derivative of the virtual point position can be expressed as

$$\dot{p}_d = \frac{\partial p_d}{\partial \gamma}\frac{\partial \gamma}{\partial t} = p_{d\gamma}\,\dot{\gamma}. \tag{4}$$

Now we can define the path-following task:

Definition 1. (Path Following Task). Consider a control system described by (1) and let $p_d(\gamma) : \Gamma \to \mathbb{R}^q$ be a path parameterized by $\gamma \in \Gamma$, with dynamics given by (3). Design a control law π such that all the relevant closed-loop signals are bounded and the path following error satisfies

$$\|y - p_d(\gamma)\|_{Q(\gamma)} \leq \varepsilon(\gamma) \;\forall t \in T, \tag{5}$$

where $y := c(x)$ is the system output given by the map $c : \mathcal{X} \to \mathbb{R}^q$. Here, $H(\gamma)$ is a parameterized positive definite matrix, and $\varepsilon(\gamma) \in \mathbb{R}^+$ is a path-dependent error bound. If condition (5) holds, then the Path-Following (PF) task holds.

This definition differs from the classic formulation of Path Following, where the goal is to design a feedback controller such that $\|y - p_d\| \to 0$ as $t \to \infty$ (Aguiar, 2017). Following the definition of the task, one can now define liveness.

Definition 2. (Liveness). Given the compact sets $\Gamma := [\gamma_0, \gamma_1]$ and $T := [t_0, t_1]$, a feedback control law π is said to have the **liveness** property if: **(1)** The Path Following task holds; **(2)** $\gamma(t_f) = \gamma_1$ for some $t_f \leq t_1$.

In this formulation, the designed controller π does not have to track exactly the reference path $p_d(\gamma)$, instead, it just has to keep the error norm induced by matrix $Q(\gamma)$ bounded by $\varepsilon(\gamma)$ for all $t \in T$, while guaranteeing that task will finish in finite time. Such that one can deviate from the reference path if it allows for better trajectories. Now, the objective of this paper is to solve a minimum-time Optimal Control Problem (OCP) that solves the PF task while minimizing t_f.

3. METHODOLOGY

Since the proposed OCP is, in general, difficult to solve, even offline, our objective is to obtain a tractable approximation that preserves the key guarantees of the original formulation. Therefore, we recast the problem as a Markov Decision Process (MDP) and find a DRL policy that minimizes lap time. While DRL can achieve near-optimal performance, standard policies lack formal safety guarantees and often fail outside their training distribution. To resolve this, we propose a multi-stage architecture (Fig. 1) that separates performance optimization from constraint enforcement. First, a DRL policy generates velocity reference commands, after which a filter stack minimally modifies them to ensure safety and feasibility.

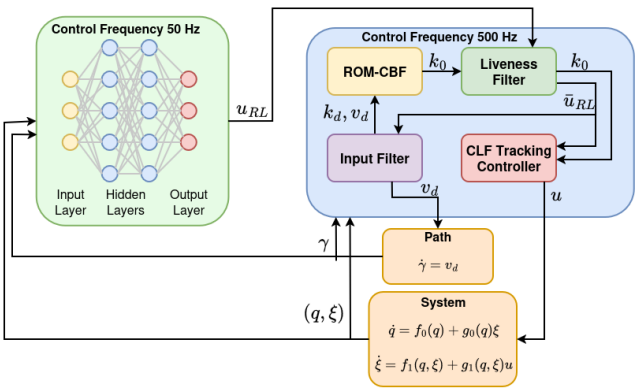


Fig. 1. Proposed Control Architecture.

3.1 Reinforcement Learning Policy and Markov Decision Process

The first stage of the framework encompasses the RL policy $\pi_{RL} : \mathcal{S} \to \mathcal{V} \times \mathcal{V}_p$, which outputs a reference velocity command u_{RL}. To find a control policy that is an (approximate) solution to the OCP, we convert the OCP into an MDP. An MDP can be described by the tuple $\langle \mathcal{S}, \mathbb{A}, \mathbf{P}, \mathbf{R}, \beta \rangle$, where $\mathcal{S}$ is the set that contains all states, $\mathbb{A}$ is the Action Space, $P : \mathcal{S} \times \mathbb{A} \to \mathcal{S}$ is the state transition function, $\mathbf{R} : \mathcal{S} \times \mathbb{A} \to \mathbb{R}$ is the Reward Function and β is the discount factor.

Considering the episodic nature of an MDP, let $G_t = \sum_{t=0}^{\infty} R_{t+1}$ be the return of a given episode. Then, the value function can be defined as the expected return when the agent is at state s and following policy $\pi : \mathcal{S} \to \mathbb{A}$:

$$v_\pi(s) = \mathbb{E}[G_t \mid S_t = s]. \tag{6}$$

Then, the goal of RL reduces to finding the policy π^* that maximizes the value function, i.e $\pi^* = \max_\pi v_\pi(s)$.

Action Space and Policy We now start modeling the MDP by defining the Action Space $\mathbb{A}$. An action is the tuple $a = u_{RL} = (k_d^{RL}, v_p^{RL}) \in \mathbb{A} := \mathcal{V} \times \mathcal{V}_p$, where $\mathcal{V}_p := [v_{p\min}, v_{p\max}]$ with $v_{p\min}, v_{p\max} > 0$ is the set of possible path velocities. Typically, $v_{p\max}$ is slightly larger than the robot's maximum linear velocity, as it provides greater control authority over path evolution.

Since the policy π is not inherently safe, the chosen action will have to be modified by the Second Stage, referred as *Filter Stage*. This originates the policy that interacts with the system to be defined as $\pi^{safe}(s) := \Phi(\pi_{RL}(s))$, where $\Phi : \mathbb{A} \to \mathbb{A}^{safe}$. In the further sections, we will further elaborate on the filter Φ.

State Space The State Space $\mathbf{S}$ is a superset of the system state, $\mathcal{X} \subseteq \mathcal{S}$. Formally, a state $s \in \mathcal{S}$ is defined as

$$s := \langle \xi, \mathbf{p_d}(\gamma), k_d, v_p, \gamma, t \rangle, \tag{7}$$

where $\xi = [v, w]$ are the robot's linear and angular velocities, respectively. The term $\mathbf{p_d}(\gamma) = [\mathcal{T} p_d(\gamma(t) + \Delta_i) \; \forall i = 0, \ldots, H]$ represents a lookahead along the reference path, for a finite horizon $H \in \mathbb{R}_{>0}$. Here, $\mathcal{T}$ denotes the transformation from the global frame to the robot's frame, and $\Delta_i \in \mathbb{R}$ is an offset that spaces consecutive points. Note that despite the pose $q = (\mathrm{x}, \mathrm{y}, \theta)$ not being directly part of the state, the information is encoded through the transform. Furthermore, the path being in the robot's reference frame facilitates generalization. The variable $k_d \in \mathcal{V}$ is the desired (filtered) velocity reference, and $v_p \in \mathcal{V}$ represents the current path speed. Finally, $\gamma \in \Gamma$ is the path parameter, and t is the corresponding time.

Reward Function The last piece of the MDP is the Reward Function. This function must encapsulate the goal of the OCP. While a single reward at episode termination, $\mathbf{R} = -t_f$, would, in theory, be equivalent to the OCP, it is unfortunately too sparse. Instead, we must shape the reward to make it denser, while keeping the original goal as unchanged as possible, resulting in

$$\begin{aligned} \mathbf{R} = &-K_c \|k_0 - k_d\|^2 - K_f \|\bar{u}_{RL}\|^2 \\ &+ K_v \min(v_p, v_{p\max} - v_{th}) + K_t \max(0, T_t - t_f), \end{aligned} \tag{8}$$

where k_0 is the safe velocity command, $\bar{u}_{RL}$ is the feasibility correction, $T_t \in \mathbb{R}_{>0}$ is a target time, $v_{th} \in \mathbb{R}_{>0}$ is a velocity threshold, and $K_c, K_f, K_v, K_t \in \mathbb{R}_{>0}$ are positive gains that weigh the importance of each event. We note that $t_f = T_t$ if the state s is not terminal, or if it was terminated before reaching the finish line. As for the gains, K_c penalizes the intervention of the safety filter, K_v incentivizes keeping a high path velocity v_p, K_f penalizes interventions of the feasibility filter, T_t is a target time and K_t is the corresponding gain, serving as a time minimization term.

3.2 Safety and Feasibility Filtering Layer

We now describe the second stage of the proposed framework, referred to as the *Filtering Layer*. This layer is responsible for transforming the nominal action proposed by the DRL policy into a command that is smooth, safe, and feasible with respect to the vehicle dynamics and constraints. It is composed of the following steps.

Step 1. Low-Pass Filter (LPF) A key requirement of the safety mechanisms employed in this work is that the reference velocity signal be smooth [1]. However, u_{RL} is inherently non-smooth. To address this issue, we introduce a first-order Low-Pass Filter (LPF), yielding the following filtered reference dynamics

$$\dot{u}_f = A_f \big(u_{RL} + \bar{u}_{RL} - u_f\big), \tag{9}$$

where $A_f = \mathrm{diag}(a_v, a_\omega, a_{vp})$ is a diagonal matrix of positive filter gains. The term $\bar{u}_{RL}$ represents a correction signal produced by a downstream feasibility filter, which will be detailed later. The filter output vector u_f is decomposed as $u_f = [k_d, v_d]$, where $k_d \in \mathcal{V}$ denotes the filtered reference vehicle velocities, and $v_d \in \mathbb{R}$ is the filtered reference path progression speed.

Step 2. Control Barrier Function Safety Filter The filtered reference commands (k_d, v_d) obtained from the LPF are not guaranteed to satisfy safety constraints. We define safety as the satisfaction of a set of path-dependent state constraints, in this case, the track boundaries. To formalize this notion, we rely on the concept of *forward invariance*. A set $\mathcal{C}(t)$ is *forward invariant* with respect to (1) if for a given initial condition $x(t_0) \in \mathcal{X}$ there exists control signal $u : T \to \mathcal{U}$ such that there exists unique solution $\boldsymbol{x} : T \to \mathcal{X}$ to (1) with $\boldsymbol{x}(t) \in \mathcal{C}(t)$ for all $t \in T$.

[1] Here, smooth means differentiable as many times as necessary.

To enforce forward invariance, we employ a Control Barrier Function (CBF)-based safety filter. However, since the full system (1) has relative degree greater than one with respect to the safety constraints, standard CBF formulations cannot be directly applied. Instead, we will use the concept of Reduced-Order Models (ROM) (Cohen et al., 2024). We begin by considering the modified top-layer dynamics

$$\dot{q} = f_0(q) + g_0(q)(k_0 + d), \tag{10}$$

where $k_0 \in \mathcal{V}$ denotes a velocity command and $d \in \mathcal{W}$ is a bounded disturbance satisfying $\|d\|_\infty \leq \delta$, for some known bound $\delta > 0$.

Let (10) be ROM for the whole vehicle (1), and let $h_0 : \mathcal{D} \times \Gamma \to \mathbb{R}$ with $\mathcal{D} \subset \mathbb{R}^3$ be a continuously differentiable function defining the path-dependent constraint sets

$$\mathcal{C}_0(\gamma(t)) := \{q \in \mathcal{D}(\gamma) \mid h_0(q, \gamma) \geq 0\}. \tag{11}$$

$$\mathcal{C}_\delta(\gamma(t)) := \{q \in \mathcal{D}(\gamma) \mid h_\delta := h_0(q, \gamma) + \iota(\delta) \geq 0\}. \tag{12}$$

In (12), $\mathcal{C}_\delta(\gamma(t))$ is an inflated safe set for a $\iota \in \mathcal{K}_\infty$ that compensates for the effect of bounded disturbances. Then, the concept of safety can be extended using the notion of Input-to-State Safety (ISSf) (Kolathaya and Ames, 2019) to accommodate disturbances.

Definition 3. (Input-to-State Safety). The system (10) is said to be input-to-state safe on a set $\mathcal{C}_0(\gamma)$ as in (11) if there exists a $\iota \in \mathcal{K}_\infty$ such that for all $d(t)$ with $\|d\|_\infty \leq \delta$ the set $\mathcal{C}_\delta(\gamma)$ defined as (12) is forward invariant.

Definition 4. (Input-to-State Safe CBF). A continuously differentiable function $h_0 : \mathcal{D} \times \Gamma \to \mathbb{R}$ defining a set $\mathcal{C}_0(\gamma)$ is said to be an ISSf CBF for (10) on $\mathcal{C}_\delta(\gamma)$ if there exists $\alpha \in \mathcal{K}_\infty$, $\eta \in \mathbb{R}_{>0}$ such that for all $x \in \mathcal{D}(\gamma)$, $\gamma \in \Gamma$ the set

$$K(q, \gamma) := \{k_0 \in \mathcal{V} \mid L_{f_0} h_0 + L_{g_0} h_0 k_0 + h_{0\gamma} v_d \geq -\alpha(h_0) + \frac{\|L_{g_0} h_0\|^2}{\eta}\}, \tag{13}$$

is non-empty. In (13) $h_{0\gamma} := \frac{\partial h_0}{\partial \gamma}$ and $L_{g_0} h_0, L_{f_0} h_0$ are the Lie derivatives of h_0 along f_0 and g_0. Moreover, if K is not a singleton for all $x \in \mathcal{D}(\gamma)$, and all $\gamma \in \Gamma$, then the ISSf-CBF is *strict*.

In this paper, we consider exclusively strict CBFs with class $\mathcal{K}$ functions of the form $\alpha(r) := \alpha r$. We further note that when $v_d = 0$, the barrier function is static and the corresponding safe sets are time-invariant.

Given an ISSf-CBF, we construct a safety filter by computing a control input k_0 that minimally modifies the reference velocity k_d and renders the set $\mathcal{C}(\gamma)$ ISSf. This leads to the following quadratic program (QP):

$$\begin{aligned} k_0 = \underset{k_0 \in \mathcal{V}}{\operatorname{argmin}} \ & \frac{1}{2}\|k_0 - k_d\|^2 \\ \textbf{s.t.} \ & L_{f_0} h_0 + L_{g_0} h_0 k_0 + h_{0\gamma} v_d \geq -\alpha h_0 + \frac{\|L_{g_0} h_0\|^2}{\eta}. \end{aligned} \tag{14}$$

Proposition 1. (Existence of a valid control). Let h_0 be a strict static ISSf CBF for (10), and let the path progression rate v_d be a controllable decision variable. Then, there exists a $v_d \geq 0$ such that $k_0 \in K \subset \mathcal{V}$ and $K \neq \emptyset$.

Proof. Given a valid strict static ISSf CBF, meaning there exists a $k_0 \in \mathcal{V}$ such that $L_{f_0} h_0 + L_{g_0} h_0 k_0 + \alpha(h_0) - \frac{\|L_{g_0} h_0\|^2}{\eta} < 0$, one can always adjust the controllable variable v_d to ensure the time-varying ISSf condition remains strictly feasible.

We note that since all the CBFs considered in this paper yield a compact safe set, then verifying the validity of a strict static ISSf can be a tractable problem, and therefore, given the controllability of v_d, the problem of guaranteeing feasibility is reduced to finding a proper $v_d(t)$ signal.

By solving (14), one obtains the minimally invasive safe controller $k_0 \in \mathcal{K}$, which is not smooth. To preserve smoothness, we employ a Softplus approximation of the unconstrained solution of (14) given by

$$\begin{aligned} k_0 &:= k_d + \lambda(a(q,\gamma), b(q,\gamma)) L_{g_0} h_0(q,\gamma)^T, \\ \lambda(a,b) &:= \begin{cases} 0 & b \leq 0 \\ \sigma \log(1 + e^{-\frac{a}{\sigma b}}) & b > 0 \end{cases} \end{aligned} \tag{15}$$

where $a := L_{f_0} h_0 + L_{g_0} h_0 k_d + h_{r\gamma} v_d + \alpha h_0$, $b := \|L_{g_0} h_0\|^2$, and $\sigma \in \mathbb{R}_{>0}$ is a tunable constant that can be decreased to achieve a tighter approximation, which is not guaranteed to satisfy the velocity constraints.

Step 3. Feasibility Filter In this step, we introduce a feasibility filter that adjusts the signal $v_d(t)$ to ensure satisfaction of the velocity constraints while preserving compliance with the ISSf condition.

Consider the discrete-time evolution of the safe command

$$k_0[n+1] := k_0[n] + \dot{k}_0[n] \Delta_f, \tag{16}$$

where Δ_f denotes the inner-loop discretization period and

$$\dot{k}_0 = \frac{\partial k_0}{\partial q}\dot{q} + \frac{\partial k_0}{\partial \gamma} v_d + \frac{\partial k_0}{\partial v_d}\dot{v}_d + \frac{\partial k_0}{\partial k_d}\dot{k}_d. \tag{17}$$

Although $k_0[n] \in \mathcal{V}$ by construction, its nominal evolution does not guarantee that $k_0[n+1]$ remains within bounds. As shown in (17), $\dot{k}_0$ depends on the terms k_d, v_d, $\dot{k}_d$, and $\dot{v}_d$. While k_d and v_d are fixed to preserve smoothness, $\dot{k}_d$ and $\dot{v}_d$ can be modified to enforce feasibility at the next timestep. Rewriting the filtered dynamics from (9),

$$\dot{k}_d = A_{k_d}\left(k_d^{RL} + \bar{k}_d - k_d\right), \ \dot{v}_d = A_{v_d}\left(v_d^{RL} + \bar{v}_d - v_d\right), \tag{18}$$

where $\bar{u}_{RL} = [\bar{k}_d, \bar{v}_d]$ denotes the corrective action produced by the feasibility filter. Grouping the remaining terms of $\dot{k}_0$ yields

$$\Lambda := \frac{\partial k_0}{\partial q}\dot{q} + \frac{\partial k_0}{\partial \gamma} v_d + \frac{\partial k_0}{\partial k_d}(k_d^{RL} - k_d) + \frac{\partial k_0}{\partial v_d}(v_d^{RL} - v_d). \tag{19}$$

The feasibility filter is then obtained by solving the QP:

$$\begin{aligned} & \underset{\bar{u}_{RL} \in \overline{\mathcal{U}}}{\arg\min} \|\bar{u}_{RL}\|_P \\ & \textbf{s.t} \ \Delta_t S \begin{bmatrix} \frac{\partial k_0}{\partial k_d} & \frac{\partial k_0}{\partial v_d} \end{bmatrix} \bar{u}_{RL} \leq D_b - \Delta_t S \Lambda \end{aligned} \tag{20}$$

where $\overline{\mathcal{U}}$ is the set of admissible corrections ($0 < v_d[n+1] \leq v_{d\max}$), $S \in \mathbb{R}^{4\times 2}$ is a selection matrix for the velocity bounds vector $D_b = [v_{\max}, v_{\min}, w_{\max}, w_{\min}]^T$, and $P = \operatorname{diag}(\sigma_v, \sigma_\omega, \sigma_{vd})$ induces a weighted L_2 norm.

Step 4. Safety Backstepping The proposed safety filter operates at the velocity level of the reduced-order model. However, the full-order vehicle dynamics (1) are actuated at the acceleration level. To bridge this gap, we employ a CLF-based backstepping layer that tracks the safety-certified velocity command k_0 using the acceleration input u. To this end, consider the Lyapunov function $V : \mathbb{R}^3 \times \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$ for the full order dynamics

$$V(q,\xi) = \frac{1}{2}(g_0\xi - g_0k_0)^T(g_0\xi - g_0k_0) = \frac{1}{2}\tilde{\xi}^T G\tilde{\xi} \quad (21)$$

$$\lambda_1\|\tilde{\xi}\|^2 \leq V(q,\xi) \leq \lambda_2\|\tilde{\xi}\|^2 \quad (22)$$

where $\tilde{\xi} = \xi - k_0$, $G = g_0^T g_0$, and λ_1, λ_2 are positive constants. A tracking controller is then obtained by solving the following CLF-QP:

$$\begin{aligned} u^\star &= \arg\min \|u\|^2 \\ \textbf{s.t } & (\dot{q} - g_0k_0)^T(\dot{g}_0(\xi - k_0) - g_0\dot{k}_0 + g_1u) \leq -\lambda V \end{aligned} \quad (23)$$

where $\dot{g}_0 = \sum_{i=1}^{n} \frac{\partial g_0}{\partial q_i}\dot{q}_i$ and $\lambda \in \mathbb{R}_{>0}$. The resulting acceleration command $u \in \mathbb{R}^2$ enforces exponential convergence of the system velocity ξ to the reference k_0 in the ideal case. In practice, however, modeling uncertainties generally prevent perfect tracking. To account for this, the residual tracking error is treated as a bounded disturbance $d := \xi - k_0(q)$, and is incorporated into the barrier function formulation. In particular, the closed-loop system is assumed to satisfy an Input-to-State Stable (ISS) condition of the form

$$\dot{V}(q,\xi) \leq -\lambda V(q,\xi) + \bar{\delta}, \quad (24)$$

for some bounded constant $\bar{\delta} > 0$. To preserve safety under this bounded tracking error, we construct the following ISSf barrier function and the corresponding safe set

$$h(q,\xi,\gamma) := h_0(q) - \frac{1}{\mu\lambda_1}\left(V(q,\xi) - \frac{\bar{\delta}}{\alpha}\right), \quad (25)$$

$$\mathcal{C}(\gamma) := \{(q,\xi) \in \mathcal{D}(\gamma) \mid h(q,\xi,\gamma) \geq 0\}. \quad (26)$$

3.3 Formal Guarantees and Complete Algorithm

This subsection establishes formal safety, feasibility, and liveness guarantees for the proposed multi-stage control architecture.

Theorem 2. Consider the full-order vehicle dynamics (2), the constraint set $\mathcal{C}_0(\gamma)$ defined in (11), and the differentiable controller $k_0 \in K(q,\gamma)$ given by (15). Let $V : \mathbb{R}^3 \times \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$ be a CLF with an associated tracking controller $u \in \mathbb{R}^2$ satisfying (22) and the ISS condition (24) for positive constants $\lambda, \lambda_1, \lambda_2$, μ, and $\bar{\delta} > 0$. If

$$\lambda \geq \alpha + \frac{\eta\mu}{4}, \quad (27)$$

then the set $\mathcal{C}(\gamma)$ defined in (26) is forward invariant for the closed-loop system (1) under the control law u obtained from the CLF-QP (23).

For complete proof, see Cohen et al. (2024).

Assembling all the components, we obtain the safety and liveness operator $\Phi : \mathcal{V} \times \mathcal{V}_p \to K \times \mathcal{V}_p^{safe}$, which maps the RL action u_{RL} to certified velocity commands $[k_0, v_d]$, subsequently tracked by the CLF-based controller generating the acceleration input u. A single outer-loop iteration is summarized in Algorithm 1.

Applying this safety block to any arbitrary policy π yields the main result.

Theorem 3. Given an arbitrary control policy $\pi : \mathcal{S} \to \mathcal{V} \times \mathcal{V}_p$, the operator $\Phi : \mathcal{V} \times \mathcal{V}_p \to K \times \mathcal{V}_p^{safe}$ defined by Algorithm 1 guarantees that the closed-loop system (1) under the modified policy $\Phi(\pi)$ satisfies path-dependent safety constraints, respects velocity bounds, and has the liveness property.

Algorithm 1 Policy Interaction with The Environment

Initialize: $k_d[0], v_p[0], s[t]$
Let: $N = f_f/f_{RL}$ ▷ Number of inner-loop steps
$(k_d^{RL}, v_p^{RL}) \leftarrow \pi(s[t])$, $v_d^{RL} \leftarrow \frac{v_p^{RL}}{\|p_{d\gamma}\|}$
for $n = 0$ **to** $N - 1$ **do** ▷ Run Inner-Loop
  $k_0[k] \leftarrow$ Compute Safe Velocity Control (15)
  $\bar{k}_d[n], \bar{v}_d[n] \leftarrow$ Compute Feasibility Filter (20)
  $u[n] \leftarrow$ Compute Acceleration using CLF (23)
  $q[n+1], \xi[n+1] \leftarrow$ Update System State (2)
  $\gamma[n+1] \leftarrow$ Update Path (3)
  $k_d[n+1], v_d[n+1] \leftarrow$ Update with Input Filter (9)
end for

Table 1. Framework Hyperparameters

Description	Symbol	Value
RL and Filter frequency	f_{RL}, f_f	50, 500 Hz
Path Domain	Γ	[0,1000]
Max/min linear velocity	$\pm v_{\max}$	$\pm$10 m/s
Max/min angular velocity	$\pm w_{\max}$	$\pm$2 rad/s
Max/min path velocity	$v_{p\,\min}, v_{p\,\max}$	0.5 , 12 m/s
Path Horizon and Offset	H, Δ_i	80, 0.2 ,
Velocity Threshold and Target	v_{th}, T_t	2.5 m/s, 60 s
Reward Gains	K_c, K_f, K_v, K_t	0.5, 0.5, 0.5, 100
Feasibility Cost Matrix	$\sigma_v, \sigma_\omega, \sigma_{v_d}$	0.1, 1.0, 0.001
CBF Parameters	$\alpha, \bar{\delta}, \eta$	4, 0.08, 15
CLF Parameters	λ, λ_1, μ	10,1,15
Input Filter Gains	a_v, a_w, a_{v_p}	10,10,20
Discount Factor	β	0.995
Network Architecture	π_θ, v_ϕ	MLP [256, 256]
Parallel Environments	N_{env}	64

Proof. From Theorem 2, the CLF-backstepped controller guarantees forward invariance of $\mathcal{C}(\gamma)$. The feasibility filter (20) ensures $k_0 \in \mathcal{V}$ for all $t \in T$. Finally, the feasibility of $v_d > 0$ follows from Proposition 1, implying task completion in finite time and hence liveness.

4. SIMULATION RESULTS

To demonstrate the capabilities of the proposed framework, we now evaluate the method in simulation. We consider a unicycle robot of scale 1 : 10, with a length of $L = 0.5$ m. For solving the MDP, we adopt Proximal Policy Optimization (PPO) (Schulman et al., 2017) as the DRL algorithm. All the hyperparameters are listed in Table 1.

For the environment, we use the F1:10 tracks Dataset from Betz et al. (2022). Each track is defined by control points specifying the centerline and track boundaries. The centerline is interpolated to obtain the parameterized reference path $p_d(\gamma)$. The track boundaries define the state constraints, which are encoded using the CBF:

$$h_0(q,\gamma) := \frac{1}{2} - \frac{1}{2}(q - p_d(\gamma(t)))^T Q(q - p_d(\gamma(t))), \quad (28)$$

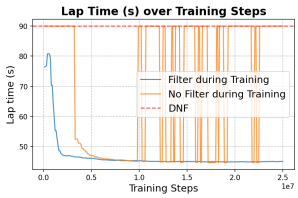
where the matrix $Q = \text{diag}(\lambda_h, \lambda_h, 0)$ parameterizes the zero super-levelset $\mathcal{C}_0$ as a circle of radius $r_0 = \frac{1}{\sqrt{\lambda_h}}$. By inflating the set with the factor of $\bar{\delta}$, the final set $\mathcal{C}(\gamma)$ has a radius of $r = 1$, corresponding to the track limits.

4.1 Training

We train all models exclusively on the Silverstone Circuit for 25 million environment steps. Two policies are considered: one trained with the proposed Filter Stage in the

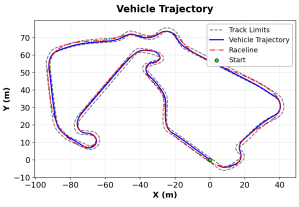


(a) Reward over training steps

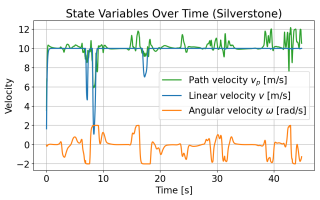


(b) Lap Time length over steps

Fig. 2. Training curves for the models with and without filter stage.



(a) Vehicle Trajectory.



(b) State Variables over Time.

Fig. 3. Vehicle trajectory and state variables.

loop, and a baseline *vanilla policy*, in which the RL outputs are directly passed through the input smoothing filter and the CLF tracking controller, without safety filtering. Any episode that violates safety constraints is terminated early, and all episodes restart from $\gamma = 0$. The training curves for reward and episode length can be visualized in Figure 2, where it is important to note that any lap-time equal to 90 seconds corresponds to an episode that Did Not Finish (DNF). The results show that the proposed framework significantly improves training stability and accelerates convergence, with cumulative reward and lap time improving almost monotonically. Conversely, while the unfiltered baseline achieved a 0.28-second faster lap time, its training was highly unstable, as it struggled to finish episodes after 10 million steps and failed catastrophically by 20 million. Furthermore, even a randomly initialized policy π equipped with the filter completes episodes in under 80 seconds. This shows that guaranteeing episode completion stabilizes training without significantly hampering performance, as evidenced by the minimal gap between lap times and rewards. We further note that, despite training on only a single track, our method generalized to all 23 tracks of the dataset with competitive lap times, whereas the vanilla DRL method finished only 2 tracks, with minimal time gains.

4.2 Trajectory Analysis

For a more thorough analysis of our proposed method, Fig. 3a illustrates the vehicle trajectory on the Silverstone Track, and Fig. 3b shows the time evolution of the state variables. One interesting aspect of the results is that the path velocity v_p is almost always equal to or higher than the vehicle velocity v. This is noticeable in high-speed corners, where the robot takes a tighter line, which in turn causes the path to increase its speed beyond $v_{\max}$ in order to keep up with the robot. This fact clearly shows that the taken path is more optimized than the nominal path, and was achieved purely by trial-and-error exploration, skipping expensive offline trajectory optimization.

5. CONCLUSION

This paper introduced a learning-based control framework for autonomous racing that integrates deep reinforcement learning with formal, model-based safety guarantees. The racing problem was formulated as a modified path-following task, using the geometric centerline of the track as a nominal reference while allowing safety-certified deviations that enable performance optimization. Safety and task completion were enforced through a multi-stage control architecture combining ISSf CBFs, a feasibility filter, and a CLF-based backstepping controller operating on the full-order vehicle dynamics. The results highlight that coupling learning-based control with formally grounded safety and feasibility mechanisms yields controllers that are not only performant but also keep the desired guarantees. Future work will include incorporating perception-based extensions for online track extraction and higher-fidelity vehicle models capturing tire and slip dynamics.

REFERENCES

Aguiar, A.P. (2017). Single and multiple motion control of autonomous robotic vehicles. In *2017 11th International Workshop on Robot Motion and Control (RoMoCo)*, 172–184.

Betz et al. (2022). Autonomous vehicles on the edge: A survey on autonomous vehicle racing. *IEEE Open Journal of Intelligent Transportation Systems*, 3, 458–488.

Brunke, L. et al. (2022). Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(Volume 5, 2022), 411–444.

Cohen, M.H., Molnar, T.G., and Ames, A.D. (2024). Safety-critical control for autonomous systems: Control barrier functions via reduced-order models. *Annual Reviews in Control*, 57, 100947.

Garrett, C.R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L.P., and Lozano-Pérez, T. (2021). Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(Volume 4, 2021), 265–293.

Hung, N. et al. (2023). A review of path following control strategies for autonomous robotic vehicles: Theory, simulations, and experiments. *Journal of Field Robotics*, 40(3), 747–779.

Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., and Scaramuzza, D. (2023). Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976), 982–987.

Kolathaya, S. and Ames, A.D. (2019). Input-to-state safety with control barrier functions. *IEEE Control Systems Letters*, 3(1), 108–113.

Reis, M.F., Jain, R.P., Aguiar, A.P., and de Sousa, J.B. (2019). Robust moving path following control for robotic vehicles: Theory and experiments. *IEEE Robotics and Automation Letters*, 4(4), 3192–3199.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. URL `https://arxiv.org/abs/1707.06347`.

Wang, Y., Liu, J., Yuan, X., and Yang, J. (2025). Safe reinforcement learning for competitive autonomous racing: Integrated state–action mapping and exploration guidance framework. *Actuators*, 14(7).