

Learning to Tune Pure Pursuit in Autonomous Racing: Joint Lookahead and Steering-Gain Control with PPO

Mohamed Elgouhary and Amr S. El-Wakeel

Abstract—Pure Pursuit (PP) is widely used in autonomous racing for real-time path tracking due to its efficiency and geometric clarity, yet performance is highly sensitive to how key parameters—lookahead distance and steering gain—are chosen. Standard velocity-based schedules adjust these only approximately and often fail to transfer across tracks and speed profiles. We propose a reinforcement-learning (RL) approach that *jointly* chooses the lookahead L_d and a steering gain g online using Proximal Policy Optimization (PPO). The policy observes compact state features (speed and curvature taps) and outputs (L_d, g) at each control step. Trained in F1TENTH Gym and deployed in a ROS 2 stack, the policy drives PP directly (with light smoothing) and requires no per-map retuning. Across simulation and real-car tests, the proposed RL-PP controller that jointly selects (L_d, g) consistently outperforms fixed-lookahead PP, velocity-scheduled adaptive PP, and an RL lookahead-only variant, and it also exceeds a kinematic MPC raceline tracker under our evaluated settings in lap time, path-tracking accuracy, and steering smoothness, demonstrating that policy-guided parameter tuning can reliably improve classical geometry-based control.

Index Terms—Autonomous racing; Path tracking; Pure Pursuit; Reinforcement learning; Proximal Policy Optimization (PPO); Dynamic lookahead; Steering gain; Sim-to-real transfer; ROS2; F1TENTH

I. INTRODUCTION

Intelligent and connected vehicles must track reference paths accurately across widely varying speeds and curvatures while preserving stability and comfort. Geometric controllers remain attractive in this setting for their transparency and low computational cost, with Pure Pursuit (PP) [1] directing the vehicle toward a lookahead point and converting geometry into a steering command suitable for real-time use.

A longstanding limitation of PP is its sensitivity to important parameters—most notably the lookahead distance and the effective steering gain. Small lookahead values yield agile cornering but can excite oscillations on straights; large values smooth the motion yet understeer in tight bends. Likewise, an overly aggressive steering gain can induce noise, whereas a conservative gain slows convergence. Rule-based schedules that scale lookahead (and sometimes gain) with velocity or curvature partially address these effects, but their fixed functional forms and tuned coefficients often fail to transfer across tracks, speed profiles, and platforms. Model-predictive control (MPC) can achieve strong performance, but typically requires more

The authors are with the Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV, USA ({mae00018@mix.wvu.edu, amr.elwakeel@mail.wvu.edu}).

modeling effort and online optimization. Instead of replacing PP, we keep its geometric steering law and learn to tune a small set of parameters to improve performance while preserving simplicity and interpretability. We propose a data-driven alternative in which a reinforcement learning (RL) policy *jointly* sequences the lookahead distance L_d and a steering gain g online. We formulate parameter sequencing as a continuous control problem and train with Proximal Policy Optimization (PPO) [2] to map compact state features—vehicle speed and curvature taps at near/mid/far horizons—into a 2-D action (L_d, g) . The policy is trained in the F1TENTH Gym [3] using a minimum-curvature raceline [4] to supply waypoints and curvature; optimization is stabilized through KL-constrained PPO updates, observation/return normalization, gradient clipping, and a scheduled learning rate chosen via Optuna [5]. In simulation, we increase task difficulty by uniformly scaling the raceline speed profile so that the policy must learn a curvature- and speed-aware schedule rather than overfitting to a single fixed speed profile.

The learned policy is integrated into a ROS 2 control stack that preserves PP’s geometric steering law and exposes two lightweight interfaces (topics) for (L_d, g) . A first-order smoother mitigates sudden action changes, and a safety “teacher” (linear $v \mapsto L_d$ and $v \mapsto g$) serves as a fallback if RL commands become stale.

To focus on the effect of learned parameter tuning while preserving PP’s interpretability and runtime efficiency, We evaluate against multiple Pure Pursuit (PP) baselines—including fixed-lookahead and velocity-scheduled variants—as well as an MPC raceline tracker. We evaluate in simulation (zero-shot across unseen tracks) and on the physical F1TENTH platform at speeds limited by safety and track size, and we monitor PPO diagnostics (approximate KL, clip fraction, action standard deviation, and value loss as illustrated in Figs. 2 and 3) to verify stable optimization consistent with the chosen learning-rate settings.

The main contributions of this paper are:

- An RL-Pure Pursuit framework that adapts the lookahead distance L_d and steering gain g online while preserving PP’s simplicity and interpretability.
- A practical training recipe (reward design, hyperparameter tuning, and stability techniques) that learns a curvature-speed-aware (L_d, g) policy with smooth, accurate tracking.
- A simulation and real-car evaluation in a ROS 2 F1TENTH stack, including ablations and comparisons to

fixed PP, velocity-scheduled PP, and a kinematic MPC tracker, showing improved lap time, lateral error, and steering smoothness.

- An interpretability study that visualizes how L_d and g vary with track geometry and speed along the raceline.

This work shows how deep reinforcement learning can complement geometric control to improve adaptability and robustness while preserving interpretable, real-time path tracking.

II. RELATED WORK

Path tracking is a core problem in autonomous driving, addressed by classical methods [6] and learning-based approaches [7]. Among geometric controllers, Pure Pursuit (PP) [1] and Stanley [8] remain popular due to their simplicity and real-time performance, including in racing settings [9]. A central limitation of PP is its sensitivity to the lookahead distance, which trades responsiveness against smoothness. Consequently, many works propose adaptive lookahead schemes based on speed and/or curvature [1], [10], but still rely on hand-designed functional forms and tuned coefficients.

Model predictive control (MPC) optimizes over a finite horizon with dynamics and constraints and is widely used as a strong baseline in autonomous racing. While MPC can achieve precise tracking and constraint handling, it typically requires accurate modeling, online optimization, and careful tuning of costs and constraints, which may be challenging on resource-limited platforms.

Reinforcement learning (RL) provides a data-driven alternative for context-dependent control [11], with methods such as DQN [12], DDPG [13], and PPO [2] applied across robotics and autonomous driving. End-to-end policies [14] can be effective but often face interpretability and transfer challenges [15]–[17]. To balance adaptability with structure, hybrid approaches use RL to tune parameters of classical controllers rather than replace them outright [18]–[20]. However, most prior work focuses on throttle or full-actuation policies, with less emphasis on learning to tune the core geometric path-following parameters in PP. While RL-driven lookahead adaptation has been explored in other domains (e.g., heavy trucks) [21], we explicitly benchmark against an *adaptive PP* baseline (a velocity-linear Pure Pursuit schedule) and include an MPC raceline tracker as a stronger model-based reference in both simulation and hardware experiments.

Finally, we use minimum-curvature racelines from TUM’s `global_racetrjectory_optimization` [4] as the shared reference for all controllers.

Prior adaptive PP methods adjust L_d with hand-designed rules (speed/curvature/preview), while RL-based tuning has been used to adapt gains of classical controllers or to learn end-to-end tracking. Our contribution differs in that we (i) keep the PP steering law fixed and interpretable, (ii) learn a *joint* continuous schedule for both L_d and a gain g using compact curvature-preview features, and (iii) evaluate zero-shot generalization to unseen tracks and deploy within a ROS 2 stack on a real F1TENTH vehicle.

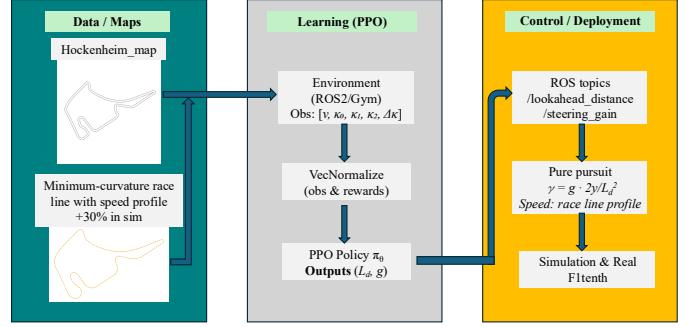


Fig. 1. Pipeline: mapping and minimum-curvature raceline feed the simulator and a baseline PP controller. A PPO agent learns (L_d, g) and integrates with PP via ROS topics. We validate in simulation and deploy on a real F1TENTH car.

III. FRAMEWORK AND METHODOLOGY

This section details the full stack used in our study. We estimate pose with a LiDAR-based Monte Carlo localization (MCL) filter, generate a global reference using minimum-curvature optimization [4], and couple a classical Pure Pursuit (PP) tracker with a learned policy that selects the lookahead distance and the steering gain online. We then specify observation and action spaces, the reward, PPO training, and how these components integrate in simulation and on the real F1TENTH platform [3] for fair comparison against multiple PP baselines (fixed-lookahead and velocity-scheduled variants) and an MPC raceline tracker.

From a controller-design perspective, our goal is to investigate how far a lightweight geometric Pure Pursuit approach can be pushed by exposing only a small set of parameters to learning. We therefore build our stack around PP and evaluate against standard tracking baselines under a common reference trajectory and localization stack.

A. Localization and Global Reference

We estimate pose using a standard LiDAR-based Monte Carlo localization (MCL) particle filter [22] over a prebuilt occupancy grid. Particles are propagated with a kinematic motion model, reweighted by a beam-based scan likelihood using grid ray-casting, and resampled when the effective sample size drops; the state estimate is taken as the weighted mean (circular mean for heading).

For the global reference, we use TUM’s `global_racetrjectory_optimization` [4] to compute a smooth minimum-curvature raceline within track boundaries. The optimizer returns a closed-loop trajectory with associated curvature $\kappa(s)$ and a friction-limited speed profile $v_{\max}(s)$, which we export as waypoints $\{x(s), y(s), \kappa(s), v_{\max}(s)\}$. This globally smooth geometry provides both the tracking reference and curvature features used by the controller and the RL policy.

B. System Overview and Experimental Setup

Fig. 1 summarizes the components. Using the F1tenth *Hockenheim* map, we load a raceline with curvature and a reference

speed profile. The raceline drives both the **Environment** and a baseline **PP** controller. Rollouts train a **PPO** agent that outputs (L_d, g) each step, which are lightly smoothed and published on ROS topics; **PP** then computes steering and speed commands.

C. Observation and Action Spaces

At time t the agent observes

$$s_t = \begin{bmatrix} v_t \\ \kappa_{0,t} \\ \kappa_{1,t} \\ \kappa_{2,t} \\ \Delta\kappa_t \end{bmatrix}, \quad (1)$$

where v_t is speed, $\kappa_{0,1,2}$ are absolute curvatures at near/mid/far horizon taps along the raceline, and $\Delta\kappa_t = \kappa_{1,t} - \kappa_{0,t}$. The action is a 2-D continuous vector,

$$a_t = \begin{bmatrix} L_{t+1} \\ g_{t+1} \end{bmatrix}, \quad L_{t+1} \in [0.35, 4.0] \text{ m}, \quad g_{t+1} \in [0.45, 1.15]. \quad (2)$$

followed by a first-order exponential smoother (per-component) before publishing:

$$\begin{aligned} \tilde{L}_{t+1} &= \beta_L L_{t+1} + (1 - \beta_L) \tilde{L}_t, \\ \tilde{g}_{t+1} &= \beta_g g_{t+1} + (1 - \beta_g) \tilde{g}_t. \end{aligned} \quad (3)$$

with $\beta_L = \beta_g = 0.2$. Conceptually, the policy implements a mapping $s_t \mapsto (L_{t+1}, g_{t+1})$ that uses current speed and previewed curvature to adjust the **PP** lookahead distance and steering gain before they are smoothed and applied.

Curvature preview taps. Let the raceline be discretized into N waypoints $\{p_i = (x_i, y_i)\}_{i=0}^{N-1}$ with associated curvature samples $\{\kappa_i\}_{i=0}^{N-1}$. At time t , we find the nearest waypoint index $i_t^* = \arg \min_i \|p_t - p_i\|_2$, where p_t is the current vehicle position. We then form three fixed preview taps using waypoint-index offsets $k \in \{0, 5, 12\}$ (with wrap-around modulo N): $\mathcal{I}_t(k) = (i_t^* + k) \bmod N$. The curvature features are the absolute curvatures at these taps, $\kappa_{j,t} = |\kappa_{\mathcal{I}_t(k_j)}|$ for $k_0=0$, $k_1=5$, $k_2=12$, and $\Delta\kappa_t = \kappa_{1,t} - \kappa_{0,t}$. With our waypoint spacing, $k = 5-12$ corresponds to approximately 1–4 m of lookahead.

D. Reward Design

The reward balances speed, agreement with teacher targets, smoothness, curvature exposure, straight behavior, progress, and safety:

$$\begin{aligned} R_t = w_v v_t - w_L | &\tilde{L}_t - L_t^* | - w_G | \tilde{g}_t - g_t^* | - w_{jL} | \tilde{L}_t - \tilde{L}_{t-1} | \\ &- w_{jG} | \tilde{g}_t - \tilde{g}_{t-1} | - w_k | \kappa_t | - w_x (\tilde{L}_t \kappa_t^{\max}) \\ &+ w_{\text{pre}} \mathbb{I}_{\text{bend}}(\kappa_t^{\max}) \mathbb{I} [\tilde{L}_t \leq \ell(v_t)] \\ &- w_c \mathbb{I}_{\text{collision}} - w_s \mathbb{I}_{\text{slow}} + w_p \Delta p_t. \end{aligned} \quad (4)$$

where $\kappa_t^{\max} = \max(\kappa_{0,t}, \kappa_{1,t}, \kappa_{2,t})$, κ_t is a smoothed local curvature, and Δp_t counts newly passed waypoints. The teacher targets are

$$\begin{aligned} L_t^* &= \text{clip}(0.50 + 0.28 v_t - 3.5 \kappa_t^{\max}, 0.35, 4.0), \\ g_t^* &= \text{clip}(m v_t + b, 0.45, 1.15). \end{aligned} \quad (5)$$

TABLE I
REWARD WEIGHTS USED FOR PPO TRAINING.

Term	Weight
w_v	1.8
w_L	3.0
w_G	0
w_{jL}	0.4
w_{jG}	0
w_k	1.5
w_x	2.0
w_{pre}	1.5
w_c	10.0
w_s	0.5
w_p	1.0

with $m = (g_{\min} - g_{\max}) / (v_{\max} - v_{\min})$ and $b = g_{\max} - m v_{\min}$ (we use $v_{\min}=3$, $v_{\max}=18$, $g_{\max}=0.9$, $g_{\min}=0.65$). We give a small bonus (w_{pre}) for *pre-shortening* lookahead before pronounced bends, and gently discourage overly large g on straights. Collisions are detected if the minimum LiDAR range < 0.2 m; we also penalize low speed and stalling. Rewards are clipped $R_t \in [-30, 100]$ for stability.

Reward weights. Table I lists the weight values used for all PPO training runs.

E. Pure Pursuit Control

Given the target point at a distance L_d along the raceline, transformed to the vehicle frame (x', y') , **PP** computes

$$\gamma = g \cdot \frac{2y'}{L_d^2}, \quad \gamma \in [-0.35, 0.35], \quad (6)$$

and publishes the steering command (we use γ directly as the steering angle command in our setup). The speed command follows the raceline's reference profile (capped on the real car for safety). If RL commands are stale beyond a short timeout, **PP** reverts to a linear teacher for (L_d, g) until fresh actions resume. Here L_d controls the lookahead distance and therefore the trade-off between responsiveness and smoothness, while g scales the curvature implied by the lookahead geometry and sets how aggressively the vehicle turns toward the target point.

Why L_d and g are non-redundant.: L_d determines the target point (preview geometry), while g scales the resulting curvature command after the target is fixed. Hence g acts as a pure multiplier, whereas L_d also moves the target point: $\partial\gamma/\partial g = 2y'/L_d^2$, whereas $\frac{d\gamma}{dL_d}$ contains both an explicit term $-4gy'/L_d^3$ and an implicit term $(2g/L_d^2) dy'/dL_d$ due to the L_d -dependent target point (x', y') . Ablations confirm the benefit of jointly adapting (L_d, g) .

F. Vehicle Dynamics Model

We model the platform with the standard kinematic bicycle dynamics:

$$\dot{x} = v \cos \theta, \quad \dot{y} = v \sin \theta, \quad (7)$$

$$\dot{\theta} = \frac{v}{L} \tan \delta, \quad \dot{v} = a, \quad (8)$$

where (x, y) denote the position, θ the heading, v the longitudinal speed, a the commanded acceleration, δ the steering

input, and L the wheelbase. Under small-slip conditions, this simplified, nonholonomic model provides an accurate description for our high-rate path-tracking tasks in simulation and is sufficiently faithful for transfer to the real F1TENTH platform.

G. Proximal Policy Optimization (PPO)

We train the stochastic policy $\pi_\theta(a_t \mid s_t)$ —which outputs the 2-D action (L_{t+1}, g_{t+1}) —using Proximal Policy Optimization (PPO) from Stable-Baselines3 [23]. PPO is a first-order policy-gradient method that constrains each update via a clipped surrogate objective, which helps prevent large, destabilizing policy shifts between iterations.

Our training setup uses on-policy rollouts of length $n_{\text{steps}} = 4096$ with a minibatch size of 256 and $n_{\text{epochs}} = 5$ optimization passes per update. We adopt $\gamma = 0.99$ for discounting and Generalized Advantage Estimation with $\lambda = 0.98$. The probability ratio is clipped with $\epsilon = 0.2$, and we monitor a target KL of 0.015 to guard against overly aggressive updates. We include an entropy bonus (coefficient 0.02) to encourage exploration and weight the value-function loss with 0.6. Gradients are clipped at a global norm of 0.7. The default learning rate follows a linear decay $\ell(f) = \ell_0 f$ with $\ell_0 = 2.4 \times 10^{-4}$.

We also evaluate a cosine schedule (enabled via `--lr-schedule cosine`), implemented as

$$\ell_{\text{cos}}(f) = \frac{\ell_0}{2} (1 + \cos(\pi(1 - f))),$$

where $f \in [1, 0]$ is the remaining training progress provided by SB3.

Observations and returns are normalized online with **VecNormalize** for stability.

PPO maximizes the clipped surrogate

$$\begin{aligned} \mathcal{L}_{\text{clip}}(\theta) = & \\ \mathbb{E}_t \left[\min(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t) \right] & \end{aligned} \quad (9)$$

where $r_t(\theta) = \pi_\theta(a_t \mid s_t) / \pi_{\theta_{\text{old}}}(a_t \mid s_t)$ and \hat{A}_t denotes the advantage. The full objective combines policy, value, and entropy terms:

$$\begin{aligned} \mathcal{L}_{\text{ppo}}(\theta, \phi) = & \\ -\mathcal{L}_{\text{clip}}(\theta) + c_v \mathbb{E}_t[(V_\phi(s_t) - \hat{R}_t)^2] & \\ -c_s \mathbb{E}_t[\mathcal{H}(\pi_\theta(\cdot \mid s_t))] & \end{aligned} \quad (10)$$

with $c_v = 0.6$ and $c_s = 0.02$ in our implementation. During training we evaluate the policy periodically on a separate evaluation environment that shares (but does not update) the normalization statistics, and we save both best-model checkpoints and intermediate snapshots.

H. Implementation and Training

Our pipeline is implemented in ROS2 using `rclpy` and trained with Stable-Baselines3 PPO. The policy outputs a two-dimensional action (L_{t+1}, g_{t+1}) at each control step; both components are lightly filtered and published on ROS topics consumed by the Pure Pursuit (PP) controller. PP

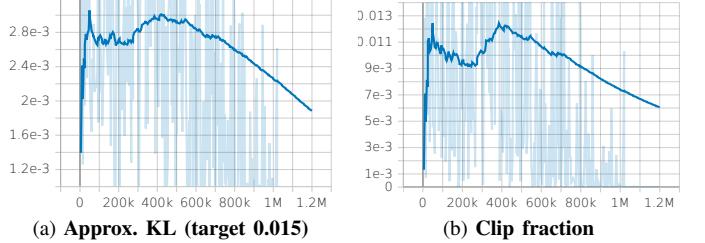


Fig. 2. PPO diagnostics (1/2): update stability metrics. Approx. KL and clip fraction remain small after the initial transient, indicating conservative policy updates.

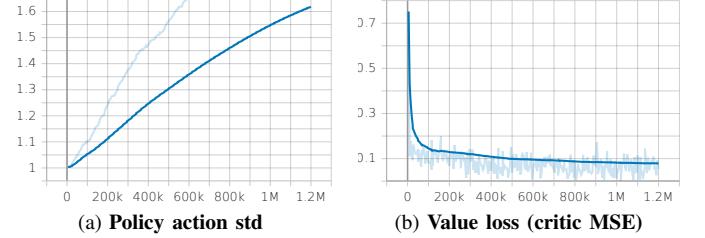


Fig. 3. PPO diagnostics (2/2): exploration and critic learning. The policy action standard deviation decreases as exploration reduces, while the critic loss stabilizes as value estimates improve.

then computes steering directly from the geometric relation $\gamma = g \cdot 2y/L^2$ and applies a raceline speed profile (safely capped on hardware). If RL messages become stale, PP falls back to a linear “teacher” rule for (L, g) until fresh actions arrive.

Runs comprise up to **1.2M** environment steps. We log to **TensorBoard** and evaluate every **5,000** steps on a separate evaluation environment that *shares* (but does not update) the normalization statistics. The best-performing snapshot is saved automatically, and additional checkpoints are written every **25,000** steps to enable recovery and ablation comparisons.

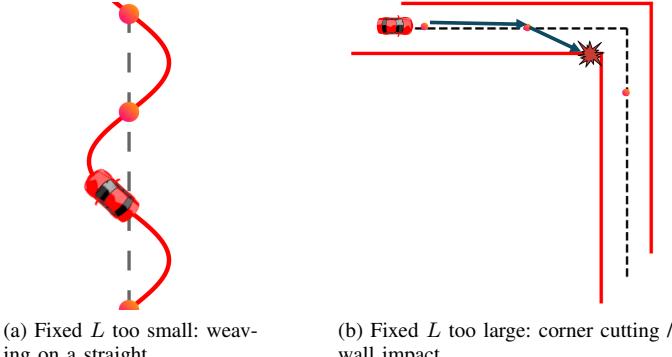
As training proceeds (Figs. 2, 3), the approximate KL and clip fraction remain small after an initial spike and trend downward later in training, the policy’s action standard deviation increases gradually, and the critic’s value loss decreases and stabilizes—together indicating conservative updates with a stabilizing critic.

The full loop executes entirely in simulation (F1TENTH Gym + ROS2), and we repeat runs to check consistency and generalization. The agent emits (L_d, g) , which are relayed to PP. PP issues steering/speed commands, and the environment returns the next state and reward. This closed-loop design allows the policy to learn curvature- and speed-aware adjustments for *both* lookahead and steering gain, and it transfers readily to the real vehicle via the same ROS interfaces.

All experiments were executed on a workstation-class desktop (Dell Precision 3660, Intel Core i9–13900, 32 GiB RAM) running Ubuntu 22.04.5 LTS (64-bit, X11). We observed typical wall-clock training times on the order of a few hours for runs of ~ 1.2 M environment steps.

IV. EXPERIMENTAL RESULTS

We evaluate in both simulation and on a physical F1TENTH vehicle. Across all experiments, we compare five controllers:



(a) Fixed L too small: weaving on a straight.
 (b) Fixed L too large: corner cutting / wall impact.

Fig. 4. Classical Pure Pursuit failure modes with fixed lookahead.

(i) Fixed PP, (ii) Adaptive PP (linear $v \mapsto L_d$), (iii) RL-PP (L_d only), (iv) RL-PP (joint (L_d, g)), and (v) a kinematic MPC raceline tracker. All methods track the same minimum-curvature raceline and share the same localization and interfaces for fair comparison.

A. Baseline: Kinematic MPC

Kinematic MPC raceline tracker (baseline).: As a stronger model-based reference, we implement a kinematic MPC waypoint tracker that follows the same minimum-curvature raceline used by PP. The MPC state is $\mathbf{x} = [x, y, v, \psi]^\top$ and the control is $\mathbf{u} = [a, \delta]^\top$. We use a horizon of $T_K = 8$ with timestep $\Delta t = 0.1$ s (0.8 s lookahead). The horizon reference $\{\mathbf{x}_t^{\text{ref}}\}_{t=0}^{T_K}$ is generated by advancing along the raceline proportional to the current speed.

At each control step, we solve a QP that penalizes tracking error, control effort, and control-rate changes:

$$\begin{aligned} \min_{\mathbf{x}_{0:T_K}, \mathbf{u}_{0:T_K-1}} & \sum_{t=0}^{T_K-1} \|\mathbf{x}_t - \mathbf{x}_t^{\text{ref}}\|_Q^2 + \|\mathbf{x}_{T_K} - \mathbf{x}_{T_K}^{\text{ref}}\|_{Q_f}^2 \\ & + \sum_{t=0}^{T_K-1} \|\mathbf{u}_t\|_R^2 + \sum_{t=0}^{T_K-2} \|\mathbf{u}_{t+1} - \mathbf{u}_t\|_{R_\Delta}^2. \end{aligned} \quad (11)$$

with $Q = Q_f = \text{diag}(13.5, 13.5, 5.5, 13.0)$ for (x, y, v, ψ) , $R = \text{diag}(0.01, 5.0)$ for (a, δ) , and $R_\Delta = \text{diag}(0.01, 5.0)$.

We impose actuator and rate limits consistent with our platform:

$$\begin{aligned} \delta &\in [-0.4189, 0.4189] \text{ rad}, \quad |a| \leq 3.0 \text{ m/s}^2, \\ |\dot{\delta}_{t+1} - \dot{\delta}_t| &\leq \dot{\delta}_{\max} \Delta t, \quad \dot{\delta}_{\max} = 180^\circ/\text{s}. \end{aligned} \quad (12)$$

The QP is solved online using CVXPY with OSQP.

Fixed-Lookahead Failure Modes.: Classical Pure Pursuit (PP) with a *fixed* lookahead L is highly sensitive to the track geometry and the chosen speed profile. If L is set too small, the vehicle tends to oscillate on straights and cut inside through bends; if L is set too large, straight-line tracking appears stable, but the car understeers and struggles in tighter corners, as shown in Fig. 4. In practice, deploying PP on a new track often requires re-tuning L , and performance can remain sensitive despite this effort. In our setting, these limitations motivate learning to adapt not only L but also the *steering gain* g online.

Challenges: Although our hardware experiments reach $v_{\max} = 6$ m/s, they still operate below the most aggressive dynamics explored in simulation (via speed-profile scaling). At higher speeds, tire slip and unmodeled nonlinearities become more prominent; thus we use the high-speed simulation stress tests to complement the real-car validation while maintaining safety on the physical track.

Setup and zero-shot evaluation.: We train a PPO policy that outputs **both** (L, g) entirely in simulation on the *Hockenheim* F1TENTH racetrack, then evaluate *without any fine-tuning* on two unseen layouts: *Montreal* and *Yas Marina* from the `f1tenths_racetracks` collection [24]. To increase task difficulty and encourage generalization, we *simulate* training with the raceline speed profile uniformly scaled by **+30%**, which yields a maximum reference speed of **15.6** m/s and a minimum of **4.16** m/s.

Track selection. Training uses *Hockenheim*; *Montreal* and *Yas Marina* are held out for zero-shot tests (Fig. 5).

Safety-teacher activation rate.: To confirm that results are driven by the learned policy rather than the linear fallback, we log whether the controller uses PPO outputs (*RL mode*) or the linear teacher (*teacher mode*) at each control step (teacher mode triggers only if RL outputs are unavailable/stale beyond the timeout). Across our evaluation runs, the teacher was never triggered on either track: Montreal 0/8261 steps (0.000%), 0 events; Yas Marina 0/11228 steps (0.000%), 0 events (max continuous teacher duration 0.000 s).

Qualitative robustness.: Fig. 6 compares representative challenging regions on *Montreal* and *Yas Marina*. For each track, the full-track plot marks the area enlarged below, where we overlay the reference raceline and all five controllers. Across both tracks, the learned RL-PP policies follow the reference closely through corner entry, mid-corner, and exit while staying within the track boundaries; the joint (L_d, g) variant is typically the closest match, while the L_d -only policy shows slightly larger offsets in the highest-curvature parts of the turn. In contrast, fixed-lookahead PP and the velocity-linear adaptive PP exhibit larger geometry-dependent deviations in the highlighted regions (e.g., earlier inside cutting or larger exit drift), consistent with their reliance on a single hand-designed scheduling rule. The kinematic MPC tracker follows the same reference and provides a model-based comparison; under our horizon and constraints it shows modest rounding/offset in the highlighted regions. Overall, these qualitative observations align with the lap-time results: adapting *both* lookahead and steering gain improves robustness relative to fixed-parameter and single-rule baselines.

B. Ablation Study: Benefit of Joint (L_d, g) Tuning

A key design choice in our framework is to adapt *both* the Pure Pursuit lookahead distance L_d and the steering gain g . To isolate the value of learning g in addition to L_d , we compare two learned variants trained under the same PPO setup and observation space:

- **RL-PP (joint)**: the policy outputs (L_d, g) .
- **RL-PP (L_d only)**: the policy outputs L_d while g is held fixed at a constant $g = g_0$.

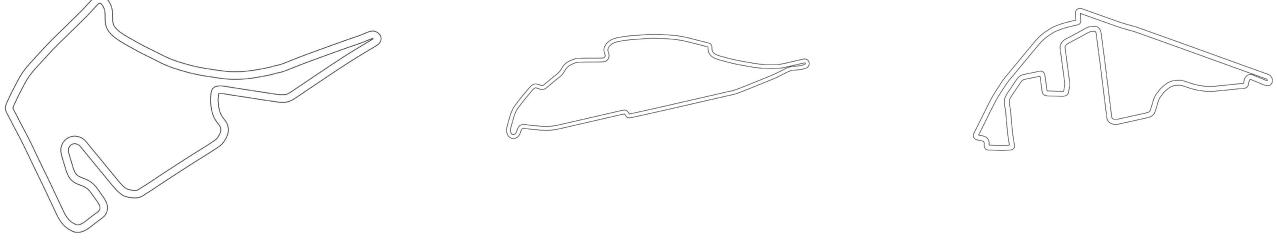


Fig. 5. Simulator maps used in our study: (a) Hockenheim for training, (b) Montreal, and (c) YasMarina for evaluation.

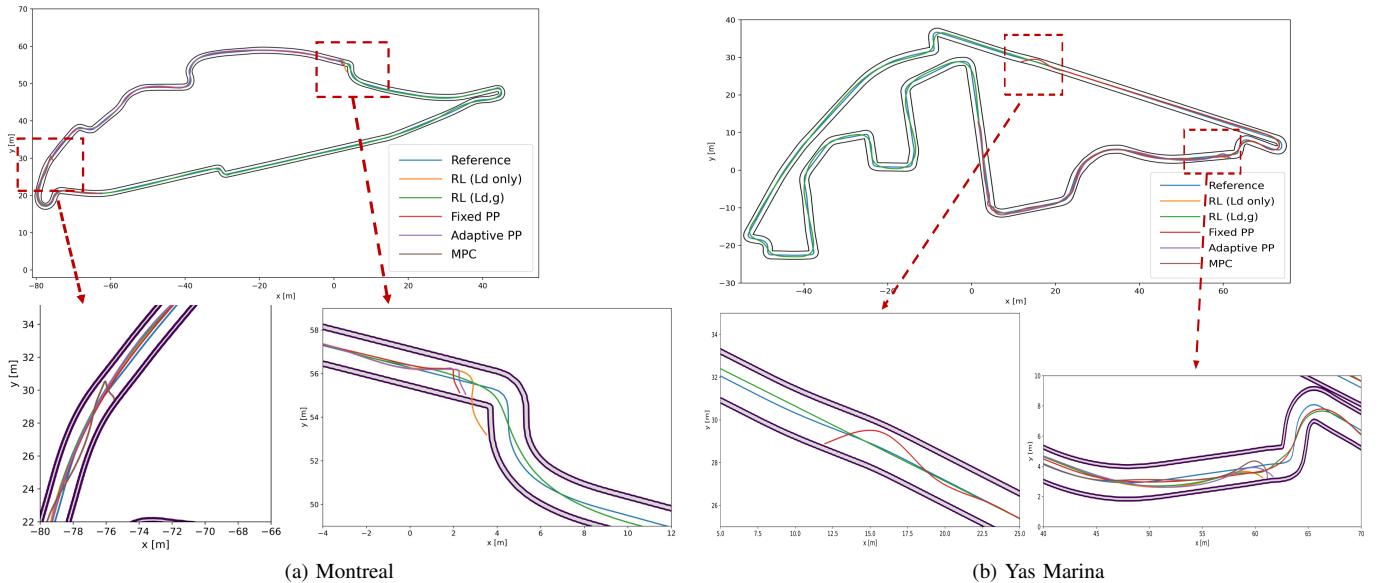


Fig. 6. Qualitative comparison in highlighted regions on two unseen tracks. For each track, the full-layout plot (top) indicates the zoom window shown below. The zoom overlays the reference raceline and all five controllers. The learned RL-PP policies follow the reference closely while remaining within the track boundaries, whereas Fixed PP and Adaptive PP (linear $v \mapsto L_d$) show larger deviations in these regions. The kinematic MPC tracker provides a model-based comparison under the same reference and constraints. Colors follow the legend.

Choice of g_0 . We set g_0 to the best-performing fixed gain selected by validation in simulation (i.e., a short sweep over a small grid of constant g values, evaluated under the same completion criterion used in our main results). This ensures the L_d -only ablation is not artificially weakened by a poor fixed gain.

Both variants are trained on *Hockenheim* and evaluated zero-shot on *Montreal* and *Yas Marina* under the same protocol as the main results. The results are reported in Tables II and III (rows **RL-PP (joint)** and **RL-PP (L_d only)**), and in the real-car evaluation in Table IV.

Overall, the joint policy achieves the best combination of lap time and stability, indicating that learning g in addition to L_d provides measurable benefit beyond adaptive lookahead alone.

Policy interpretability.: To understand what the learned policy encodes, we log (L_d, g) along the raceline and correlate them with curvature and speed. Empirically, the policy reduces L_d in high-curvature segments (tighter preview) and increases L_d on straights. The gain g varies more mildly, remaining near a narrow range while adjusting slightly with context. We provide plots of (L_d, g) versus κ_t^{\max} and v_t in Fig. 8(a), and show the along-lap evolution of (L_d, g) in Fig. 8(b).

TABLE II

MONTRÉAL F1TENTH RACETRACK: LAP TIMES OVER 10 CONSECUTIVE LAPS. FOR THE 10/10 COMPLETION OBJECTIVE, EACH CONTROLLER USES THE BEST SPEED MULTIPLIER THAT YIELDS 10/10 LAPS IN OUR SETUP.

Adaptive PP (linear) APPLIES A LINEAR $v \mapsto L_d$ RULE WITH BOUNDS $L_d \in [1.0, 2.5]$ m.

Controller	Mean	Std	Min	Max
RL-PP (joint L_d, g , +16%)	32.85	0.24	32.18	32.88
RL-PP (L_d only, +13%)	33.23	0.13	33.05	33.49
Adaptive PP (linear $v \mapsto L_d$, +10%)	34.25	0.04	34.18	34.34
Fixed PP (L_d fixed, -10%)	41.35	0.09	41.24	41.57
MPC raceline tracker (-15%)	48.06	0.08	47.96	48.25

Montreal F1TENTH racetrack: quantitative results (10 laps).: We evaluate five controllers on *Montreal* over 10 consecutive laps. To compare methods under a common completion criterion, each controller uses the best speed-profile multiplier that achieves 10/10 completed laps in our setup: RL-PP (joint (L_d, g)) at +16%, RL-PP (L_d only) at +13%, Adaptive PP (linear $v \mapsto L_d$) at +10%, Fixed PP at -10%, and the MPC raceline tracker at -15%. The resulting lap-time statistics are summarized in Table II.

From Table II, RL-PP (joint (L_d, g)) achieves a mean lap time of 32.85 ± 0.24 s over ten laps at the +16% profile (range

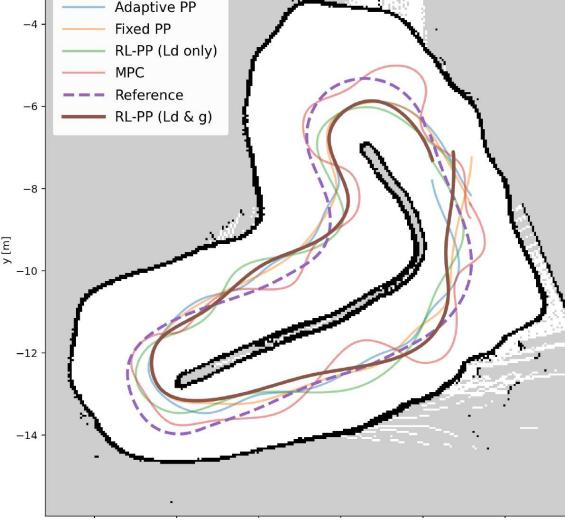


Fig. 7. Real-car tracking comparison using a shared raceline speed profile capped at $v_{\max} = 6$ m/s: overlaid trajectories for all five controllers against the same reference raceline. Using the same capped profile emphasizes differences in lateral tracking and stability while keeping longitudinal commands comparable across controllers.

TABLE III

YAS MARINA F1TENTH RACETRACK: LAP TIMES OVER 10 CONSECUTIVE LAPS. FOR THE 10/10 COMPLETION OBJECTIVE, EACH CONTROLLER USES THE BEST SPEED MULTIPLIER THAT YIELDS 10/10 LAPS IN OUR SETUP.

Adaptive PP (linear) APPLIES A LINEAR $v \mapsto L_d$ RULE WITH BOUNDS $L_d \in [1.0, 2.5]$ m.

Controller	Mean	Std	Min	Max
RL-PP (joint L_d, g , +17.2%)	45.10	0.07	44.96	45.20
RL-PP (L_d only, +15%)	46.04	0.10	45.93	46.21
Adaptive PP (linear $v \mapsto L_d$, [+15%])	46.41	0.12	46.24	46.57
Fixed PP (L_d fixed, orig.)	52.97	0.21	52.79	53.44
MPC raceline tracker ([−10%])	64.01	0.25	63.60	64.51

32.18–32.88 s). The RL-PP (L_d only) variant at +13% records 33.23 ± 0.13 s (range 33.05–33.49 s), and the Adaptive PP (linear $v \mapsto L_d$) baseline at +10% achieves 34.25 ± 0.04 s (range 34.18–34.34 s). Fixed PP requires a reduced profile (−10%) to complete all laps and yields 41.35 ± 0.09 s. The MPC raceline tracker attains 10/10 completion at −15% with a mean time of 48.06 ± 0.08 s. Overall, the joint RL-PP variant provides the fastest times under our completion criterion, improving over the linear adaptive baseline while maintaining low lap-to-lap variability.

Yas Marina F1TENTH racetrack: quantitative results (10 laps).: We apply the same protocol as in Montreal: each controller uses the best speed-profile multiplier that achieves 10/10 completed laps in our setup. On *Yas Marina*, RL-PP (joint (L_d, g)) runs at +17.2%, RL-PP (L_d only) at +15%, Adaptive PP (linear $v \mapsto L_d$) at +15%, Fixed PP uses the original (non-accelerated) profile, and the MPC raceline tracker completes 10/10 laps at −10%. The lap-time statistics (seconds) are reported in Table III.

Across ten laps, the RL policy again leads, with a lower mean time and comparably small dispersion relative to the linear dynamic baseline, and a substantial margin over fixed- L .

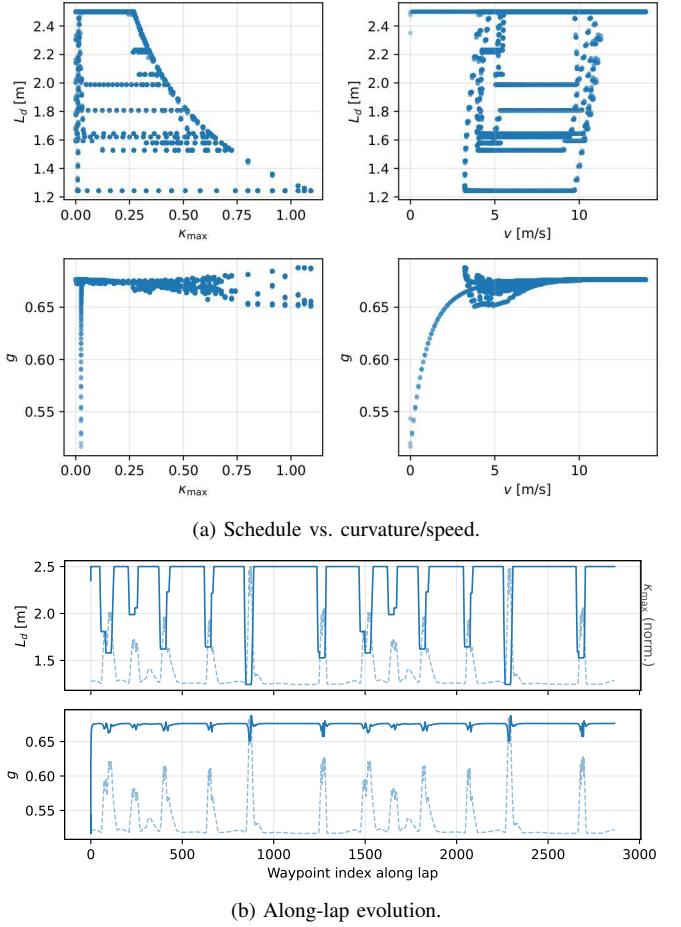


Fig. 8. Policy interpretability on Montreal. (a) Parameter schedule versus curvature and speed along the raceline. (b) Along-lap evolution of L_d and g plotted against waypoint index, with normalized κ_{\max} (dashed) highlighting high-curvature segments.

Real-car validation (raceline speed profiles capped for safety).: We evaluated the learned controller on a physical 1:10 racecar (F1TENTH class) [3] equipped with a VESC-based drivetrain, a 2D LiDAR (UST-10LX class), and onboard IMU/odometry running Ubuntu 20.04 with ROS 2 Foxy. A SLAM-based map [25] was first generated for the test track; during experiments the vehicle localized against this map using scan matching and an MCL particle filter [22]. A raceline for the test track was computed offline using a minimum-curvature optimizer [4] and converted into a reference speed profile.

A lightweight ROS 2 wrapper supplies the policy with the same observation channels as in simulation (speed, curvature along the raceline, and heading error to the lookahead point) and returns the actions (L_d, g), preserving observation/action parity with the simulator and reducing Sim-to-Real drift due to interface mismatch.

Lap-time evaluation (Table IV). All controllers track the same raceline speed profile capped at $v_{\max} = 6$ m/s for safety and a fair comparison. Table IV summarizes the resulting lap times; RL-PP (joint L_d, g) achieves the best mean performance (9.46 ± 0.23 s; min 9.09, max 9.82), followed by RL-PP (L_d only) and Adaptive PP (linear $v \mapsto L_d$). Fixed PP is slightly slower, while the MPC raceline tracker is substantially slower in our hardware setting.

TABLE IV
REAL-CAR LAP TIMES USING A RACELINE SPEED PROFILE UNIFORMLY CAPPED AT $v_{\max} = 6$ m/s.

Controller	Mean	Std	Min	Max
RL-PP (joint L_d, g)	9.46	0.23	9.09	9.82
RL-PP (L_d only)	9.61	0.58	8.94	10.51
Adaptive PP (linear $v \mapsto L_d$)	9.72	0.27	9.34	10.40
Fixed PP (L_d fixed)	9.85	0.43	9.32	10.55
MPC raceline tracker	15.42	0.47	14.48	16.17

Trajectory overlays (Fig. 7). To visualize tracking behavior under a more conservative cap, we generate the overlays in Fig. 7 using the same raceline speed profile capped at $v_{\max} = 6$ m/s. All five controllers are overlaid against the reference raceline, highlighting that the learned policies follow the reference smoothly with small lateral deviation.

V. CONCLUSION

We presented a hybrid Pure Pursuit (PP) controller in which a PPO policy *jointly* selects the lookahead L_d and steering gain g from curvature–speed features. The design preserves PP’s geometric steering law and adds only lightweight interfaces (two ROS topics) with a linear fallback, yielding a deployable and interpretable module.

Across simulation tracks, the RL-tuned PP consistently improves lap time, path-tracking accuracy, and steering smoothness over both fixed- L , velocity-linear baselines, and MPC raceline tracker (see Tables II and III). On hardware with a capped raceline speed profile, RL-PP (joint (L_d, g)) achieves the best mean lap time among the evaluated controllers (Table IV) and exhibits smooth tracking behavior in the qualitative zooms and trajectory overlays (Figs. 7).

Overall, adaptive selection of (L_d, g) reduces per-track retuning and transfers from simulation to the real car, though performance depends on a high-quality global reference raceline/localization and may degrade under extreme dynamics, motivating future work on robustness to imperfect references and highly dynamic scenarios.

REFERENCES

- [1] A. L. Garrow, D. L. Peters, and T. A. Panchal, “Curvature sensitive modification of pure pursuit control,” *ASME Letters in Dynamic Systems and Control*, vol. 4, no. 2, p. 021002, 2024.
- [2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” arXiv:1707.06347, 2017.
- [3] M. O’Kelly, H. Zheng, D. Karthik, and R. Mangharam, “F1tenth: An open-source evaluation environment for continuous control and reinforcement learning,” in *Proc. NeurIPS Competition and Demonstration Track*, vol. 123, 2020, pp. 77–89.
- [4] A. Heilmeier, A. Wischnewski, L. Hermansdorfer, J. Betz, M. Lienkamp, and B. Lohmann, “Minimum curvature trajectory planning and control for an autonomous race car,” *Vehicle System Dynamics*, vol. 58, no. 10, pp. 1497–1527, 2020.
- [5] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proc. ACM SIGKDD*, 2019, pp. 2623–2631.
- [6] S. Rahman, X. Jiang, Y. Zhang, M. Wang, and C. Chen, “A survey on trajectory planning and path tracking for autonomous vehicles,” *Results in Engineering*, vol. 23, p. 102439, 2025.
- [7] Y. Beycimen and A. Duran, “Unmanned ground vehicles: A comprehensive review of perception, planning, and control,” *Journal of Field Robotics*, 2023.
- [8] A. AbdElmoniem and M. Osama, “A path-tracking algorithm using predictive stanley lateral controller,” *International Journal of Advanced Robotic Systems*, vol. 17, no. 6, pp. 1–14, 2020.
- [9] R. Al Fata and N. Daher, “Adaptive pure pursuit with deviation model regulation for trajectory tracking in small-scale racecars,” in *Proc. Int. Conf. Control, Automation, and Instrumentation (IC2AI)*, 2025, pp. 1–6.
- [10] F. Tarhini, R. Talj, and M. Doumiati, “Adaptive look-ahead distance based on an intelligent fuzzy decision for an autonomous vehicle,” in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, 2023, pp. 1–8.
- [11] J. B. Kimball, B. DeBoer, and K. Bubbar, “Adaptive control and reinforcement learning for vehicle suspension control: A review,” *Annual Reviews in Control*, vol. 58, p. 100974, 2024.
- [12] V. Mnih, A. P. Badia, M. Mirza et al., “Asynchronous methods for deep reinforcement learning,” in *Proc. Int. Conf. Machine Learning (ICML)*, 2016, pp. 1928–1937.
- [13] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” arXiv:1509.02971, 2015.
- [14] M. Ganesan, S. Kandhasamy, B. Chokkalingam, and L. Mihet-Popa, “A comprehensive review on deep learning-based motion planning and end-to-end learning for self-driving vehicle,” *IEEE Access*, vol. 12, pp. 66 031–66 067, 2024.
- [15] S. Milani, N. Topin, M. Veloso, and F. Fang, “Explainable reinforcement learning: A survey and comparative review,” *ACM Comput. Surv.*, vol. 56, no. 7, 2024.
- [16] A. Wachi, W. Hashimoto, X. Shen, and K. Hashimoto, “A generalized formulation and algorithms for safe exploration in reinforcement learning,” in *Proc. Int. Joint Conf. Artificial Intelligence (IJCAI)*, 2024.
- [17] X. Hu, S. Li, T. Huang, B. Tang, R. Huai, and L. Chen, “How simulation helps autonomous driving: A survey of sim2real, digital twins, and parallel intelligence,” *IEEE Trans. Intelligent Vehicles*, vol. 9, no. 2, pp. 593–616, 2024.
- [18] R. Muduli, D. Jena, and T. Moger, “Application of reinforcement learning-based adaptive pid controller for automatic generation control of multi-area power system,” *IEEE Trans. Automation Science and Engineering*, vol. 22, pp. 1057–1068, 2025.
- [19] T. Kobayashi, “Reward bonuses with gain scheduling inspired by iterative deepening search,” *Results in Control and Optimization*, vol. 12, p. 100244, 2023.
- [20] B. D. Evans, H. W. Jordaan, and H. A. Engelbrecht, “Comparing deep reinforcement learning architectures for autonomous racing,” *Machine Learning with Applications*, vol. 14, p. 100496, 2023.
- [21] Z. Han, P. Chen, B. Zhou, and G. Yu, “Hybrid path tracking control for autonomous trucks: Integrating pure pursuit and deep reinforcement learning with adaptive look-ahead mechanism,” *IEEE Trans. Intelligent Transportation Systems*, vol. 26, no. 5, pp. 7098–7112, 2025.
- [22] C. Walsh and S. Karaman, “Cddt: Fast approximate 2d ray casting for accelerated localization,” arXiv:1705.01167, 2017.
- [23] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dorrmann, “Stable-baselines3: Reliable reinforcement learning implementations,” *J. Mach. Learn. Res.*, vol. 22, no. 268, pp. 1–8, 2021.
- [24] “F1tenth racetracks,” GitHub repository, 2020, accessed 2025.
- [25] W. Hess, D. Kohler, H. Rapp, and D. Andor, “Real-time loop closure in 2d lidar slam,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.