Reinforcement Learning-Enhanced Adaptive NMPC for Safe Autonomous Driving

Sheng Jin and Joel Y. Y. Loh *

Posted Date: 28 May 2026

doi: 10.20944/preprints202605.1986.v1

Keywords: autonomous vehicles; control barrier function (CBF); nonlinear model predictive control (NMPC); proximal policy optimization (PPO); reinforcement learning (RL)

 doi:10.20944/preprints202605.1986.v1

Article

Reinforcement Learning-Enhanced Adaptive NMPC for Safe Autonomous Driving

Sheng Jin and Joel Y. Y. Loh *

Department of Electrical and Electronic Engineering, University of Manchester, Oxford Road, Manchester, M13 9PL, UK

* Correspondence: joel.loh@manchester.ac.uk

Abstract

Nonlinear Model Predictive Control (NMPC) has garnered significant attention in autonomous systems due to its ability to predict future states and manage complex vehicle dynamics. However, the adaptability of existing NMPC methods is constrained by having to manually set the weight coefficients in the NMPC cost function. This study aims to explore a novel approach that integrates NMPC with Reinforcement Learning (RL), specifically employing Proximal Policy Optimization (PPO), to dynamically adjust NMPC weight matrices. The investigation begins by establishing a physics-based model for a two wheeled differential drive vehicle. A PPO model is then trained and deployed real-time to adapt to the NMPC weight matrices, achieving a 71% reduction in tracking error compared with the NMPC baseline. Importantly, the performance gain arises from PPO's ability to reshape the NMPC cost function real-time, amplifying both orientation and lateral penalties in curves while relaxing them on straights, thereby enabling adaptive trade-offs between accuracy and control effort that static-weight NMPC cannot achieve. To enhance safety, the controller is integrated with a Control Barrier Function (CBF) layer for real-time obstacle avoidance, while PPO's real-time weight adaptation contributes to improved tracking performance relative to NMPC+CBF. Finally, robustness evaluations under friction uncertainty, sensor noise, and path disturbances demonstrate that the PPO+NMPC+CBF method maintains reliable tracking accuracy and safety margins.

Keywords: autonomous vehicles; control barrier function (CBF); nonlinear model predictive control (NMPC); proximal policy optimization (PPO); reinforcement learning (RL)

1. Introduction

Path tracking is a fundamental problem in autonomous driving, where the controller must ensure accurate trajectory following while satisfying vehicle dynamics and safety constraints. In recent years, various control strategies have been explored for path tracking. Traditional PID controllers, favored for their simplicity, are widely applied to lane keeping and parking, with nested PID improving accuracy [1,2]. Sliding Mode Control (SMC) enhances robustness through nonlinear switching, while fuzzy PID and adaptive fuzzy SMC improve adaptability via online gain tuning [3–5]. Yet these controllers cannot explicitly handle multi-variable constraints, limiting their scalability in autonomous driving. Model Predictive Control (MPC) overcomes this by embedding dynamics and constraints into an optimization problem [6]. Extensions with state lattice planners [7] and refined cost functions [8], improve planning and comfort, but linear approximations limit fidelity in dynamic scenarios [9]. To address this shortcoming, Nonlinear Model Predictive Control (NMPC) directly incorporates nonlinear vehicle dynamics into the predictive model, allowing more accurate state evolution and improved performance in complex and highly dynamic driving conditions [10]. It enables joint steering and torque control for obstacle avoidance at high speeds and enhances rollover prevention for four-wheel independent drive vehicles [11,12]. In addition, NMPC supports automatic weight tuning via genetic algorithms and robust control through Gaussian process modeling of uncertainty [13,17]. However, its effectiveness is highly sensitive to factors such as vehicle speed,

 doi:10.20944/preprints202605.1986.v1

road curvature, and sampling frequency. Prior studies report that high vehicle speed or road curvature significantly amplifies prediction errors and can destabilize the optimizer if weights are not carefully adjusted [18,19]. Similarly, insufficient sampling density degrades prediction fidelity, while excessive sampling increases solver iterations and computational time, leading to response delays [20]. As a result, NMPC often requires trade-offs between responsiveness, stability, and computational cost. The latter grows approximately quadratically with prediction horizon and number of constraints, making real-time operation challenging for embedded platforms [21].Furthermore, traditional NMPC approaches rely on fixed weights and parameters that are typically tuned via expert knowledge or costly trials, making them time-consuming and difficult to scale [22].

Several adaptive tuning strategies have been proposed to reduce this dependence on manual NMPC weight selection. Genetic algorithms have been used to tune nonlinear MPC parameters for autonomous vehicle velocity and steering control [13], while Bayesian optimization has been applied to MPC weight adaptation in connected and automated vehicle scenarios [14]. Other approaches use curvature-dependent or gain-scheduled MPC weights so that tracking penalties change with the local path geometry [15]. Recent weight-varying MPC methods also combine optimization-based weight selection with reinforcement learning to adjust controller priorities online [16]. These approaches show that adaptive weighting is an active research direction, but they differ in the source of adaptation. Offline optimizers such as genetic algorithms and Bayesian optimization can reduce manual tuning but may require repeated scenario evaluations. Gain-scheduled and curvature-dependent methods are computationally efficient but depend on predefined scheduling rules. The present work instead learns an online mapping from local tracking features to NMPC weight scaling coefficients, while retaining the NMPC solver for constrained control action generation.

To overcome the limitations of classical controllers, recent work has explored Reinforcement Learning (RL) for greater adaptability. RL learns optimal strategies through interaction with the environment, avoiding explicit models or manual tuning. Most control applications adopt the Actor–Critic (AC) architecture, where the Actor maps observations to actions and the Critic evaluates long-term rewards [23]. This end-to-end mechanism enables low-latency inference, making RL suitable for high-frequency or resource-constrained systems [24,25]. Therefore, to better handle complex dynamic environments, integrating RL with traditional controllers has gained increasing attention. Studies have combined RL with PID to adjust weights in real time, reducing AGV tracking errors and improving adaptability to complex paths [26,27]. In MPC, RL has been used to tune cost functions and prediction horizons online, enhancing trajectory tracking without increasing computational load [28,29], while neural networks have been applied to learn vehicle dynamics for improved prediction accuracy and robustness [30]. For NMPC, a RL-based frameworks like Re4MPC dynamically select models, constraints, and cost functions, significantly reducing computational overhead [31], and Actor-Critic structures have been employed to optimize weights and initialization for better convergence and stability [32]. Additionally, real-time tuning methods combining digital twins and RL improve control accuracy in uncertain environments such as surface systems [33–35]. However, most RL-based controllers adopt an end-to-end design, which limits interpretability and weakens constraint enforcement [36]. They also lack formal safety guarantees, restricting their applicability in safety-critical autonomous driving.

Since the weighting matrices in NMPC directly determine the trade-off among tracking accuracy and energy consumption, adaptive tuning becomes a natural entry point. The core challenge is to integrate RL into NMPC in a way that preserves transparency, respects constraints, and ensures real-time safety. This work addresses these issues by proposing a three-layer framework which can adapt NMPC weights online with a Control Barrier Function (CBF) layer enforcing safety via real-time projection. The performance of the proposed approach is illustrated in **Figure 1**, which presents a trajectory comparison highlighting its advantages over baseline methods.

 doi:10.20944/preprints202605.1986.v1

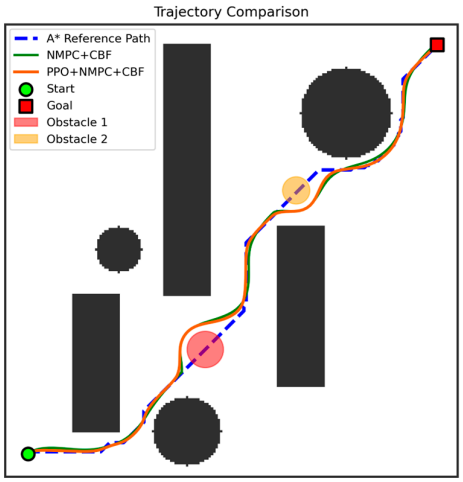


Figure 1. Trajectory comparison among the A* reference path, NMPC+CBF, and the proposed PPO+NMPC+CBF framework. The proposed method achieves closer adherence to the reference while ensuring safe clearance from obstacles.

This paper is organized as follows. Section II formulates the autonomous vehicle trajectory tracking problem and defines the control objectives. Section III presents the proposed PPO+NMPC+CBF framework, detailing the integration of PPO-based adaptive weight tuning with NMPC optimization and safety enforcement via CBF. The simulation setup and experimental results are presented, including evaluations of tracking accuracy, safety, and robustness in Section IV. Finally, conclusion remarks are provided in Section V.

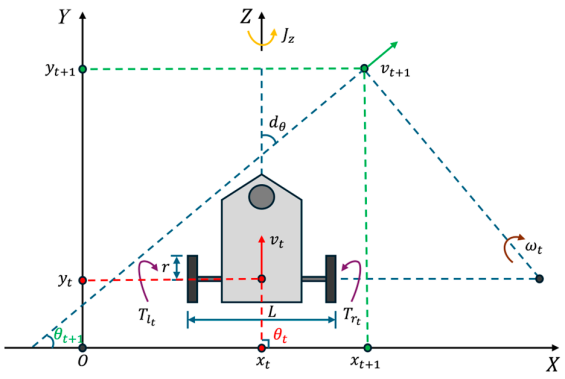
2. Problem Formulation

Section II formulates the trajectory-tracking problem for a differential-drive vehicle. Both the dynamic model of the system and the associated NMPC controller are presented to establish the basis for the proposed PPO-augmented safe control framework.

2.1. Vehicle Dynamic Model

This section systematically derives the kinematic and dynamic models of a two-wheel differential drive robot with a front caster and rear Ackermann wheels [37]. A nonlinear state-space representation is subsequently established to facilitate the design of advanced control strategies.

Consider a global inertial frame (X, Y) and a local body-fixed frame (x, y) attached to the vehicle's center of mass, as illustrated in **Figure 2**. Denoting v as the forward linear velocity and ω as the yaw rate about the Z-axis, the kinematic equations are given by:



 doi:10.20944/preprints202605.1986.v1

Figure 2. Differential-drive vehicle model showing position (x, y), velocity v, yaw rate ω, and wheel torques T_l, T_r. The yaw angle θ is measured counterclockwise from the global X-axis The variable J_z represents the yaw moment of inertia about the vertical Z-axis.

$$\dot{Z} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \quad (1)$$

The dynamics terms of the differential-drive vehicle are derived by applying Newton's second law in the longitudinal direction and torque balance about the yaw axis. The resulting nonlinear five-state model captures both translational and rotational dynamics terms, considering wheel torques, rolling resistance, aerodynamic drag, and yaw damping. The compact state-space $\Theta = [x\ y\ \theta\ v\ \omega]^T$ form is expressed as follows:

$$\dot{\Theta} = \mathcal{F}(\Theta, u) = \begin{bmatrix} \dot{Z} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} \quad (2)$$

$$= \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \\ \dfrac{[(T_l + T_r)/r - c_{rr} mg\,sgn(v) - c_v v)]}{m} \\ \dfrac{L(T_r - T_l)/2r - c_\omega \omega}{J_z} \end{bmatrix}$$

where T_l, T_r denote the left and right wheel torques, r is the wheel radius, c_{rr} and c_v represent the rolling resistance and aerodynamic drag coefficients, L is the wheelbase, c_ω is the yaw damping coefficient, m is the vehicle mass, ω denotes the yaw rate of the vehicle, and J_z is the yaw inertia. The nonlinear function $\mathcal{F}(\cdot)$ encodes the kinematic and dynamic relations of the vehicle, while $u = [T_l\ T_r]^T$ represents the input vector of wheel torques. A detailed derivation of the vehicle kinematic and dynamics are provided in **Appendix-A**.

2.2. NMPC Controller for Trajectory Tracking

Nonlinear Model Predictive Control extends the classical MPC framework by incorporating nonlinear vehicle dynamics into the prediction process. While retaining the quadratic cost structure of linear MPC, NMPC accounts for nonlinear state transitions, thereby improving fidelity in highly dynamic conditions.

To solve the nonlinear constrained optimization problem, a Sequential Quadratic Programming (SQP) scheme is employed. At each control step, the nonlinear dynamics and inequality constraints are locally linearized around the current iterate $(\Theta_i^{guess}, u_i^{guess})$. A quadratic programming (QP) subproblem is then constructed:

$$QP_{NMPC} = arg\min_{\Delta\Theta, \Delta u} \sum_{k=0}^{N-1} \frac{1}{2} \begin{bmatrix} \Delta\Theta_{i,k} \\ \Delta u_{i,k} \end{bmatrix}^T H_{i,k} \begin{bmatrix} \Delta\Theta_{i,k} \\ \Delta u_{i,k} \end{bmatrix} \quad (3a)$$

s.t.

$$\Delta\Theta_{i,0} = \widehat{\Theta}_i - \Theta_{i,0}^{guess} \quad (3b)$$

$$\Delta\Theta_{i,k+1} = A_{i,k}\Delta\Theta_{i,k} - B_{i,k}\Delta u_{i,k} \quad (3c)$$

$$C_{i,k}\Delta\Theta_{i,k} + D_{i,k}\Delta u_{i,k} + h_{i,k} \leq 0 \quad (3d)$$

 doi:10.20944/preprints202605.1986.v1

The SQP update variables are defined by $\Theta_{i,k} = \Theta_{i,k}^{\text{guess}} + \Delta\Theta_{i,k}$ and $u_{i,k} = u_{i,k}^{\text{guess}} + \Delta u_{i,k}$, where the superscript guess denotes the current SQP iterate and the delta terms are the QP decision variables.

Where $A_{i,k} = \partial\mathcal{F}/\partial\Theta$ and $B_{i,k} = \partial\mathcal{F}/\partial u$ are Jacobians of the dynamics, while $C_{i,k}, D_{i,k}$ are the Jacobians of inequality constraints, and $h_{i,k} = h\left(\Theta_i^{guess}, u_i^{guess}\right)$ is the constraint offset. The Hessian of the Lagrangian is approximated using a Gauss–Newton scheme, yielding a positive semi-definite weighting matrix $H_{i,k} \approx W_{i,k}$, which reduces computational burden while preserving descent directions. The Newton directions $(\Delta\Theta_i, \Delta u_i)$ are updated via line search until convergence [38], and the first control input is applied in a receding horizon fashion.

The inequality in Eq. (3d) is the first-order approximation of the nonlinear constraint function written in the standard SQP form $g(\Theta, u) \leq 0$. The sign convention ≤ 0 means that admissible trajectories correspond to non-positive constraint values, while positive values indicate constraint violation. Thus, Eq. (3d) enforces the locally linearized state, input, input-rate, and safety constraints within the QP subproblem.

The linearization in Eq. (3) is used as a local SQP approximation rather than as a global approximation of the nonlinear vehicle dynamics. Its validity depends on the predicted state and input trajectories remaining close to the current SQP iterate over the prediction horizon. In this study, the vehicle operates under relatively smooth low-speed path-tracking conditions, with bounded inputs, a short prediction horizon, and repeated re-linearization at every receding-horizon step. These conditions limit the deviation between the nonlinear dynamics and their first-order approximation over each short horizon. The approximation may become less accurate during aggressive maneuvers, high-speed operation, large initial tracking errors, abrupt obstacle interactions, severe actuator saturation, or longer prediction horizons. Thus, the linearized QP subproblem is used only as a local computational step inside the SQP-based NMPC solver, while the closed-loop controller relies on repeated online re-linearization to remain consistent with the nonlinear model.

The trajectory tracking problem with embedded safety can be written as a finite-horizon optimal control problem:

Before writing the finite-horizon problem, we define $\mathcal{X}$ as the admissible state set, $\mathcal{U}$ as the admissible wheel-torque input set, and $\Delta\mathcal{U}$ as the admissible input-increment set. These calligraphic set symbols are used consistently in the constraints below.

$$\min_{\{u_k\}_{k=0}^{N-1}} \sum_{k=0}^{N-1} \left(e_k{}^T Q e_k + u_k{}^T R u_k\right) \quad (4a)$$

s.t.

$$\Theta_k = \widehat{\Theta}_k \quad (4b)$$

$$\Theta_{k+1} = \mathcal{F}(\Theta_k, u_k),\ k = 1, \dots, N-1 \quad (4c)$$

$$\Theta_k \in X,\ k = 1, \dots, N-1 \quad (4d)$$

$$u_k \in U,\ k = 0, \dots, N-2 \quad (4e)$$

$$\Delta u_k \coloneqq u_k - u_{k-1} \in \Delta U,\ k = 0, \dots, N-2 \quad (4f)$$

where $e_k = y_k - r_k$ is the tracking error and Q, R weight tracking accuracy and control effort. The admissible state and input sets are given by X and U, while ΔU constrains the rate of input variation to improve feasibility and smoothness.

The weighting matrices are chosen to satisfy the standard quadratic-regulator definiteness conditions. Specifically, $Q \succcurlyeq 0$ is positive semi-definite so that tracking errors are penalized with non-negative weights, while $R \succ 0$ is positive definite so that the control-effort term remains strictly convex with respect to the input. These conditions are maintained throughout the simulations.

In this formulation, the predictive optimizer enforces system dynamics and feasibility, while the PPO agent adaptively tunes the cost weights real-time. This separation allows NMPC to remain interpretable, while RL and CBF adapts the cost and enforces safety respectively in real time.

 doi:10.20944/preprints202605.1986.v1

The controller should be interpreted as a receding-horizon numerical optimizer rather than a globally optimal regulator. At each sampling instant, the NMPC solver computes a finite-horizon solution under local SQP linearization, a Gauss-Newton Hessian approximation, finite solver tolerances, and PPO-adapted cost weights. The resulting input is therefore locally optimized, or suboptimal when the SQP iterations are terminated before exact convergence, rather than globally optimal for the original nonlinear problem.

For the same reason, this work does not claim a formal closed-loop stability guarantee in the Lyapunov or asymptotic sense. Such a guarantee would require additional terminal costs, invariant terminal sets, recursive-feasibility arguments, or a dedicated Lyapunov analysis. The simulations instead evaluate bounded tracking behavior, constraint satisfaction in the tested scenarios, and robustness under selected disturbances.

3. Construction of PPO+NMPC+CBF Framework

This section presents the construction of the PPO+NMPC+CBF framework, which integrates reinforcement learning, predictive optimization, and safety-constrained control. The PPO module is first introduced to generate adaptive weight coefficients for NMPC, followed by the PPO-adapted NMPC formulation. Finally, a Control Barrier Function layer is incorporated to impose obstacle-avoidance constraints. Together, these components form a closed-loop controller in which PPO adapts the NMPC cost weights, NMPC enforces the vehicle dynamics and input limits, and the CBF layer adds obstacle-avoidance constraints. Strict forward invariance of the safe set is obtained only when the CBF constraints are feasible and the slack variables introduced later remain inactive. When relaxation is required, the controller should instead be interpreted as providing practical safety preservation with penalized safety-margin violations.

3.1. PPO Model Training

PPO is a policy gradient algorithm that improves stability by optimizing a clipped surrogate objective, preventing destructive policy updates [39]. At each step of the rollout, i.e., during the policy's interaction with the environment, the agent receives a compact feature vector summarizing kinematic states, path-relative geometry, task progress, and a simplified cost summary. The Actor head outputs scaling coefficients that rescale the NMPC weighting matrices: Q, which penalizes state deviation from the reference, and R, which penalizes control effort. The NMPC solver uses these adapted weights to compute the optimal control input for the plant. The Critic head evaluates the current state by predicting its long-term value, guiding the policy update through advantage estimates. Advantages are computed using Generalized Advantage Estimation (GAE), and the policy is optimized by maximizing the clipped surrogate objective:

$$L^{CLIP}(\theta) = \hat{\boldsymbol{E}}_t\left[min\left(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\right)\right] \quad (5)$$

where $r_t(\theta)$ is the probability ratio between the updated and old policies, and $\hat{A}_t$ denotes the advantage estimator [40]. The full PPO objective integrates the clipped surrogate with a value function penalty and an entropy bonus to balance exploration and stability [39]:

$$L^{PPO}(\theta) = \hat{\boldsymbol{E}}_t\left[{L_t}^{CLIP}(\theta) - c_1{L_t}^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right] \quad (6)$$

where c_1, c_2 are scalar hyperparameters, ${L_t}^{VF}(\theta)$ is the squared value error $(V_\theta(s_t) - {V_t}^{targ})^2$, and $S[\pi_\theta]$ is the entropy term.

In Eq. (5), $r_t(\theta)$ is the policy probability ratio and $\hat{A}_t$ is the advantage estimate. The clipping parameter ϵ limits the policy update ratio to the interval $[1-\epsilon, 1+\epsilon]$, which prevents a single batch from producing an excessively large actor update. Equation (6) then combines the clipped policy objective, value-function fitting, and entropy regularization, so the PPO update balances policy improvement, critic accuracy, and exploration.

The PPO Neural Network (NN) reweights the predictive controller's quadratic penalties instead of commanding torques directly. As illustrated in **Figure 3**, In each step, the PPO NN reads a compact

 doi:10.20944/preprints202605.1986.v1

feature vector which summarizes kinematic states, path-relative geometry, task progress, and a simplified summary of the cost function. The policy returns a few non-negative coefficients. These coefficients scale the diagonal terms of Q and R.

The shared backbone extracts features used by both the actor and critic. The policy head applies Softplus so the scaling coefficients remain non-negative before they are passed to the NMPC cost matrices, while the value head estimates the state value for critic training.

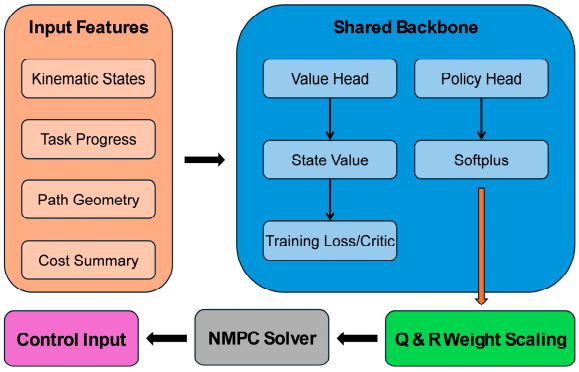


Figure 3. PPO weight-adaptation module. The actor outputs non-negative scaling coefficients for the NMPC weighting matrices Q and R, while the critic estimates the state value for PPO training.

During training, the environment returns a scalar reward and penalty function that combines five terms:

$$r_t = \omega_{pos} \cdot norm_{dist_t} + \omega_{head} \cdot head_t + \omega_{speed} \cdot speed_t + \omega_{goal} \cdot goal_t + \omega_{shape} \cdot shape_t. \quad (7)$$

These components jointly encourage accurate, smooth, and efficient path tracking. (i) The position term $norm_dist_t$ represents the lateral deviation of the vehicle's center from the reference path, normalized to reflect tracking accuracy; (ii) The heading term $head_t$ denotes the angular error between the vehicle heading and the tangent of the path, capturing steering alignment; (iii) The speed term $speed_t$ is the difference between the current forward velocity and the nominal reference speed v_{ref}, reflecting speed stability; (iv) The goal term $goal_t$ measures either the distance to the target or a terminal indicator of goal attainment, enforcing convergence to the final point; (v) The shaping term $shape_t$ represents the reduction in distance-to-goal between consecutive steps, acting as a potential-based shaping reward to encourage gradual progress toward the target [42]. The relative influence of each component is determined by the corresponding weights $\omega_{pos}, \omega_{head}, \omega_{speed}, \omega_{goal}, \omega_{shape}$.

Algorithm 1 summarizes the offline PPO training framework used to adapt the NMPC weight matrices. In each iteration, the policy generates scaling factors for the cost matrices, which the NMPC solver exploits to compute control actions. Collected trajectories are used to compute rewards and advantage estimates, and the policy is updated via the clipped surrogate objective with entropy regularization. The resulting policy is then deployed to provide adaptive weight tuning during execution.

In Algorithm 1, π_θ denotes the actor policy, V_ϕ denotes the critic network, the reference-path environment supplies rollout observations and rewards, and the NMPC solver converts PPO-scaled weights into feasible wheel-torque commands. The variables o_t and a_t denote the observation and PPO action at time step t, respectively.

Algorithm 1: Offline PPO Training Framework.

Input: Initial policy π_θ, value network V_ϕ, reference path environment, NMPC solver.
Output: Trained policy π_θ^*.

 doi:10.20944/preprints202605.1986.v1

Initialize networks and empty rollout buffer $\mathcal{D}$;
while training budget not exhausted do
  Reset environment; obtain o_0;
  while episode not finish do
    Policy outputs a_t to rescale (Q, R) diagonals;
    NMPC solves horizon problem; apply first control;
    Simulate environment to obtain o_{t+1} and reward r_t;
    Store $(o_t, a_t, r_t, o_{t+1}, \log \pi_\theta(a_t|o_t), V_\phi(o_t))$ in $\mathcal{D}$;
  end
  Estimate GAE and value targets from $\mathcal{D}$;
  Update π_θ via clipped surrogate with entropy;
  Update V_ϕ by regression to value targets;
end
Return trained policy π_θ^*.

In summary, the PPO algorithm has been described in terms of its surrogate objective, AC architecture, and reward shaping. This completes the formulation of PPO training, which will serve as the learning component in the proposed control framework.

3.2. PPO+NMPC

This section introduces a framework that integrates Proximal Policy Optimization with Nonlinear Model Predictive Control. The proposed framework enhances nonlinear predictive control by introducing an adaptive weight-shaping mechanism. At each decision step, the PPO policy receives a compact observation vector that summarizes vehicle motion states, path-following deviations, and task progress. Instead of producing control inputs directly, the policy outputs a set of non-negative coefficients that scale the diagonal elements of the NMPC weighting matrices:

$$Q' = diag(\alpha_1 q_1, \alpha_2 q_2, \dots) \quad (8)$$

$$R' = diag(\beta_1 r_1, \beta_2 r_2, \dots) \quad (9)$$

Here, Q' and R' are the PPO-adapted versions of the nominal NMPC cost matrices Q and R in Eq. (4). The prime notation is used only to distinguish the online scaled matrices from the fixed nominal matrices. The coefficients α_i and β_j are generated by the PPO policy and scale the corresponding diagonal entries of Q and R, thereby reshaping the quadratic tracking and control-effort penalties. Thus, the mapping from observation to action can be expressed as:

The nominal coefficients satisfy $q_i \geq 0$ and $r_j > 0$. The Softplus activation and clipping bounds in Eq. (10) keep the PPO scaling coefficients within admissible ranges, so the adapted matrices preserve $Q' \succcurlyeq 0$ and $R' \succ 0$. PPO therefore changes the relative importance of tracking and control effort without violating the definiteness requirements of the quadratic NMPC cost.

$$(\alpha_i, \beta_j) = clip(softplus(W_2 \phi(W_1 o_k + b_1) + b_2), c_{min}, c_{max}) \quad (10)$$

$$u_k = \kappa_{NMPC}\left(\Theta_k;\ Q'(\alpha_i), R'(\beta_j)\right) \quad (11)$$

where o_k is the observation vector at step k; W_1, W_2, b_1, b_2 are policy network parameters; $\phi(\cdot)$ is the hidden-layer activation function; c_{min}, c_{max} are lower and upper bounds applied to both α_i and β_j; Θ_k is the system state at step k; and $\kappa_{NMPC}(\cdot)$ denotes the NMPC solver that computes feasible control input u_k under adapted weights and physical constraints.

The use of PPO as an online weight-adaptation layer also differs from offline tuning and rule-based scheduling. Offline tuning methods select a fixed set of weights before deployment, whereas curvature-scheduled or gain-scheduled designs update weights according to predefined variables or rules. In the proposed framework, the policy updates the weighting coefficients at each control step

 doi:10.20944/preprints202605.1986.v1

using the current tracking state, path-relative features, and task information. The controller therefore retains the constrained NMPC structure, while the relative emphasis on lateral error, heading error, and control effort changes with the local driving context.

This mechanism enables the controller to change its priorities dynamically. When the reference path involves sharp curves, the policy increases the relative weights associated with orientation and lateral deviation, which reduces tracking error and improves path fidelity. Conversely, when the vehicle moves along straighter segments, these penalties are relaxed while more emphasis is placed on minimizing control effort, resulting in smoother actuation and greater energy efficiency. By doing so, the NMPC solver always optimizes against a cost function that reflects real-time driving demands.

An important feature of this design is the separation of responsibilities. Reinforcement learning provides adaptability by adjusting cost weights online, while NMPC ensures interpretability and constraint satisfaction by solving the predictive optimization problem under nonlinear dynamics and physical limits. This division preserves feasibility and stability, while improving robustness compared to fixed-weight formulations.

Nonetheless, the current framework only adapts to path geometry and internal states. It cannot directly address the presence of moving obstacles or other safety-critical scenarios. To overcome this limitation, the next section introduces a Control Barrier Function layer that augments the adaptive controller with real-time safety constraints.

3.3. PPO+NMPC+CBF

Control Barrier Functions provide a formal condition for forward invariance of a designated safety distance set $\mathbb{C}$ [40]. For a control-affine system $\dot{x} = f(x) + g(x)u$, a continuously differentiable function $h: \mathbb{R}^n \to \mathbb{R}$ defines $\mathbb{C} = \{x | h(x) \geq 0\}$. Safety is preserved if there exists an extended class-K_∞function $\alpha(\cdot)$ such that:

$$L_f h(x) + L_g h(x)u + \alpha\big(h(x)\big) \geq 0 \quad (12)$$

which restricts the evolution of $h(x)$ to prevent the state from crossing the boundary $h(x) = 0$. This condition generalizes Nagumo's theorem to controlled dynamical systems and provides a sufficient condition for set invariance [44]. In discrete-time settings, the CBF condition is often approximated by:

$$h_k - h_{k-1} + \gamma h_{k-1} \geq 0 \quad (13)$$

where $\gamma > 0$ controls the allowable approach rate to the constraint boundary [45]. In this obstacle avoidance task, the safety function is defined as:

Equation (12) is the continuous-time CBF condition. The safe set is $\mathcal{C} = \{x \mid h(x) \geq 0\}$, and the terms $L_f h(x)$ and $L_g h(x)$ are Lie derivatives of the safety function along the drift and input vector fields. The inequality limits the decrease of $h(x)$ so that, when the constraint is feasible, a trajectory that starts inside $\mathcal{C}$ cannot cross the boundary $h(x) = 0$.

Equation (13) is the discrete-time counterpart used over the NMPC prediction horizon. For $0 < \gamma \leq 1$, it can be written as $h(x_k) \geq (1-\gamma)h(x_{k-1})$. Hence, if $h(x_{k-1}) \geq 0$ and the constraint is enforced without relaxation, the next predicted state remains inside the safe set. The parameter γ controls how fast the trajectory may approach the boundary, with smaller values enforcing a more conservative approach rate.

$$h(x) = d(x, x_{obs}) - (r_{obs} + \delta) \quad (14)$$

where $d(\cdot)$ denotes the Euclidean distance between the vehicle and the obstacle center, r_{obs} is the obstacle radius, and δ is a safety margin. If condition (12) is feasible and enforced without relaxation, the predicted trajectory maintains the prescribed minimum clearance from the obstacles over the prediction horizon.

By embedding the safety function into the predictive optimization, the complete control problem becomes an NMPC formulation with adaptive cost shaping from PPO and safety constraints imposed by CBF. At each control step, the PPO policy outputs scaling coefficients α_i, β_j that adapt the

 doi:10.20944/preprints202605.1986.v1

diagonal entries of the cost matrices Q and R. The resulting finite-horizon optimization problem is expressed as

$$\min_{\{u_k\}_{k=0}^{H-1},\{s_k{}^{(o)}\}_{k=1}^{H}} \sum_{k=0}^{H}(e_k{}^T Q' e_k) + \sum_{k=0}^{H-1}(u_k{}^T R' u_k)$$

$$+\rho \sum_{o\in\mathcal{O}}\sum_{k=1}^{H}\left\|s_k{}^{(o)}\right\|_1 \quad (15a)$$

s.t.

$$\Theta_{k+1} = \mathcal{F}(\Theta_k, u_k),\ k = 0,\dots,H-1 \qquad (15b)$$

$$\Theta_k \in X, u_k \in U, \Delta u_k \in \Delta U, \quad (15c)$$

$$h^{(o)}(\Theta_k) - h^{(o)}(\Theta_{k-1}) + \gamma h^{(o)}(\Theta_{k-1}) \geq -s_k{}^{(o)}, \qquad (15d)$$

$$s_k{}^{(o)} \geq 0, k = 1,\dots,H. \qquad (15f)$$

Here, Q', R' are the PPO-adapted weight matrices and $s_k{}^{(o)}$ are slack variables penalized with factor $\rho > 0$. The functions $h^{(o)}(\Theta_k)$ represent safety functions defined for each obstacle $o \in \mathcal{O}$. Introducing slack variables ensures feasibility even in narrow passages or under strong disturbances, but it also opens the possibility of constraint violations. Therefore, each slack variable is penalized in the cost function by a large factor ρ, discouraging unnecessary relaxation of the safety constraints.

Constraint (15d) applies the discrete CBF condition in Eq. (13) to each obstacle o and each prediction step k. The safety function $h\hat{}(o)(Theta\_k)$ is positive when the predicted vehicle position is outside the obstacle radius plus the margin, zero on the safety boundary, and negative inside the forbidden region. With $s\_k\hat{}(o) = 0$, Eq. (15d) enforces the discrete CBF condition directly. With $s\_k\hat{}(o) > 0$, the lower bound is relaxed to preserve feasibility, and the resulting trajectory may temporarily reduce the nominal safety margin.

It is important to distinguish the strict CBF condition from the relaxed implementation used here. If s_k^(o) = 0 for all obstacles and prediction steps, the discrete CBF condition is enforced directly and the safe set remains forward invariant under the model assumptions. If s_k^(o) > 0, the constraint is relaxed to maintain feasibility, and the safety margin may be temporarily reduced. The penalty term on s_k^(o) therefore does not guarantee zero violation, but makes relaxation costly so that it is used only when strict feasibility cannot be maintained.

To better illustrate the overall architecture, **Figure 4** provides a schematic overview of the proposed framework. Thus, the PPO+NMPC+CBF framework combines adaptive cost shaping, predictive optimization, and safety enforcement in a unified loop. PPO provides online weight adjustment, NMPC enforces dynamic feasibility, and the CBF layer promotes obstacle avoidance through safety constraints. When the relaxed CBF constraints require nonzero slack, the method preserves feasibility while minimizing safety-margin violations rather than guaranteeing strict obstacle clearance. The following section evaluates its effectiveness through simulation experiments.

 doi:10.20944/preprints202605.1986.v1

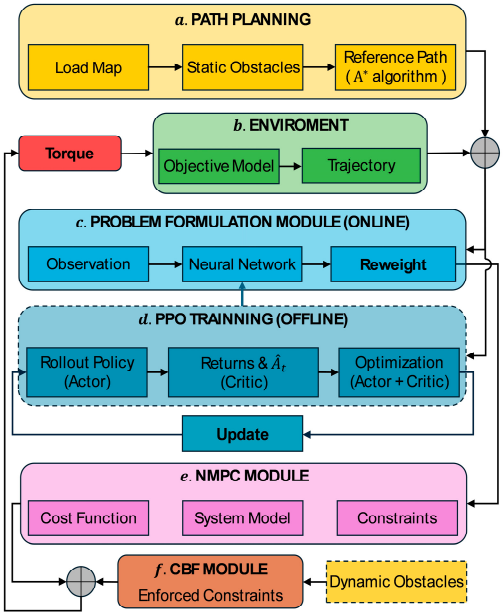


Figure 4. Workflow of the proposed PPO+NMPC+CBF control framework. (*a*) A* generates a reference path from maps and static obstacles. (*b*) The environment simulates vehicle response using torque inputs. (*c*) Real-time module observes states and adjusts NMPC weights via a neural network. (*d*) Offline PPO trains Actor–Critic networks to optimize the weight policy. (*e*) NMPC solves constrained optimal control. (*f*) CBF imposes real-time safety constraints for dynamic obstacle avoidance. Solid lines represent data flow; dashed borders indicate external modules.

4. Experiments

This section presents comprehensive simulation experiments to validate the proposed PPO+NMPC+CBF framework. The experiments evaluate tracking accuracy, real-time obstacle avoidance, and robustness under uncertainties, demonstrating the framework's performance relative to baseline controllers in terms of error reduction, safety enforcement, and control stability. Results confirm the framework's effectiveness in the tested simulations, with PPO+NMPC achieving approximately 71% tracking error reduction and PPO+NMPC+CBF maintaining zero collisions in the evaluated dynamic-obstacle scenarios. All simulations are conducted within a unified environment, and the controller parameters are kept consistent unless otherwise specified.

4.1. Simulation Setup

The simulation environment was implemented in Python 3.10, using SciPy (SLSQP solver) for numerical computation, and *PyTorch* for PPO implementation. Development was executed on an Intel Core i7-13650HX CPU with 8 GB RAM and an NVIDIA RTX 4060 Laptop GPU. Experiments were conducted on a 200×200 grid map representing structured urban environments (see **Appendix-B**), with dynamic obstacles introduced at $(4.4, 2.8, r = 0.4\,m)$ in Step 65 and $(6.4, 6.3, r = 0.3\,m)$ in Step 150. These experiments are tested on Case 9 in the dataset. The key parameters utilized in the simulation environment are summarized in **Table I**.

The reported simulations were executed in an offline Python environment and were not profiled as an embedded real-time implementation. Therefore, the current hardware specification should be interpreted as the computational platform used for simulation, rather than as evidence of guaranteed real-time deployment performance. The short prediction horizon and 0.1 s sampling time used in the simulations are also consistent with the local-validity assumption of the SQP linearization used in the NMPC solver.

 doi:10.20944/preprints202605.1986.v1

The present evaluation is limited to numerical simulation. The reported results therefore validate the proposed control architecture under controlled simulation conditions, but they do not constitute hardware-in-the-loop or physical robot validation. Real-world deployment would introduce additional effects that are not fully captured in the current model, including actuator delay, wheel slip, localization error, sensor latency, communication delay, and embedded-computation limits. These factors may affect both NMPC feasibility and the timing of PPO-based weight adaptation.

Table I. SIMULATION PARAMETERS.

Parameter	Value
Vehicle mass (m)	$10\ kg$
Inertia (J_z)	$0.4\ kg \cdot m^2$
Wheel Radius (r)	$0.05\ m$
Wheelbase (L)	$0.30\ m$
Rolling Friction coef. (c_{rr})	0.01
Velocity Damping coef. (c_v)	1.0
Angular Damping coef. (c_ω)	0.05
NMPC Horizon (N)	10
Time Step (d_t)	$0.1\ s$
Max Velocity (v_{max})	$2.0\ m/s$
Max Torque (T_{max})	$2.0\ \ N \cdot m$

4.2. Simulation Results and Analysis

The PPO model was trained externally using a self-constructed dataset of 32 maps (excluding Case 9 for testing), with each dataset a 200×200 grid representing structured urban environments. The maps were procedurally generated to cover a spectrum of layouts, ranging from straight paths with sparse obstacles to dense clusters, providing varied training conditions for the policy.

The training environment used a 10-dimensional state vector and a 7-dimensional action vector. The policy network employed an Actor–Critic architecture with a hidden layer of 128 units, Softplus activation for non-negative outputs, and clipped surrogate objectives for stability. Training lasted 500 episodes with a batch size of 32 and a learning rate of 0.001 using Adam optimizer.

The PPO policy was trained on 32 procedurally generated grid maps and evaluated on Case 9, which was excluded from training. This setup tests whether the learned weight-tuning strategy can be reused on a new map within the same simulation environment. The result should therefore be interpreted as generalization within the tested map class, not as evidence of broad applicability to arbitrary driving environments. The policy observes local tracking and task features, rather than the full map, so it can adjust the NMPC weights when the vehicle encounters different local path conditions such as straight segments, turns, and obstacle-induced deviations. However, the training set remains limited, and the controller has not yet been evaluated under substantially different road layouts, vehicle parameters, reference speeds, sensor delays, actuator dynamics, or hardware constraints. Future work will extend the evaluation to larger and more diverse map sets, out-of-distribution scenarios, and hardware-in-the-loop or real-vehicle experiments.

As shown in **Figure 5**, the cumulative reward increased from approximately -65 in early episodes to stabilize near -10 by episode 500, reflecting improved tracking accuracy, speed regulation, and goal approach. The policy began to exhibit convergence after roughly 180 episodes, where the reward curve leveled off and variability reduced, indicating consistent performance and effective balancing of the dense reward components across diverse maps.

 doi:10.20944/preprints202605.1986.v1

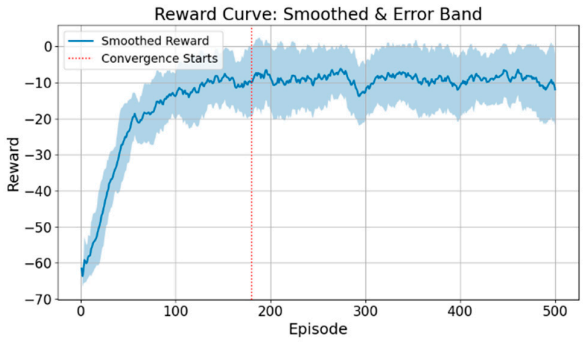


Figure 5. PPO Training Reward Across 500 Episodes (smoothed with error band). The red line marks the onset of the steady phase at around 180 episodes.

To evaluate the benefits of PPO-driven adaptive weight tuning, we compared fixed-weight NMPC with PPO+NMPC on Case 9 without dynamic obstacles, using the pre-trained PPO model. Both controllers shared the same nonlinear dynamics and A* reference path, differing only in cost adaptation: NMPC employed static weights $(Q = R = diag(1))$, while PPO+NMPC dynamically scaled Q and R according to state observations.

The metrics comparison, as illustrated in **Figure 6**, shows that PPO+NMPC significantly improves positional and heading accuracy compared to fixed-weight NMPC, leading to more precise path following. Specifically, the mean position error decreases from 76.2 mm to 22.3 mm, and the mean absolute heading error reduces from 11.14° to 7.31° demonstrating substantial gains in tracking fidelity. Here, the heading error is reported as the mean of absolute angular deviations, providing a robust measure of orientation accuracy across diverse trajectories. These accuracy gains are accompanied by slight reductions in step count and path length, but they also come with higher total energy consumption and lower energy efficiency. The radar chart highlights this contrast, with PPO+NMPC scoring higher on accuracy-related metrics but lower on energy-related metrics. The result indicates that the PPO policy does not improve all performance dimensions simultaneously. Instead, it shifts the controller toward tighter tracking and faster correction at the expense of actuation effort. This behaviour is consistent with the learned weight adaptation because larger lateral and heading penalties in curved or high-error regions cause the NMPC solver to apply stronger corrective torques. Overall, **Figure 6** illustrates how PPO-driven adaptation prioritizes tracking fidelity and stability rather than energy minimization.

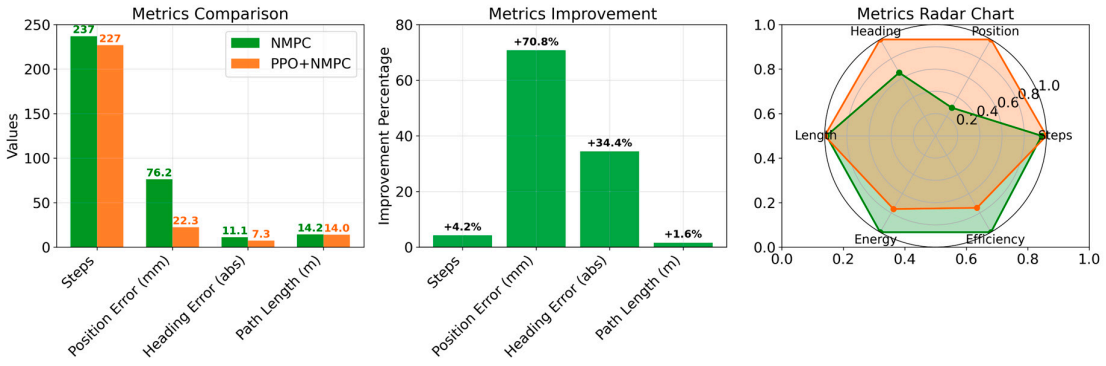


Figure 6. Comparison of NMPC and PPO+NMPC across key performance metrics. The left and middle subplots show absolute values and relative improvements. The radar chart visualizes normalized metrics, with Energy defined as the time-integrated wheel torques and Efficiency as the path length–to–energy ratio. All radar values are normalized by setting the larger value of each metric to 1, ensuring direct visual comparison between the two controllers.

 doi:10.20944/preprints202605.1986.v1

TABLE II SUMMARIZES THE QUANTITATIVE PERFORMANCE COMPARISON ACROSS ALL CONTROLLERS ABOVE. RESULTS SHOW THAT PPO AUGMENTATION CONSISTENTLY REDUCES MEAN POSITION AND HEADING ERRORS AND SLIGHTLY DECREASES STEP COUNT, WHILE THE CBF LAYER ADDS SAFETY CONSTRAINTS FOR OBSTACLE AVOIDANCE IN THE TESTED SCENARIOS.

TABLE II. PERFORMANCE METRICS FOR BASELINE AND PPO-AUGMENTED CONTROLLERS.

Metrics	NMPC	PPO+NMPC	NMPC+CBF	PPO+NMPC+CBF
Steps	237	227	250	**240**
Mean position error (m)	0.076	0.022	0.101	**0.086**
Path length (m)	14.22	13.99	16.26	**15.48**
Total energy (J)	34.88	45.94	38.79	**49.72**
Energy efficiency (m/J)	0.408	0.305	0.419	**0.311**

The energy-related metrics in Table II show the cost of adaptive weight tuning. PPO+NMPC and PPO+NMPC+CBF both reduce tracking error, but both also require higher total energy than their fixed-weight counterparts. The CBF layer enforces obstacle-avoidance constraints, while PPO shifts the NMPC cost toward more aggressive tracking corrections. As a result, the proposed controller is best interpreted as an accuracy- and safety-oriented controller rather than an energy-optimal controller. In applications where energy efficiency is the primary objective, the PPO reward function or the NMPC cost should include a stronger energy penalty so that the learned policy balances tracking accuracy, safety, and actuation effort more evenly.

Real-time computational performance is an important requirement for deploying the proposed controller on an autonomous platform. In the present study, the controller was evaluated in simulation and the implementation was not instrumented to separately record NMPC solve time, PPO inference time, SQP iteration count, maximum control-loop latency, or achieved closed-loop control frequency. PPO inference is expected to add only a small overhead because the policy network is a compact feedforward model, whereas the dominant computational cost is expected to come from the NMPC optimization and the CBF-constrained solve. However, without explicit profiling, the present results should be interpreted as simulation-based validation of the control architecture rather than a complete real-time implementation benchmark. A full runtime study should report the mean and maximum NMPC solve time, PPO inference time, total control-loop latency, average solver iterations, and achieved control frequency under representative hardware and embedded deployment conditions.

The present comparison focuses on fixed-weight NMPC, PPO+NMPC, NMPC+CBF, and PPO+NMPC+CBF. A direct benchmark against other adaptive NMPC strategies, such as genetic-algorithm-tuned NMPC, Bayesian-optimization-tuned NMPC, curvature-dependent weight scheduling, and adaptive MPC, is outside the scope of the current simulation study. Such a comparison would be useful because these methods adapt the controller through different mechanisms and may lead to different trade-offs among tracking accuracy, energy use, computational cost, and safety-margin preservation. Future work will evaluate these adaptive tuning strategies under a common vehicle model, map set, and obstacle-avoidance benchmark.

Figure 7 compares position and heading tracking errors between NMPC+CBF and PPO+NMPC+CBF. The error curves (left) show that PPO+NMPC+CBF not only yields smaller deviations, especially after obstacle encounters, but also returns to the reference trajectory more quickly, indicating faster recovery. The transient response following disturbances is noticeably smoother, suggesting stronger robustness compared to fixed-weight NMPC. The boxplots (right) further highlight this improvement: both the mean (from 0.100 m to 0.08 m) and median of the position error decrease (from 0.05 m to 0.03 m), while the heading error distribution becomes

 doi:10.20944/preprints202605.1986.v1

narrower with fewer outliers. Beyond reducing error magnitudes, the reduced variance and tighter distributions demonstrate more consistent tracking

Building upon the adaptive tracking improvements demonstrated with PPO+NMPC, the next step is to evaluate PPO+NMPC+CBF for scenarios involving dynamic obstacles. The evaluation focuses on scenarios with dynamic obstacles introduced during the trajectory. Both utilize the same nonlinear dynamics model, A* reference path, and CBF constraints ($\gamma = 0.7, d_{safe} = 0.15$ m). The difference lies in weight tuning: NMPC+CBF maintains fixed weights, while PPO+NMPC+CBF employs dynamic adaptation via the pre-trained PPO policy. This setup enables a direct comparison of safety and tracking performance under identical conditions.

and more reliable convergence, lowering the risk of extreme deviations.

Following the tracking error analysis, **Figure 8** illustrates how PPO dynamically adjusts the Q and R matrices within the PPO+NMPC+CBF framework by showing weight adjustment factors over 250 time steps. Rather than treating all terms equally, the policy adapts specific weights to changing trajectory demands. For example, $Q_1 - Q_3$ and R_2 show clear variability, indicating active regulation of lateral, longitudinal, and heading errors as well as torque balance between wheels. This reflects the controller's attempt to prioritize accuracy and stability under different operating phases. By contrast, Q_4, Q_5 and R_1 remain nearly constant, suggesting that the policy deliberately keeps angular-rate and certain input penalties stable to preserve safety margins and prevent excessive control effort. Notably, R_2 shows greater variability than R_1, indicating that the policy exploits the right-wheel torque penalty as a flexible degree of freedom for fine-tuning trajectory corrections. Meanwhile, the left-wheel penalty remains stable, serving as a baseline to preserve overall control balance. The overall pattern highlights a division of labor: some weights are flexibly tuned to handle trajectory complexity, while others remain fixed to guarantee robust constraint enforcement. This balance demonstrates how PPO adapts the cost landscape in real time, providing interpretability for why certain performance trade-offs emerge.

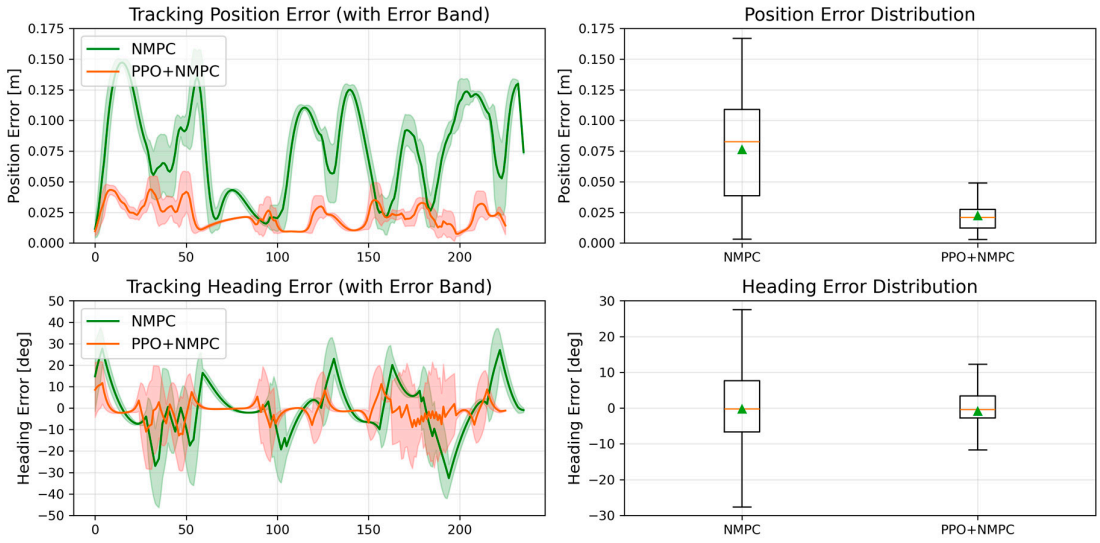


Figure 7. Comparative Analysis of Tracking Errors for NMPC+CBF and PPO+NMPC+CBF Frameworks, showing error trajectories with uncertainty bands (left) and error distributions with boxplots (right).

 doi:10.20944/preprints202605.1986.v1

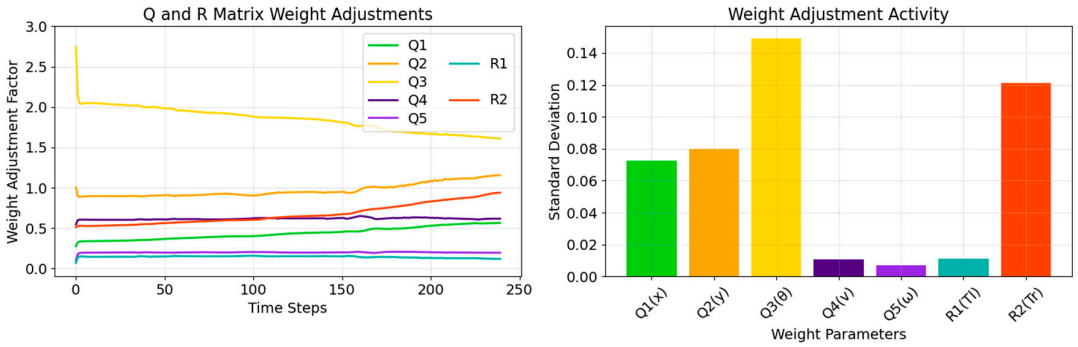
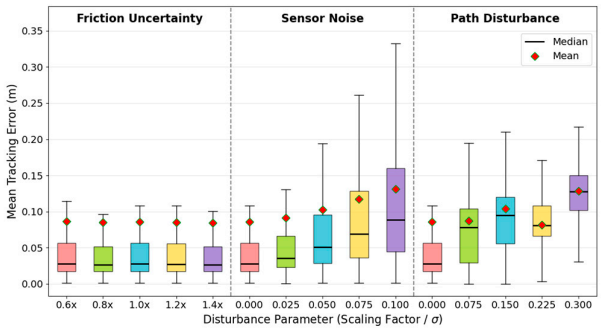


FIGURE 8. DYNAMICS AND ACTIVITY OF Q AND R MATRIX WEIGHT ADJUSTMENTS IN PPO+NMPC+CBF FRAMEWORK. EACH CURVE CORRESPONDS TO A SPECIFIC COST COMPONENT: Q_1 (LATERAL ERROR), Q_2 (LONGITUDINAL ERROR), Q_3(HEADING ERROR), Q_4AND Q_5 (VELOCITY AND ANGULAR RATE), AND R_1 AND R_2 (CONTROL EFFORT ON LEFT AND RIGHT WHEEL TORQUES).

4.3. Robustness Evaluation

Real-world autonomous driving is inevitably affected by uncertainties such as model mismatch, sensor imperfections, and path disturbances. To evaluate the robustness of the proposed PPO+NMPC+CBF framework, three representative disturbances were considered: (i) friction uncertainty, introduced by scaling the rolling resistance and velocity damping coefficients; (ii) sensor noise, simulated by injecting Gaussian noise of varying standard deviations into state observations; and (iii) path disturbance, generated by perturbing the reference trajectory with random deviations. Each scenario was tested over 20 independent trials using the pre-trained PPO policy in Section IV-B and the nonlinear vehicle model in Section II-A, with mean tracking error adopted as the primary metric. Here, σ represents the standard deviation in the robustness evaluation.

The boxplots in **Figure 9** highlights the selective strengths of the PPO+NMPC+CBF framework. Its limited sensitivity to friction mismatch suggests that the learned weight adaptation remains stable under the tested parameter variations, although this does not by itself establish generalization to different vehicle models or driving environments. The gradual degradation under sensor noise reflects the limits of observation quality, but the errors remain bounded, showing that the controller avoids noise amplification. For path disturbances, the wider error spreads suggest more challenging recovery, yet the absence of large outliers indicates that the CBF layer helps prevent unsafe divergence in the tested scenarios. The mean and median tracking errors remain low ($\leq$ 0.13 m) across all disturbance scenarios, which reflects stable and bounded system behavior rather than any catastrophic deviation. In contrast, prior studies have reported that conventional NMPC controller exhibit larger deviations and wider error distributions when subjected to observation noise or model mismatch [46,47]. It reflects its sensitivity to parameter uncertainty and external disturbances.



 doi:10.20944/preprints202605.1986.v1

Figure 9. Robustness analysis of PPO+NMPC+CBF under friction uncertainty, sensor noise, and path disturbances. Boxplots illustrate mean tracking errors across varying disturbance levels. The results show stable performance with limited noise sensitivity and moderate disturbance effects.

Overall, the experiments show that PPO+NMPC improves tracking accuracy compared with fixed-weight baselines, while the integration of the CBF layer adds safety constraints for obstacle avoidance and improves recovery after disturbances in the tested scenarios. Robustness evaluations further confirm stable performance under friction, noise, and path uncertainties, with tracking errors remaining bounded. PPO+NMPC+CBF therefore shows a simulation-level balance between adaptability, accuracy, safety constraints, and control effort.

The robustness tests provide an initial indication of closed-loop behaviour under selected simulated uncertainties, but they do not replace experimental validation. A more complete validation should include hardware-in-the-loop testing and physical experiments on a mobile robot platform. In such tests, the same PPO+NMPC+CBF pipeline should be evaluated under real-time control constraints, with measured actuator response, localization uncertainty, obstacle-detection latency, and solver execution time included in the loop.

5. Conclusions

This paper proposed a PPO-augmented nonlinear MPC framework with a CBF safety layer for autonomous vehicle trajectory tracking. The design addresses adaptability, safety-constrained control, and robustness by combining dynamic weight adjustment with predictive optimization. Extensive simulations demonstrated that PPO-enhanced NMPC achieves notable reductions in position and heading errors compared with linear MPC and fixed-weight NMPC, but these gains are obtained with higher control effort and lower energy efficiency. Thus, the present policy favours tracking accuracy and safety over energy minimization. The CBF layer improves obstacle avoidance and enables smoother recovery after disturbances. Strict safety-constraint satisfaction holds when the CBF constraints remain feasible without slack relaxation, while the implemented relaxed formulation provides practical safety preservation by penalizing any required reduction of the safety margin. Robustness tests under friction uncertainty, sensor noise, and path disturbances further confirmed stable performance, with both mean and median tracking errors remaining within a small range across all tested scenarios. Collectively, these results validate the proposed PPO+NMPC+CBF framework as an effective and interpretable simulation framework for adaptive trajectory tracking and safety-constrained navigation within the tested environment class. Future work will include larger and more diverse map evaluations, out-of-distribution scenarios, explicit real-time profiling, hardware-in-the-loop validation, and physical experiments on a mobile robotic platform. These tests will be used to quantify NMPC solve time, PPO inference time, control-loop latency, solver iteration count, achievable control frequency, tracking error, and safety-margin preservation under real sensing, actuation, and computation constraints.

Accordingly, the reported results should be read as simulation evidence of bounded closed-loop behavior under the tested conditions, rather than as a proof of global optimality or formal asymptotic stability of the complete learning-enhanced controller.

Acknowledgments: This work acknowledges grant support from Royal Society RGS\R2\242489 and from the Dame Kathleen Ollernshaw Fellowship.

Conflicts of Interest: The authors declare no conflict of interests.

Appendix A

DERIVATION OF VEHICLE MODEL

The derivations in Appendix A are not claimed as a new theoretical contribution. They follow the standard Newton-Euler force and torque balance for differential-drive vehicles, consistent

 doi:10.20944/preprints202605.1986.v1

with the background model in [37], and are included to make the simulation model, assumptions, and parameter definitions self-contained.

The linear acceleration of the differentia-drive vehicle in the forward direction is obtained by balancing the driving force generated by the wheel torques against the principal resistive forces. Applying Newton's second law to the vehicle mass yields the net force balance from which the instantaneous linear acceleration $\dot{v}$ can be directly computed as:

$$\dot{v} = \frac{[(T_l + T_r)/r - c_{rr} mgsgn(v) - c_v v)]}{m} \quad (16)$$

where T_l, T_r denote the driving torques of left and right wheels with the wheel radius r. Opposing this motion are the rolling resistance and the viscous drag which are proportional to coefficients c_{rr} and c_v respectively. The angular acceleration of the differential-drive vehicle about its Z-axis is obtained by balancing the net driving torque generated by the wheel torque difference against the viscous damping torque. The net torque balance on the yaw inertia J_z gives the instantaneous angular acceleration $\dot{\omega}$ as:

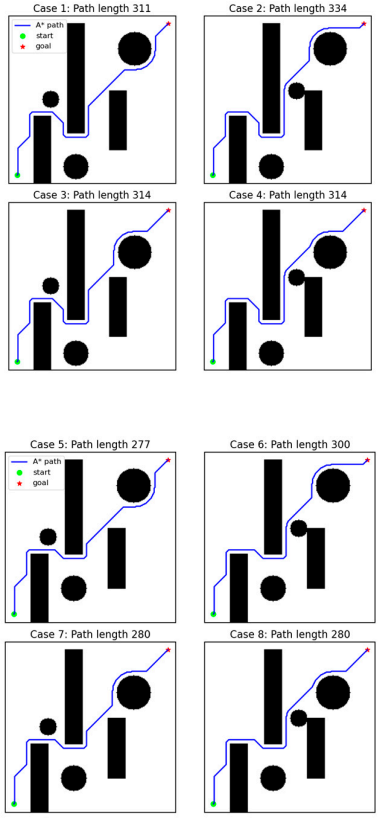
$$\dot{\omega} = \frac{L(T_r - T_l)/2r - c_\omega \omega}{J_z} \quad (17)$$

where L denotes the wheelbase. The coefficient c_ω represents the yaw damping coefficient and the variable ω denotes the yaw rate of the vehicle.

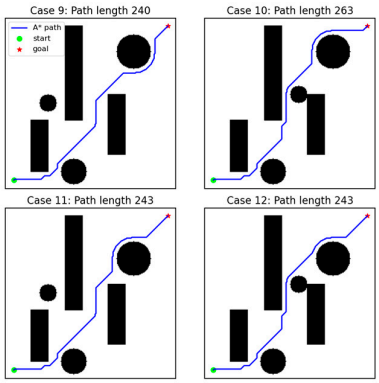
Appendix B

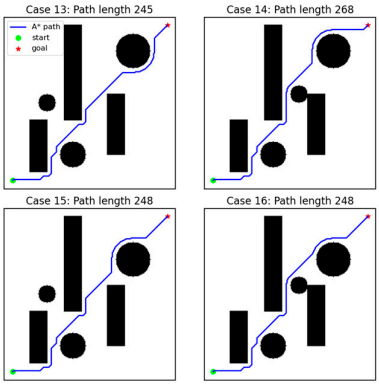
TRAINING DATASET

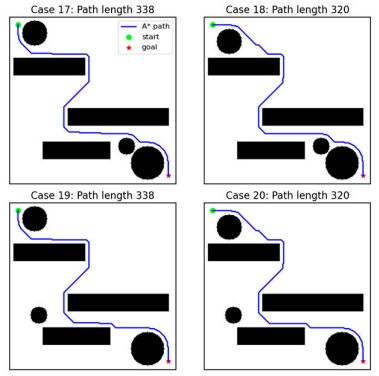
The training dataset of 32 A* reference paths, constructed on 200×200 grids to emulate urban environments, is provided :

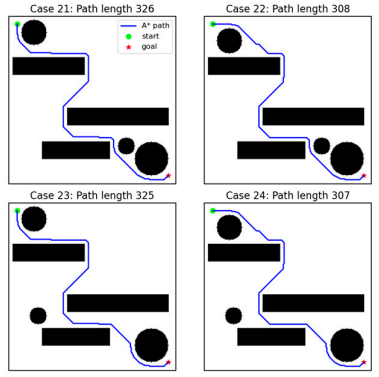


 doi:10.20944/preprints202605.1986.v1

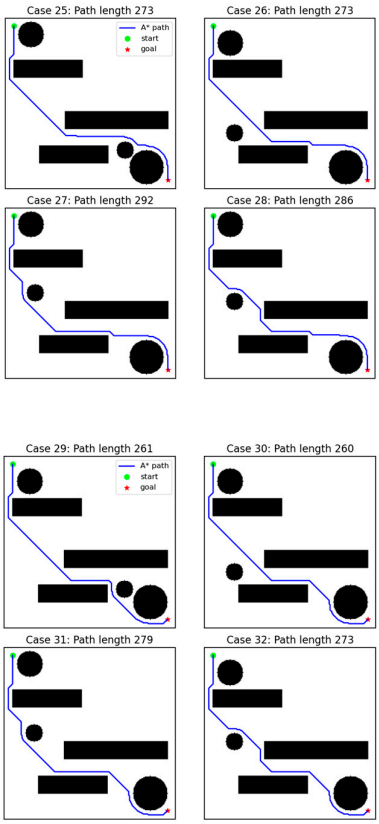








 doi:10.20944/preprints202605.1986.v1



References

1. R. Marino, S. Scalzi, G. Orlando, and M. Netto, “A nested PID steering control for lane keeping in vision-based autonomous vehicles,” in Proc. Amer. Control Conf., Jun. 2009, pp. 2885–2890.
2. X. Du, K. K. Tan, and K. K. K. Htet, “Vision approach towards fully self-reverse parking system,” in Proc. IEEE Int. Conf. Mechatronics Autom., Aug. 2014, pp. 186–191.
3. X. Du and K. K. Tan, “Autonomous reverse parking system based on robust path generation and improved sliding mode control,” IEEE Trans. Intell. Transp. Syst., vol. 16, no. 3, pp. 1225–1237, Jun. 2015.
4. X. Chen, Q. Bao, and B. Zhang, “Research on 4WIS electric vehicle path tracking control based on adaptive fuzzy PID algorithm,” in Proc. Chinese Control Conf., Jul. 2019, pp. 6753–6760.
5. Y.-C. Yeh, T.-H. S. Li, and C.-Y. Chen, “Adaptive fuzzy sliding-mode control of dynamic model-based car-like mobile robot,” Int. J. Fuzzy Syst., vol. 11, no. 4, pp. 272–281, Dec. 2009.
6. D. Limon, A. Ferramosca, I. Alvarado, and T. Alamo, “Model predictive control for setpoint tracking,” 2024, arXiv:2403.02973.
7. C. Zhang, D. Chu, S. Liu, Z. Deng, C. Wu, and X. Su, “Trajectory planning and tracking for autonomous vehicle based on state lattice and model predictive control,” IEEE Intell. Transp. Syst. Mag., vol. 11, no. 2, pp. 29–40, Summer 2019.
8. H. Wang, B. Liu, X. Ping, and Q. An, “Path tracking control for autonomous vehicles based on an improved MPC,” IEEE Access, vol. 7, pp. 161064–161073, Oct. 2019.
9. P. Stano, U. Montanaro, D. Tavernini, M. Tufo, G. Fiengo, L. Novella, et al., “Model predictive path tracking control for automated road vehicles: A review,” Annu. Rev. Control, vol. 55, pp. 194–236, Apr. 2023.
10. J. Köhler, M. A. Müller, and F. Allgöwer, “A nonlinear tracking model predictive control scheme for dynamic target signals,” Automatica, vol. 118, Art. no. 109030, Aug. 2020.
11. H. Yuan, X. Sun, and T. Gordon, “Unified decision-making and control for highway collision avoidance using active front steer and individual wheel torque control,” Veh. Syst. Dyn., vol. 57, no. 8, pp. 1188–1205, Aug. 2019.

 doi:10.20944/preprints202605.1986.v1

12. G. Zhu, H. Jie, and W. Hong, "NMPC-based path tracking control strategy for 4WID autonomous vehicle considering handling stability under extreme conditions," in Proc. 7th CAA Int. Conf. Veh. Control Intell., Oct. 2023, pp. 1–6.
13. X. Du, K. K. K. Htet, and K. K. Tan, "Development of a genetic-algorithm-based nonlinear model predictive control scheme on velocity and steering of autonomous vehicles," IEEE Trans. Ind. Electron., vol. 63, no. 11, pp. 6970–6977, Nov. 2016.
14. V. Le and A. Malikopoulos, "Controller adaptation via learning solutions of contextual Bayesian optimization," IEEE Robot. Autom. Lett., vol. 10, pp. 8308–8315, 2025, doi: 10.1109/LRA.2025.3585716.
15. E. Alcalá, V. Puig, J. Quevedo, and U. Rosolia, "Autonomous racing using Linear Parameter Varying-Model Predictive Control (LPV-MPC)," Control Eng. Pract., vol. 95, Art. no. 104270, 2020, doi: 10.1016/j.conengprac.2019.104270.
16. B. Zarrouki, M. Spanakakis, and J. Betz, "A safe reinforcement learning driven weights-varying model predictive control for autonomous vehicle motion control," in Proc. IEEE Intell. Vehicles Symp. (IV), 2024, pp. 1401–1408, doi: 10.1109/IV55156.2024.10588747.
17. C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Robust constrained learning-based NMPC enabling reliable mobile robot path tracking," Int. J. Robot. Res., vol. 35, no. 13, pp. 1547–1563, Nov. 2016.
18. J. Pannek and M. Gerdts, "Performance of sensitivity based NMPC updates in automotive applications," 2014, arXiv:1401.3548.
19. M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," in Nonlinear Model Predictive Control: Towards New Challenging Applications, L. Magni, D. M. Raimondo, and F. Allgöwer, Eds., Berlin, Heidelberg: Springer, 2009, pp. 391–417.
20. E. Kayacan, W. Saeys, H. Ramon, C. Belta, and J. M. Peschel, "Experimental validation of linear and nonlinear MPC on an articulated unmanned ground vehicle," IEEE/ASME Trans. Mechatron., vol. 23, no. 5, pp. 2023–2030, Oct. 2018.
21. L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos, "A simple and efficient algorithm for nonlinear model predictive control," in Proc. 56th IEEE Conf. Decis. Control, Dec. 2017, pp. 1939–1944.
22. G. C. Goodwin, M. G. Cea, M. M. Seron, D. Ferris, R. H. Middleton, and B. Campos, "Opportunities and challenges in the application of nonlinear MPC to industrial problems," in Proc. IFAC World Congr., Aug. 2012, pp. 39–49.
23. Y. P. Pane, S. P. Nageshrao, and R. Babuška, "Actor-critic reinforcement learning for tracking control in robotics," in Proc. 55th IEEE Conf. Decis. Control, Dec. 2016, pp. 5819–5826.
24. K. Kosta, M. A. Anwar, P. Panda, A. Raychowdhury, and K. Roy, "RAPID-RL: A reconfigurable architecture with preemptive-exits for efficient deep-reinforcement learning," in Proc. IEEE Int. Conf. Robot. Autom., May 2022, pp. 7492–7498.
25. M. Riemer, G. Subbaraj, G. Berseth, and I. Rish, "Enabling realtime reinforcement learning at scale with staggered asynchronous inference," 2024, arXiv:2412.14355.
26. Y. Shan, B. Zheng, L. Chen, L. Chen, and D. Chen, "A reinforcement learning-based adaptive path tracking approach for autonomous driving," IEEE Trans. Veh. Technol., vol. 69, no. 10, pp. 10581–10595, Oct. 2020.
27. J. E. Sierra-Garcia and M. Santos, "Combining reinforcement learning and conventional control to improve automatic guided vehicles tracking of complex trajectories," Expert Syst., vol. 41, no. 2, Art. no. e13076, 2023.
28. G. Bellegarda and K. Byl, "An online training method for augmenting MPC with deep reinforcement learning," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Oct. 2020, pp. 5453–5459.
29. Z. Chen, J. Lai, P. Li, O. I. Awad, and Y. Zhu, "Prediction horizon-varying model predictive control (MPC) for autonomous vehicle control," Electronics, vol. 13, no. 8, Art. no. 1442, Apr. 2024.
30. M. Rokonuzzaman, N. Mohajer, S. Nahavandi, and S. Mohamed, "Model predictive control with learned vehicle dynamics for autonomous vehicle path tracking," IEEE Access, vol. 9, pp. 128233–128249, 2021.
31. N. Ü. Akmandor, S. Prajapati, M. Zolotas, and T. Padır, "Re4MPC: Reactive nonlinear MPC for multi-model motion planning via deep reinforcement learning," 2025, arXiv:2506.08344.
32. R. Reiter, A. Ghezzi, K. Baumgärtner, J. Hoffmann, R. D. McAllister, and M. Diehl, "AC4MPC: Actor-critic reinforcement learning for nonlinear model predictive control," 2024, arXiv:2406.03995.

 doi:10.20944/preprints202605.1986.v1

33. B. Martinsen, A. M. Lekkas, and S. Gros, "Reinforcement learning-based NMPC for tracking control of ASVs: Theory and experiments," Control Eng. Pract., vol. 120, Art. no. 105024, Mar. 2022.
34. H. S. Berg, D. Menges, T. Tengesdal, and A. Rasheed, "Digital twin syncing for autonomous surface vessels using reinforcement learning and nonlinear model predictive control," Sci. Rep., vol. 15, no. 1, Art. no. 9344, Mar. 2025.
35. M. Mehndiratta, E. Camci, and E. Kayacan, "Automated tuning of nonlinear model predictive controller by reinforcement learning," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Oct. 2018, pp. 3016–3021.
36. G. Ceusters, L. R. Camargo, R. Franke, A. Nowé, and M. Messagie, "Safe reinforcement learning for multi-energy management systems with known constraint functions," 2022, arXiv:2207.03830.
37. S. K. Malu and J. Majumdar, "Kinematics, localization and control of differential drive mobile robot," Glob. J. Res. Eng., vol. 14, no. H1, pp. 1–7, Jan. 2014.
38. H. Yang, F. Deng, Y. He, D. Jiao, and Z. Han, "Robust nonlinear model predictive control for reference tracking of dynamic positioning ships based on nonlinear disturbance observer," Ocean Eng., vol. 215, Art. no. 107885, Nov. 2020.
39. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, arXiv:1707.06347.
40. J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, arXiv:1506.02438.
41. M. Han, L. Zhang, J. Wang, and W. Pan, "Actor-critic reinforcement learning for control with stability guarantee," IEEE Robot. Autom. Lett., vol. 5, no. 4, pp. 6217–6224, Oct. 2020.
42. S. Devlin and D. Kudenko, "Dynamic potential-based reward shaping," in Proc. Int. Conf. Autonomous Agents Multiagent Syst., Jun. 2012, pp. 433–440.
43. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in Proc. Eur. Control Conf., Jun. 2019, pp. 3420–3431.
44. Z. Horváth, Y. Song, and T. Terlaky, "Invariance conditions for nonlinear dynamical systems," 2016, arXiv:1607.01107.
45. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," IEEE Trans. Autom. Control, vol. 62, no. 8, pp. 3861–3876, Aug. 2017.
46. E. Suwartadi, V. Kungurtsev, and J. Jäschke, "Sensitivity-Based Economic NMPC with a Path-Following Approach," Processes, vol. 5, no. 1, p. 8, Mar. 2017.
47. H. Zhang, P. Li, and C. E. García, "Robust stability of nonlinear model predictive control based on extended Kalman filter," J. Process Control, vol. 22, no. 1, pp. 82–89, Jan. 2012.

Sheng Jin received the B.E. degree in Automation from Guangdong University of Technology, Guangzhou, China, in 2024. He is currently pursuing the M.Sc. degree in Advanced Control and Systems Engineering with the Department of Electrical and Electronic Engineering at the University of Manchester, Manchester, U.K. His research interests include learning-based control and reinforcement learning.

Joel Loh received the Ph.D. degree in Electrical and Computer Engineering from the University of Toronto, Toronto, Canada, in 2020. He is currently a Dame Kathleen Ollernshaw Fellow (Assistant Professor) with the Department of Electrical and Electronic Engineering, University of Manchester, Manchester, U.K. His research interests include metamaterials, memristors, chemical sensing, and artificial intelligence.