

PCARNN–DCBF: Minimal-Intervention Geofence Enforcement for Ground Vehicles

Yinan Yu and Samuel Scheidegger

Abstract—Runtime geofencing for ground vehicles is rapidly emerging as a critical technology for enforcing Operational Design Domains (ODDs). However, existing solutions struggle to reconcile high-fidelity learning with the structural requirements of verifiable control. We address this by introducing PCARNN-DCBF, a novel pipeline integrating a Physics-encoded Control-Affine Residual Neural Network with a preview-based Discrete Control Barrier Function. Unlike generic learned models, PCARNN explicitly preserves the control-affine structure of vehicle dynamics, ensuring the linearity required for reliable optimization. This enables the DCBF to enforce polygonal keep-in constraints via a real-time Quadratic Program (QP) that handles high relative degree and mitigates actuator saturation. Experiments in CARLA across electric and combustion platforms demonstrate that this structure-preserving approach significantly outperforms analytical and unstructured neural baselines.

Index Terms—geofencing, discrete control barrier functions, minimal intervention, control-affine bicycle model, physics-informed machine learning, physics-encoded modeling, polygonal signed distance function, emergency braking, automatic emergency steering

I. INTRODUCTION

GEOFENCING is a technique that uses virtual geographic boundaries to regulate vehicle operation within designated areas. It is the specification and runtime enforcement of spatial policies by predicting impending non-compliance and applying necessary interventions to stay compliant. It shares similarities with other AD/ADAS functions, such as lane keeping, collision avoidance, and road-edge protection, but its purpose is distinct: it governs *where* the vehicle is allowed to operate at all, not only *how* it should follow lanes or avoid objects. In practice, geofencing serves as a policy and safety envelope that complements perception and planning by continuously assessing proximity to a prescribed region and applying the minimal correction required to avoid a boundary breach [1].

a) Representations: A *geofence* is a geometric region defined in a global or local coordinate frame. Two fundamental modes are *keep-in* (remain inside the region) and *keep-out* (avoid entering the region) [2]. A geofence are commonly expressed as a polygon, a corridor (the set between two boundaries), a level-set of a distance function, or a union of such sets.

- **Polygon:** This is the most common representation for static, hard boundaries. Research in both ground and aerial vehicles frequently defines geofences using polygonal

horizontal boundaries [3]. While simple, this method must account for complex geometries; algorithms have been developed to handle "concave geozones" [4], as well as "multi-island" and "multi-hole" polygons [5]. In their simplest form, enforcement is achieved via a Point-in-Polygon check [6], though this is insufficient for predictive intervention.

- **Corridor:** This representation is intrinsically linked to motion planning. The literature on Model Predictive Control (MPC) for autonomous driving, for example, explicitly uses this concept. An integrated approach may feature a "corridor path planner" that defines the "constraints on vehicle position" [7]. This corridor is then fed into the MPC controller, which optimizes the vehicle's trajectory within those constraints. Here, the geofence is not a hard boundary but the dynamic input to an optimization-based controller.
- **Level-set:** This is a more formal and flexible, though computationally intensive, approach. In robotics practice, such implicit representations are commonly instantiated as Signed Distance Fields (SDFs), which encode obstacle geometry with distances and gradients that integrate naturally with control, trajectory optimization, and learning methods [8], [9], [10]. A level-set method defines the geofence boundary as the zero-value contour of a higher-dimensional implicit function [5]. In robotics, this is used for path planning by defining an anisotropic cost function (e.g. risk) and solving for the optimal, minimum-risk path using methods that solve the Eikonal or Hamilton-Jacobi-Bellman equations [11]. Its primary advantage is in representing complex, dynamic boundaries, as the boundary can be evolved over time by updating the level-set function [12].

b) Functional taxonomy: The functional taxonomy of geofencing characterizes geofences along three independent axes: static versus dynamic, hard versus soft, and passive versus active, each describing a different operational property of how the geofence is defined, interpreted, and enforced.

- **Static / dynamic:** Static fences are fixed regions encoded in digital maps. This is the baseline assumption for most SAE Level 4 (L4) Operational Design Domain (ODD) deployments [17] and for safety-critical test tracks. Dynamic fences are temporary, context-dependent areas. The literature strongly associates this capability with Vehicle-to-everything (V2X) and Cooperative Intelligent Transport Systems (C-ITS) [18]. Research has demonstrated V2X-aided autonomous driving systems that use Traveler

Yinan Yu is with the Department of Computer Science and Engineering, Chalmers University of Technology, Sweden and Asymptotic AI, Sweden. E-mail: yinan@chalmers.se.

Samuel Scheidegger is with Asymptotic AI, Sweden.

Information Messages (TIM) to "detou[r] around a construction site ahead", a successful real-world execution of a dynamic geofence.

- *Hard / soft*: Hard fences define strict invariance conditions that the vehicle must never violate. Soft fences introduce buffer zones or graded boundaries in which the system can modulate its response based on estimated risk or uncertainty. Soft implementations are often used when small deviations are tolerable or when measurement uncertainty warrants conservative, layered responses.
- *Passive / active*: Passive fences issue warnings or advisories but leave the final action to the driver or upstream planner. Active fences initiate automatic intervention, such as braking, steering, or a coordinated maneuver, when a violation is predicted. Active enforcement requires actuator authority to reside on board the vehicle; external services may transmit geofencing constraints or stop requests via V2X, but a safety-rated on-board controller remains the sole command owner for brake and steering execution.
- *Ground / aerial / maritime*: For ground vehicles, geofencing typically acts in the planar x - y space with heading and speed dynamics considered when predicting boundary crossings and planning evasive actions. For aerial systems, geofences are 3D volumes in (x, y, z) with altitude floors/ceilings and climb/sink rate constraints [13], [4]. For maritime and underwater vehicles, they are surface regions in (x, y) with shoreline and traffic-separation buffers or 3D (x, y, z) volumes with depth limits and bathymetry constraints. In all cases, attitude/turn-rate and speed dynamics inform prediction and the choice of feasible interventions.

c) *Violations*: Geofence violations may originate from different classes of events: (i) *human factors*: driver distraction, drowsiness, or misjudgment; (ii) *autonomy faults*: perception dropouts, localization drift, stale or mismatched maps, planner failures, or degraded actuation; (iii) *security events*: spoofed commands, GNSS manipulation, or malicious path injection that would drive the vehicle out of bounds, as exemplified in the broader automotive security literature [14], [15]. In all cases, geofencing aims to detect impending boundary violations early and apply the smallest safe correction.

d) *Actuation choices*: When a geofence violation is predicted, Automatic Emergency Braking (AEB) and Automatic Emergency Steering (AES) are both valid responses, but they address different geometries and margins. AEB is appropriate when the residual buffer allows the vehicle to stop within tire-road limits, or when lateral space is scarce or uncertain. AES is preferable when there is sufficient lateral clearance to redirect the vehicle while staying within stability envelopes (yaw rate, sideslip, rollover) and without creating new hazards. In practice, the controller may need to choose between braking, steering, or a coordinated combination (e.g. brake-to-stabilize, then steer) based on real-time feasibility under friction limits and surrounding traffic conditions.

The context also matters. If the vehicle experiences a critical system failure, continuing operation is not an option and a controlled stop via AEB is the appropriate response. This type of brake-to-stop fallback is similar in spirit to the

Minimum Risk Maneuvers (MRMs) discussed in automotive safety assessments built around standards such as ISO 26262 and ISO/SAE 21434, where automation transitions the vehicle toward a reduced-risk stopped condition when it can no longer guarantee safe operation (e.g. due to ODD exit or a critical failure) [16], [17], [18]. In our setting, we focus on this control-level conservative fail-safe stop rather than a full, standards-defined MRM. By contrast, for scenarios such as autonomous testing or routine operation within a bounded service area, path correction through AES may be preferable to preserve continuity of motion. Across all cases, a guiding principle is *minimal intervention*: apply the smallest deviation necessary to guarantee compliance with the geofence while avoiding disruptive or unsafe maneuvers. Actuator command ownership must be unambiguous. Two deployment patterns cover most cases. (a) *Vehicle-owned actuation*: the geofencing function runs on board or issues intervention requests (e.g. stop, lateral redirect with bounds) to the vehicle motion controller, which converts them into brake and steering commands under stability and friction limits. (b) *Service-assisted geofencing*: an off-board service computes constraints or issues a stop request, but an on-board safety controller is the only entity permitted to actuate; external services never bypass this controller. In both patterns, actuator commands flow only through the on-board command owner; external inputs are advisory or supervisory and are validated before execution.

e) *Architectural view*: A geofencing function can be understood in two stages: a *predictive layer* that estimates whether the current state and control will lead to a boundary breach, and a *control layer* that decides how to intervene, through braking, steering, or a combination, while respecting vehicle dynamics and safety constraints. Detailed design options for both stages are discussed in the background section. Within this two-stage view, only the control layer holds actuator authority. The predictive layer may request an intervention (brake, steer, or coordinated) with target setpoints or bounds, but it does not issue torque or steering commands directly. If a geofencing service is off board, it interfaces at the request level; final actuation remains on board.

f) *Applications*: Geofencing is directly connected to the ODD: it enforces the spatial component of the ODD at runtime [6]. Beyond commercial deployments that restrict automated operation to validated service areas, several concrete scenarios motivate an explicit geofencing layer:

- **Proving grounds and test tracks.** Keep a test vehicle within a validated area (e.g. a proving ground), stopping or steering when simulations or experiments push the vehicle toward an unsafe perimeter.
- **Geofenced automated services.** Low- to medium-speed automated shuttles, valet parking, campus logistics, mines, ports, and warehouses where the site boundary defines where autonomy is permitted.
- **Temporary or dynamic zones.** Road works, special events, school zones, and emergency scenes where the allowable region changes over time or moves with a convoy or escort.
- **Infrastructure safety margins.** Bridges, waterfronts, and drop-offs where boundary violations have high conse-

quence and must trigger decisive action.

g) *Relation to adjacent Autonomous Driving (AD)/Advanced Driver-Assistance Systems (ADAS) topics:*

Geofencing intersects but is not identical to lane departure prevention, collision avoidance, and road-edge detection. Lane departure focuses on markings and rules; geofencing governs arbitrary polygons and corridors. Collision avoidance manages moving/static objects; geofencing treats the workspace boundary as the constraint. Road-edge protection uses perception of drivable limits; geofencing can be defined via localization even without visual cues. At the supervisory level, geofencing aligns with ODD compliance, formal safety monitoring, and security measures that reject commands leading outside the allowed set.

In this paper, we consider ground vehicles with static, hard keep-in geofences given as polygons, and encode safety via a signed-distance level-set barrier $h(x) = d(p)$ on the vehicle position. We focus primarily on braking and steering as the actuation strategy.

Our primary contribution is a novel geofencing pipeline, Physics-encoded Control-Affine Residual Neural Network + Discrete Control Barrier Function (PCARNN-DCBF), which integrates a structured, physics-encoded dynamics model with a predictive barrier-function controller for reliable, optimization-based enforcement of polygonal keep-in constraints under complex vehicle dynamics. Specifically, we introduce:

- Physics-encoded Control-Affine Residual Neural Network, a hybrid model that learns corrections for both the drift and control-effectiveness terms of the vehicle dynamics. This architecture strictly preserves the control-affine structure essential for tractable, optimization-based control.
- A tractable, preview-based Discrete Control Barrier Function controller that *builds on* sampled-data and zero-order Control Barrier Functions (CBFs) for safety-critical control [19], [20], [21], [22] in the polygonal geofencing setting. It enforces a terminal barrier constraint on a numerically integrated zero-order-hold map, and combines a minimal-deviation Quadratic Programming (QP) objective, and saturation-aware bound–secant linearization near actuator limits to achieve real-time geofence enforcement with minimal intervention.
- We introduce a set of specialized metrics (Containment F1 Score (CF1), False Positive Rate (FPR), Median Containment Distance with Signed Preference (MCD⁺)) to address the key requirements of accurate intervention, successful containment, and minimal correction. Using this framework, we conduct a comprehensive evaluation in the Car Learning to Act (CARLA) simulator, where we benchmark the PCARNN-DCBF pipeline against analytical, Neural Ordinary Differential Equation (ODE), and standard physics-encoded residual models. We analyze performance by examining control linearity and robustness across different vehicle types (electric vs. combustion) and challenging driving regimes.

The paper is structured as follows: Section II reviews the problem setting, vehicle dynamics models, and background on control barrier functions. Section III surveys related work in

discrete-time CBFs and learned dynamics. Section Section IV details our proposed method, including the PCARNN dynamics model and the preview-based DCBF controller. Section V presents the experimental setup, benchmark comparisons, and a detailed analysis of the results.

II. BACKGROUND

A. Problem setting and preliminaries

We model geofencing as the runtime enforcement of a spatial constraint, defined by a region $S(t) \subseteq \mathbb{R}^2$ in the world frame. The set $S(t)$ denotes the admissible operating area at time t . For *static* geofences, $S(t)$ reduces to a fixed region $S \subseteq \mathbb{R}^2$, such as a polygon extracted from a digital map, and the vehicle must remain inside S for all time.

Let the vehicle position in the world frame be $\mathbf{p} = [p_x, p_y]^\top \in \mathbb{R}^2$, where p_x and p_y denote the Cartesian coordinates in the world frame. Under *keep-in* policies, admissible states satisfy $\mathbf{p} \in S$.

The full vehicle state in the world frame is defined as

$$x = [p_x, p_y, \psi, v_x, v_y, \omega, \delta]^\top \in \mathbb{R}^7 \quad (1)$$

where $\psi \in [-\pi, \pi]$ is the yaw angle, v_x and v_y are the longitudinal and lateral velocities in the body frame, ω is the yaw rate, and δ is the front-wheel steering angle. This stacks the vehicle pose and dynamic bicycle states in a form consistent with standard dynamic single-track models [23], [24], [25].

We define a geometric signed distance $d(p)$ on position and a state barrier $h(x)$ on the full state. Specifically, $d : \mathbb{R}^2 \rightarrow \mathbb{R}$ gives the signed distance from the position $p = [p_x, p_y]^\top$ to the geofence boundary. Its zero level set $\{p \in \mathbb{R}^2 : d(p) = 0\}$ coincides with the boundary, with $d(p) > 0$ for points inside the admissible region S and $d(p) < 0$ for points outside. The barrier on the full world-frame state x is the composition

$$h(x) := d(p), \quad p = \begin{bmatrix} p_x \\ p_y \end{bmatrix}.$$

Constructing control barrier functions directly from signed distance fields has been proposed as a practical strategy for collision avoidance, since the signed distance provides a scalar safety margin and its gradient supplies a natural avoidance direction that can be embedded in control synthesis [26], [27]. For simple polygonal fences, $d(p)$ is the minimum Euclidean distance to the polygon edges, with the sign determined by an even–odd (ray-crossing) inside test (orientation-independent) [28].

Since d is positive inside the safe zone, ∇d points inward (the gradient is undefined only at measure-zero sets such as vertices). We define the outward and inward unit normals as $\mathbf{n}_{\text{out}} = -\nabla d / \|\nabla d\|$ and $\mathbf{n}_{\text{in}} = +\nabla d / \|\nabla d\|$, respectively.

Symbols and definitions used in this paper can be found in Table I for convenience.

B. Vehicle dynamics

1) *Dynamic bicycle model:* The motion of a ground vehicle can be described by dynamic equations that relate the vehicle's state to its control inputs, capturing how velocities and

Table I: Symbols and definitions

States and Parameters	
World frame	A global coordinate system fixed to the environment, used to express the vehicle's absolute position and orientation.
Body frame	A local coordinate system attached to the vehicle, with the x -axis pointing forward (longitudinal) and y -axis pointing to the left (lateral), used to express velocities and dynamic states.
p_x, p_y $\mathbf{p} \in \mathbb{R}^2$	Vehicle coordinates in the world frame. Location vector of the vehicle $\mathbf{p} = [p_x, p_y]^\top$ in the world frame.
m	Vehicle mass.
v_x	Longitudinal velocity of the vehicle in its body frame (m/s).
v_y	Lateral velocity of the vehicle in its body frame (m/s).
ω	Yaw rate of the vehicle (rad/s), representing rotation around the vertical axis.
δ	Front wheel steering angle (rad).
$\psi \in [-\pi, \pi]$	Vehicle's yaw angle in the world frame.
$R(\psi)$	Rotation matrix from body to world frame: $\begin{bmatrix} \cos \psi & -\sin \psi \\ \sin \psi & \cos \psi \end{bmatrix}$. The transpose of this matrix is the rotation matrix from world to body frame.
$x \in \mathbb{R}^7$	Vehicle's full state in the world frame: $x = [p_x, p_y, \psi, v_x, v_y, \omega, \delta]^\top$.
\dot{x}	Derivative of the world frame state.
$d(\mathbf{p})$	Signed distance from \mathbf{p} to a simple polygon geofence: the minimum Euclidean edge distance with sign from an orientation-independent even-odd (ray-crossing) inside test [28]; positive inside, negative outside.
$\mathbf{n}_{\text{out}}(\mathbf{p})$	Outward unit normal at \mathbf{p} : $\mathbf{n}_{\text{out}} = -\nabla d(\mathbf{p}) / \ \nabla d(\mathbf{p})\ $.
$h(x)$	State barrier defined as $h(x) = d(\mathbf{p})$.
\mathbf{x}	Vehicle's dynamic state in its body frame. The exact composition depends on the model. This composition could be, for example, $[v_x, v_y, \omega, \delta]^\top$.
$\dot{\mathbf{x}}$	Derivative of the body-frame state \mathbf{x} .
r	Parameters of physics models (e.g. for the bicycle model, $r = (\text{mass}, I_z, C_f, \dots)$).
α_f, α_r	Slip angles at the front and rear wheels.
F_{yf}, F_{yr}	Lateral tire forces at the front and rear wheels.
Control Model	
$\mathbf{u} \in \mathbb{R}^2$	Control input $\mathbf{u} = \begin{bmatrix} \dot{\delta} \\ F_x \end{bmatrix}$, where $\dot{\delta}$ is the commanded steering rate (rad/s) and F_x is the applied longitudinal force (N).
$f(\mathbf{x}, r)$ or $f(\mathbf{x})$	Drift vector field: the natural evolution of the system with parameters r when no control input is applied ($\mathbf{u} = 0$).
$g(\mathbf{x}, r)$ or $g(\mathbf{x})$	Control effectiveness matrix: maps each control input in \mathbf{u} to its influence on the state dynamics.
Control-affine model	A general nonlinear system structure where dynamics separate into a drift vector field and a control effectiveness matrix multiplying the control input \mathbf{u} : $\dot{x} = f(\cdot) + g(\cdot)\mathbf{u}$.
Neural Network	
θ	Learnable weights and biases of the neural network.
$NN_\theta(\cdot)$	Neural network parameterized by θ . Its specific inputs and outputs differ by architecture.
$\dot{\mathbf{x}}_{\text{pred}}$	Predicted state derivative.
$\dot{\mathbf{x}}_{\text{gt}} \in \mathbb{R}^4$	Ground truth state derivatives from real-world or simulated driving scenarios.
λ_{data}	Weight of the data loss term (hyperparameter).
λ_{phys}	Weight of the physics loss or regularization term (hyperparameter).

orientation evolve under steering and longitudinal actuation.

In this work, we model the body-frame dynamics; the global pose kinematics (position and heading) are handled separately.

We adopt the *dynamic bicycle model* as our representation of the vehicle dynamics. The bicycle model is a widely used reduced-order single-track abstraction that aggregates the left and right wheels of each axle into a single equivalent wheel, while preserving the essential lateral and longitudinal behavior relevant for control design [23], [24], [25]. This model provides a tractable yet expressive approximation of the vehicle's lateral-longitudinal dynamics for control and safety-critical applications. However, this simplified single-track representation remains an approximation. For low- to moderate-lateral-acceleration regimes, dynamic bicycle models have been shown to capture the dominant coupling between longitudinal, lateral, and yaw dynamics with sufficient accuracy for control-oriented modeling and prediction [23], [24], [25]. Beyond these nominal regimes, it neglects effects such as load transfer, additional body degrees of freedom, and coupled longitudinal-lateral tire dynamics that become important at higher speeds and in more aggressive transients [24], [25]. Comparative studies indicate that low-order kinematic or single-track models perform well mainly under such nominal conditions with limited lateral acceleration, whereas more detailed multi-DoF models that include, e.g., combined longitudinal-lateral tire effects and load transfer are preferable as lateral acceleration approaches the vehicle's handling limits or during rapid transient maneuvers [24], [25]. Recent residual-learning work further shows that simple 3-DoF physics-based models tend to drift in long-horizon state prediction, and that learning residual corrections on top of such a base model can substantially reduce prediction error [29]. Taken together, these observations motivate our choice to retain a low-order dynamic bicycle model for its tractability and analytic structure, while augmenting it with learned residual terms rather than replacing it with a high-dimensional physics model.

The resulting bicycle model captures the longitudinal, lateral, and yaw motions of the vehicle. We adopt the standard approximation that lateral tire forces depend on slip angle but are independent of the commanded longitudinal force F_x . Under this assumption, the dynamics can be expressed in control-affine form as

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}, \quad (2)$$

where $\mathbf{x} = [v_x, v_y, \omega, \delta]^\top$ is the dynamic state in the body frame and $\mathbf{u} = [\dot{\delta}, F_x]^\top$ is the control input consisting of steering rate and longitudinal force. We model the steering angle δ as a state, which makes the steering rate $\dot{\delta}$ the control input. This is a standard technique in vehicle handling and optimal-control formulations [55], [56], as it provides a simple way to capture steering-actuator dynamics and enforce steering-rate limits. In our setting, this choice also preserves the control-affine structure of the dynamics, which we will exploit in the controller design of Sec. IV-C.

Further, we define the slip angles and lateral tire forces for the bicycle model. The front and rear slip angles α_f, α_r are given by

$$\alpha_f = \text{atan2}(v_y + \ell_f \omega, v_{x,\text{safe}}) - \delta, \quad \alpha_r = \text{atan2}(v_y - \ell_r \omega, v_{x,\text{safe}}),$$

Table II: Comparison of models combining physics and machine learning (Physics-informed machine learning (PIML)) for dynamical systems. These models are illustrated primarily in the context of ODEs, where $t \in [0, T]$ is the time coordinate, $x(t)$ denotes the system's state vector, $\dot{x}(t)$ is its time derivative, $u(t)$ is the control function, and x_k denotes the measurement indexed by k . Moreover, we use r to denote the set of parameters in the governing physics model, and θ denotes the parameters in the neural networks.

Category	Domain Knowledge Requirement	Generalizable to Unseen Scenarios	Mathematical Formulation	Pros and Cons
Physics-Informed Neural Network (PINN) [30], [31], [32], [33], [34], [35], [36], [37] Networks trained with physics-based loss terms enforcing ODE/PDE constraints at collocation points sampled throughout the domain.	High Requires the explicit physics equations (e.g. the PDE/ODE).	Low The trained network learns the solution $x(t)$ for one specific instance (one set of fixed initial/boundary conditions and specific controls).	System: $\dot{x}(t) = f(x(t), u(t), t; r)$ (known physics) NN learns: $\hat{x}(t; \theta) \approx x(t)$ Inputs: $t, u(t)$, boundary/initial conditions Loss (physics): $\mathcal{L}_{\text{phys}}(\hat{x}(t_j; \theta), f(\hat{x}(t_j; \theta), u(t_j), t_j; r))$ Loss (data): $\mathcal{L}_{\text{data}}(\hat{x}(t_k; \theta), x_k)$ $\hat{\dot{x}}(t)$ is computed through auto differentiation.	Suitable for: Known physics constraints; limited data Advantages: Enforces known physics; data-efficient Limitations: Require known physics; new training required for unseen scenarios (e.g. new boundary conditions)
Surrogate Models [38] "Black-box" neural network approximates expensive physics simulations.	Low Requires a large dataset of input-output pairs. No explicit equations are needed.	Low Primarily used for interpolation within the training data domain.	System: $\hat{x}(t + \Delta t; \theta) = f_{\theta}(x(t), u(t), t)$ NN Learns: f_{θ} Inputs: $x(t), u(t), t$ Loss (data): $\mathcal{L}_{\text{data}}(x_{t+\Delta t}, f_{\theta}(x(t), u(t), t))$	Suitable for: Replacing expensive simulators; usually discrete (typical setup of simulators) Advantages: Fast surrogate approximations, require large training datasets Limitations: Poor generalizability/extrapolation beyond training scenarios; may violate physics
Differentiable Physics Simulations [39], [40] Differentiable physics simulators combined with neural networks for parameter, material, or control policy optimization through gradient-based learning.	High Requires explicit governing equations or a differentiable physics engine implementing f ; full model structure must be specified by the user.	Medium Low when calibrating parameters to a single experiment, but can be high when learning shared parameters or control policies across many operating conditions, within the range represented in the training data.	System: $\dot{x}(t) = f(x(t), u(t), t; r)$ NN learns: parameters r_{θ} (system identification) and/or a control policy $u_{\theta}(t)$ or $u_{\theta}(x(t), t)$ Inputs: current state $x(t)$ (and possibly t) to the policy network; initial conditions and model f to the simulator Loss (data): $\mathcal{L}_{\text{data}}(x_k, \hat{x}(t_k; p_{\theta}, u_{\theta}))$ where \hat{x} is obtained by integrating f with p_{θ} and u_{θ}	Suitable for: Parameter calibration, system identification, and optimal control when high-fidelity differentiable simulators are available. Advantages: Guarantees that trajectories follow the specified physics; enables end-to-end gradient-based optimization of parameters and control policies; leverages mature numerical solvers. Limitations: Requires accurate and differentiable physics models; computationally expensive due to repeated forward and backward simulations; cannot recover missing physics structure beyond f .
Physics-guided Models [41], [42], [43] Hybrid models that combine empirical neural networks with physics-based regularization terms and/or physics-derived features so that predictions respect known constraints or low-order mechanistic relationships.	Medium Requires partial physics knowledge (e.g. constraint function g , invariants, or a coarse mechanistic model), but not the full ODE/PDE.	Medium Physics constraints improve extrapolation compared to purely data-driven models, but performance is limited to regimes where the encoded physics remains valid.	System: $\dot{y} = f_{\theta}(x)$ NN learns: f_{θ} (the empirical mapping) Inputs: features x Loss (physics): $\mathcal{L}_{\text{phys}}(g(f_{\theta}(x), x))$ Loss (data): $\mathcal{L}_{\text{data}}(y, f_{\theta}(x))$	Suitable for: Systems with partially known physics or reliable constraints where full governing equations are unavailable or too complex. Advantages: Uses physics to regularize the hypothesis space; can reduce data needs; improves interpretability and physical plausibility; flexible to incorporate diverse constraint types. Limitations: Requires expert specification of constraints or mechanistic terms; misspecified g can bias learning; still relies on training data coverage and may extrapolate poorly outside the range where the physics assumptions hold.
Neural ODEs [44], [45], [46] Neural networks parameterize the right-hand side of an ODE, and a numerical ODE solver is used as part of the computational graph so that continuous-time dynamics are learned directly from data.	Low Does not require explicit physics equations; only observed trajectories are needed, although domain knowledge can be encoded via architecture or regularization.	Medium Generalizes well to new initial conditions and time grids within the state and control regimes seen during training; extrapolation to qualitatively new regimes is limited.	System: $\hat{\dot{x}}(t; \theta) = f_{\theta}(x(t), u(t), t)$ NN learns: f_{θ} (the dynamics) Inputs: $u(t), t$, boundary/initial conditions Loss (data): $\mathcal{L}_{\text{data}}(x(t), \hat{x}(t; \theta))$ \hat{x} is typically obtained by integrating the derivative $\hat{\dot{x}}$ through a numerical ODE solver (RK4, Dormand–Prince, adaptive solvers, etc.)	Suitable for: Modeling continuous-time dynamics and irregularly sampled time series; learning latent dynamical systems. Advantages: Handles variable-step integration and irregular sampling; parameter sharing over time; can be combined with latent-variable models for complex systems. Limitations: Training can be computationally expensive due to repeated ODE solves; sensitivity to stiff dynamics and solver choices; does not automatically enforce physical constraints unless additional structure or penalties are introduced.
Residual Models [47], [48], [49], [50] Neural networks augment a known physics model by learning residual corrections to account for model mismatch, unmodeled effects, or empirical adjustments.	Medium Requires a baseline physics model f that captures coarse dynamics; residual network is used to correct remaining discrepancies.	High Can generalize well across operating conditions where the baseline model remains qualitatively valid, since the main structure is provided by f and the NN only learns corrections.	System: $\hat{\dot{x}}(t; \theta) = f(x(t), u(t), t; r) + \Delta f_{\theta}(x(t), u(t), t)$ NN learns: Δf_{θ} (residual term) that models the discrepancy between the baseline physics and observed dynamics Inputs: $x(t), u(t), t$, initial/boundary conditions Loss (data): $\mathcal{L}_{\text{data}}(x_k, \hat{x}(t_k; \theta))$	Suitable for: Improving forecasts when a trusted but imperfect physics model exists. Advantages: Preserves interpretability and structure of the baseline model; focuses learning capacity on systematic errors; often improves extrapolation relative to purely data-driven models. Limitations: Performance depends on the quality of the baseline model; residual may absorb structural errors and become hard to interpret; extrapolation fails if the underlying physics model is invalid in new regimes.
Operator Learning Models [51], [52], [53] Networks learn a solution operator that maps problem definitions (for example parameters, forcing terms, and boundary or initial conditions) to full trajectories or fields, rather than predicting state evolution step by step.	Medium Requires specifying a family of ODE/PDE problems, including the type of equation and ranges of parameters and conditions used to generate training data, but not necessarily the closed-form operator.	High Once trained on a distribution of parameters and conditions, the learned operator can generalize to unseen configurations within that distribution, enabling rapid evaluation for new scenarios.	Formulation: $x(t) = \mathcal{G}(u(\cdot), r, t)$, NN learns: $\hat{x}(t; \theta) = \mathcal{G}_{\theta}(u(\cdot), r, t)$ Inputs: discretized or encoded forcing/control functions $u(\cdot)$, problem parameters r , initial/boundary conditions, and query times t Loss (data): $\mathcal{L}_{\text{data}}(x_k, \hat{x}(t_k; \theta))$ (often integrated over space and time for PDEs)	Suitable for: Fast emulation of families of physics problems, rapid evaluation under varying parameters or conditions, and surrogate modeling when many training solves are available offline. Advantages: Learns global mappings from problem setup to solution; amortizes the cost of expensive solvers over many queries; well suited for real-time or many-query scenarios. Limitations: Requires large and diverse training datasets that cover the parameter and condition space; extrapolation outside the training distribution is unreliable; still depends on the correctness of the simulator or data source used for training.
Data-driven Discovery [54] Methods that infer explicit governing equations directly from data, often via sparse regression or symbolic neural approaches, to recover interpretable dynamical models.	High Requires designing a candidate library of basis functions (for example polynomials, nonlinear interactions, control terms) and appropriate sparsity or regularization assumptions; physics structure is then inferred from data.	Medium Discovered equations can generalize within the regime represented in the data, but reliability depends strongly on data quality, noise levels, and whether the true dynamics lie in the chosen function library.	Formulation: $\hat{\dot{x}}(t) = \Theta(x(t), u(t), t) \xi$ where $\Theta(\cdot)$ is a library of candidate nonlinear terms evaluated from data and ξ is a coefficient vector. NN / optimizer learns: a sparse coefficient vector ξ (and possibly the structure of Θ) that selects which terms form the governing equations. Inputs: measured trajectories $x(t)$, controls $u(t)$, and estimated derivatives $\dot{x}(t)$ (or smoothed approximations) User sets: the candidate function library Θ , sparsity-promoting penalties, and selection thresholds Loss (data + sparsity): $\mathcal{L}(\hat{\dot{x}}, \Theta(x, u, t) \xi) + \lambda \ \xi\ _1$ (or related sparse model-selection objectives)	Suitable for: Hypothesis generation, discovering compact analytic models from experimental or simulation data, and gaining mechanistic insight into previously unknown dynamics. Advantages: Produces interpretable equations rather than only black-box predictors; can reveal dominant physical mechanisms; useful when explicit models are unavailable or uncertain. Limitations: Sensitive to noise, differentiation errors, and the choice of candidate library; often requires dense, high-quality data; less useful when governing equations are already well established and trusted.

where ℓ_f, ℓ_r are the distances from the center of gravity to the front and rear axles, and $v_{x,\text{safe}}$ is a safeguarded longitudinal velocity defined as $v_{x,\text{safe}} = \text{sign}(v_x) \max(|v_x|, \varepsilon)$ with $\varepsilon > 0$ and $\text{sign}(0) = +1$. We compute lateral forces using the Pacejka Magic Formula (lateral-only) with the convention that positive slip yields negative lateral force [57] (parameters μ friction, C shape, E curvature):

$$F_y(\alpha) = -D \sin\left(C \arctan\left(B\alpha - E(B\alpha - \arctan(B\alpha))\right)\right) \quad (3)$$

Consistent with our implementation, we set the per-axle peak factor to

$$D = \mu F_z, \quad F_z = \frac{mg}{2} \text{ (where } g \text{ is gravitational acceleration),}$$

and derive front/rear stiffness factors from identified cornering stiffnesses (C_f, C_r ; front/rear cornering stiffness, N/rad):

$$B_f = \frac{C_f}{C D}, \quad B_r = \frac{C_r}{C D}. \quad (4)$$

The natural evolution of the dynamics without control (i.e., the drift vector field) is

$$f(\mathbf{x}) = \begin{bmatrix} -\frac{F_{yf} \sin \delta}{I_z} + v_y \omega \\ \frac{F_{yr} + F_{yf} m \cos \delta}{I_z} - v_x \omega \\ \frac{\ell_f F_{yf} m \cos \delta - \ell_r F_{yr}}{I_z} \\ 0 \end{bmatrix}, \quad (5)$$

where m is the vehicle mass and I_z is the yaw inertia. The contribution of the control input \mathbf{u} (i.e., the control effectiveness matrix) is expressed as

$$g(\mathbf{x}, r) = \begin{bmatrix} 0 & \frac{\cos \delta}{m} \\ 0 & \frac{\sin \delta}{m} \\ 0 & \frac{\ell_f \sin \delta}{I_z} \\ 1 & 0 \end{bmatrix}. \quad (6)$$

This formulation ensures that the steering rate $\dot{\delta}$ is directly controlled, while the longitudinal force F_x is projected through the current steering angle δ , influencing both lateral motion and yaw dynamics, assuming F_x acts at the front axle along the wheel heading.

2) *Physics-informed machine learning*: Beyond purely analytical dynamics, a growing body of work explores PIML as a means to reconcile first-principles modeling with data-driven learning. The core objective is to leverage known physical structure, such as governing equations and kinematic constraints, while utilizing machine learning to capture unmodeled or highly nonlinear phenomena that analytical models often fail to represent.

Table II provides a taxonomy of prominent PIML families. These range from PINNs, which embed physical laws directly into the loss function as constraints for a data-driven Partial Differential Equation (PDE)/ODE solver, to surrogate models that approximate expensive simulators, and hybrid residual approaches that learn corrections to a nominal physics model. Other categories include physics-regularized networks, Neural ODEs that integrate neural networks within continuous-time solvers, differentiable simulators for parameter identification, and operator-learning frameworks that approximate solution mappings.

These approaches differ fundamentally in their reliance on domain knowledge, their generalization capabilities, and the extent to which they preserve the underlying system structure. For instance, purely data-driven surrogate models maximize computational efficiency but often fail to generalize outside their training distribution. Conversely, operator learning can generalize across broad families of PDEs but typically demands large, diverse datasets.

In this work, we adopt a *hybrid residual* strategy rooted in the dynamic bicycle model (Sec. II-B1). Specifically, we design a structure-preserving architecture that retains the control-affine form of the dynamics while learning residual corrections from data. This approach balances physical interpretability with data-driven fidelity, ensuring the resulting model remains compatible with the safety-critical constraints required for geofence enforcement.

C. Control models for geofence enforcement

Given the vehicle dynamics defined above, we now describe two control models for geofencing intervention: a reactive braking-only model based on Time-To-Collision (TTC), and an active approach that exploits both braking and steering.

1) *Time-To-Collision*: The TTC method represents a conservative, braking-only baseline. We reference the standard notion of time-to-collision from automotive practice (e.g. ISO 15623) and its origins and engineering use in collision-avoidance systems [58], [59], [60], while here evaluating safety via a max-braking rollout rather than a constant-relative-speed TTC formula. This simulation of a fail-safe maneuver is conceptually related to formal verification methods that use reachability analysis to compute forward-reachable sets (often including designated fail-safe trajectories to a standstill) and verify whether a safe state can be maintained [61], [62].

It determines whether a safe stop can be achieved from the current state by applying a fixed Emergency Braking (EB) control policy held constant during the rollout (zero steering rate, maximum-magnitude braking) $\mathbf{u}_{\text{EB}} = \begin{bmatrix} 0 \\ F_{x,\text{min}} \end{bmatrix}$, where $F_{x,\text{min}}$ is the most negative admissible longitudinal force (maximum-magnitude braking). A forward rollout is computed by numerically integrating the dynamics on a fixed grid $t = 0, 0.1, \dots, 5.0$ [s] under \mathbf{u}_{EB} , terminating at the first index k_{stop} where the body-frame forward velocity $v_x(k) \leq 0$. If there exists any sampled step $k \leq k_{\text{stop}}$ such that $d(\mathbf{p}_k) < -\varepsilon$ with $\mathbf{p}_k = [p_{x,k}, p_{y,k}]^\top$ and $\varepsilon = 0.5$ m (boundary tolerance), the current state is deemed unsafe. This binary assessment captures whether an emergency stop is sufficient to guarantee containment. This approach is deliberately conservative and may lead to unnecessary stops or false interventions, a known issue with braking-only and other conservative geofencing methods that can trigger needless intervening actions or excessively restrict the operational area [6]. It is important to note that this “braking-only rollout” is closely aligned with classical TTC-based safety evaluation [58], [59], [60], even though we implement it via a forward max-braking rollout rather than a closed-form constant-relative-speed TTC formula. In parallel, a separate line of modern work investigates the direct use of time-based indicators such as TTC (or its

inverse) as the safety constraint $h(x)$ within a CBF-based optimization framework for longitudinal collision avoidance and traffic control [63]. These time-based CBFs are particularly powerful for car-following scenarios, whereas our controller adopts the classical rollout as a conservative baseline and focuses on enforcing safety via a geometric signed-distance barrier $h(x) = d(\mathbf{p})$, which is more naturally suited to arbitrary polygonal keep-in constraints.

2) *Discrete Control Barrier Function*: Let $d(\mathbf{p})$ be the geometric signed distance (positive inside), and define the barrier $h(x) := d(\mathbf{p})$ where $\mathbf{p} = [p_x, p_y]^\top$ is extracted from the world-frame state x . For continuous-time control-affine dynamics, a standard (exponential) CBF condition enforces

$$\dot{h}(x) + \kappa h(x) \geq 0,$$

where $\kappa > 0$ sets the convergence rate to the safe set.

While this condition defines safety, it does not specify the control law. The seminal work of [64] established the Control Barrier Function-based Quadratic Program (CBF-QP) as the de facto framework for operationalizing this guarantee. In this paradigm, the CBF condition is formulated as an affine inequality constraint on the control input \mathbf{u} and embedded in a real-time QP that mediates safety with a performance objective (e.g. a Control Lyapunov Function or a min-deviation term) [64]. Following the CBF-QP safety-filter template, and *departing from the usual CLF-soft/CBF-hard convention in [64]*, we adopt a *heavily penalized CBF slack* in discrete time to preserve feasibility under preview linearization and model mismatch; when $\xi = 0$, we recover strict CBF enforcement.

While classical CBFs are posed in continuous time, real implementations sample the dynamics at fixed intervals. We therefore adopt a discrete CBF formulation that acts on the numerically integrated next-state map $x_{k+1} = \Phi(x_k, \mathbf{u}_k)$. We use the widely adopted discrete exponential surrogate:

$$h(x_{k+1}) \geq e^{-\kappa \Delta t} h(x_k) =: \beta_k, \quad (7)$$

with $1 - \gamma = e^{-\kappa \Delta t}$ (cf. the discrete exponential surrogate $\Delta h \geq -\gamma h$ in [19]). With a fixed preview horizon t_h , we parameterize $\kappa = -\ln(1 - \gamma)/t_h$ so the terminal target $\beta(t_h)$ used in Sec. IV-C is consistent with (7).

Notation. We use $h(x) = d(\mathbf{p})$ with safe set $\{x : h(x) \geq 0\}$; the discrete contraction factor is $\gamma \in [0, 1)$ and the preview horizon is t_h ; the terminal target is $\beta(t_h) = \max\{h_{\text{target}}, h(x_0)e^{-\kappa t_h}\}$ with $\kappa = -\ln(1 - \gamma)/t_h$.

D. Numerical integration

The vehicle dynamics are specified in continuous time as an ODE, but controllers and safety checks operate in discrete time. Numerical integration bridges this gap by approximating the evolution $\dot{\mathbf{x}} = f(\mathbf{x}, r) + g(\mathbf{x}, r)\mathbf{u}$ with a discrete map $\mathbf{x}_{k+1} = \Phi(\mathbf{x}_k, \mathbf{u}_k)$ under Zero-Order Hold (ZOH) inputs. A *rollout* is obtained by repeatedly applying this update, starting from an initial state, to generate a trajectory $\{x_0, x_1, \dots, x_K\}$ over a horizon.

Depending on the task, integration must balance *accuracy* (e.g. for training and validation) and *speed/stability* (e.g. for real-time safety checks). To accommodate both needs we employ two complementary one-step schemes:

a) *Classical Runge–Kutta (RK4)*: A fourth-order method that achieves high accuracy by averaging four slope evaluations per step. It is used when fidelity and determinism are paramount, such as in training rollouts or stopping-distance safety checks.

b) *Semi-implicit (symplectic) Euler*: A first-order scheme, but qualitatively more stable than Forward Euler for mechanical states partitioned into positions and velocities. Its low cost makes it well suited to the dense, short-horizon rollouts needed for predictive control barrier functions. This trade-off mirrors practice in automotive simulation, where specially adapted semi-implicit Euler schemes are used to achieve stable, real-time-capable vehicle dynamics in Hardware-in-the-Loop (HIL) test stands [65]. Further background on the accuracy, stability, and error properties of these one-step schemes can be found in standard numerical analysis texts [66], [67], [68].

III. RELATED WORK

Current geofencing strategies generally fall into geometric, reachability-based, or control-theoretic categories. Geometric methods, such as the Anticipatory Range Control (ARC) approach in [69], construct turning-circle envelopes to handle acute polygon corners, but, although originally developed for UAVs, typically assume constant turning radii that neglect vehicle-dynamics effects such as sideslip or tire slip when applied to ground vehicles. To guarantee containment under dynamic constraints, a Model Predictive Geofencing scheme based on Backward Reachable Tubes (BRTs) is proposed in [6], where the BRT for a given target set and dynamics is computed once offline and stored as a discretized tensor, yielding $\mathcal{O}(1)$ online execution but tying the safety guarantees to fixed model parameters (e.g., mass, surface friction, and braking limits). Closure-rate constraints for geofence violation prevention are derived in [1] to ensure stopping distances are respected near a geofence, while an Explicit Reference Governor (ERG) is employed in [70] to adjust trajectory references upstream of the controller so that position and velocity constraints are never violated; however, such formally verified or ERG-based designs often require conservative braking profiles and linearized dynamics, which can limit operational efficiency or degrade fidelity at higher speeds.

On the other hand, real-time safety filters based on CBFs and reachability have been developed for high-performance aerial vehicles. Implicit backup sets are used in conjunction with CBF-based filtering in [71] to provide an onboard safety monitor for high-speed racing drones, and backstepping-based and high-order CBFs for cascaded fixed-wing dynamics enabling simultaneous collision avoidance and geofencing for a Dubins-type aircraft model are designed in [72]. Probabilistic extensions also exist; for example, [73] casts human motion prediction as a Hamilton–Jacobi reachability problem in a joint human–belief state space and computes belief-augmented forward reachable sets using grid-based HJ PDE solvers, although the focus there is on human motion rather than vehicle dynamics. Taken together, these approaches span geometric heuristics, precomputed reachable tubes and sets, and analytically derived barrier or governor conditions tailored to simplified models, highlighting a trade-off between computational tractability,

model fidelity, and the ease with which safety constraints can be enforced in real time.

a) CBF controller design: Digital controllers face the “inter-sample gap”: a system verified to be safe at discrete time steps may still violate constraints between samples as the physical system evolves continuously. Under ZOH implementation over a finite sampling interval, naively applying continuous-time CBF designs can therefore create both inter-sample safety gaps and modeling gaps between exact and numerically integrated discrete-time maps [20], [21]. Extending CBFs to discrete time on a general nonlinear next-state map Φ can also yield nonconvex synthesis problems; under exponential DT-CBF conditions, the problem becomes a QCQP [19]. For control-affine systems with mild block structure, DT-CBF invariance conditions that are affine in the input restore tractability and admit Boolean/piecewise-safe-set compositions via mixed-integer encodings [74]. Sampled-data CBF (SD-CBF) formulations explicitly address ZOH implementations: [20] formalize practical safety and show how SD-CBFs can be synthesized on approximate Runge–Kutta maps with consistency guarantees, while [21] derive ZOH-aware, control-affine conditions that reduce conservativeness and embed cleanly in QPs. Complementary zero-order formulations enforce safety directly on the numerically integrated next-sample map under ZOH, sidestepping Lie-derivative bookkeeping and accommodating high relative degree and state–input-dependent constraints [22], and high-order/adaptive DT-CBFs provide additional tools for feasibility under tightening bounds [75]. For geofence enforcement under digital control, these discrete and sampled-data formulations identify how safety constraints should be posed on the numerically integrated map so that constraint sets remain invariant despite sampling and modeling errors.

Position-based safety constraints (such as keeping a vehicle within a geofence) typically have high relative degree: actuation affects acceleration or curvature, and position only changes after integrating velocity and acceleration. For such high-relative-degree constraints, continuous-time CBF designs introduce additional structure so that the input appears in the barrier condition. Early work used *backstepping/dynamic extension* to handle relative degree > 1 within a CBF–CLF–QP framework [76]. *Exponential CBFs (ECBFs)* provide a *systematic, pole-placement* recipe to handle *arbitrary* relative-degree constraints and embed them in QPs [77]. Alternatively, sampled-data approaches enforce barrier conditions on a previewed terminal state obtained by numerically integrating the dynamics over one sampling interval. By constraining the discrete barrier difference at that terminal state, these methods implicitly resolve high-relative-degree effects and can accommodate state–input-dependent constraints without recursive Lie-derivative constructions [22].

b) Geofence barrier parameterization and keep-in enforcement: A fundamental distinction in safety filtering is the definition of the hazard. Two commonly adopted parameterizations are geometric-based and time-based. An extensive body of work uses time-based quantities such as time headway or time-to-collision as the core safety constraint $h(x)$ within CBF-based longitudinal controllers for connected and automated

vehicles [63]. However, time-based metrics are ill-posed for general 2D confinement, as the “time to collision” varies discontinuously with steering. Thus, spatial enforcement typically relies on geometric signed-distance barriers $h(x) = d(\mathbf{p})$. Such time-based CBFs are highly effective for one-dimensional car-following and traffic scenarios, but are less directly applicable to maintaining invariance within a general two-dimensional polygonal region. For geofencing and related 2D confinement problems, this motivates the use of signed-distance-based barriers that directly encode the margin to polygonal boundaries.

Greedy, single-step safety filters suffer from myopia, “potentially steering the system into a dead end” (inevitable collision) that is not immediately visible without a prediction horizon. *Predictive safety filters* use short-horizon optimization with a *terminal safe set*. *MPC–CBF* formulations enforce *DT–CBF* constraints *across the horizon*, powerful but requiring a *finite-horizon* (generally nonlinear) program at each cycle [78]. *Predictive Barrier Functions* introduce an *always-feasible soft-constrained* auxiliary problem with a *terminal CBF* and show its value function is itself a *predictive CBF*, yielding a *recovery mechanism* [79].

c) PIML models for vehicle dynamics and control: Neural network vehicle models have been embedded in feedback control for high-performance driving. [80] replaces a simple physics-based model in a feedforward–feedback architecture with a neural model that takes a short history of past states and inputs as input, trained on real driving data, achieving human-competitive path tracking at the friction limit and implicitly adapting to changing road surfaces without explicit friction estimation. [81] estimates tire, drivetrain, and inertia parameters of a single-track racecar model using a physics-constrained network with a guard layer that restricts coefficients to physically meaningful ranges, while [82] use a sparse Gaussian-process residual on top of a nominal bicycle model inside an MPC controller, improving lap times and accounting for model uncertainty. [83] and [29] further develop residual-correction frameworks in which a deep encoder with an SVGP (DRF) or a Transformer (DyTR) learns trajectory- or state-prediction residuals on top of baseline vehicle models, substantially reducing prediction error, and [84] combine a mechanism-based estimator with an LSTM for lateral-velocity estimation that transfers across vehicles. [85] embed a Pacejka tire model and a dynamic bicycle model as differentiable layers inside a neural network, using latent tire-force features from the physics-embedded model to drive a risk-aware MPC that adapts to unknown friction. Overall, these works target high-fidelity prediction or racing performance, but their learned components typically act as generic residuals or parameter correctors and do not enforce an explicit control architecture, such as control-affine structures. This limitation hinders the design of verifiable safety-critical controllers, as the highly nonlinear dependency on the control input complicates the derivation of formal stability or safety guarantees.

PIML have also been proposed to explicitly adapt to control architectures. [86] replace an AGV kinematic model inside NMPC with a PINN that enforces ODE residuals in the loss, and [87] introduce PINC, a PINN-style architecture that supports control inputs and long-horizon simulation for

nonlinear benchmark systems. In parallel, control-structure-aware neural identifiers explicitly align network structure with controller needs: CA-NNARX models are surrogate neural networks parameterized to be affine in the control input and come with δ ISS conditions enabling efficient IMC design [88], and stable GRU models have been trained for system identification and embedded in nonlinear MPC with offset-free tracking guarantees [89].

IV. METHOD

This section presents our geofencing pipeline. The design follows two principles: (i) a hybrid residual dynamics model that preserves a *control-affine* structure to enable CBF-QP-style safety filtering, and (ii) a sampled-data, *preview-discrete* Discrete Control Barrier Function controller that delivers minimal, stability-aware interventions. We first relate the background to the concrete geofencing task, then detail the model, the data-driven parameter refinement, and the controller, and finally summarize the full pipeline in algorithmic form.

A. Overview

The background (vehicle dynamics, barrier functions, and numerical integration) is instantiated for geofence enforcement via a two-stage architecture:

Predictive layer (model-based preview). From the current world-frame state $x = [p_x, p_y, \psi, v_{x,w}, v_{y,w}, \omega, \delta]^\top$, we generate short-horizon rollouts under candidate constant inputs $\mathbf{u} = [\dot{\delta}, F_x]^\top$ using the body-frame bicycle dynamics $\dot{\mathbf{x}} = f(\mathbf{x}, r) + g(\mathbf{x}, r)\mathbf{u}$ with Zero-Order Hold. We integrate with a semi-implicit (symplectic) Euler preview scheme with a few substeps (for stability and speed); high-fidelity rollouts (e.g. for validation) use RK4 (cf. Sec. II-D). Body-frame accelerations are mapped to the world frame to update the state, and barriers are evaluated on the world position $\mathbf{p} = [p_x, p_y]^\top$.

The preview evaluates terminal barrier at the end of the horizon via the signed-distance barrier $h(x) = d(\mathbf{p})$ (see Sec. IV-C), using the target $\beta(t_h)$ from Eq. (10), consistent with Eq. (7). To make the sensitivity well-posed near actuator limits, steering-rate sensitivities are computed by finite differences with $\dot{\delta}$ steps *clamped* to bounds, and the longitudinal-force column uses a bound-secant toward *full braking* $F_x = u_{\min}$.

Control layer (minimal intervention). Given the preview, we enforce the terminal constraint $h(\Phi_{t_h}(x_k, \mathbf{u}_k)) \geq \beta(t_h)$ by solving a small 2-variable QP in $[\dot{\delta}, F_x]$ that minimally perturbs a nominal input \mathbf{u}^{nom} . We linearize the terminal constraint about $\mathbf{u}_k^{\text{nom}}$ using the sensitivity matrix $\mathbf{J}_k \approx \partial(h \circ \Phi_{t_h})/\partial \mathbf{u}$:

$$h(\Phi_{t_h}(x_k, \mathbf{u}_k)) \approx h_{\text{nom}} + \mathbf{J}_k (\mathbf{u}_k - \mathbf{u}_k^{\text{nom}}),$$

where $h_{\text{nom}} := h(\Phi_{t_h}(x_k, \mathbf{u}_k^{\text{nom}}))$. Including a slack variable $\xi \geq 0$ for soft-constraint feasibility, the constraint becomes:

$$h_{\text{nom}} + \mathbf{J}_k (\mathbf{u}_k - \mathbf{u}_k^{\text{nom}}) + \xi \geq \beta(t_h).$$

We write this in the standard affine form $\mathbf{A}\mathbf{u}_k + \xi \geq \mathbf{b}$, with:

$$\mathbf{A} = \mathbf{J}_k, \quad \mathbf{b} = \beta(t_h) - h_{\text{nom}} + \mathbf{J}_k \mathbf{u}_k^{\text{nom}}.$$

Minimal intervention is enforced by (i) an early exit when the terminal target is already satisfied (Algorithm 2), and (ii)

an ℓ_2 -closest control to \mathbf{u}^{nom} subject to linearized safety and actuator box constraints (Eq. (15)). This QP instantiates the CBF-QP framework [64] in a discrete-time, preview-terminal setting: unlike the common continuous-time setup (hard CBF, soft CLF), we use a proximal objective around \mathbf{u}^{nom} and a *heavily penalized slack* on the CBF rows to preserve feasibility, together with actuator box constraints.

Only the control layer has actuator authority; the predictive layer proposes but does not actuate.

Compared with (i) TTC/EB-only baselines that brake even when a gentle steer suffices, and (ii) continuous-time CBFs that require higher-order Lie derivatives and are brittle under sampling and saturation, our *preview-discrete* DCBF operates on the numerically integrated map, uses the target $\beta(t_h)$, and builds saturation-aware Jacobians—improving feasibility and reducing unnecessary stops.

B. Predictive layer: PCARNN for modeling vehicle dynamics

We retain a control-affine structure in the body frame (cf. 2), but f and g are represented by a combination of physics and neural networks. More specifically, to preserve the control-affine form, PCARNN separates learned corrections into drift and gain terms:

$$f(\mathbf{x}) = f_{\text{phys}}(\mathbf{x}) + \Delta f_{\text{NN}}(\mathbf{x}), \quad (8)$$

$$g(\mathbf{x}) = g_{\text{phys}}(\mathbf{x}) + \Delta g_{\text{NN}}(\mathbf{x}), \quad (9)$$

where $f_{\text{phys}}, g_{\text{phys}}$ are the analytical bicycle terms, while the neural heads Δf_{NN} and Δg_{NN} learn structured residuals that capture tire nonlinearities, load transfer, and parametric drift. We *zero-initialize* Δg_{NN} and bind the last row of g_{total} to $[1 \ 0]$ so that $\dot{\delta}$ remains directly actuated and the model always stays control-affine in $(\dot{\delta}, F_x)$.

Novelty (dynamics). In the context of polygonal geofencing with sampled-data safety filters, we instantiate a physics-encoded, control-affine residual architecture with three key properties: (i) it is *hybrid residual*—data corrects an analytic dynamic-bicycle model instead of replacing it; (ii) it is *explicitly control-affine* in $\mathbf{u} = [\dot{\delta}, F_x]^\top$, so that the safety constraint's sensitivity \mathbf{J}_k remains numerically stable and the resulting QP-based controller (Eq. (15)) stays as a small, two-variable problem that behaves reliably in the sampled-data CBF setting [19], [74], [22]; and (iii) it is *gracefully degradable*: setting $\Delta f_{\text{NN}} = \mathbf{0}, \Delta g_{\text{NN}} = \mathbf{0}$ reverts to the physics baseline. Unlike generic PIML vehicle models and black-box control-affine identifiers used in model-based control, this PCARNN keeps an analytic dynamic-bicycle backbone with interpretable parameters and explicit steering-rate and braking channels, tailored to the geofence safety filter.¹

While generic learned models (e.g. Neural ODEs) yield a nonlinear next-state map $\Phi(x, \mathbf{u})$ that can make safety filtering ill-conditioned or intractable, the proposed PCARNN preserves separability of f and g (Eqs. (6)–(7)), ensuring the terminal-barrier linearization (Eq. (15)) defines a small, well-posed QP suitable for fast, tractable enforcement.

¹See Sec. III for a detailed comparison with data-driven vehicle models and control-affine neural identifiers.

Algorithm 1 Dynamic model calibration with data-driven parameter refinement

```

1: Inputs: trajectory dataset  $\mathcal{D} = \{(\mathbf{x}_t, \mathbf{u}_t, \dot{\mathbf{x}}_t)\}_{t=1}^N$ ; measurements
   (mass, geometry); tire priors  $(C_f, C_r, C, E)$ ; training horizon  $T$ 
2: Outputs: calibrated parameters  $p^* = \{C_f^*, C_r^*, C^*, E^*\}$ 
3: Initialize: set nominal  $(C_f, C_r, C, E)$  and measured geometry/mass to form  $p$ 
4: for epoch = 1, 2, ... do
5:   for mini-batch  $(\mathbf{x}, \mathbf{u}, \dot{\mathbf{x}}) \subset \mathcal{D}$  do
6:     Predict  $\dot{\mathbf{x}}_{\text{pred}} = f_{\text{phys}}(\mathbf{x}; p) + g_{\text{phys}}(\mathbf{x}; p) \mathbf{u}$ 
7:     Loss (L2)  $\mathcal{L} = \|\dot{\mathbf{x}}_{\text{pred}} - \dot{\mathbf{x}}\|_2^2$ 
8:     Update apply Adam [90] to  $\log C_f, \log C_r, \log C$  and to  $E$ 
9:   Validate held-out rollouts of length  $T$  (RK4); early-stop on forecast and barrier-prediction error
10: return  $p^*$ 

```

For integration, we use RK4 for high-fidelity training rollouts and semi-implicit Euler for fast preview inside the controller; positions and velocities are consistently mapped between body and world frames using $R(\psi)$.

Accurate physics terms depend on parameters p (mass m , inertia I_z , axle distances ℓ_f, ℓ_r , tire stiffnesses C_f, C_r , friction μ , etc.). We combine measurement, formula-based derivation, literature priors, and a targeted data-driven refinement loop that optimizes primarily the tire parameters while keeping the structure in Eq. (2) intact. This preserves control-affinity and improves preview accuracy near handling limits.

Novelty (calibration). The physics parameters and residual heads are *co-trained*, but under architectural constraints that keep the dynamics in the control-affine form required by the safety filter. This geofencing-oriented co-calibration improves preview accuracy (especially in tire-limited regimes) while preserving the structure needed for tractable, safety-filter-compatible control.

The neural network residual heads, Δf_{NN} and Δg_{NN} , are implemented as feedforward MLPs that take the current body-frame dynamic state \mathbf{x} as input. Following modern deep learning practice, we use the SiLU (Sigmoid-weighted Linear Unit) activation in the hidden layers [91], [92]. The exact network architectures (depth, width) and whether parameters for f and g are shared or split are treated as hyperparameters; the specific configurations used in our experiments are reported in Sec. V.

C. Control layer: Discrete Control Barrier Function for geofence enforcement

(i) *Terminal preview and schedule.* We evaluate the barrier at a fixed preview time $\tau = t_h$ by integrating the bicycle model with semi-implicit Euler (constant \mathbf{u}) and n_{sub} substeps: $x(\tau) = \Phi_\tau(x_0, \mathbf{u})$. We enforce a target schedule

$$\beta(\tau) = \max\{h_{\text{target}}, h(x_0)e^{-\kappa\tau}\}, \kappa = \frac{-\ln(1-\gamma)}{t_h}, \quad (10)$$

so that $h(x(\tau)) \geq \beta(\tau)$ guarantees a monotonic approach toward a positive margin h_{target} (cf. the discrete exponential DCLF-DCBF surrogate $h_{k+1} \geq (1-\gamma)h_k$ in [19]). In all experiments, unless noted otherwise, we use $t_h = 0.30$ s,

$n_{\text{sub}} = 3$, $\varepsilon_{\delta} = 0.25$ rad/s, a longitudinal bound-secant toward full braking, $\gamma = 0.4$, and clip sensitivity magnitudes at $\Gamma_{\text{clip}} = 10^6$.

a) *On high relative degree.* By enforcing the barrier at a finite preview time τ , the current input \mathbf{u} directly influences the terminal state $x(\tau)$ under zero-order hold, and thus $h(x(\tau))$. Classical continuous-time CBF constructions handle high-relative-degree position constraints by introducing chains of Lie derivatives via backstepping/dynamic extension [76] or pole-placement formulations such as Exponential CBFs [77]. In contrast, we follow the sampled-data / zero-order CBF viewpoint [19], [20], [21], [22]: rather than enforcing conditions on continuous-time derivatives, we act directly on the numerically integrated preview map $h \circ \Phi_\tau$ and impose a discrete-time inequality on this map. This avoids recursive high-order CBF constructions and yields a terminal barrier condition that is *locally* well-approximated as affine in \mathbf{u} when combined with our control-affine PCARNN dynamics. The resulting constraint enters the QP as a single linear row (Eq. (15)), with the polygonal signed-distance barrier, its speed-aware extension, and the saturation-aware sensitivities from the preview-rollout linearization tailoring the sampled-data condition to the polygonal geofencing and actuator-limited setting.

(ii) *Polygonal SDF with inward normal.* Let $d(\mathbf{p})$ be the signed distance to a simple polygon $\Omega \subset \mathbb{R}^2$, positive inside. The inward unit normal is $\mathbf{n}_{\text{in}}(\mathbf{p}) = \nabla_{\mathbf{p}} d(\mathbf{p})$. Since the state barrier is $h(x) = d(\mathbf{p})$ with $\mathbf{p} = [p_x, p_y]^\top$ and $x = [p_x, p_y, \psi, v_{x,w}, v_{y,w}, \omega, \delta]^\top$, the state-gradient is

$$\nabla_x h(x) = [\nabla_{\mathbf{p}} d(\mathbf{p})^\top \quad \mathbf{0}_{1 \times 5}].$$

(iii) *Input-space linearization by preview rollouts.* We build the *barrier sensitivity matrix* $\mathbf{J} = [J_{h,\delta} \quad J_{h,F_x}]$ at the preview time $\tau = t_h$ using short, zero-order-hold rollouts of the numerically integrated map Φ_τ . Let $\mathbf{u}_{\text{nom}} = [\delta_{\text{nom}}, F_{x,\text{nom}}]^\top$. Define the elementwise clipping operator $\text{clip}(a; a_{\text{min}}, a_{\text{max}}) = \min\{\max\{a, a_{\text{min}}\}, a_{\text{max}}\}$ and the steering perturbations

$$\dot{\delta}_{\pm} = \text{clip}(\dot{\delta}_{\text{nom}} \pm \varepsilon_{\delta}; \dot{\delta}_{\text{min}}, \dot{\delta}_{\text{max}}), \quad \mathbf{u}_{\pm} = [\dot{\delta}_{\pm}, F_{x,\text{nom}}]^\top,$$

together with a full-braking input $\mathbf{u}_{\text{brk}} = [\dot{\delta}_{\text{nom}}, F_{x,\text{min}}]^\top$.

To capture the nonlocal safety effect of decisive braking, we use a bound-secant for the longitudinal entry:

$$J_{h,F_x} \approx \frac{h(\Phi_\tau(x_0, \mathbf{u}_{\text{brk}})) - h(\Phi_\tau(x_0, \mathbf{u}_{\text{nom}}))}{F_{x,\text{min}} - F_{x,\text{nom}}}. \quad (11)$$

This “bound-secant” approach (Eq. (11)) intentionally deviates from a standard local tangent approximation. It is a pragmatic heuristic designed to address the well-known challenges that actuator saturation poses for optimization-based controllers, which can lead to QP infeasibility or performance degradation when the linearized model’s predictions diverge from the system’s true capabilities [93]. Such saturation-induced infeasibilities are a known challenge in control-allocation formulations [94]. By using a secant to the saturation limit $F_{x,\text{min}}$, we provide the safety-filter QP with a more conservative, nonlocal approximation of the true control authority available, improving feasibility and ensuring that

the “true safety effect of hard braking” is reflected in the optimization.

We first attempt a central difference using the clamped perturbations $\dot{\delta}_{\pm}$. If the central-difference interval collapses (i.e., $\dot{\delta}_{+} = \dot{\delta}_{-}$ because both perturbations clamp to the *same* bound), we revert to a one-sided secant from the nominal input to that active bound (cases below):

If $\dot{\delta}_{+} > \dot{\delta}_{-}$,

$$J_{h,\dot{\delta}} \approx \frac{h(\Phi_{\tau}(x_0, \mathbf{u}_{+})) - h(\Phi_{\tau}(x_0, \mathbf{u}_{-}))}{\dot{\delta}_{+} - \dot{\delta}_{-}}. \quad (12)$$

If $\dot{\delta}_{+} = \dot{\delta}_{-} = \dot{\delta}_{\min}$,

$$J_{h,\dot{\delta}} \approx \frac{h(\Phi_{\tau}(x_0, [\dot{\delta}_{\min}, F_{x,\text{nom}}]^{\top})) - h(\Phi_{\tau}(x_0, [\dot{\delta}_{-}, F_{x,\text{nom}}]^{\top}))}{\dot{\delta}_{\min} - \dot{\delta}_{-}}, \quad (13)$$

If $\dot{\delta}_{+} = \dot{\delta}_{-} = \dot{\delta}_{\max}$,

$$J_{h,\dot{\delta}} \approx \frac{h(\Phi_{\tau}(x_0, [\dot{\delta}_{+}, F_{x,\text{nom}}]^{\top})) - h(\Phi_{\tau}(x_0, [\dot{\delta}_{\min}, F_{x,\text{nom}}]^{\top}))}{\dot{\delta}_{+} - \dot{\delta}_{\min}}. \quad (14)$$

For conditioning, we clip derivative magnitudes:

$$J_{h,\dot{\delta}} \leftarrow \text{clip}(J_{h,\dot{\delta}}; -\Gamma_{\text{clip}}, \Gamma_{\text{clip}}), \quad J_{h,F_x} \leftarrow \text{clip}(J_{h,F_x}; -\Gamma_{\text{clip}}, \Gamma_{\text{clip}})$$

with a fixed $\Gamma_{\text{clip}} > 0$. With the row for the terminal barrier and bounds stacked, we solve the minimal-deviation QP:

b) QP instantiation.: In our implementation we solve the softened CBF-QP in scaled solver variables $z = [v; \xi]$, with $v = S\mathbf{u}$ and $S = \text{diag}(1, 10^{-3})$:

$$\begin{aligned} \min_{v, \xi \geq 0} \quad & \frac{1}{2} \|v - v_{\text{nom}}\|_{\Lambda}^2 + \frac{1}{2} \rho_{\text{slack}} \|\xi\|_2^2, \\ \text{s.t.} \quad & AS^{-1}v + \xi \geq b, \quad S\mathbf{u}_{\min} \leq v \leq S\mathbf{u}_{\max}, \end{aligned}$$

where $v_{\text{nom}} = S\mathbf{u}^{\text{nom}}$, $\Lambda = \text{diag}(\lambda_{\dot{\delta}}, \lambda_{F_x})$, and $\rho_{\text{slack}} \gg 0$. We log $\|\xi\|_{\infty}$ to monitor any residual violation; when $\xi = 0$ the constraints are met strictly.

Weight/scaling neutrality. The relative influence of steering vs. braking in the proximal term is determined by the combination of variable scaling S and solver weights Λ . Choosing (S, Λ) is equivalent to choosing $W = S^{\top} \Lambda S$ in the unscaled form; i.e., any “preference” is a modeling choice rather than an inherent bias of the method. In the original (unscaled) variables we write:

$$\begin{aligned} \min_{\mathbf{u}, \xi \geq 0} \quad & \frac{1}{2} \|\mathbf{u} - \mathbf{u}^{\text{nom}}\|_W^2 + \frac{1}{2} \rho \|\xi\|_2^2 \\ \text{s.t.} \quad & \mathbf{A}\mathbf{u} + \xi \geq \mathbf{b}, \\ & \mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}, \end{aligned} \quad (15)$$

where \mathbf{A} stacks the linearized barrier sensitivities (e.g. the row from h at $\tau = t_h$) and \mathbf{b} stacks the corresponding right-hand sides (e.g. $\beta(t_h) - h_{\text{nom}} + \mathbf{J}\mathbf{u}^{\text{nom}}$). Actuator bounds $\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}$ are retrieved online from the simulator, which encodes state-dependent steering-rate limits near mechanical stops and speed-dependent longitudinal force limits.

The above linearization acts on the *previewed, sampled-data* terminal map $h \circ \Phi_{\tau}$ obtained by numerical integration under ZOH. This aligns with the sampled-data CBF viewpoint: it directly enforces a discrete barrier difference on the next-sample map, avoiding Lie-derivative bookkeeping (cf. the zero-order perspective [22]) and is consistent with SD-CBF formulations that justify designing safety filters on *approximate* discrete-time models (e.g. Runge–Kutta) with practical-safety guarantees [20]. In parallel, ZOH-aware CBF conditions for sampled-data systems provide control-affine, QP-enforceable constraints that reduce conservativeness relative to earlier emulation-style margins [21]. Our finite-horizon, terminal enforcement follows this sampled-data logic while adding saturation-aware sensitivities and a slack-penalized feasibility mechanism for robustness to model mismatch.

Algorithm 2 Runtime geofencing with PCARNN-DCBF

- 1: **Inputs:** current state x_0 , polygonal geofence Ω , nominal input \mathbf{u}^{nom} , model (p^*, θ^*) , bounds $\mathbf{u}_{\min}, \mathbf{u}_{\max}$, horizon τ , substeps n_{sub} , targets $(h_{\text{target}}, \kappa)$ [κ]
- 2: **Output:** commanded input \mathbf{u}^*
- 3: propagate preview under \mathbf{u}^{nom} : $x(\tau) \leftarrow \Phi_{\tau}(x_0, \mathbf{u}^{\text{nom}})$
- 4: evaluate terminal barrier h and schedule $\beta(\tau)$
- 5: **if** $h(x(\tau)) \geq \beta(\tau)$ **then**
- 6: **return** \mathbf{u}^{nom} (early exit: no intervention)
- 7: build Jacobian \mathbf{J} by short rollouts:
 - (a) central differences in $\dot{\delta}$ around \mathbf{u}^{nom}
 - (b) bound–secant in F_x between \mathbf{u}^{nom} and $F_{x,\min}$
- 8: assemble linearized constraint (\mathbf{A}, \mathbf{b}) for h
- 9: solve QP Eq. (15) with bounds $\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}$
- 10: **if** QP feasible **then**
- 11: **return** \mathbf{u}^* (minimal correction; pass to on-board motion controller)
- 12: **else**
- 13: **return** $\mathbf{u}_{\text{EB}} = [0, F_{x,\min}]^{\top}$ (safety fallback)

Novelty (controller). Within the sampled-data / zero-order CBF framework [19], [20], [21], [22], our controller makes three design contributions tailored to polygonal geofencing and actuator-limited ground vehicles: (i) it instantiates a finite-horizon, preview-terminal barrier condition on the polygonal signed-distance map $h \circ \Phi_{\tau}$, using a geofence-specific schedule $\beta(\tau)$ and the learned PCARNN dynamics so that the constraint enters the QP as a single linear row tailored to the keep-in constraint; (ii) it introduces a simple “bound–secant” linearization heuristic near the full-braking limit to improve QP feasibility under actuator saturation, informed by control-allocation perspectives [94], [93]; and (iii) it solves a scaled minimal-deviation QP with an early-exit condition that avoids solving the QP when the nominal control is already safe, thereby reducing both intervention and computational load. Taken together, the preview-terminal enforcement on $h \circ \Phi_{\tau}$, the saturation-aware bound–secant sensitivities, and the scaled minimal-deviation QP with early exit yield a single, tractable safety filter specialized to polygonal geofencing with actuator-limited, learned vehicle dynamics.

D. End-to-end runtime pipeline

The end-to-end geofencing algorithm is presented in Algorithm 2 and the overall architecture is shown in Figure 1. If the QP is infeasible (line 12), the controller is designed to revert to a predefined fallback policy, commanding a full emergency brake \mathbf{u}_{EB} . This provides a simple, conservative fail-safe: when the safety filter cannot find a minimally corrective control input that guarantees constraint satisfaction, the vehicle is brought to a stop inside the current geofence.

V. EXPERIMENTS AND RESULTS

A. Experimental Setup

1) *Trajectory Creation in CARLA:* All training and evaluation trajectories were generated using the CARLA simulator (version 0.9.15) with a custom Python interface for high-fidelity control and state extraction [95]. Each trajectory was constructed to ensure kinematic consistency between position, velocity, and acceleration measurements and to capture realistic dynamic behavior across different operating regimes. Further details on the simulator configuration, dataset generation, and test scenario labeling are provided in Appendix A.

2) *Operating Regimes and Test Scenarios:* The distribution of safe and unsafe cases differs across both vehicles and operating regimes. A summary can be found in Table III. These trajectories were designed to evaluate how each model handles scenarios of varying

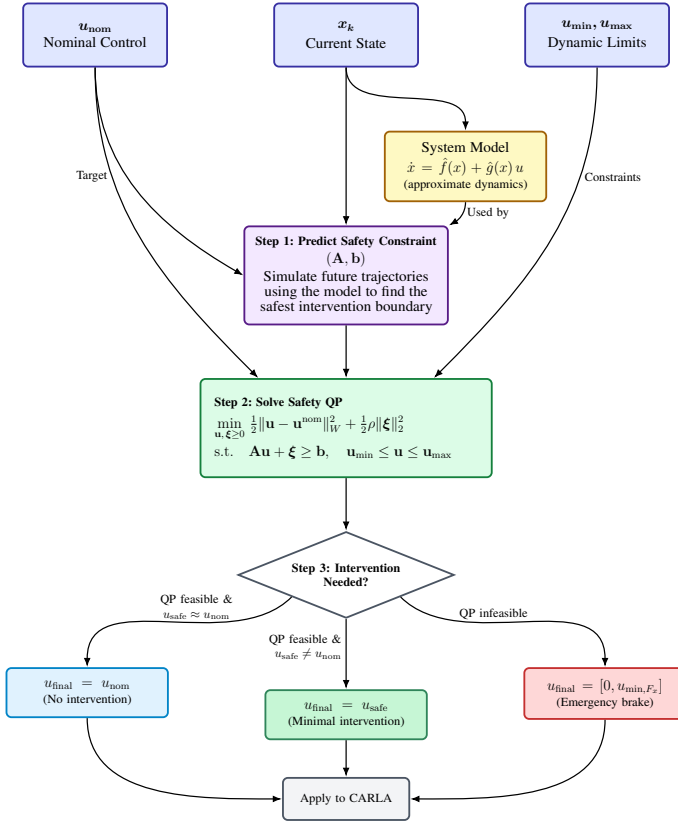


Figure 1: PCARNN-DCBF geofencing architecture.

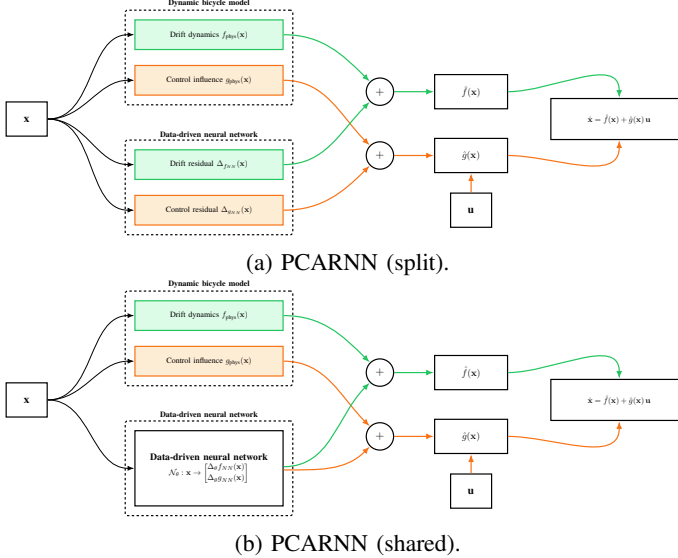


Figure 2: System model: PCARNN split model and PCARNN shared model.

difficulty, defined by combinations of vehicle speed and steering curvature.

In general, **low-speed-straight** regimes represent the easiest operating conditions. Vehicle dynamics are slow, lateral drift is minimal, and small corrective inputs are sufficient to maintain containment within the geofence. Similarly, **low-speed-sharp** trajectories, while involving higher steering inputs, remain relatively easy to control due to limited kinetic energy and lower momentum, which provide larger control margins and longer reaction times.

By contrast, **high-speed-straight** regimes are among the most challenging. Even small deviations from the boundary can quickly accumulate due to limited lateral authority and high inertial forces, leaving little room for corrective steering or braking. Finally, **high-speed-sharp** trajectories pose compounded difficulties: high curvature demands rapid and precise steering control, while increased velocity amplifies centrifugal forces and narrows the feasible control envelope. These scenarios are thus critical for evaluating the controller's responsiveness, stability, and ability to respect safety constraints under tight dynamic coupling between steering and braking.

Overall, this stratification of test scenarios enables a systematic assessment of how well each model generalizes across operating conditions that vary in control difficulty and physical constraints.

Table III: Counts of safe and unsafe scenarios by operating regime for each vehicle. Regimes are defined using thresholds on vehicle speed ($v_x = 8.0$ m/s) and steering input ($\delta = 0.35$ rad).

Vehicle	Operating Regime	Safe	Unsafe	Total
Lincoln MKZ	Low-Straight	49	44	93
	Low-Sharp	65	28	93
	High-Straight	17	18	35
	High-Sharp	29	8	37
	Total	160	98	258
Audi E-tron	Low-Straight	63	58	121
	Low-Sharp	98	16	114
	High-Straight	48	49	97
	High-Sharp	43	22	65
	Total	252	145	397

3) *Models for Benchmarking:* To benchmark different models and evaluate the effectiveness of the PCARNN-DCBF, we implemented and tested various {model}-DCBF architectures, where

$\text{model} \in \{\text{Dynamic Bicycle Model, Neural ODE, Hybrid Residual Model, PCARNN (shared-network), PCARNN (split-network)}\}$.

a) *Dynamic Bicycle Model:* The analytical baseline follows the kinematic bicycle formulation described in Section II-B1. For comparison, we also include the *Bicycle Ackermann* variant (adopted in [6]). In this model, the front axle follows classical Ackermann steering: in a steady turn, the inner and outer front wheels are oriented so that their wheel axes intersect at a common centre of rotation, with the inner wheel commanded to a larger steer angle so that both wheels roll on circular paths without lateral slip in low-speed manoeuvres [96], [97]. An equivalent single steering angle δ for the bicycle abstraction is then computed from the inner and outer wheel angles using this Ackermann relation, yielding a more geometrically accurate low-speed kinematic baseline.

b) *Neural ODE:* A neural network $NN_\theta(x, u)$ directly models the system dynamics by predicting the state derivative:

$$\dot{x} = NN_\theta(x, u).$$

This formulation maximizes expressiveness but abandons the control-affine structure, providing no guarantee of separability between system dynamics and control input.

c) *Hybrid Residual Model:* The residual model augments the analytical dynamics with a learned correction term:

$$\dot{x} = f_{\text{phys}}(x) + g_{\text{phys}}(x)u + \Delta_\theta(x, u),$$

where Δ_θ is a neural residual. This approach anchors learning to known physics while improving expressiveness. However, the addition of Δ_θ breaks the strictly control-affine form since it may depend nonlinearly on u .

Table IV: Summary of best-performing variants (highest CF1) from each model group (under braking + steering settings) for both vehicles. One small ($\sim 50K$) and one large ($\sim 150K$) configuration are selected per variant type. Lower is better for FPR and MCD^+ , higher is better for CF1.

Vehicle	Model Variant	Controller	Size (f, g)	Params	CF1 \uparrow	FPR \downarrow	$MCD^+\downarrow$
Audi E-tron	PCARNN (shared)	DCBF	$4 \times 128 / 5 \times 192$	51.3K / 150K	0.915 / 0.918	0.082 / 0.077	0.438 / 0.498
	PCARNN (split)	DCBF	$(3 \times 90, 4 \times 104) / (5 \times 135, 5 \times 135)$	51K / 149K	<u>0.928</u> / 0.935	0.087 / <u>0.066</u>	0.672 / <u>0.470</u>
	Residual	DCBF	$4 \times 128 / 5 \times 192$	50.9K / 150K	0.906 / 0.913	0.082 / 0.060	0.760 / 0.489
	Neural ODE	DCBF	$4 \times 128 / 5 \times 192$	50.9K / 150K	0.910 / 0.906	0.077 / 0.082	0.734 / 0.746
	Bicycle	DCBF	N/A	N/A	0.850	0.077	0.617
	Bicycle Ackermann	DCBF	N/A	N/A	0.896	0.120	1.080
Lincoln MKZ	PCARNN (shared)	DCBF	$4 \times 128 / 5 \times 192$	51.3K / 150K	0.922 / <u>0.913</u>	<u>0.050</u> / 0.062	0.356 / 0.569
	PCARNN (split)	DCBF	$(3 \times 90, 4 \times 104) / (4 \times 128, 5 \times 156)$	51K / 150K	0.908 / <u>0.913</u>	0.069 / 0.062	0.647 / 0.535
	Residual	DCBF	$4 \times 128 / 5 \times 192$	50.9K / 150K	0.737 / 0.748	0.044 / 0.062	0.293 / <u>0.341</u>
	Neural ODE	DCBF	$4 \times 128 / 5 \times 192$	50.9K / 150K	0.718 / 0.771	0.062 / 0.056	0.344 / 0.359
	Bicycle	DCBF	N/A	N/A	0.832	0.044	0.457
	Bicycle Ackermann	DCBF	N/A	N/A	0.905	0.088	0.999

d) *PCARNN (shared-network)*: In this formulation, we learn residuals for both f and g using a shared neural network structure. A single network outputs both the residual components for f and g , with the first part of the output $\Delta_\theta f(\mathbf{x})$ corresponding to the drift term and the second $\Delta_\theta g(\mathbf{x})$ to the control matrix. The final dynamics remain control-affine:

$$\dot{\mathbf{x}} = (f_{\text{phys}}(\mathbf{x}) + \Delta_\theta f(\mathbf{x})) + (g_{\text{phys}}(\mathbf{x}) + \Delta_\theta g(\mathbf{x}))\mathbf{u}.$$

We evaluate both small ($\sim 51.3K$ parameters) and large ($\sim 150K$ parameters) configurations.

e) *PCARNN (split-network)*: This variant employs two independent neural networks for Δf_{NN} and Δg_{NN} , allowing greater specialization for the drift and control components, respectively. The resulting model preserves the control-affine structure:

$$\dot{\mathbf{x}} = (f_{\text{phys}}(\mathbf{x}) + \Delta f_{NN}(\mathbf{x})) + (g_{\text{phys}}(\mathbf{x}) + \Delta g_{NN}(\mathbf{x}))\mathbf{u}.$$

As with the shared-network variant, we evaluate two model sizes, approximately 51K and 150K parameters, corresponding to compact and large-scale configurations.

4) *Network Architectures*: The Δf_{NN} and Δg_{NN} components were implemented as standard feedforward neural networks. We experimented with several architectures to analyze the impact of model capacity and parameter sharing, including:

- **Shared-network**: A single MLP backbone that outputs corrections for both f and g .
- **Split-network**: Two independent MLPs, one for Δf_{NN} and one for Δg_{NN} .

These shared and split control-affine architectures are summarized schematically in Figure 2. For both variants, we tested two network sizes, referred to as “small” ($\sim 50K$ parameters) and “large” ($\sim 150K$ parameters). For example, a large configuration (e.g. big_5x192) consists of 5 hidden layers with 192 nodes each, as shown in our ablation studies (Table VIII). Following modern practice, we use the SiLU activation in hidden layers [91], [92]. All models were trained with the AdamW optimizer [98] and a cosine-annealing learning-rate schedule [99]. In addition, we apply spectral normalization to the linear layers of the residual networks to control their operator 2-norms and thereby bound the Lipschitz gains of the learned corrections [100]. In the split PCARNN, spectral normalization is applied only to the Δg_{NN} network, while in the shared PCARNN a single spectrally normalized backbone produces both $\Delta f_{NN}(\mathbf{x})$ and $\Delta g_{NN}(\mathbf{x})$.

B. Evaluation metrics

The effectiveness of geofence enforcement is evaluated based on three key criteria: (1) it should successfully *detect* potential breaches and *prevent* the vehicle from breaching the geofence once an intervention is applied (high intervention success), (2) the controller should *accurately detect* only when an intervention is necessary (low false alarm), and (3) the *intervention should be minimal*, avoiding unnecessary or excessive control actions (minimal

intervention). To capture these aspects quantitatively, we employ three complementary metrics: FPR, CF1, and MCD^+ . At each control cycle we cast the decision as a binary classification. The predicted label is *positive* if the controller issues any intervention (brake and/or steer), otherwise *negative*. The ground-truth label is *positive* if the nominal, no-intervention rollout would violate the geofence within the lookahead horizon, otherwise *negative*. Let TP, FP, TN, FN denote the corresponding counts, and let CF be the number of *containment failures*, i.e. ground-truth positive cases where the controller *did* intervene but the geofence was still breached downstream (“try to intervene but failed”).

1) **CF1**: The Containment F1 score augments the standard F1 with a penalty for failed containments among required interventions. Define the success ratio among true positives as

$$s = \frac{TP - CF}{TP}, \quad (\text{set } s = 0 \text{ if } TP = 0).$$

Then

$$CF1 = F1 \cdot s = F1 \cdot \frac{TP - CF}{TP}.$$

Higher values indicate better containment performance, rewarding both correct intervention timing and effective containment. This is the main evaluation metric.

2) **FPR**: The false positive rate measures unnecessary interventions:

$$FPR = \frac{FP}{FP + TN}.$$

Lower values indicate fewer unnecessary interventions.

3) **MCD^+** : The median containment distance measures how close trajectories come to the geofence boundary under the closed-loop controller. For each episode, we compute the minimum signed distance to the polygon over time,

$$d_{\min} = \min_t d(\mathbf{p}(t)),$$

where $d(\cdot) > 0$ when the vehicle is inside the fence and $d(\cdot) < 0$ when it is outside. We then report the median of these minimum distances across all episodes:

$$\text{MedianContainmentDistance}(MCD) = \text{median}(\{d_{\min}^{(i)}\}_i) \text{ [m]}.$$

A positive MCD indicates that the vehicle remains within the geofence, where smaller positive values correspond to minimal but safe containment margins. A negative MCD indicates boundary violation, and smaller absolute values $|MCD|$ correspond to less severe breaches. We use MCD^+ to denote this signed performance interpretation, where smaller positive values are preferred as a proxy for minimal intervention, and negative values (especially large negative ones) are undesirable (hence the “+” sign).

C. Research Questions

We investigate the following research questions in this paper.

- RQ1.** *How do different PIML models and analytical models perform on the geofencing task?* This question compares the performance of various physics-informed machine learning models and analytical models in maintaining containment within geofenced boundaries.
- RQ2.** *How do different operating regimes (speed and steering curvature) influence the performance of different models?* This question examines how various models perform across four distinct regimes: high-speed-sharp-steering, high-speed-straight, low-speed-sharp-steering, and low-speed-straight.
- RQ3.** *How does the control-affine network structure affect control linearity and safety performance?* This question investigates whether decoupling the residual learning of f (drift dynamics) and g (control influence) improves control linearity and containment performance compared to a conventional residual neural network. Specifically, it compares architectures where f and g share the same network against those where they are modeled by separate subnetworks within the control-affine formulation.
- RQ4.** *How do key hyperparameters, including contraction rate (γ) and model size, influence the performance?* This question explores how tuning γ and scaling model size affect the performance metrics and whether optimal γ values generalize across models and vehicles.

D. Main Results (RQ1)

Table III summarizes performance across vehicles, hyperparameters, and model variants. Overall, PCARNN variants dominate on containment safety: they achieve the strongest CF1 on both vehicles while keeping FPR near the best and MCD competitive. This indicates that preserving the control-affine structure and learning residuals for both drift and control yields better calibrated decision boundaries than purely data-driven dynamics or PIML with combined residuals.

Second, the split vs. shared design interacts with vehicle dynamics. On the Audi E-tron, the split variant benefits from extra capacity, suggesting that independently tailoring drift and control residuals helps a vehicle with richer actuation-dynamics couplings. On the Lincoln MKZ, the smaller shared network peaks in CF1, implying that parameter sharing can regularize learning when a simpler inductive bias matches the vehicle's dynamics. In both cases, capacity is useful only to the extent that it respects the control-affine inductive bias; scaling without structure does not close the gap.

Third, alternative baselines expose a safety-conservatism trade-off. Residual models without control-affine constraints tend to be conservative, often improving FPR and MCD locally, but they compromise CF1, pointing to under-detection of unsafe cases. Neural ODE loses separability between state and input, which harms overall safety classification despite expressive function approximation. Classical bicycle models trail in CF1 and are less consistent on MCD, highlighting the limitation of fixed-form physics without learned corrections.

Findings (RQ1) Control-affine architectures (PCARNN) consistently outperform both analytical and other PIML baselines in containment safety, achieving the best balance between CF1, FPR, and MCD. Preserving separability between drift and control enables more stable and interpretable safety boundaries than residual or neural ODE models. Model capacity and sharing strategy interact with vehicle dynamics: electric drivetrains (Audi) benefit from split f - g specialization, while combustion drivetrains (Lincoln) favor shared representations that regularize the coupled propulsion-braking behavior.

E. Performance on Different Driving Scenarios (RQ2)

The stratified analysis in Figure 3 highlights how containment, intervention rate, and control smoothness vary across operating regimes and vehicle types. For the Audi E-tron, split control-affine models with

separate f and g components achieve the highest containment (CF1 \approx 0.93–0.94) and lowest intervention rates, particularly under high-speed and sharp-turn scenarios. This reflects the electric drivetrain's rapid and linear torque response, where modeling f and g independently improves stability and precision. For the Lincoln MKZ, shared-network architectures perform best overall, especially in sharp-turn conditions, suggesting that the coupled throttle and braking dynamics of petrol vehicles benefit from a jointly learned control-affine representation of drift and control effects. Overall, these results confirm that *the optimal model structure depends on the vehicle's actuation characteristics and dynamic coupling*, demonstrating the adaptability of the control-affine formulation across heterogeneous platforms.

Findings (RQ2) Performance varies systematically across driving regimes. Split PCARNN models excel on the Audi E-tron, particularly in high-speed or sharp-turn conditions, where independent modeling of f and g captures the vehicle's fast, decoupled actuation. The Lincoln MKZ achieves stronger containment with shared networks, especially in nonlinear low-speed regimes, reflecting tighter coupling between throttle and braking.

F. The Impact of the Control-Affine Structure (RQ3)

As established in Sec. III, a key challenge for discrete-time CBFs is that generic nonlinear dynamics lead to intractable optimizations [19]. A known remedy is to enforce a control-affine structure, which recovers a tractable and well-behaved QP [74].

The DCBF controller (Eq. (15)) relies on this control-affine assumption: small variations in $u = [\delta, F_x]^\top$ should induce an approximately linear change in the terminal barrier h . If this does not hold, the optimization may use inaccurate sensitivities, leading to constraint violations or undue conservatism. In this section, we directly test our hypothesis: does the PCARNN architecture, by design, produce a more linear control response (and thus better DCBF compatibility) than a generic residual or neural ODE baseline?

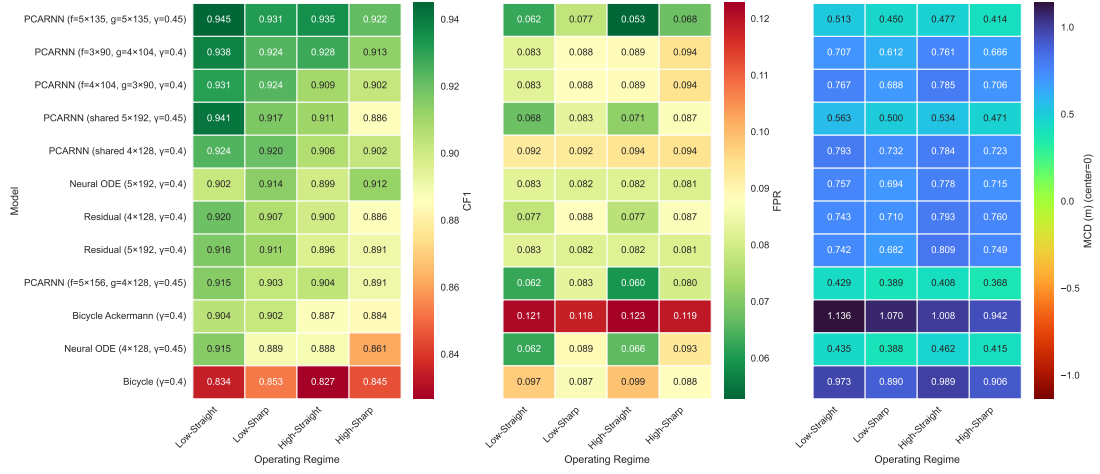
1) Gain of the control-affine structure with and without split f and g networks: We conduct an ablation study to assess the contribution of the control-affine formulation to the geofencing task. Specifically, we compare a standard residual PIML network (without the control-affine structure) against variants that incorporate control-affine decomposition, where the drift dynamics f and control influence g are either represented by a shared network or by separate subnetworks. Table V summarizes the results. Numbers in parentheses indicate performance changes relative to the residual baseline, with improvements highlighted in green and declines in red.

Across all tested configurations, introducing the control-affine structure markedly improved the containment safety metric (CF1) compared to the residual baseline. Among the control-affine variants, the split-network design achieved the highest overall performance, indicating that explicitly modeling the control influence g as a linear component enhances both stability and responsiveness. The split formulation offers greater flexibility by allowing the drift dynamics f and control influence g to specialize in distinct behaviors: f can better capture the nonlinear passive dynamics of the vehicle, while g focuses on the control-dependent response, improving adaptation under varying actuation modes (e.g. braking-only vs. braking+steering). This separation reduces interference between the learned dynamics and control pathways, resulting in smoother barrier updates and more consistent containment across vehicles. Overall, the results confirm that the control-affine formulation provides a substantial performance gain for geofencing applications without increasing model complexity.

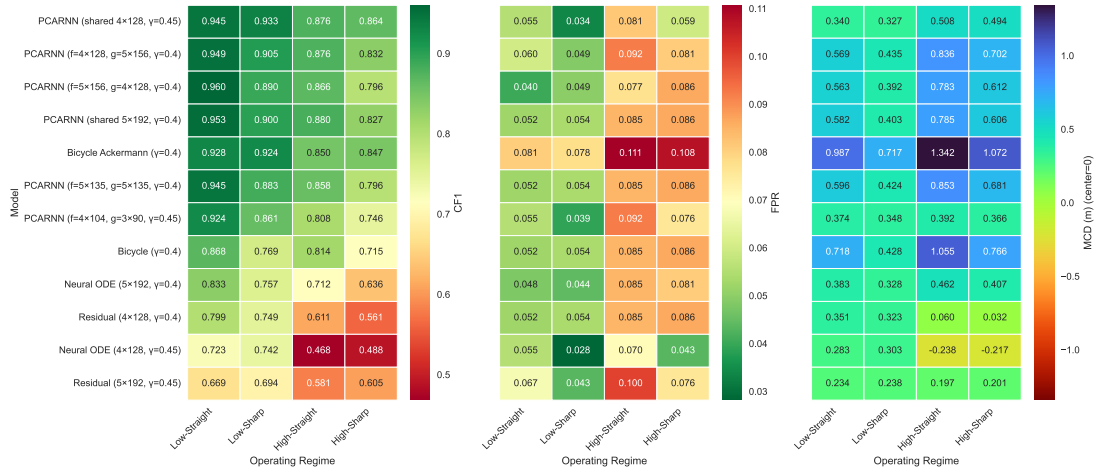
2) Control linearity analysis for DCBF compatibility: To test to what extent that each learned dynamics model satisfies the linearity assumption, we perform a *control linearity analysis*. This test evaluates whether the terminal barrier value h changes approximately linearly in response to small control perturbations. Specifically, we measure how well the first-order approximation

$$\Delta h \approx J \cdot \Delta u,$$

holds, where $J = \nabla_u h$ is the Jacobian of h with respect to the control inputs u .



(a) Stratified multi-metric analysis for the Audi E-tron across operating regimes. Each heatmap shows CF1, FPR, and MCD^+ scores for different model architectures under varying actuation regimes. Higher CF1 and lower FPR/ MCD^+ indicate better containment and smoother control responses.



(b) Stratified multi-metric analysis for the Lincoln MKZ across operating regimes. Results show the interaction between model architecture, actuation regime, and control linearity in determining containment performance.

Figure 3: Comparison of containment performance, intervention rate, and control smoothness across operating regimes for the Audi E-tron and Lincoln MKZ. Each panel illustrates how different architectures and control-affine formulations perform under low/high speed and steering conditions.

For each state in the test dataset, we estimate J numerically via central differences about the nominal control u_{nom} . The perturbations are applied over a 0.3 s preview horizon with $\Delta\delta = \pm 0.25$ rad/s and $\Delta F_x = \pm 800$ N. We then propagate the full nonlinear dynamics under both u_{nom} and $u_{nom} + \Delta u$ to obtain the actual change:

$$\Delta h_{actual} = h(\Phi(s_0, u_{nom} + \Delta u)) - h(\Phi(s_0, u_{nom})),$$

where Φ denotes the system rollout. The predicted change is given by

$$\Delta h_{predicted} = J \cdot \Delta u,$$

and the deviation between the two defines the *linearity error*:

$$\epsilon_{lin} = |\Delta h_{actual} - \Delta h_{predicted}|.$$

Lower values of ϵ_{lin} indicate that the model exhibits a stable, approximately linear response to control perturbations, aligning with the DCBF's control-affine assumption. Models with high ϵ_{lin} , on the other hand, violate this assumption and may produce unreliable or unstable barrier updates when deployed with the DCBF controller.

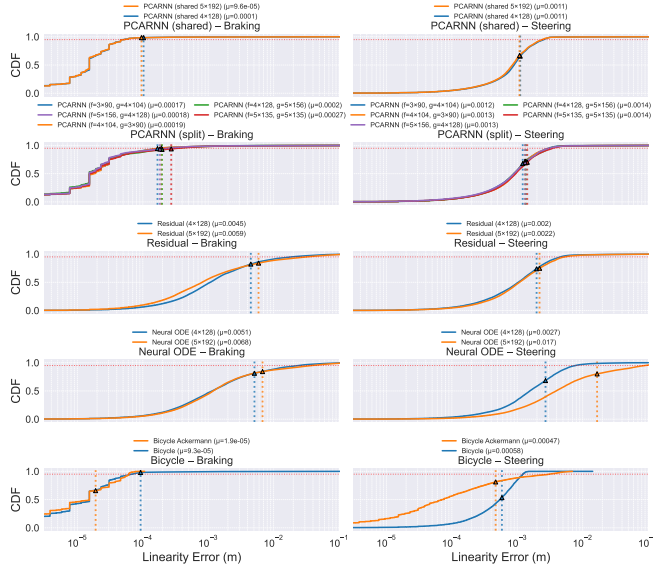
Figure 4 shows the cumulative distributions of linearity error ϵ_{lin} under braking and steering perturbations. Lower ϵ_{lin} values indicate a more linear, control-affine response, aligning better with the DCBF assumption.

Across both vehicles, braking and steering responses behave differently. **Braking** perturbations exhibit larger dispersion in ϵ_{lin} , especially for learned models, reflecting the stronger nonlinear coupling between longitudinal force and vehicle dynamics. **Steering** perturbations, by contrast, show much tighter CDFs with mean errors often one to two orders of magnitude smaller, indicating that lateral dynamics are locally smoother and more linear.

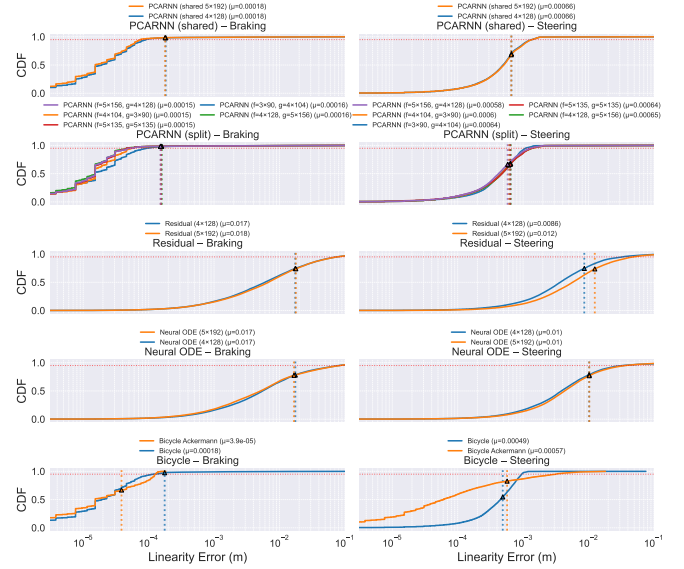
For the *Audi E-tron* (cf. Figure 4a), both shared and split PCARNN variants achieve the lowest linearity errors across braking and steering, maintaining $\mu \approx 10^{-4}$ – 10^{-3} m. The split architecture provides slightly tighter consistency, suggesting that explicitly modeling f and g separately helps capture the EV's decoupled torque and steering dynamics. The analytical bicycle model performs comparably well, confirming that the control-affine assumption remains valid for this drivetrain. In contrast, residual and neural ODE models display broader error distributions—especially under braking—indicating

Table V: Ablation study on the effect of control-affine components. The residual PIML network serves as the baseline. Variants include control-affine formulations where the drift dynamics f and control influence g are either modeled by a shared network or by separate subnetworks of comparable parameter count. Numbers in parentheses denote changes relative to the baseline; green indicates improvement and red indicates decline.

Variant	Size	Params	Braking + Steering		Braking			
			Lincoln MKZ	Audi E-tron	Lincoln MKZ		Audi E-tron	
			CBF	CBF	TTC	CBF	TTC	CBF
Residual	5×192	150K	0.748	0.913	0.623	0.624	0.449	0.571
w/ Control-affine (shared network)	5×192	150K	0.913 (+0.165)	0.918 (+0.005)	0.806 (+0.183)	<u>0.634</u> (+0.010)	<u>0.802</u> (+0.353)	0.595 (+0.024)
w/ Control-affine (split networks)	$f=5 \times 135, g=5 \times 135$	149K	0.873 (+0.125)	0.935 (+0.022)	0.562 (-0.061)	<u>0.634</u> (+0.010)	<u>0.794</u> (+0.345)	<u>0.603</u> (+0.032)
w/ Control-affine (split networks)	$f=4 \times 128, g=5 \times 156$	150K	0.913 (+0.165)	<u>0.919</u> (+0.006)	0.536 (-0.087)	0.640 (+0.016)	0.803 (+0.354)	0.612 (+0.041)
w/ Control-affine (split networks)	$f=5 \times 156, g=4 \times 128$	150K	<u>0.907</u> (+0.159)	<u>0.919</u> (+0.006)	0.552 (-0.071)	0.640 (+0.016)	0.798 (+0.349)	0.609 (+0.038)



(a) CDF of control linearity ϵ_{lin} for the vehicle Audi E-tron. The electric drivetrain exhibits tighter control-affine behavior, particularly under higher γ .



(b) CDF of control linearity ϵ_{lin} for the vehicle Lincoln MKZ. Lower values indicate better local linearity and more stable control responses.

Figure 4: Comparison of control linearity distributions for the Lincoln MKZ and Audi E-tron under varying γ and model configurations. Each CDF shows how well the discrete control barrier function (DCBF) maintains local linearity during operation.

Table VI: Ablation on the hyperparameter γ for the PCARNN (split) + DCBF model.

Size	Vehicle	γ	CF1 \uparrow	FPR \downarrow	MCD $^{+}\downarrow$
$f = 5 \times 135$ $g = 5 \times 135$	Audi E-tron	0.40	0.911	0.087	0.697
		0.44	<u>0.922</u>	0.071	0.496
		0.45	0.935	0.066	0.470
		0.46	0.901	0.077	0.416
		0.48	0.896	<u>0.060</u>	<u>0.355</u>
		0.60	0.803	0.055	0.122

higher nonlinearity and reduced DCBF compatibility.

For the *Lincoln MKZ* (cf. Figure 4b), overall linearity errors are higher, particularly under braking, reflecting the more complex drivetrain dynamics of the combustion vehicle. Here, the shared PCARNN variant achieves the best linearity performance ($\mu \approx 10^{-5}$ – 10^{-4} m), while the split version shows modest degradation. This suggests that jointly learning f and g stabilizes the interaction between throttle and brake channels, which are inherently coupled in the petrol configuration. Residual and neural ODE models again exhibit the largest variability, confirming their limited ability to maintain consistent linear response under nonlinear longitudinal coupling.

Findings (RQ3) Braking introduces greater nonlinearity than steering across all models. Control-affine formulations (analytical and PCARNN) maintain low and consistent ϵ_{lin} , validating their suitability for DCBF-based control. Electric drivetrains (Audi) benefit from separated f – g learning, as their torque response is fast and largely decoupled from steering, enabling each subnetwork to specialize in distinct control effects. Combustion drivetrains (Lincoln) achieve better stability with shared architectures, since throttle, braking, and load transfer are inherently coupled and require joint representation of drift and control dynamics.

G. Contraction Rate γ and Model Size (RQ4)

The sweep over contraction rates γ and model sizes highlights consistent trade-offs between safety enforcement, responsiveness, and control smoothness. Values around $\gamma = 0.45$ deliver the best overall containment performance for both vehicles, achieving strong safety enforcement without excessive conservatism. Lower γ values make the controller more permissive, sometimes allowing delayed reactions, while higher values ($\gamma > 0.5$) produce over-constrained, abrupt interventions that reduce responsiveness and increase containment oscillations.

Model capacity also affects performance stability. Compact networks (~ 50 K parameters) perform competitively, benefiting from the structured control-affine design that efficiently separates drift

Table VII: Ablation on PCARNN model architecture design with the DCBF controller. For each (Size f , Size g) configuration, only the γ yielding the highest CF1 is shown (breaking ties with lower FPR and MCD⁺). Lower is better for FPR and MCD⁺; higher is better for CF1. Values in parentheses indicate improvement relative to the shared baseline at the same γ .

Vehicle	Model	DCBF γ	Size f	Size g	Params	CF1 \uparrow	FPR \downarrow	MCD ⁺ \downarrow
Audi E-tron	PCARNN (shared)	0.40		4×128	51.3K	0.915	0.071	0.758
	PCARNN (split)	0.40	3×90	4×104	51K	0.928 (+0.013)	0.087 (+0.016)	0.672 (-0.086)
	PCARNN (split)	0.40	4×104	3×90	51K	0.921 (+0.006)	0.087 (+0.016)	0.612 (-0.146)
	PCARNN (shared)	0.45		5×192	150K	0.918	0.077	0.498
	PCARNN (split)	0.45	5×135	5×135	149K	0.935 (+0.017)	0.066 (-0.011)	0.470 (-0.028)
	PCARNN (split)	0.40	4×128	5×156	150K	0.919 (+0.004)	0.087 (+0.016)	0.658 (+0.160)
	PCARNN (split)	0.45	5×156	4×128	150K	0.905 (-0.013)	0.071 (-0.006)	0.396 (-0.102)
	PCARNN (split)	0.45	5×156	4×128	150K	0.905 (-0.013)	0.071 (-0.006)	0.396 (-0.102)
Lincoln MKZ	PCARNN (shared)	0.45		4×128	51.3K	0.922	0.050	0.356
	PCARNN (split)	0.45	3×90	4×104	51K	0.908 (-0.014)	0.069 (+0.019)	0.647 (+0.291)
	PCARNN (split)	0.45	4×104	3×90	51K	0.903 (-0.019)	0.062 (+0.012)	0.536 (+0.180)
	PCARNN (shared)	0.40		5×192	150K	0.913	0.062	0.569
	PCARNN (split)	0.40	5×135	5×135	149K	0.898 (-0.015)	0.062 (+0.000)	0.552 (-0.017)
	PCARNN (split)	0.40	4×128	5×156	150K	0.913 (+0.000)	0.062 (+0.000)	0.535 (-0.034)
	PCARNN (split)	0.40	5×156	4×128	150K	0.882 (-0.031)	0.050 (-0.012)	0.318 (-0.251)
	PCARNN (split)	0.40	5×156	4×128	150K	0.882 (-0.031)	0.050 (-0.012)	0.318 (-0.251)

and control effects. Larger networks ($\sim 150K$ parameters) improve containment (CF1) and reduce false positives (FPR), particularly when paired with well-tuned γ values, but can exhibit mild over-sensitivity under high contraction rates.

Findings (RQ4) Optimal contraction rates cluster around $\gamma \approx 0.45$ across models and vehicles, representing a robust equilibrium between safety and responsiveness. Moderate network sizes provide the most stable and generalizable performance.

VI. CONCLUSION

We presented PCARNN-DCBF, a geofencing framework that integrates a hybrid control-affine dynamics model with a preview-based safety filter to enforce polygonal keep-in constraints. By explicitly preserving the control-affine structure within the learning loop, our pipeline ensures that the safety constraint remains linear in the control input. This structural prior enables the use of a fast, interpretable QP for real-time intervention, avoiding the tractability issues common to unstructured neural models while correcting the fidelity gaps of purely analytical baselines.

Our comparative analysis highlighted the trade-offs inherent in data-driven safety filtering. While Neural ODEs offer high flexibility, they obscure the direct influence of actuation, leading to inconsistent safety margins. The PCARNN approach proved superior by maintaining the certifiable backbone of the bicycle model while learning targeted residuals for tire and load-transfer effects. This "structure-preserving" strategy yielded the highest containment reliability (CF1) with modest data requirements.

Furthermore, we found that drivetrain topology dictates the optimal learning architecture. Decoupled electric powertrains favored split drift/control networks to maximize control authority, whereas coupled combustion dynamics required shared representations to regularize the interaction between propulsion and braking.

Despite the observed gains, these findings are subject to several limitations. We target static keep-in sets and do not model moving agents or social driving rules. The dynamics are a bicycle style abstraction validated in simulation; real vehicle effects from tires, load transfer, and low friction can be stronger. The preview length and exponential target in the DCBF are engineered choices rather than formal recursive feasibility proofs.

Future work will extend the framework to dynamic environments with moving obstacles and incorporate robust state estimation to handle localization noise. As a direct next step, we aim to validate the full stack on a physical vehicle, specifically targeting low-friction winter conditions to test the generalization limits of the learned residuals.

ACKNOWLEDGMENT

We thank Sweden's Innovation Agency, Vinnova, for funding Grant No. 2021-05052, making the herein-described work possible.

REFERENCES

- [1] S. Balachandran, A. Narkawicz, C. Muñoz, and M. Consiglio, "A geofence violation prevention mechanism for small uas," in *Congress of the International Council of the Aeronautical Sciences*, no. NF1676L-27651, 2018.
- [2] J. Cho and Y. Yoon, "How to assess the capacity of urban airspace: A topological approach using keep-in and keep-out geofence," *Transportation Research Part C: Emerging Technologies*, vol. 92, pp. 137–149, 2018.
- [3] M. N. Stevens and E. M. Atkins, "Generating airspace geofence boundary layers in wind," *Journal of Aerospace Information Systems*, vol. 17, no. 2, pp. 113–124, 2020.
- [4] P. R. Thomas and P. Sarhadi, "Geofencing motion planning for unmanned aerial vehicles using an anticipatory range control algorithm," *Machines*, vol. 12, no. 1, p. 36, 2024.
- [5] X. Hong, G. Xiao, Y. Zhang, and J. Zhou, "A new path planning strategy based on level set function for layered fabrication processes," *The International Journal of Advanced Manufacturing Technology*, vol. 119, no. 1, pp. 517–529, 2022.
- [6] S. Thorén, L. Wikander, V. Jarlow, and T. Kero, "Model predictive geofencing for vehicle containment," in *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2024, pp. 65–70.
- [7] M. G. Plessen, D. Bernardini, H. Esen, and A. Bemporad, "Spatial-based predictive control and geometric corridor planning for adaptive cruise control coupled with obstacle avoidance," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 1, pp. 38–50, 2017.
- [8] S. T. Bukhari, D. Lawson, and A. H. Qureshi, "Differentiable composite neural signed distance fields for robot navigation in dynamic indoor environments," 2025. [Online]. Available: <https://arxiv.org/abs/2502.02664>
- [9] J. Ortiz, A. Clegg, J. Dong, E. Sucar, D. Novotny, M. Zollhoefer, and M. Mukadam, "isdf: Real-time neural signed distance fields for robot perception," 2022. [Online]. Available: <https://arxiv.org/abs/2204.02296>
- [10] Y. Li, X. Chi, A. Razmjoo, and S. Calinon, "Configuration space distance fields for manipulation planning," 2024. [Online]. Available: <https://arxiv.org/abs/2406.01137>
- [11] T. Wang and J. Sui, "Robot path planning method based on level set," in *Journal of Physics: Conference Series*, vol. 1621, no. 1. IOP Publishing, 2020, p. 012015.
- [12] M. W. Otte, "A survey of machine learning approaches to robotic path-planning," *University of Colorado at Boulder, Boulder*, 2015.
- [13] J. Kim and E. Atkins, "Airspace geofencing and flight planning for low-altitude, urban, small unmanned aircraft systems," *Applied Sciences*, vol. 12, no. 2, p. 576, 2022.
- [14] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *20th USENIX Security Symposium (USENIX Security 11)*. San Francisco, CA: USENIX Association, Aug. 2011. [Online]. Available: <https://www.usenix.org/conference/usenix-security-11/comprehensive-experimental-analyses-automotive-attack-surfaces>

Table VIII: Ablation on model architecture variants with the DCBF controller. Lower is better for FPR and MCD, and higher is better for CF1.

Vehicle	Variant	Size	Params	CF1 \uparrow	FPR \downarrow	MCD ⁺ \downarrow
Audi E-tron	PCARNN (shared) ($\gamma = 0.40$)	4×128	51.3K	0.915	0.071	0.758
	PCARNN (shared) ($\gamma = 0.40$)	5×192	150K	0.907	0.093	0.798
	PCARNN (shared) ($\gamma = 0.45$)	4×128	51.3K	0.915	0.082	0.438
	PCARNN (shared) ($\gamma = 0.45$)	5×192	150K	0.918	0.077	0.498
	PCARNN (split) ($\gamma = 0.40$)	$f = 3 \times 90, g = 4 \times 104$	51K	0.928	0.087	0.672
	PCARNN (split) ($\gamma = 0.40$)	$f = 5 \times 135, g = 5 \times 135$	149K	0.911	0.087	0.697
	PCARNN (split) ($\gamma = 0.40$)	$f = 4 \times 128, g = 5 \times 156$	150K	0.919	0.087	0.658
	PCARNN (split) ($\gamma = 0.45$)	$f = 3 \times 90, g = 4 \times 104$	51K	0.915	0.071	0.458
	PCARNN (split) ($\gamma = 0.45$)	$f = 5 \times 135, g = 5 \times 135$	149K	0.935	0.066	0.470
	PCARNN (split) ($\gamma = 0.45$)	$f = 4 \times 128, g = 5 \times 156$	150K	0.914	0.071	0.410
	Residual ($\gamma = 0.40$)	4×128	50.9K	0.906	0.082	0.706
	Residual ($\gamma = 0.40$)	5×192	150K	0.906	0.082	0.731
	Residual ($\gamma = 0.45$)	4×128	50.9K	0.892	0.066	0.412
	Residual ($\gamma = 0.45$)	5×192	150K	0.913	0.060	0.489
	Neural ODE ($\gamma = 0.40$)	4×128	50.9K	0.910	0.077	0.734
	Neural ODE ($\gamma = 0.40$)	5×192	150K	0.906	0.082	0.746
	Neural ODE ($\gamma = 0.45$)	4×128	50.9K	0.893	0.077	0.425
	Neural ODE ($\gamma = 0.45$)	5×192	150K	0.887	0.066	0.617
	Bicycle ($\gamma = 0.40$)	N/A	N/A	0.839	0.093	0.964
	Bicycle ($\gamma = 0.45$)	N/A	N/A	0.850	0.077	0.617
	Bicycle Ackermann ($\gamma = 0.40$)	N/A	N/A	0.896	0.120	1.080
Lincoln MKZ	PCARNN (shared) ($\gamma = 0.40$)	4×128	51.3K	0.913	0.062	0.558
	PCARNN (shared) ($\gamma = 0.40$)	5×192	150K	0.913	0.062	0.569
	PCARNN (shared) ($\gamma = 0.45$)	4×128	51.3K	0.922	0.050	0.356
	PCARNN (shared) ($\gamma = 0.45$)	5×192	150K	0.896	0.056	0.502
	PCARNN (split) ($\gamma = 0.40$)	$f = 3 \times 90, g = 4 \times 104$	51K	0.908	0.069	0.647
	PCARNN (split) ($\gamma = 0.40$)	$f = 5 \times 135, g = 5 \times 135$	149K	0.898	0.062	0.552
	PCARNN (split) ($\gamma = 0.40$)	$f = 4 \times 128, g = 5 \times 156$	150K	0.913	0.062	0.535
	PCARNN (split) ($\gamma = 0.45$)	$f = 3 \times 90, g = 4 \times 104$	51K	0.906	0.056	0.534
	PCARNN (split) ($\gamma = 0.45$)	$f = 5 \times 135, g = 5 \times 135$	149K	0.873	0.050	0.343
	PCARNN (split) ($\gamma = 0.45$)	$f = 4 \times 128, g = 5 \times 156$	150K	0.892	0.050	0.340
	Residual ($\gamma = 0.40$)	4×128	50.9K	0.728	0.062	0.309
	Residual ($\gamma = 0.40$)	5×192	150K	0.748	0.062	0.341
	Residual ($\gamma = 0.45$)	4×128	50.9K	0.737	0.044	0.293
	Residual ($\gamma = 0.45$)	5×192	150K	0.653	0.062	0.235
	Neural ODE ($\gamma = 0.40$)	4×128	50.9K	0.718	0.062	0.344
	Neural ODE ($\gamma = 0.40$)	5×192	150K	0.771	0.056	0.359
	Neural ODE ($\gamma = 0.45$)	4×128	50.9K	0.660	0.044	0.253
	Neural ODE ($\gamma = 0.45$)	5×192	150K	0.716	0.050	0.284
	Bicycle ($\gamma = 0.40$)	N/A	N/A	0.816	0.062	0.684
	Bicycle ($\gamma = 0.45$)	N/A	N/A	0.832	0.044	0.457
	Bicycle Ackermann ($\gamma = 0.40$)	N/A	N/A	0.905	0.088	0.999

- [15] S. Gupta, C. Maple, and R. Passerone, "An investigation of cyber-attacks and security mechanisms for connected and autonomous vehicles," *IEEE Access*, vol. 11, pp. 90 641–90 669, 2023.
- [16] Volkswagen Group of America, "Volkswagen group of america voluntary safety self-assessment (vssa)," 2023, available at: <https://media.vw.com/assets/documents/original/17321-VolkswagenVSSAV12023.pdf>.
- [17] United Nations Economic Commission for Europe (UNECE), "Draft guideline regarding safety technology for automated vehicles," 2023, document: WP29-180-10e. Available at: <https://unece.org/DAM/trans/doc/2020/wp29/WP29-180-10e.pdf>.
- [18] K. Balakrishnan, "Functional safety concept of "minimum risk maneuver" in conditional driving automation (level 3) vehicles," SAE Technical Paper 2022-28-0301, 2022, 10th SAE India International Mobility Conference, SAE International. [Online]. Available: <https://doi.org/10.4271/2022-28-0301>
- [19] A. Agrawal and K. Sreenath, "Discrete Control Barrier Functions for Safety-Critical Control of Discrete Systems with Application to Bipedal Robot Navigation," in *Robotics: Science and Systems XIII*. Robotics: Science and Systems Foundation, Jul. 2017. [Online]. Available: <http://www.roboticsproceedings.org/rss13/p73.pdf>
- [20] A. J. Taylor, V. D. Dorobantu, R. K. Cosner, Y. Yue, and A. D. Ames, "Safety of Sampled-Data Systems with Control Barrier Functions via Approximate Discrete Time Models," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. Cancun, Mexico: IEEE, Dec. 2022, pp. 7127–7134. [Online]. Available: <https://ieeexplore.ieee.org/document/9993226/>
- [21] J. Breeden, K. Garg, and D. Panagou, "Control Barrier Functions in Sampled-Data Systems," *IEEE Control Systems Letters*, vol. 6, pp. 367–372, 2022. [Online]. Available: <https://ieeexplore.ieee.org/document/9417092/>
- [22] X. Tan, E. Das, A. D. Ames, and J. W. Burdick, "Zero-Order Control Barrier Functions for Sampled-Data Systems with State and Input Dependent Safety Constraints," Apr. 2025, arXiv:2411.17079 [eess]. [Online]. Available: <http://arxiv.org/abs/2411.17079>
- [23] R. Rajamani, "Vehicle dynamics and control," *Mechanical Engineering Series*, 2012.
- [24] P. Polack, F. Althché, B. Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" 06 2017, pp. 812–818.
- [25] V. Patil, "Generic and complete vehicle dynamic models for open-source platforms," 2017.
- [26] S. Wu, Y. Fang, N. Sun, B. Lu, X. Liang, and Y. Zhao, "Optimization-free smooth control barrier function for polygonal collision avoidance," *IEEE Transactions on Cybernetics*, 2025.
- [27] M. Koptev, N. Figueroa, and A. Billard, "Neural joint space implicit signed distance functions for reactive robot manipulator control," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 480–487, 2023.
- [28] K. Hormann and A. Agathos, "The point in polygon problem for arbitrary polygons," *Computational Geometry*, vol. 20, no. 3, pp.

- 131–144, 2001. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925772101000128>
- [29] J. Miao, R. Yan, B. Zhang, T. Wen, J. Li, Z. Fu, K. Jiang, M. Yang, J. Huang, Z. Zhong *et al.*, “Residual learning towards high-fidelity vehicle dynamics modeling with transformer,” *IEEE Robotics and Automation Letters*, 2025.
- [30] A. Farea, O. Yli-Harja, and F. Emmert-Streib, “Understanding Physics-Informed Neural Networks: Techniques, Applications, Trends, and Challenges,” *AI*, vol. 5, no. 3, pp. 1534–1557, Aug. 2024. [Online]. Available: <https://www.mdpi.com/2673-2688/5/3/74>
- [31] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, “Scientific Machine Learning Through Physics-Informed Neural Networks: Where we are and What’s Next,” *Journal of Scientific Computing*, vol. 92, no. 3, p. 88, Sep. 2022. [Online]. Available: <https://link.springer.com/10.1007/s10915-022-01939-z>
- [32] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, “Physics-informed neural networks (PINNs) for fluid mechanics: a review,” *Acta Mechanica Sinica*, vol. 37, no. 12, pp. 1727–1738, Dec. 2021. [Online]. Available: <https://link.springer.com/10.1007/s10409-021-01148-1>
- [33] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, “Physics-Informed Neural Networks for Heat Transfer Problems,” *Journal of Heat Transfer*, vol. 143, no. 6, p. 060801, Jun. 2021. [Online]. Available: <https://asmedigitalcollection.asme.org/heattransfer/article/143/6/060801/1104439/Physics-Informed-Neural-Networks-for-Heat-Transfer>
- [34] F. Sahli Costabal, Y. Yang, P. Perdikaris, D. E. Hurtado, and E. Kuhl, “Physics-Informed Neural Networks for Cardiac Activation Mapping,” *Frontiers in Physics*, vol. 8, p. 42, Feb. 2020. [Online]. Available: <https://www.frontiersin.org/article/10.3389/fphy.2020.00042/full>
- [35] G. S. Misyris, A. Venzke, and S. Chatzivasileiadis, “Physics-Informed Neural Networks for Power Systems,” in *2020 IEEE Power & Energy Society General Meeting (PESGM)*. Montreal, QC, Canada: IEEE, Aug. 2020, pp. 1–5. [Online]. Available: <https://ieeexplore.ieee.org/document/9282004/>
- [36] M. Raissi, P. Perdikaris, and G. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *Journal of Computational Physics*, vol. 378, pp. 686–707, Feb. 2019. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0021999118307125>
- [37] J. Cho, S. Nam, H. Yang, S.-B. Yun, Y. Hong, and E. Park, “Separable Physics-Informed Neural Networks.”
- [38] Y. Zhao, C. Jiang, M. A. Vega, M. D. Todd, and Z. Hu, “Surrogate Modeling of Nonlinear Dynamic Systems: A Comparative Study,” *Journal of Computing and Information Science in Engineering*, vol. 23, no. 1, p. 011001, Feb. 2023. [Online]. Available: <https://asmedigitalcollection.asme.org/computingengineering/article/23/1/011001/1139337/Surrogate-Modeling-of-Nonlinear-Dynamic-Systems-A>
- [39] R. Newbury, J. Collins, K. He, J. Pan, I. Posner, D. Howard, and A. Cosgun, “A Review of Differentiable Simulators,” *IEEE Access*, vol. 12, pp. 97 581–97 604, 2024. [Online]. Available: <https://ieeexplore.ieee.org/document/10589638/>
- [40] J. Degrave, M. Hermans, J. Dambre, and F. Wyffels, “A Differentiable Physics Engine for Deep Learning in Robotics,” *Frontiers in Neurorobotics*, vol. 13, p. 6, Mar. 2019. [Online]. Available: <https://www.frontiersin.org/article/10.3389/fnbot.2019.00006/full>
- [41] A. Daw, A. Karpatne, W. Watkins, J. Read, and V. Kumar, “Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling,” Sep. 2021, arXiv:1710.11431 [cs]. [Online]. Available: <http://arxiv.org/abs/1710.11431>
- [42] X. Jia, Y. Xie, S. Li, S. Chen, J. Zwart, J. Sadler, A. Appling, S. Oliver, and J. Read, “Physics-Guided Machine Learning from Simulation Data: An Application in Modeling Lake and River Systems,” in *2021 IEEE International Conference on Data Mining (ICDM)*. Auckland, New Zealand: IEEE, Dec. 2021, pp. 270–279. [Online]. Available: <https://ieeexplore.ieee.org/document/9679022/>
- [43] S. Pawar, O. San, B. Aksoylu, A. Rasheed, and T. Kvamsdal, “Physics guided machine learning using simplified theories,” *Physics of Fluids*, vol. 33, no. 1, p. 011701, Jan. 2021. [Online]. Available: <https://pubs.aip.org/pof/article/33/1/011701/1018204/Physics-guided-machine-learning-using-simplified>
- [44] A. Tong, T. Nguyen-Tang, D. Lee, D. Nguyen, T. Tran, D. Hall, C. Kang, and J. Choi, “NEURAL ODE TRANSFORMERS: ANALYZING INTER- NAL DYNAMICS AND ADAPTIVE FINE-TUNING,” 2025.
- [45] J. M. Worsham and J. K. Kalita, “A guide to neural ordinary differential equations: Machine learning for data-driven digital engineering,” *Digital Engineering*, vol. 6, p. 100060, Sep. 2025. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S29505550X25000263>
- [46] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, “Neural Ordinary Differential Equations,” Dec. 2019, arXiv:1806.07366 [cs]. [Online]. Available: <http://arxiv.org/abs/1806.07366>
- [47] K. El Haloui, N. Thome, and N. Sisourat, “Combining Physics and Machine Learning: Hybrid Models for Predicting Interatomic Potentials,” *Atoms*, vol. 13, no. 11, p. 89, Nov. 2025. [Online]. Available: <https://www.mdpi.com/2218-2004/13/11/89>
- [48] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, and A. Edelman, “Universal Differential Equations for Scientific Machine Learning,” Nov. 2021, arXiv:2001.04385 [cs]. [Online]. Available: <http://arxiv.org/abs/2001.04385>
- [49] F. E. Bock, S. Keller, N. Huber, and B. Klusemann, “Hybrid Modelling by Machine Learning Corrections of Analytical Model Predictions towards High-Fidelity Simulation Solutions,” *Materials*, vol. 14, no. 8, p. 1883, Apr. 2021. [Online]. Available: <https://www.mdpi.com/1996-1944/14/8/1883>
- [50] Y. Yin, V. Le Guen, J. Dona, E. de Bézenac, I. Ayed, N. Thome, and P. Gallinari, “Augmenting physical models with deep networks for complex dynamics forecasting,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2021, no. 12, p. 124012, dec 2021. [Online]. Available: <https://doi.org/10.1088/1742-5468/ac3ae5>
- [51] B. Raonić, R. Molinaro, T. De Ryck, T. Rohner, F. Bartolucci, R. Alaifari, S. Mishra, and E. de Bézenac, “Convolutional Neural Operators for robust and accurate learning of PDEs,” 2023, version Number: 3. [Online]. Available: <https://arxiv.org/abs/2302.01178>
- [52] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, “Fourier Neural Operator for Parametric Partial Differential Equations,” May 2021, arXiv:2010.08895 [cs]. [Online]. Available: <http://arxiv.org/abs/2010.08895>
- [53] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, “Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators,” *Nature Machine Intelligence*, vol. 3, no. 3, pp. 218–229, Mar. 2021. [Online]. Available: <https://www.nature.com/articles/s42256-021-00302-5>
- [54] S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Discovering governing equations from data: Sparse identification of nonlinear dynamical systems,” *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, Apr. 2016, arXiv:1509.03580 [math]. [Online]. Available: <http://arxiv.org/abs/1509.03580>
- [55] J. HENDRIKX, T. MEIJLINK, and R. KRIENS, “Application of optimal control theory to inverse simulation of car handling,” *Vehicle System Dynamics*, vol. 26, no. 6, pp. 449–461, 1996.
- [56] M. Massaro and D. J. N. Limebeer, “Minimum-lap-time optimisation and simulation,” *Vehicle System Dynamics*, vol. 59, no. 7, pp. 1069–1113, 2021.
- [57] H. B. Pacejka and E. Bakker, “The magic formula tyre model,” *Vehicle system dynamics*, vol. 21, no. S1, pp. 1–18, 1992.
- [58] International Organization for Standardization, “Intelligent transport systems — forward vehicle collision warning systems — performance requirements and test procedures,” Standard ISO 15623:2013, 2013.
- [59] D. N. Lee, “A theory of visual control of braking based on information about time-to-collision,” *Perception*, vol. 5, no. 4, pp. 437–459, 1976.
- [60] A. Van der Horst and J. Hogema, “Time-to-collision and collision avoidance systems,” *Verkeersgedrag in Onderzoek*, 1994.
- [61] M. Althoff and J. M. Dolan, “Online verification of automated road vehicles using reachability analysis,” *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.
- [62] C. Pek and M. Althoff, “Fail-safe motion planning for online verification of autonomous vehicles using convex optimization,” *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 798–814, 2021.
- [63] C. Zhao, H. Yu, and T. G. Molnar, “Safety-critical traffic control by connected automated vehicles,” *Transportation Research Part C: Emerging Technologies*, vol. 154, p. 104230, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0968090X23000219X>
- [64] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, “Control Barrier Function Based Quadratic Programs for Safety Critical Systems,” *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017. [Online]. Available: <http://ieeexplore.ieee.org/document/7782377/>
- [65] M. Vögel, O. von Stryk, and R. Bulirsch, “Real-time simulation of vehicle dynamics: On-line control and handling investigations,” in *Proceedings of ECMI ’98, European Conference on Industrial and Applied Mathematics*, 1998, available at: <http://www.math.chalmers.se/Conf/ECMI98/standard/Abstracts/Vehicles-aerodynamics/voegel.pdf>.

- [66] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving ordinary differential equations I (2nd revised. ed.): nonstiff problems*. Berlin, Heidelberg: Springer-Verlag, 1993.
- [67] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, 3rd ed. Hoboken, New Jersey: John Wiley & Sons, 2016. [Online]. Available: <https://onlinelibrary.wiley.com/doi/book/10.1002/9781119121534>
- [68] E. Hairer, C. Lubich, and G. Wanner, *Geometric numerical integration*, 2nd ed., ser. Springer Series in Computational Mathematics. Springer-Verlag, Berlin, 2006, vol. 31, structure-preserving algorithms for ordinary differential equations.
- [69] P. R. Thomas and P. Sarhadi, “Geofencing Motion Planning for Unmanned Aerial Vehicles Using an Anticipatory Range Control Algorithm,” *Machines*, vol. 12, no. 1, p. 36, Jan. 2024. [Online]. Available: <https://www.mdpi.com/2075-1702/12/1/36>
- [70] E. Hermand, T. W. Nguyen, M. Hosseinzadeh, and E. Garone, “Constrained Control of UAVs in Geofencing Applications,” in *2018 26th Mediterranean Conference on Control and Automation (MED)*. Zadar, Croatia: IEEE, Jun. 2018, pp. 217–222. [Online]. Available: <https://ieeexplore.ieee.org/document/8443035/>
- [71] A. Singletary, A. Swann, Y. Chen, and A. D. Ames, “Onboard Safety Guarantees for Racing Drones: High-Speed Geofencing With Control Barrier Functions,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2897–2904, Apr. 2022. [Online]. Available: <https://ieeexplore.ieee.org/document/9691815/>
- [72] T. G. Molnar, S. K. Kannan, J. Cunningham, K. Dunlap, K. L. Hobbs, and A. D. Ames, “Collision Avoidance and Geofencing for Fixed-wing Aircraft with Control Barrier Functions,” Jan. 2025, arXiv:2403.02508 [eess]. [Online]. Available: <http://arxiv.org/abs/2403.02508>
- [73] S. Bansal, A. Bajcsy, E. Ratner, A. D. Dragan, and C. J. Tomlin, “A Hamilton-Jacobi Reachability-Based Framework for Predicting and Analyzing Human Motion for Safe Planning,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France: IEEE, May 2020, pp. 7149–7155. [Online]. Available: <https://ieeexplore.ieee.org/document/9197257/>
- [74] M. Khajenejad, M. Cavorsi, R. Niu, Q. Shen, and S. Z. Yong, “Tractable Compositions of Discrete-Time Control Barrier Functions with Application to Driving Safety Control,” in *2021 European Control Conference (ECC)*. Delft, Netherlands: IEEE, Jun. 2021, pp. 1303–1309. [Online]. Available: <https://ieeexplore.ieee.org/document/9777251/>
- [75] Y. Xiong, D.-H. Zhai, M. Tavakoli, and Y. Xia, “Discrete-Time Control Barrier Function: High-Order Case and Adaptive Case,” *IEEE Transactions on Cybernetics*, vol. 53, no. 5, pp. 3231–3239, May 2023. [Online]. Available: <https://ieeexplore.ieee.org/document/9777251/>
- [76] S.-C. Hsu, X. Xu, and A. D. Ames, “Control barrier function based quadratic programs with application to bipedal robotic walking,” in *2015 American Control Conference (ACC)*. Chicago, IL, USA: IEEE, Jul. 2015, pp. 4542–4548. [Online]. Available: <http://ieeexplore.ieee.org/document/7172044/>
- [77] Q. Nguyen and K. Sreenath, “Exponential Control Barrier Functions for enforcing high relative-degree safety-critical constraints,” in *2016 American Control Conference (ACC)*. Boston, MA, USA: IEEE, Jul. 2016, pp. 322–328. [Online]. Available: <http://ieeexplore.ieee.org/document/7524935/>
- [78] J. Zeng, B. Zhang, and K. Sreenath, “Safety-Critical Model Predictive Control with Discrete-Time Control Barrier Function,” in *2021 American Control Conference (ACC)*. New Orleans, LA, USA: IEEE, May 2021, pp. 3882–3889. [Online]. Available: <https://ieeexplore.ieee.org/document/9483029/>
- [79] K. P. Wabersich and M. N. Zeilinger, “Predictive control barrier functions: Enhanced safety mechanisms for learning-based control,” May 2022, arXiv:2105.10241 [eess]. [Online]. Available: <http://arxiv.org/abs/2105.10241>
- [80] N. A. Spielberg, M. Brown, N. R. Kapania, J. C. Kogelman, and J. C. Gerdes, “Neural network vehicle models for high-performance automated driving,” *Science Robotics*, vol. 4, no. 28, p. eaaw1975, Mar. 2019. [Online]. Available: <https://www.science.org/doi/10.1126/scirobotics.aaw1975>
- [81] J. Chrosniak, J. Ning, and M. Behl, “Deep dynamics: Vehicle dynamics modeling with a physics-constrained neural network for autonomous racing,” *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5292–5297, 2024.
- [82] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, “Learning-Based Model Predictive Control for Autonomous Racing,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, Oct. 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8754713/>
- [83] S. Jiang, Y. Wang, W. Lin, Y. Cao, L. Lin, J. Miao, and Q. Luo, “A High-accuracy Framework for Vehicle Dynamic Modeling in Autonomous Driving,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Prague, Czech Republic: IEEE, Sep. 2021, pp. 6680–6687. [Online]. Available: <https://ieeexplore.ieee.org/document/9636861/>
- [84] G. Chen, J. Yao, Z. Gao, Y. Zhao, C. Liu, S. Song, and M. Hua, “Lateral Velocity Estimation Utilizing Transfer Learning Characteristics by a Hybrid Data-mechanism-driven Model,” in *2024 IEEE Intelligent Vehicles Symposium (IV)*. Jeju Island, Korea, Republic of: IEEE, Jun. 2024, pp. 460–465. [Online]. Available: <https://ieeexplore.ieee.org/document/10588663/>
- [85] T. Kim, H. Lee, and W. Lee, “Physics Embedded Neural Network Vehicle Model and Applications in Risk-Aware Autonomous Driving Using Latent Features,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Kyoto, Japan: IEEE, Oct. 2022, pp. 4182–4189. [Online]. Available: <https://ieeexplore.ieee.org/document/9981303/>
- [86] Y. Li and L. Liu, “Physics-Informed Neural Network-Based Nonlinear Model Predictive Control for Automated Guided Vehicle Trajectory Tracking,” *World Electric Vehicle Journal*, vol. 15, no. 10, p. 460, Oct. 2024. [Online]. Available: <https://www.mdpi.com/2032-6653/15/10/460>
- [87] E. A. Antonelo, E. Camponogara, L. O. Seman, E. R. d. Souza, J. P. Jordanou, and J. F. Hubner, “Physics-Informed Neural Nets for Control of Dynamical Systems,” *Neurocomputing*, vol. 579, p. 127419, Apr. 2024, arXiv:2104.02556 [cs]. [Online]. Available: <http://arxiv.org/abs/2104.02556>
- [88] J. Xie, F. Bonassi, and R. Scattolini, “Learning control affine neural narx models for internal model control design,” *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 8137–8149, 2025.
- [89] F. Bonassi, C. F. O. Da Silva, and R. Scattolini, “Nonlinear MPC for Offset-Free Tracking of systems learned by GRU Neural Networks,” *IFAC-PapersOnLine*, vol. 54, no. 14, pp. 54–59, 2021. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S240589632101733X>
- [90] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations (ICLR)*, 2015, arXiv:1412.6980. [Online]. Available: <https://arxiv.org/abs/1412.6980>
- [91] D. Hendrycks, “Gaussian error linear units (gelus),” *arXiv preprint arXiv:1606.08415*, 2016.
- [92] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for Activation Functions,” Oct. 2017, arXiv:1710.05941 [cs]. [Online]. Available: <http://arxiv.org/abs/1710.05941>
- [93] R. de Castro and J. Brembeck, “Lyapunov-based control allocation for over-actuated nonlinear systems,” in *2019 American Control Conference (ACC)*, 2019, pp. 5033–5038.
- [94] T. A. Johansen and T. I. Fossen, “Control allocation—a survey,” *Automatica*, vol. 49, no. 5, pp. 1087–1103, 2013. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0005109813000368>
- [95] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator,” in *Conference on robot learning*. PMLR, 2017, pp. 1–16.
- [96] A. Corominas, M. Idrees, and F. Karlsson, “Four wheel steering: Comarom with two wheel steering,” Bachelor’s thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2014. [Online]. Available: <http://kth.diva-portal.org/smash/get/diva2:713538/FULLTEXT01.pdf>
- [97] S. S. Verma, K. Rohith, A. S. Prasad, E. H. Teja, and M. V. Sai, “Design and comparative analysis of ackermann and anti-ackermann steering system,” *International Journal of Research and Analytical Reviews*, vol. 6, no. 1, pp. 156–167, 2019.
- [98] I. Loshchilov and F. Hutter, “Decoupled Weight Decay Regularization,” Jan. 2019, arXiv:1711.05101 [cs]. [Online]. Available: <http://arxiv.org/abs/1711.05101>
- [99] —, “SGDR: Stochastic Gradient Descent with Warm Restarts,” May 2017, arXiv:1608.03983 [cs]. [Online]. Available: <http://arxiv.org/abs/1608.03983>
- [100] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral normalization for generative adversarial networks,” *arXiv preprint arXiv:1802.05957*, 2018.
- [101] CARLA Simulator Team, “CARLA python api reference: carla.VehiclePhysicsControl,” 2024, accessed: 2025-11-13. Available at: https://carla.readthedocs.io/en/0.9.15/python_api/#carla.VehiclePhysicsControl.
- [102] —, “Carla vehicle blueprint library (v0.9.15),” https://carla.readthedocs.io/en/0.9.15/bp_library/, accessed: 2025-11-13.

- [103] OpenCOOD Project, “Opencood smartdata model: Carla 0.9.15 vehicle parameters (town06),” <https://lisha.ufsc.br/Opencood+-+SmartData+Model>, accessed: 2025-11-13.
- [104] R. Xu, H. Xiang, X. Xia, X. Han, J. Li, and J. Ma, “Opv2v: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication,” in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2583–2589.
- [105] MathWorks, “Steering system,” <https://www.mathworks.com/help/vdynblks/ref/steeringsystem.html>, vehicle Dynamics Blockset Documentation, accessed: 2025-11-13.

APPENDIX

A. Data Fidelity and Simulation Environment

The experimental framework employs the CARLA simulator (version 0.9.15) [95] with a custom Python interface to generate high-fidelity, kinematically consistent datasets for training the PCARNN. This predictive model forms the core of the safety framework.

1) *CARLA Physics Customization*: To ensure the simulator’s vehicle dynamics align with the analytical bicycle model assumptions and to avoid simulator-specific artifacts, CARLA’s physics configuration was extensively customized:

- **Actuator authority**: The default speed-dependent steering attenuation was flattened to allow full steering range (δ_{\max}) at all speeds. This was implemented by setting the `steering_curve` field of the `carla.VehiclePhysicsControl` object to a constant value, replacing the default speed-dependent curve (e.g. `[[0.0, 1.0], [10.0, 0.5]]`) documented in the CARLA physics API [101]. This ensures that the controller maintains unrestricted authority.
- **Powertrain characterization**: The vehicle physics were parameterized to provide a consistent mapping between the controller’s commanded longitudinal force (F_x) and CARLA’s throttle/brake inputs. For the *Audi e-tron*, we used the default CARLA vehicle blueprint (`vehicle.audi.etron`) and its stock EV powertrain parameters [95], [102]. For the *Lincoln MKZ (2017)*, we used powertrain and transmission parameters from the OpenCOOD SmartData Model [103], [104].
- **Fidelity alignment**: Tire friction coefficients, brake torque limits, and stiffness parameters were tuned so that the handling limits (maximum lateral acceleration and deceleration) matched the analytical assumptions used in the Discrete Control Barrier Function (DCBF) formulation. The Audi E-tron retained CARLA’s default chassis and tire setup, while the Lincoln MKZ used the OPV2V-derived physical parameters.

2) *Kinematically Consistent State and Derivative Extraction*: Rather than relying on CARLA’s instantaneous acceleration outputs (which can exhibit numerical noise and drift), all ground-truth derivatives were computed via finite differencing to ensure consistency between \mathbf{x} , $\dot{\mathbf{x}}$, and $\ddot{\mathbf{x}}$:

- **State definition**: The system state \mathbf{x} includes position (p_x, p_y) , yaw ψ , body-frame velocities (v_x, v_y) , yaw rate ω , and effective front wheel steering angle δ . The effective δ is computed using a standard Ackermann steering relation from the individual left/right wheel angles to provide a kinematically equivalent bicycle model input [105].
- **Derivative calculation**: State derivatives $\dot{\mathbf{x}} = [\dot{p}_x, \dot{p}_y, \dot{\psi}, \dot{\omega}, \dot{\delta}]^T$ are obtained using a three-point central or forward finite-difference scheme over a fixed time step $\Delta t = 0.02$ s. This ensures that \mathbf{x}_k , \mathbf{u}_k , and $\dot{\mathbf{x}}_k$ are mathematically coupled, producing a consistent training signal for residual and ODE-based dynamics learning.

B. Dataset Generation and Stratified Sampling

The training dataset is generated through a **Robust Systematic Scenario Generator** that samples vehicle states and control sequences across a wide dynamic range, including near-tire-saturation regimes relevant to extrapolation testing.

1) *Systematic Scenario Space*: Three control dimensions were discretely sampled to ensure comprehensive coverage of operating conditions:

- 1) **Initial longitudinal speed (v_x)**: Six representative speeds ranging from 0.00 m/s to 35.00 m/s.
- 2) **Steering input (δ)**: Time-varying profiles including ramp steer (47.9%), sinusoidal steer (41.7%), constant-rate turns, and step inputs.
- 3) **Longitudinal force (F_x)**: Force commands covering step (23.3%), constant (21.7%), ramp (19.8%), sinusoidal (18.1%), and multi-phase sequences (17.1%) simulating mixed braking and acceleration maneuvers.

2) Dataset Structuring and Filtering:

- **Data size**: The final dataset includes 420 unique scenarios, yielding approximately 1.25 million data points for dense coverage of the vehicle’s dynamic envelope.
- **Inversion for symmetry**: Each scenario was mirrored laterally (sign inversion for lateral states and control inputs) to double the effective dataset and enforce left–right steering symmetry in the learned dynamics.
- **Realism filter**: Scenarios requiring non-physical inputs (e.g. excessively high-frequency steering at low speed or instantaneous stops) were excluded. The resulting dataset enforces physically realistic control limits, with maximum braking $F_x \approx -11,979$ N and maximum acceleration $F_x \approx 7,000$ N.

3) *Stratified Splitting for Generalization*: To ensure representative coverage across different motion patterns, the dataset is divided using stratified sampling:

$$\text{Train:Val:Test} \approx 75\% : 12.5\% : 12.5\%.$$

Stratification is performed across:

- Steering family (e.g. sine vs. ramp),
- Force family (e.g. constant vs. multi-phase),
- Speed bucket (e.g. low, medium, high).

This ensures that validation and test sets contain balanced dynamic diversity, providing a robust assessment of PCARNN generalization to unseen regimes.

C. Geofence Test Scenario Generation and Labeling

The final test dataset used for closed-loop evaluation is created using a **guaranteed-solvable random sampling algorithm** designed to generate dynamically valid containment scenarios.

- 1) **Randomized initialization**: The vehicle’s initial position (p_x, p_y, ψ) is uniformly sampled within the geofence region Ω , producing diverse proximity and approach angles relative to the boundary.
- 2) **Solvability check**: Each sampled initial state undergoes a feasibility test using a *panic brake baseline*. The scenario is retained only if applying maximum braking $F_{x,\min}$ can bring the vehicle to a complete stop within Ω , ensuring physical solvability.
- 3) **Test trajectory labeling**: Each valid scenario is executed under nominal dynamics (without DCBF intervention) and labeled:
 - **Unsafe**: if the trajectory exits Ω at any time.
 - **Safe**: otherwise.
- 4) **Final state capture**: Each run concludes with a forced braking phase until the vehicle comes to rest. The complete trajectory, including states, controls, and safety labels, is stored as a Parquet file for computing CF1 and FPR metrics during evaluation.

Table IX: Summary of best-performing variants (highest CF1) from each model group under **braking-only** settings for both vehicles. One small ($\sim 50K$) and one large ($\sim 150K$) configuration are selected per variant type. Lower is better for FPR and MCD, higher is better for CF1.

Vehicle	Model Variant	Controller	Size (f, g)	Params	CF1 \uparrow	FPR \downarrow	MCD $^+\downarrow$
Audi E-tron	PCARNN (shared)	TTC / TTC	$4 \times 128 / 5 \times 192$	51.3K / 150K	0.800 / 0.802	0.164 / 0.175	2.575 / 2.690
	PCARNN (split)	TTC / TTC	$(3 \times 90, 4 \times 104) / (5 \times 135, 5 \times 135)$	51K / 149K	0.787 / 0.803	0.257 / 0.186	4.294 / 1.800
	Residual	DCBF / DCBF	$4 \times 128 / 5 \times 192$	50.9K / 150K	0.604 / 0.571	0.082 / 0.066	0.207 / 0.236
	Neural ODE	TTC / DCBF	$4 \times 128 / 5 \times 192$	50.9K / 150K	0.705 / 0.522	0.131 / 0.060	-0.912 / 0.136
	Bicycle	TTC	N/A	N/A	0.816	0.137	2.084
	Bicycle Ackermann	BRT	N/A	N/A	0.727	0.284	2.379
Lincoln MKZ	PCARNN (shared)	TTC / TTC	$4 \times 128 / 5 \times 192$	51.3K / 150K	0.714 / 0.806	0.088 / 0.106	0.466 / 0.574
	PCARNN (split)	TTC / DCBF	$(3 \times 90, 4 \times 104) / (5 \times 156, 4 \times 128)$	51K / 150K	0.718 / 0.640	0.088 / 0.050	0.466 / 0.411
	Residual	DCBF / DCBF	$4 \times 128 / 5 \times 192$	50.9K / 150K	0.657 / 0.624	0.069 / 0.056	0.452 / 0.389
	Neural ODE	DCBF / TTC	$4 \times 128 / 5 \times 192$	50.9K / 150K	0.657 / 0.651	0.069 / 0.106	0.452 / 0.229
	Bicycle	TTC	N/A	N/A	<u>0.783</u>	0.131	0.673
	Bicycle Ackermann	BRT	N/A	N/A	0.672	0.206	0.778

Table X: Comparison of model architecture variants under different controllers (TTC and DCBF) for braking only scenarios. Lower is better for FPR and MCD, and higher is better for CF1.

Vehicle	Variant	Size	Controller	Params	CF1 \uparrow	FPR \downarrow	MCD $^+\downarrow$
Audi E-tron	PCARNN (shared)	4×128	TTC	51.3K	0.800	0.164	2.575
	PCARNN (shared)	5×192	TTC	150K	0.802	0.175	2.690
	PCARNN (split)	$f = 3 \times 90, g = 4 \times 104$	TTC	51K	0.787	0.257	4.294
	PCARNN (split)	$f = 5 \times 135, g = 5 \times 135$	TTC	149K	0.794	0.175	1.643
	PCARNN (split)	$f = 5 \times 156, g = 4 \times 128$	TTC	150K	0.798	0.180	1.827
	PCARNN (split)	$f = 4 \times 128, g = 5 \times 156$	TTC	150K	0.803	0.186	1.800
	Residual	4×128	TTC	50.9K	0.287	0.131	-0.192
	Residual	5×192	TTC	150K	0.449	0.093	-0.023
	Neural ODE	4×128	TTC	50.9K	0.705	0.131	0.403
	Neural ODE	5×192	TTC	150K	0.350	0.109	-0.098
	Bicycle	N/A	TTC	N/A	0.816	0.137	2.084
	PCARNN (shared)	4×128	DCBF ($\gamma = 0.40$)	51.3K	0.624	0.093	0.475
	PCARNN (shared)	5×192	DCBF ($\gamma = 0.45$)	150K	0.595	0.071	0.411
	PCARNN (split)	$f = 3 \times 90, g = 4 \times 104$	DCBF ($\gamma = 0.45$)	51K	0.609	0.071	0.336
	PCARNN (split)	$f = 5 \times 135, g = 5 \times 135$	DCBF ($\gamma = 0.45$)	149K	0.603	0.066	0.308
	PCARNN (split)	$f = 5 \times 156, g = 4 \times 128$	DCBF ($\gamma = 0.45$)	150K	0.609	0.071	0.319
	PCARNN (split)	$f = 4 \times 128, g = 5 \times 156$	DCBF ($\gamma = 0.45$)	150K	0.612	0.071	0.284
	Residual	4×128	DCBF ($\gamma = 0.45$)	50.9K	0.537	0.060	0.088
	Residual	4×128	DCBF ($\gamma = 0.40$)	50.9K	0.604	0.082	0.207
	Residual	5×192	DCBF ($\gamma = 0.45$)	150K	0.571	0.066	0.236
	Neural ODE	4×128	DCBF ($\gamma = 0.45$)	50.9K	0.528	0.060	0.113
	Neural ODE	4×128	DCBF ($\gamma = 0.40$)	50.9K	0.556	0.077	0.203
	Neural ODE	5×192	DCBF ($\gamma = 0.45$)	150K	0.522	0.060	0.136
	Bicycle	N/A	DCBF ($\gamma = 0.40$)	N/A	0.622	0.098	0.454
	Bicycle Ackermann	N/A	BRT	N/A	0.727	0.284	2.379
Lincoln MKZ	PCARNN (shared)	4×128	TTC	51.3K	0.714	0.088	0.466
	PCARNN (shared)	5×192	TTC	150K	0.806	0.106	0.574
	PCARNN (split)	$f = 3 \times 90, g = 4 \times 104$	TTC	51K	0.718	0.081	0.564
	PCARNN (split)	$f = 5 \times 135, g = 5 \times 135$	TTC	149K	0.562	0.094	0.249
	PCARNN (split)	$f = 5 \times 156, g = 4 \times 128$	TTC	150K	0.552	0.106	0.293
	PCARNN (split)	$f = 4 \times 128, g = 5 \times 156$	TTC	150K	0.536	0.081	0.155
	Residual	4×128	TTC	50.9K	0.431	0.081	-0.068
	Residual	5×192	TTC	150K	0.623	0.106	0.215
	Neural ODE	4×128	TTC	50.9K	0.546	0.125	0.113
	Neural ODE	5×192	TTC	150K	0.651	0.106	0.229
	Bicycle	N/A	TTC	N/A	0.783	0.131	0.673
	PCARNN (shared)	4×128	DCBF ($\gamma = 0.40$)	51.3K	0.676	0.069	0.486
	PCARNN (shared)	5×192	DCBF ($\gamma = 0.45$)	150K	0.634	0.056	0.435
	PCARNN (split)	$f = 3 \times 90, g = 4 \times 104$	DCBF ($\gamma = 0.45$)	51K	0.634	0.056	0.477
	PCARNN (split)	$f = 5 \times 135, g = 5 \times 135$	DCBF ($\gamma = 0.45$)	149K	0.634	0.056	0.407
	PCARNN (split)	$f = 5 \times 156, g = 4 \times 128$	DCBF ($\gamma = 0.45$)	150K	0.640	0.050	0.411
	PCARNN (split)	$f = 4 \times 128, g = 5 \times 156$	DCBF ($\gamma = 0.45$)	150K	0.640	0.050	0.422
	Residual	4×128	DCBF ($\gamma = 0.45$)	50.9K	0.598	0.056	0.345
	Residual	4×128	DCBF ($\gamma = 0.40$)	50.9K	0.657	0.069	0.452
	Residual	5×192	DCBF ($\gamma = 0.45$)	150K	0.624	0.056	0.389
	Neural ODE	4×128	DCBF ($\gamma = 0.45$)	50.9K	0.601	0.056	0.389
	Neural ODE	4×128	DCBF ($\gamma = 0.40$)	50.9K	0.657	0.069	0.483
	Neural ODE	5×192	DCBF ($\gamma = 0.45$)	150K	0.615	0.056	0.402
	Bicycle	N/A	DCBF ($\gamma = 0.40$)	N/A	0.705	0.069	0.499
	Bicycle Ackermann	N/A	BRT	N/A	0.672	0.206	0.778