Student Conference Proceedings
Vol 2, No 1, 2026, Article ID 2562, 4 Pages

Student Conference Lübeck

*Proceedings Article*

# ParkingPPOEnv: A Curriculum-Based Reinforcement Learning Framework for Autonomous Parking

Batbandi Gerelkhuu [a,*] · Max Studt [b] · Georg Schildbach [b]

[a] Study program Robotics and Autonomous Systems, Universität zu Lübeck, Lübeck, Germany
[b] Institute for Electrical Engineering in Medicine, Universität zu Lübeck, Lübeck, Germany
*Corresponding author, email: batbandi.gerelkhuu@student.uni-luebeck.de; m.studt@uni-luebeck.de; georg.schildbach@uni-luebeck.de

Received 11 February 2026; Accepted 17 August 2026; Published online 25 August 2026



**Abstract**

Autonomous parking presents a unique challenge in robotics, requiring the synthesis of strategic decision-making and precise non-holonomic control. Traditional rule-based planners often lack the flexibility to handle dynamic environments, while end-to-end Reinforcement Learning (RL) approaches struggle with sample efficiency and convergence in dense obstacle scenarios.

This paper addresses these challenges through the development of *ParkingPPOEnv*, a high-fidelity Gymnasium environment for training autonomous agents. The system integrates a kinematic bicycle model, a 16-ray planar LIDAR perception system, and an Oriented Bounding Box (OBB) collision detection engine. To facilitate learning, we implemented a multi-phase curriculum strategy that progressively increases environmental complexity from 0 % to 90 % obstacle occupancy. This paper details the system architecture, the custom reward function design, and analyzes the agent's performance. Initial results demonstrate successful maneuvering in static environments, though challenges remain in dense obstacle avoidance, highlighting the need for future hierarchical control integration.

## I. Introduction

The deployment of autonomous vehicles (AVs) in unstructured environments, such as parking lots, remains an open research problem. Unlike highway driving, which is primarily longitudinal, parking requires complex lateral maneuvers, frequent direction changes, and precise positioning within centimeters of obstacles.

While Model Predictive Control (MPC) offers safety guarantees and precise trajectory tracking, it often requires high computational resources and struggles with high-level tactical decisions when the horizon is limited [1]. Conversely, Reinforcement Learning (RL) has shown promise in learning complex policies from experience but suffers from the "sparse reward" problem in long-horizon navigation tasks [2].

### I.I. Related Work

Recent literature has explored various approaches to the autonomous parking problem. Paden et al. [3] provide a comprehensive survey of motion planning, highlighting that classical graph-search methods (like A* or Hybrid A*) often struggle to efficiently satisfy the non-holonomic constraints of car-like vehicles in real-time.

In the domain of learning-based control, Chen et al. [4] demonstrate that model-free Deep RL can solve urban driving tasks end-to-end. However, their approach may lack stability in tight, cluttered spaces where precision is paramount. To address these limitations, Naveed et

DOI 10.18416/SCP.2026.2562 

al. [5] and Lu et al. [6] propose Hierarchical Reinforcement Learning (HRL). These architectures decompose the problem into high-level tactical decisions (e.g., "park left") and low-level execution, significantly improving sample efficiency. Similarly, Shi et al. [7] combine MPC with RL, using the learner to guide the optimization process to avoid local minima. Our work builds upon these foundations by creating a lightweight yet physically rigorous testbed to evaluate Curriculum Learning strategies for end-to-end parking policies without requiring heavy hierarchical structures from the outset.

### I.II. Objective

The primary objective of this paper is to build a robust simulation framework that bridges the gap between simplified grid-worlds and computationally heavy simulators (like CARLA). We introduce *ParkingPPOEnv*, a custom environment built on the Gymnasium interface [8].

## II. Methods and Materials

The simulation is designed to be computationally efficient to allow for millions of training steps while maintaining physical fidelity.

### II.I. System Architecture

The ego vehicle is modeled using a non-holonomic Kinematic Bicycle Model. This model captures the essential constraints of car-like robots, specifically the inability to move sideways (non-holonomic constraint). The simulation operates at a discrete time step of $\Delta t = 0.1\,\mathrm{s}$. The state evolution is defined by:

$$\dot{x} = v\cos(\psi) \tag{1}$$

$$\dot{y} = v\sin(\psi) \tag{2}$$

$$\dot{\psi} = \frac{v}{L}\tan(\delta) \tag{3}$$

Where $x$ and $y$ represent the position of the center of the rear axle in meters, $\psi$ is the vehicle heading angle in radians, $v$ is the longitudinal velocity in meters per second, and $\delta$ is the steering angle in radians. The vehicle parameters are fixed to a wheelbase of $L = 4\,\mathrm{m}$ and a width of $W = 2\,\mathrm{m}$.

The control inputs are constrained by physical limits:

$$v \in [-5.0, 5.0]\,\mathrm{m\,s^{-1}}, \qquad \delta \in [-0.785, 0.785]\,\mathrm{rad}$$

To prevent the agent from overfitting to a single map, the environment procedurally generates a parking lot with four distinct geometric sections: Upper (Perpendicular), Left (Angled $-135°$), Middle (Vertical), and Right (Angled $-45°$). Each section contains approximately 8 to 12 individual parking slots, each sized $2.5\,\mathrm{m} \times 5.5\,\mathrm{m}$ with a 6 m wide driving lane, resulting in a total lot size of approximately $40\,\mathrm{m} \times 30\,\mathrm{m}$. The goal slot and the vehicle's initial position are randomized within the lot on every episode reset.

### II.II. Perception and Physics

A critical contribution of this project is the implementation of precise collision physics. Standard grid-world environments often approximate vehicles as circles, creating "wasted space" at the corners and preventing tight maneuvers. We implement an Oriented Bounding Box (OBB) collision engine. This utilizes the Separating Axis Theorem (SAT) to detect overlaps between the rotated rectangle of the ego vehicle and static obstacles.

The agent perceives its surroundings via a simulated 16-ray planar LIDAR system (360° field of view). The observation space ($S_t \in \mathbb{R}^{23}$) consists of:

- **Navigation (7):** goal relative position $(\Delta x, \Delta y)$, heading $(\cos\psi, \sin\psi)$, velocity $(v)$, and goal orientation $(\cos\alpha, \sin\alpha)$.
- **Lidar (16):** Normalized distances to the nearest OBB intersection along 16 radial vectors.

### II.III. Reward Function Design

RL agents often struggle with "sparse rewards" in navigation (getting +1 only at the goal and 0 otherwise). To accelerate convergence, we implement a dense reward signal

$$R_{total} = (1 - \sqrt[3]{C_{prox}}) + R_{collision} + R_{success}, \tag{4}$$

where $C_{prox}$ is a weighted proximity cost based on the agent's state error:

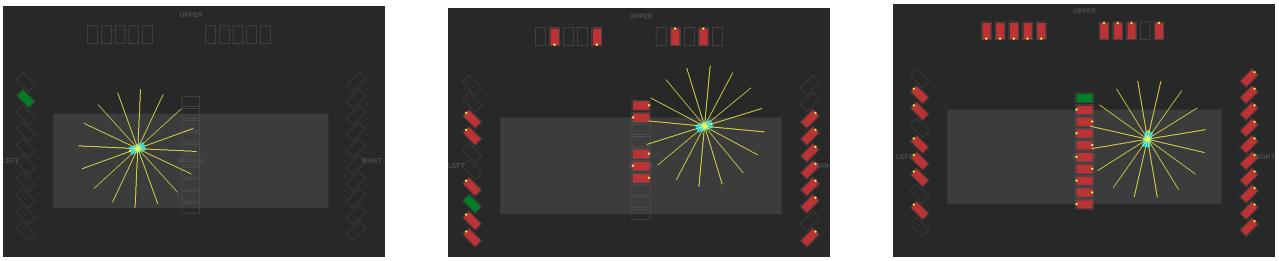
$$C_{prox} = w_x|\Delta x| + w_y|\Delta y| + w_\alpha|\Delta\alpha| + w_v|v|, \tag{5}$$

The cubic root transformation ($\sqrt[3]{\cdot}$) is selected empirically to ensure high precision. This results in a steeper gradient when the agent is near the target, forcing the policy to make fine-grained adjustments for perfect alignment rather than settling for a "close enough" position. The specific weights used in our experiments are detailed in Table 1.

**Table 1:** Reward Function Configuration

| Component | Weight/Value | Description |
|---|---|---|
| X-pos. weight ($w_x$) | 0.35 | Penalty for $\Delta x$ error |
| Y-pos. weight ($w_y$) | 0.35 | Penalty for $\Delta y$ error |
| Angle weight ($w_\alpha$) | 0.20 | Penalty for orientation error |
| Vel. weight ($w_v$) | 0.10 | Penalty for excess speed |
| Success reward | +1000 | Threshold $< 4 \cdot 10^{-4}$ |
| Collision penalty | −200 | Fixed penalty on impact |

DOI 10.18416/SCP.2026.2562



**(a)** 0% Occupancy **(b)** 40% Occupancy **(c)** 90% Occupancy

**Figure 1:** Curriculum progression: The agent starts in an empty lot (a) to learn kinematics, and is progressively exposed to denser environments (b, c) to learn obstacle avoidance.

This reward formulation is chosen to balance global convergence with local precision. In preliminary experiments, purely quadratic penalties result in premature convergence with residual pose error, whereas the proposed shaping encourages consistent fine-scale corrections near the goal.

## II.IV. Curriculum Learning Strategy

Directly training an agent in a dense environment often leads to policy collapse, where the agent learns to remain stationary to avoid collision penalties. To mitigate this, we employ a fixed-schedule Curriculum Learning approach:

**Phase Structure:** The training is divided into six distinct phases, each consisting of 2 million time steps, for a total of 12 million steps.

**Complexity Progression:** Phases correspond to increasing obstacle density levels from 0 % to 90 %, allowing the agent to stabilize its control policy in open space before refining it for complex collision avoidance.

**Termination Conditions:** Episodes terminate upon a collision (OBB overlap), reaching the target threshold, or exceeding a 1,000-step limit.

**Success Threshold:** A successful trial requires a combined state error $C_{prox} < 4 \cdot 10^{-4}$. This equates to high-precision positioning with a Euclidean distance error within 1 cm and orientation disalignment within 1°.

# III. Results and Discussion

The agent is trained using Proximal Policy Optimization (PPO) [9] for 12 million steps. We utilize the standard hyperparameters provided by Stable Baselines3 (Learning Rate: $3 \cdot 10^{-4}$, Batch Size: 64, Gamma: 0.99).

## III.I. Performance Metrics

We evaluated the agent's performance across the curriculum phases. Figure 2 presents the key metrics extracted from TensorBoard.

1. **Reward Convergence:** The Mean Episode Reward (Fig. 2a) shows a rapid increase during Phase 1 (0–2M steps), indicating the agent successfully learned to reach the parking spot. The reward curve oscillates in later phases, reflecting the difficulty of higher densities.
2. **Episode Length:** The Mean Episode Length (Fig. 2b) decreases initially as the agent learns to drive directly to the target. In higher occupancy phases, the length oscillates slightly, reflecting the more complex path planning required to navigate around obstacles.
3. **Success Rate Stability:** The Success Rate (Fig. 2c) reaches near 0.80 (80 %) for the empty environment. As occupancy rises, the success rate remains robust. However, a drop is observed in the 90 % density phase, suggesting the current observation space (16 LIDAR rays) may be insufficient for extremely tight gaps. Furthermore, a portion of the successful episodes stems from favorable initializations where no obstacles block the path between the vehicle and the goal, rather than from active avoidance maneuvers. The per-phase breakdown is summarized in Table 2.

**Table 2:** Success Rate per Curriculum Phase

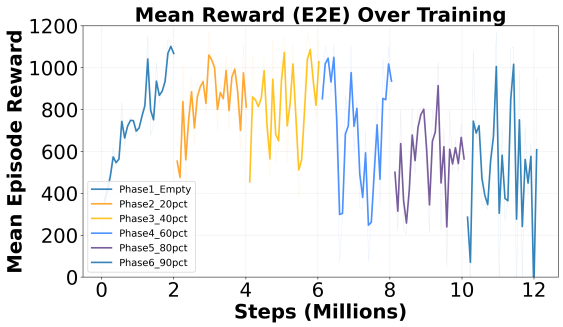
| Phase | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Occupancy | 0 % | 20 % | 40 % | 60 % | 80 % | 90 % |
| Success Rate | 0.20 | 0.52 | 0.65 | 0.64 | 0.59 | 0.62 |

## III.II. Analysis of Failure Modes

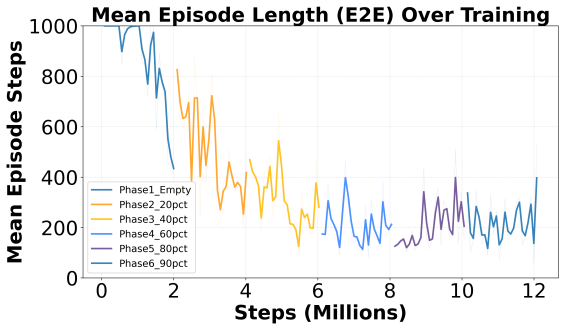
Visual analysis confirms that the failures in dense phases are largely due to "grazing" collisions. The agent tends to optimize for the shortest path to the +1000 reward, ignoring the risk of close-proximity maneuvers. This behavior highlights a known issue in RL navigation: *reward hacking*. The agent accepts a small probability of collision to minimize the path length. As noted by Bernhard et al. [10], pure RL often lacks the safety guarantees of

DOI 10.18416/SCP.2026.2562 

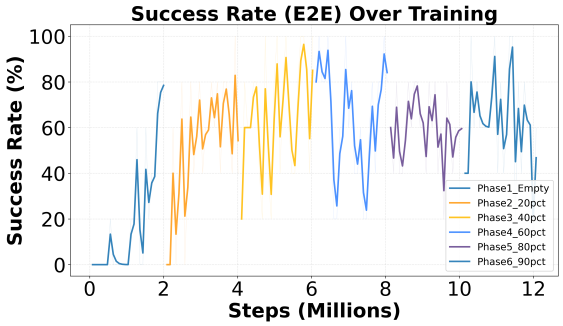
classical planners. Attempts to mitigate this by simply increasing negative rewards often result in the agent refusing to move ("freezing") to avoid risk. Therefore, a more sophisticated approach is required.




**(a)** Mean Episode Reward




**(b)** Mean Episode Length




**(c)** Success Rate

**Figure 2:** Training Metrics: (a) Rewards stabilize early but fluctuate with obstacle introduction. (b) Trajectories become efficient over time. (c) Success rate stabilizes around 70% even in moderate density.

# IV. Conclusion

We successfully developed *ParkingPPOEnv*, a high-fidelity environment for autonomous parking. By integrating precise OBB physics and a curriculum pipeline, we created a robust testbed for RL algorithms. The results demonstrate that while Curriculum Learning effectively bootstraps the policy, pure end-to-end RL struggles with safety in high-density scenarios. Future work will therefore focus on integrating hierarchical or hybrid planning methods to combine the efficiency of RL with the safety guarantees of classical approaches.

## IV.I. Future Work

Future iterations of this project will focus on two areas:

1. **Adaptive Reward Shaping:** Since naïve penalties caused freezing behavior, future work will investigate curriculum-based safety margins that tighten as the agent's precision improves.
2. **Hierarchical Control:** The RL policy will generate high-level target states, tracked by an MPC layer enforcing hard safety constraints [7].

# Acknowledgments

The work has been carried out at the Institute for Electrical Engineering in Medicine, Universität zu Lübeck, supervised by Prof. Dr. Georg Schildbach. We thank Max Studt for his mentorship and technical guidance.

# Author's statement

Conflict of interest: Authors state no conflict of interest.

# References

[1] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, Kinematic and dynamic vehicle models for autonomous driving control design, in *IEEE Intell. Vehicles Symp. (IV)*, 1094–1099, 2015. doi:10.1109/IVS.2015.7225830.

[2] Z. Zhu and H. Zhao. A survey of deep RL and IL for autonomous driving policy learning. *IEEE Trans. Intell. Transp. Syst.*, 23(9):14043–14065, 2021, doi:10.1109/TITS.2021.3134702.

[3] B. Paden, M. Cap, S. Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans. Intell. Vehicles*, 1(1):33–55, 2016, doi:10.1109/TIV.2016.2578706.

[4] J. Chen, B. Yuan, and M. Tomizuka, Model-free deep reinforcement learning for urban autonomous driving, in *IEEE Intell. Transp. Syst. Conf. (ITSC)*, 2765–2771, 2019. doi:10.1109/ITSC.2019.8917306.

[5] K. Naveed, Z. Qiao, and J. Dolan, Trajectory planning for autonomous vehicles using hierarchical reinforcement learning, in *IEEE Intell. Transp. Syst. Conf. (ITSC)*, 601–606, 2021. doi:10.1109/ITSC48978.2021.9564412.

[6] X. Lu, F. Fan, and T. Wang, Action and trajectory planning for urban autonomous driving with hierarchical reinforcement learning, in *IEEE/RSJ Intell. Robots Syst. (IROS)*, 2020.

[7] J. Shi, K. Li, C. Piao, J. Gao, and L. Chen. Model-based predictive control and reinforcement learning for planning vehicle-parking trajectories. *Sensors*, 23(16):7124, 2023, doi:10.3390/s23167124.

[8] M. Towers *et al.* Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2303.04117*, 2023.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[10] J. Bernhard, R. Gieselmann, K. Esterle, and A. Knoll, Experience-based heuristic search: Robust motion planning with deep Q-learning, in *IEEE Intl. Conf. Robotics and Automation (ICRA)*, 2021.