ParkRL: Learning Generalizable Parking Policies via Deep Reinforcement Learning

Raine Cui, Georgia Lin, James Liu, Tianyao Ren
December 5th, 2025

Abstract

Autonomous parking is a critical component of self-driving systems, requiring precise control and robust generalization across diverse scenarios. We present ParkRL, a deep reinforcement learning approach that learns a single end-to-end policy capable of handling perpendicular, angled, and parallel parking in randomized environments with dense obstacles. Our key contributions include: (1) a novel reward design using harmonic mean and exponential decay that prevents local minima and encourages precise alignment, (2) a systematic three-stage curriculum learning strategy essential for convergence, and (3) comprehensive ablation studies demonstrating the impact of sensor resolution, reward shaping, and training curriculum. The learned policy exhibits emergent human-like behaviors such as reverse parking. We deploy the trained model to an interactive web demonstration, providing real-time inference. Demo: `https://pkucuipy.github.io/rl-final`

1 Introduction

Autonomous parking represents a fundamental challenge in self-driving systems, requiring vehicles to navigate from arbitrary starting positions to precise target poses within confined environments.

The task is technically challenging for several reasons. First, vehicles are subject to **non-holonomic constraints**—unlike omnidirectional robots, cars cannot slide sideways and must perform complex multi-point maneuvers to adjust lateral position. Second, the task requires **continuous control** over both throttle and steering, operating in a high-dimensional action space. Third, parking scenarios involve **dense obstacle environments** with walls, other vehicles, and barriers that require precise spatial awareness to avoid collisions. Finally, the task exhibits **sparse rewards**: naive formulations provide no feedback until the vehicle is perfectly parked, making exploration extremely difficult.

Furthermore, real-world parking systems must generalize across diverse scenarios. Parking lots exhibit three common configurations: perpendicular (90°), angled (45°), and parallel (0°) parking, each requiring fundamentally different maneuvering strategies. Traditional approaches often require separate hand-tuned controllers for each type, limiting their practical deployment.

Our approach. We formulate autonomous parking as a continuous control reinforcement learning problem and train a single neural policy that generalizes across all parking types through environment randomization. Our key contributions are:

- **Novel reward design**: We introduce a reward function using harmonic mean to enforce simultaneous satisfaction of multiple objectives, and exponential decay curves that prevent "good enough" local minima.
- **Systematic curriculum learning**: We demonstrate that a carefully designed three-stage curriculum (low speed $\rightarrow$ progressive speed increase $\rightarrow$ obstacle density increase) is *essential* for convergence. Training from scratch fails to learn even basic behaviors.

- **Comprehensive ablations**: We conduct systematic studies on sensor resolution (8 vs 64 LiDAR rays), reward curve design (linear vs exponential decay), and training strategies (curriculum vs from-scratch), providing insights into crucial design decisions.
- **Emergent behaviors**: The learned policy independently discovers human-like strategies such as reverse bay parking and reverse parallel parking, matching techniques taught in driving schools without any explicit programming.
- **Complete deployment pipeline**: We implement a full training-to-deployment system (Python training $\rightarrow$ ONNX $\rightarrow$ JavaScript inference) with an interactive web demonstration.

2 Related Work

Traditional methods. Autonomous parking has traditionally been addressed through **decoupled planning and control** approaches. These methods separate the problem into two stages: first computing a geometric path using motion primitives, optimization-based planners, or analytical solutions, then tracking this path with a low-level controller such as Model Predictive Control (MPC) [2]. While these approaches can provide theoretical guarantees in structured scenarios, they face several limitations. First, they typically require **hand-tuned rules specific to each parking type**—perpendicular, angled, and parallel parking each need different planning strategies. Second, the rigid separation between planning and control limits adaptability; replanning is computationally expensive and struggles with real-time constraints. Third, these methods often lack robustness to variations in parking lot layouts, requiring extensive manual tuning for different configurations.

Reinforcement learning approaches. Reinforcement learning offers an alternative through end-to-end learning, where a single neural policy maps sensor observations directly to control actions. Existing RL environments for parking, such as `parking-v0` in the Highway-Env suite [3], provide standardized benchmarks but exhibit key limitations. These environments typically use simple kinematic observations (vehicle position, velocity, heading) without rich spatial sensing, operate in **fixed scenarios** with predetermined obstacle configurations, and lack systematic studies of generalization to unseen layouts. Policies trained in these settings may overfit to specific parking lot configurations.

Our focus. This work addresses the generalization challenge through **systematic environment randomization** during training. Every episode presents a completely different parking lot with random layouts, obstacle placements, and target positions, forcing the policy to learn generalizable strategies rather than memorizing specific scenarios. We investigate key design decisions through ablation studies: How does sensor resolution (Section 4.3) impact policy robustness? How does reward curve design (Section 4.4) affect convergence and final performance? Is curriculum learning (Section 4.5) necessary for multi-objective tasks? These systematic investigations provide insights into crucial architectural and algorithmic choices for learning robust parking policies.

3 Method

3.1 Problem Formulation

We formulate autonomous parking as a continuous control Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$.

State space $\mathcal{S}$ consists of a 71-dimensional observation vector encoding:

- **LiDAR sensor** (64 dimensions): We deploy 64 rays uniformly distributed over 360° around the vehicle, each with a maximum range of 10 meters. Each ray returns the normalized distance to the nearest obstacle: $r_i \in [0, 1]$ where 0 indicates maximum range (no obstacle detected) and 1 indicates immediate proximity.
- **Ego-motion** (2 dimensions): Current vehicle speed $v \in [-5, 5]$ m/s (normalized) and steering angle $\theta \in [-\pi/4, \pi/4]$ radians (normalized).
- **Target pose** (4 dimensions): The target parking spot's position and orientation relative to the vehicle in the ego-centric coordinate frame:
  - Relative position: forward distance dx and lateral distance dy
  - Relative heading: $(\cos \Delta\theta, \sin \Delta\theta)$ to ensure continuity at angle wraparound

- **Stopping progress** (1D): Normalized count of consecutive frames with speed < 0.1 m/s.

All observations are in the vehicle's ego-centric coordinate frame, meaning the state representation is invariant to global translation and rotation. This design choice dramatically improves generalization—the policy doesn't need to learn separate strategies for different absolute positions.

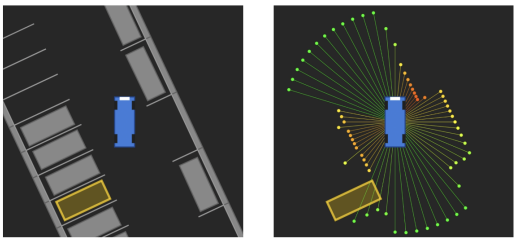

Figure 1: Observations of an Example Scene

Action space $\mathcal{A}$ consists of 2-dimensional continuous control normalized to $[-1, 1]^2$:

$$\mathbf{a} = [\text{target\_speed\_norm}, \text{target\_steering\_norm}] \in [-1, 1]^2 \tag{1}$$

These normalized actions are scaled to physical ranges: target speed to $[-5, 5]$ m/s and target steering to $[-\pi/4, \pi/4]$ radians.

Importantly, actions represent *targets* rather than instantaneous commands. The vehicle's actual speed and steering angle transition smoothly toward these targets with rate limits:

$$\dot{v} \leq 5 \text{ m/s}^2 \quad \text{(max acceleration)} \tag{2}$$

$$\dot{\theta} \leq \pi/4 \text{ rad/s} \quad \text{(max steering rate)} \tag{3}$$

This rate-limited design prevents unrealistic high-frequency oscillations (e.g., rapid switching between full left and full right steering) and better reflects real vehicle dynamics.

Dynamics $\mathcal{T}$ uses a kinematic bicycle model with the rear axle as the reference point:

$$\dot{x} = v\cos(\psi), \quad \dot{y} = v\sin(\psi), \quad \dot{\psi} = \frac{v\tan(\theta)}{L}$$

where (x, y) is the rear axle position, ψ is the vehicle heading, θ is the steering angle, and $L = 3$ m is the wheelbase. The vehicle has length 4.5 m and width 1.8 m.

Collision detection uses the Separating Axis Theorem to check for intersections between the vehicle's oriented bounding box and all obstacles. Upon collision, the vehicle's velocity is reversed and halved: $v \leftarrow -0.5v$. Episodes do not terminate on collision to allow the policy to learn recovery behaviors.

Reward function $\mathcal{R}$ is detailed in Section 3.3.

3.2 Environment Design: Randomized Parking Lots

To enforce generalization, we train exclusively on procedurally generated parking lots. Each episode samples a completely new environment with the following randomization:

Parking lot layout: The environment consists of a central road (width randomly sampled from 6-10 meters) with parking areas on both sides. Each side contains a variable number of parking spots of one of three types: **perpendicular** (90°): standard bay parking; **angled** (45°): diagonal parking; and **parallel** (0°): street parking. The type is randomly sampled per episode with configurable weights (we use 1:1:3 to oversample parallel parking which is inherently more difficult).

Obstacle placement: Each parking spot has a probability $p_{\text{obs}} \in [0, 1]$ of containing an obstacle vehicle. Additionally, barriers are randomly placed adjacent to spots with probability 0.5.

Target selection: One empty spot is randomly designated as the target. The vehicle is initialized on the central road with random position, heading $\psi_0 \in [-\pi/2, \pi/2]$, and initial speed $v_0 \in [-1, 1]$ m/s.

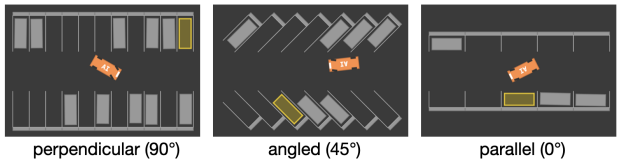

Figure 2: Examples of three parking lot types with different configurations

3.3 Reward Design

The reward function combines a sparse success reward with dense shaping rewards to guide learning.

Success reward (up to +100 points) is granted when the vehicle remains stationary (speed < 0.1 m/s) for a threshold duration (0.2-1.0 seconds depending on curriculum stage; e.g., 0.2s = 2 frames at 10 FPS). The reward evaluates parking quality using three metrics:

1. **Overlap ratio** r_{overlap}: Fraction of vehicle area overlapping the target spot, computed via Monte Carlo sampling (1000 random points within vehicle bounds).
2. **Distance score** r_{dist}: Exponential decay based on center-to-center distance d:
$$r_{\text{dist}} = \exp(-3 \cdot d/d_{\text{max}}) \tag{4}$$
where d_{max} is the diagonal of the parking spot.
3. **Angle score** r_{angle}: Exponential decay based on heading error $\Delta\theta$, considering both forward and reverse orientations (since either direction is acceptable):
$$r_{\text{angle}} = \exp(-5 \cdot \min(|\Delta\theta|, |\Delta\theta - \pi|)/\pi) \tag{5}$$

The final success reward uses the **harmonic mean** of these three scores:

$$R_{\text{success}} = 100 \cdot \frac{3}{\frac{1}{r_{\text{overlap}}} + \frac{1}{r_{\text{dist}}} + \frac{1}{r_{\text{angle}}}} \tag{6}$$

We use harmonic mean rather than arithmetic mean because it exhibits a "short-board effect"—all three metrics must be high to achieve a high overall score. For example, perfectly centered with zero distance but perpendicular to the spot yields near-zero reward, correctly penalizing poor parking.

The choice of exponential decay with decay rates 3 and 5 is crucial and validated in Section 4.4. These steep curves prevent "good enough" local minima where the agent settles for mediocre alignment.

Intermediate rewards provide "breadcrumbs" for exploration:

- **Distance shaping** ± 1 point per meter: Reward for increasing/decreasing distance to target.
- **Time penalty** -0.5 points per second: Encourages efficient parking.
- **Collision penalty** $-(1 + 3v^2)$ points: Discourages crashes, with quadratic dependence on velocity to heavily penalize high-speed collisions.
- **Gear-shift penalty** -1 point: Triggered when velocity changes sign (forward $\leftrightarrow$ backward), discouraging rapid oscillations that simulate unrealistic gear shifting.

3.4 Training Algorithm and Architecture

We employ Proximal Policy Optimization (PPO) [5] with the following hyperparameters:

Training is performed using Stable-Baselines3 [4]. We use a standard MLP policy, where a shared feature extractor feeds into separate policy head (outputs action mean $\mu_\theta(\mathbf{s})$ and learnable log-std) and value head. The policy is a squashed Gaussian: $\mathbf{a} \sim \tanh(\mathcal{N}(\mu_\theta(\mathbf{s}), \sigma_\theta^2))$.

We experimented with 1D-CNN architectures to process the 64-ray LiDAR as a circular image (using circular padding), but found MLP sufficient for this task and more stable during training.

Hyperparameter	Value
Learning rate	3×10^{-4}
Rollout buffer size	4096 steps
Mini-batch size	1024 samples
Training epochs per update	10
Discount factor γ	0.99
GAE parameter λ	0.95
Clip range	0.2
Target KL divergence	0.05 (early stopping)
Entropy coefficient	0.01
Policy std initialization	$\log \sigma_0 = -1.0$ ($\sigma_0 \approx 0.37$)
Policy std clamping	$[\exp(-5), \exp(-1)] \approx [0.0067, 0.37]$

Table 1: PPO hyperparameters used for training.

3.5 Curriculum Learning Strategy

A critical finding (validated in Section 4.5) is that training from scratch with the final task configuration *fails to converge*. We instead employ a carefully designed three-stage curriculum [1]:

Stage 1: Learn basic parking (2 hours)

- Maximum speed: 1 m/s, Obstacle density: 20%
- Stopping threshold: 0.2 seconds (2 frames at 10 FPS)
- *Goal*: Learn to approach target spot and stop.

Stage 2: Progressive speed adaptation (40 minutes, 4 rounds)

- Speed sequence: $1.0 \rightarrow 1.5 \rightarrow 2.25 \rightarrow 3.375 \rightarrow 5.0$ m/s
- Each fine-tuning round: $\sim$10 minutes
- *Goal*: Gradually decrease relative vehicle responsiveness.

Stage 3: Increase obstacle robustness (3 hours)

- Obstacle density: $0.2 \rightarrow 0.4 \rightarrow 0.8$
- Stopping threshold: $0.2 \rightarrow 0.3 \rightarrow 1.0$ seconds
- *Goal*: Learn collision avoidance and precise maneuvering in cluttered spaces.

3.6 Deployment Pipeline

We implement a complete training-to-deployment system:

1. **Training**: Python environment using Gymnasium API with PPO (Stable-Baselines3 library).
2. **Export**: Trained policy exported to ONNX format using deterministic action selection.
3. **Web deployment**: Interactive JavaScript implementation with in-browser inference.

The Python and JavaScript environments share identical physics configurations. The web demo provides an interactive experience where users can toggle AI control and visualize sensor observations.

4 Experimental Results

4.1 Training Convergence

Figure 3 shows the training progression across all three curriculum stages. Starting from an average reward of -50 (random exploration with collisions and timeouts), the policy steadily improves:

Final performance metrics (evaluated with converged policy):

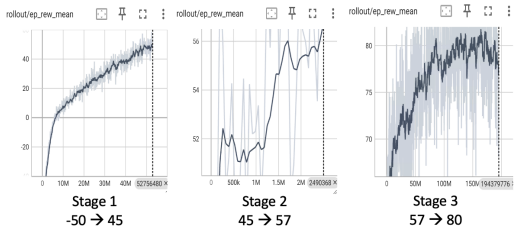

Figure 3: Training curves showing reward progression across three curriculum stages.

- Average reward: 79.8
- Success rate: 96.2%
- Average parking speed: 3.1 m/s (distance / time for successful episodes)
- Collision rate: 0.146 times per episode

4.2 Emergent Behaviors

A remarkable finding is that the learned policy independently discovers parking strategies matching human driving school techniques, despite *no explicit programming or demonstration*. We observe:

- **Reverse bay parking**: For perpendicular (90°) and angled (45°) spots, the policy preferentially backs into the spot rather than driving forward.
- **Reverse parallel parking**: For parallel (0°) spots, the policy approaches from the front, backs past the target spot, then reverses in with precise angle control. This mirrors the "reverse parallel parking" maneuver taught to human drivers.

These behaviors emerge purely from reward optimization—the reward function does not specify *how* to park, only the desired final state (proper alignment with target spot). The fact that RL discovers the same strategies as human experts suggests these may indeed be optimal solutions to the parking problem under the given vehicle dynamics and constraints.

Interestingly, the reward function is direction-agnostic (allows both forward and reverse parking via $\min(|\Delta\theta|, |\Delta\theta - \pi|)$ in angle scoring), yet the policy learns a strong preference for reversing. This likely arises from the non-holonomic constraints: backing into a spot allows finer lateral adjustment using the front wheels' larger turning radius.

4.3 Ablation Study 1: LiDAR Resolution

Research question: Is dense LiDAR coverage (64 rays) necessary, or can the policy operate with sparser sensors (8 rays)?

Setup: We train two policies with identical configurations except for LiDAR resolution.

Finding: The 8-ray policy exhibits severe failure modes due to **blind spots**, leading to:

1. **Unexpected collisions**: The policy approaches obstacles that are invisible in its current observation, then collides when the obstacle enters sensor view.
2. **Jumpy, unstable control**: When an obstacle suddenly appears between time steps t and $t + 1$ (moving from a blind spot into sensor range), the policy's actions change abruptly, causing oscillatory behavior.

This problem is exacerbated by our **stateless policy** design—the policy has no memory of previously observed obstacles. If an obstacle moves into a blind spot, the policy "forgets" its existence. With dense 64-ray coverage, blind spots are minimized, and obstacles remain consistently visible.

Conclusion: Dense sensor coverage is crucial for stateless policies. The 64-ray configuration provides sufficient spatial awareness to enable stable, collision-aware control.

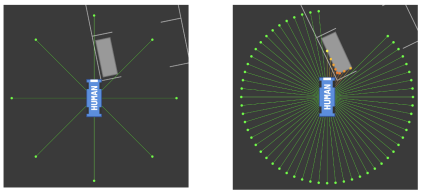

Figure 4: Illustration of blind spot problem with 8 rays vs 64 rays

4.4 Ablation Study 2: Reward Curve Design

Research question: How does the choice of reward decay function (linear vs exponential) affect learning and final performance?

Setup: We compare two reward configurations for the distance and angle scores:

- **Linear decay**: $r = \max(0, 1 - k \cdot \text{error})$
- **Exponential decay** (ours): $r = \exp(-k \cdot \text{error})$ with $k = 3$ for distance, $k = 5$ for angle

Counterintuitive finding: Making the reward *harder to achieve* leads to *better final performance*.

With linear decay, the policy converges to an average reward of $\sim$50 points. Inspection reveals a "good enough" parking strategy: the vehicle enters the spot with moderate alignment (e.g., 30-45° angle error, 0.3-0.5 m lateral offset), achieving 50-60% overlap. This provides sufficient reward to outweigh exploration costs, creating a local minimum.

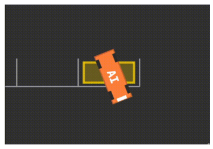

Figure 5: "Good enough" parking with linear decay shows mediocre alignment.

With exponential decay, even 30° misalignment yields very low reward, providing insufficient reinforcement for the "good enough" strategy. This forces the policy to discover more sophisticated maneuvers—notably, reverse parking approaches that allow finer alignment adjustments. The converged policy achieves average reward $\sim$80 points with near-perfect alignment in most cases.

Conclusion: The choice of reward curve significantly impacts converged policy quality. Exponential decay with appropriate rates prevents premature convergence to suboptimal strategies, ultimately achieving higher final performance despite being "harder."

4.5 Ablation Study 3: Necessity of Curriculum Learning

Research question: Can we train from scratch with final configuration (5 m/s, 80% obstacles, 1s stopping)? **Observed failure modes**: **Failure 1: Never learns to stop.** Policy navigates to target but rolls back/forth indefinitely. Root cause: 1s stopping threshold too strict for random exploration—agent never receives success reward. **Failure 2: Only learns obstacle avoidance.** Agent circles parking lot avoiding obstacles but never attempts parking. Root cause: High obstacle density makes success extremely rare—collision penalties dominate reward signal. **Conclusion**: For multi-objective tasks, curriculum learning is *essential* to balance sparse high-value rewards with dense low-value rewards.

4.6 Generalization Performance

The policy demonstrates robust zero-shot generalization to unseen parking lot configurations. Key evidence: **single policy for all types**: One network handles perpendicular, angled, and parallel parking without mode switching; **unseen layouts**: Evaluation episodes use random seeds disjoint

from training, generating completely novel obstacle arrangements and target positions; **interactive demo**: Users can manually create arbitrary parking scenarios in the web interface, and the policy adapts in real-time.

Generalization is enabled by two key design choices: (1) aggressive environment randomization preventing overfitting, and (2) ego-centric observations providing translation/rotation invariance.

5 Limitations and Future Work

5.1 Current Limitations

Soft collision constraints. Our approach uses reward penalties to discourage collisions but provides no hard safety guarantee. While the converged policy achieves low collision rates ($\sim$0.14 per episode), it is theoretically possible for the stochastic policy to sample collision-inducing actions. A potential solution is **beam search over rollouts**: at each time step, sample multiple action sequences, simulate their outcomes, and select collision-free paths with highest expected return.

No temporal memory. The current policy is stateless (feedforward MLP), lacking memory of past observations. This can cause inefficient exploration—e.g., trying a blocked path, backing out, then trying the same path again because it "forgot" the previous failure. Recurrent architectures (LSTM, Transformers with positional encoding) could maintain temporal context, encoding information like "I already explored left and found obstacles."

Limited model capacity. Our policy uses a small 2-layer MLP (128-128-64 neurons). While sufficient for the current task, more complex real-world scenarios (e.g., multi-floor parking garages, moving pedestrians) may benefit from larger networks or pre-trained visual feature extractors.

Idealized sensors. The simulation uses perfect LiDAR with no noise, occlusion, or sensor failures. Real-world deployment requires modeling sensor noise and potentially integrating multiple sensor modalities (cameras, ultrasonic sensors) for robustness.

5.2 Future Directions

Robustness to sensor noise. Augment training with simulated sensor noise (Gaussian noise on ray distances, random ray dropouts, false detections) to improve sim-to-real transfer.

Multi-agent scenarios. Extend to dynamic environments with moving obstacles (pedestrians, other vehicles entering/exiting spots) requiring reactive planning.

Tighter integration with perception. Current work assumes perfect localization of the target pose. Real-world deployment requires perception modules to estimate target positions from sensor data.

6 Conclusion

We presented ParkRL, a deep reinforcement learning system for autonomous parking that achieves robust generalization across diverse scenarios through systematic environment randomization and curriculum learning. Our key contributions include a novel reward design using harmonic mean and exponential decay that prevents local minima, a three-stage curriculum strategy demonstrated to be essential for convergence, and comprehensive ablation studies providing insights into sensor resolution, reward shaping, and training strategies.

The learned policy exhibits emergent human-like behaviors such as reverse parking, achieving 96% success rate with precise alignment. We deployed the system to an interactive web demonstration, enabling real-time inference at 60 FPS. Our ablation studies reveal counterintuitive findings—making rewards harder to achieve leads to better final performance, and curriculum learning is not merely helpful but *necessary* for learning multi-objective tasks.

This work demonstrates that end-to-end reinforcement learning can discover sophisticated control strategies for complex non-holonomic systems, matching and potentially exceeding human-designed solutions, given appropriate reward design and training curriculum.

Interactive demo: `https://pkucuipy.github.io/rl-final`

References

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48, 2009.

[2] Paolo Falcone, Francesco Borrelli, Jahan Asgari, H Eric Tseng, and Davor Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on Control Systems Technology*, 15(3):566–580, 2007.

[3] Edouard Leurent. An environment for autonomous driving decision-making. In *GitHub repository*, 2018.

[4] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.