Model-Based vs Model-Free Reinforcement Learning for Autonomous Parking

Team 16 Project Report

Ayush Agrawal, Anmol Gupta, Mehak Singal, Rachita Rajesh, Saharsh Goenka

CS 269 Fall ’25

1 Introduction

Autonomous parking is a challenging control problem because it requires high-precision maneuvering in constrained spaces where small errors can cause collisions. Unlike highway driving, parallel parking demands fine-grained modulation of steering and throttle under strict geometric and safety constraints. As urban density increases and shared-mobility fleets expand, RL-based parking systems must reliably reach target poses, adapt to diverse layouts, handle dynamic obstacles, and satisfy multiple competing objectives.

Recent work shows that reinforcement learning (RL) can acquire parking behaviors through closed-loop interaction, but most approaches rely on *model-free* RL, which requires large amounts of data, struggles with sparse terminal rewards, and generalizes poorly to new obstacle configurations. *Model-based* RL (MBRL) offers improved sample efficiency and interpretability through explicit or learned dynamics, yet existing parking studies typically evaluate only simplified scenarios and rarely compare MBRL and model-free methods under matched conditions.

These gaps motivate our study, which builds a unified simulation framework to directly compare a model-based controller (learned dynamics + MPC) and a model-free baseline (SAC+HER) on the same parallel-parking tasks. We focus on two questions: how their data requirements differ, and how well each generalizes to unseen parking layouts and obstacle configurations. Our evaluation reveals clear efficiency–robustness trade-offs that underscore when model-based planning or model-free flexibility is more advantageous.

2 Related Work

Automated parking has progressed from geometric planning to data-driven control, with Reinforcement Learning (RL) becoming a central approach. Prior work is typically categorized by whether it uses model-based or model-free RL and by the scope of behaviors addressed. As summarized in Table 5 (Appendix), most methods excel along one dimension but retain significant limitations elsewhere.

Model-Based RL with Limited Scope. Model-based RL leverages vehicle dynamics to improve predictive planning and sample efficiency. For example, [1] combines a learned transition model with Monte Carlo Tree Search but evaluates only terminal slot-entry in a static environment. Broader sequences and dynamic obstacles remain unaddressed, reflecting a recurring trade-off: interpretability and data efficiency achieved by narrowing the task domain.

Model-Free RL for End-to-End Maneuvers. Model-free deep RL is widely used to learn full parking trajectories directly from interaction. Works such as [2] and [3] achieve strong simulated performance. However, as shown in Table 5, these methods suffer from high sample complexity, require substantial reward shaping, show limited robustness, and rarely assess generalization to unseen obstacle layouts or slot configurations.

Advanced Architectures for Specific Challenges. Several studies introduce specialized frameworks for sparse rewards or high-dimensional observations. [4] applies hierarchical RL and federated training for constrained spaces but omits dynamic obstacles and remains non-model-based. [5] employs a transformer-based policy for dynamic agents, yet continues to demand large datasets and lacks interpretable planning structure.

Synthesized Research Gaps. Table 5 reveals three persistent gaps:

1. **Paradigm Comparison Gap:** No prior work conducts a controlled, head-to-head evaluation of model-based and model-free RL on identical tasks.
2. **Incomplete Metrics:** Most studies emphasize success rates over sample efficiency and generalization.
3. **Scope–Efficiency Trade-off Unresolved:** Model-based RL restricts scope, while model-free RL trades efficiency for flexibility.

These gaps motivate a systematic, unified evaluation of both RL paradigms across parking scenarios.

3 Problem Formulation

Autonomous parallel parking is a constrained optimal-control problem in which the ego-vehicle must reach a desired terminal pose within a parking slot while avoiding both static and dynamic obstacles. Unlike highway or lane-following tasks, parking requires fine-grained low-speed maneuvering, multiple direction reversals, and strict collision avoidance within tight geometric boundaries.

We formalize the autonomous parking task as a finite-horizon Markov Decision Process (MDP), defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$. Detailed definitions of the state representation, action parameterization, and bicycle-model transition dynamics are included in Appendix 8.2.

Objective: The objective is to determine an optimal policy π^* that maximizes the cumulative discounted reward. The reward function $r(s_t, a_t)$ encourages reaching a goal configuration s_{goal} within tolerance ϵ, and penalizes collisions, unsafe proximity to obstacles, and unnecessary control effort:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right]$$

The maneuver must satisfy $d(s_t, \mathbf{O}_t) > 0$ for all t, ensuring obstacle clearance.

To solve this MDP, we examine two learning paradigms commonly used in autonomous parking:

1. **MBRL**: A model-based approach that either uses known vehicle dynamics or learns an approximation of $f(s_t, a_t)$, enabling trajectory prediction and optimization.
2. **MFRL**: A model-free approach in which the agent does not approximate f explicitly but instead learns the control policy $\pi(s_t)$ directly via interaction, leveraging Hindsight Experience Replay to improve learning under sparse terminal rewards.

The limitations outlined in Section 2 lead to two central research questions that guide this study:

[RQ1] Sample Efficiency Comparison: How do MBRL (Learned Dynamics + Model Predictive Control) and MFRL (Soft Actor-Critic + Hindsight Experience Replay) differ in the amount of data and the number of training scenarios required to reach comparable parking performance?

[RQ2] Generalization and Robustness: Given a fixed dataset size, which method provides stronger robustness and navigation to unseen parking layouts and obstacle configurations?

This comparative evaluation aims to clarify the trade-off between data-intensive learning and structure-aware planning, ultimately supporting more reliable end-to-end autonomous parking.

4 Method

This section outlines the experimental design, simulation setup, learning architectures, and evaluation protocol used to study RQ1 (data efficiency) and RQ2 (generalization). Our methodology follows a pipeline designed to fairly compare Model-Based (MBRL) and Model-Free (MFRL) reinforcement learning strategies for autonomous parallel parking.

4.1 Environment Design and Dynamics

We developed six custom environments using a Gymnasium-style simulator to capture increasing behavioral complexity: (i) Reverse Parking (No Obstacles), (ii) Reverse Parking (Static), (iii) Reverse Parking (Dynamic), (iv) Parallel Parking (No Obstacles), (v) Parallel Parking (Static), and (vi) Parallel Parking (Dynamic). Representative visualizations are included in Appendix 8.3.

To better approximate realistic low-speed vehicle behavior, we implemented a **Kinematic Bicycle Model**. This model is particularly important for parking tasks because it enforces non-holonomic constraints, preventing lateral motion without forward or reverse velocity. The vehicle state evolves under:

$$\dot{x} = v\cos(\theta), \quad \dot{y} = v\sin(\theta), \quad \dot{\theta} = \frac{v}{L}\tan(\delta), \quad \dot{v} = a \tag{1}$$

where L denotes wheelbase length, and the control actions are acceleration a and steering angle δ. Gaussian process noise is added to both actuation and sensing channels to approximate real-system uncertainty.

- **State Space ($\mathcal{S}$):** $s_t = [x, y, \theta, v, \mathbf{O}_t]^T$, where $\mathbf{O}_t$ indicates obstacle states.
- **Action Space ($\mathcal{A}$):** $a_t = [acc, steer]^T$, defined over a continuous, normalized range of $[-1, 1]$.

4.2 Algorithm Implementation

We analyze two fundamentally different decision-making pipelines:

(i) **MBRL: MPC with Learned Dynamics**:

Our model-based framework follows a "Learn-then-Plan" approach. First, we train a multilayer perceptron (3 hidden layers, 128 units) to approximate the transition function $f(s, a)$. Next, a Model Predictive Control (MPC) planner uses this learned model for real-time trajectory optimization with a prediction horizon of $H = 25$ and 100 gradient-based optimization iterations. MPC was chosen because its use of gradient information enables rapid convergence to locally optimal trajectories, which is important for real-time parking.

(ii) **MFRL: SAC + HER**:

For the model-free baseline, we implement Soft Actor-Critic (SAC) augmented with Hindsight Experience Replay (HER). Both the Actor and Critic networks use 3 layers of 512 units each, a replay buffer of size 10^6, and a batch size of 1024. HER is essential for sparse-reward parking problems because it converts unsuccessful rollouts into informative goal-conditioned transitions, improving sample efficiency and policy stability.

4.3 Evaluation Procedure

To measure the performance trade-offs between both paradigms, we adopt the following protocol:

(i) **RQ1 (Data Efficiency):** We vary the dataset size $|\mathcal{D}|$ across $\{100, 500, 1000, 5000, 10000\}$ trajectories. MBRL and MFRL are trained using identical scenario distributions for fair comparison.

(ii) **RQ2 (Generalization):** Fully trained policies are tested on 50–100 held-out scenarios with unseen obstacle layouts and start configurations, providing a measure of robustness and out-of-distribution generalization.

Per episode success of the parking manoeuvre is determined by comparing the achieved reward and the set success goal reward threshold. A comprehensive review on the definition of reward model and its use in deciding the success of the parking manoeuvre is given in Appendix 8.6.

5 Experiments

We evaluate our MBRL and MFRL approaches across a grid of twelve controlled scenarios (see Appendix 8.4). A trial is counted as a success if the ego vehicle completes the full parking maneuver and reaches the goal pose within a $0.12\,\text{m}$ tolerance without any collision.

Before settling on our final algorithms, we examined two additional baselines: a CEM-based planner for the MBRL pipeline and PPO for the MFRL pipeline. The CEM controller showed instability in tight obstacle configurations, and PPO struggled to learn reliable reversing trajectories even with extended training schedules. These preliminary findings motivated our adoption of MPC for MBRL and SAC+HER for MFRL.

Data Efficiency Study (RQ1)
To assess how much experience each method requires to achieve high reliability, we track the minimum dataset size (for MBRL) or number of interaction steps (for MFRL) at which the success rate nears 100% in both the empty-lot and static-obstacle environments. Full RQ1 hyperparameters and sweep settings are listed in Appendix 8.9.

Generalization to unseen scenarios (RQ2)
To evaluate robustness, we test each controller on a held-out collection of unseen start poses and novel obstacle configurations not present during training. Both policies are evaluated under an identical training budget (10,000 interaction steps for SAC+HER; a 5,000-trajectory dataset for dynamics learning). Performance on these out-of-distribution tasks reflects each method's ability to generalize under structural and geometric variability. Experimental details are provided in Appendix 8.9.

Note: Appendix 8.7 contains a log screenshot illustrating part of the SAC+HER training process, from which our recorded results were derived.

6 Results and Discussion

6.1 Dynamics Model Performance

Training initially stalled, but adding weight decay, learning-rate scheduling, and gradient clipping reduced validation error by nearly two orders of magnitude and produced stable long-horizon rollouts, leading to more reliable MPC performance in tight obstacle configurations. Additional results are shown in Appendix 8.5.

6.2 Trajectory Quality and Maneuver Characteristics

Across reverse and parallel parking, SAC+HER and MPC exhibited different motion profiles. SAC+HER generated smooth, continuous trajectories with slowly varying steering inputs, reflecting policy-gradient optimization over long-horizon value estimates. In contrast, MPC yielded segmented motions with sharper heading adjustments driven by re-optimization at every timestep (see Figure 4 in Appendix 8.8). Although less visually smooth, these corrective updates often produced higher terminal pose accuracy and tighter slot alignment.

In static-obstacle environments, SAC+HER occasionally approached obstacles too closely during early training, highlighting its lack of explicit geometric constraints. MPC maintained safer margins through penalty-weighted rewards but remained sensitive to planning-horizon tuning. In MPC experiments with dynamic obstacles, short horizons caused delayed reactions to moving agents, while longer horizons improved anticipatory avoidance at increased computational load (illustrated in Figure 7 in Appendix 8.8).

Despite these differences, **both approaches completed full parking maneuvers and reliably achieved accurate final poses**. A video showing alignment behavior is available here.

6.3 Data Efficiency (RQ1)

Table 1: MBRL vs. MFRL: comparison of data requirements in reverse parking.

Rollout Metric	SAC+HER	MPC
Mean Episode Length	22.9	170
Mean Episode Reward	-7.27	-0.26
Training Time Steps	28,486	5,000
Achieved Success %	99%	100%

Table 2: SAC+HER data demand as environment complexity increases.

Rollout Metric	0 Obstacles	6 Obstacles
Mean Episode Reward	-15.8	-15.6
Training Time Steps	11,178	78,197
For Success Rate	45%	45%

Tables 1 and 2 quantify the data-efficiency gap between MBRL and MFRL. As shown in Table 1, SAC+HER requires over 28,000 interaction steps in an empty-lot environment to achieve 99% success, whereas MPC reaches 100% success with a dynamics model trained on only 5,000 trajectories. This reflects a core structural advantage: by decoupling model learning from control, MBRL amortizes data cost upfront and requires minimal additional interaction to achieve near-perfect performance.

Table 2 highlights how data demand for MFRL scales with task complexity. Increasing the obstacle count from 0 to 6 raises SAC+HER's training requirement from 11,178 to 78,197 steps while yielding only a 45% success rate. This represents nearly a sevenfold increase in data under an unchanged reward formulation. In contrast, MPC performance is largely unchanged once the dynamics model is learned, since planning explicitly accounts for obstacles at test time. The primary trade-off then becomes computational cost: SAC+HER executes a single forward pass, while MPC optimizes online at every control step.

6.4 Generalization to Unseen Scenarios (RQ2)

Rollout Metric	SAC+HER	MPC
Mean Episode Length	53.4	170
Mean Episode Reward	-16.9	-0.26
Success Rate	37%	100%

Table 3: MBRL vs. MFRL: comparison of success rates under a fixed training budget.

Rollout Metric	0 Obstacles	6 Obstacles
Mean Episode Length	53.4	56.7
Mean Episode Reward	-16.9	-18.4
Success Rate	37%	1%

Table 4: SAC+HER success rate as environment complexity increases.

Tables 3 and 4 evaluate how MBRL and MFRL generalize under a fixed training budget. As shown in Table 3, MPC sustains a 100% success rate even when tested on unseen initial poses and novel obstacle layouts. Under the same budget, SAC+HER reaches only 37% success in the empty-lot setting, indicating limited robustness beyond the training distribution.

Table 4 further shows that adding six static obstacles reduces SAC+HER performance from 37% to 1%, reflecting strong sensitivity to distribution shift and sparse geometric constraints. MPC avoids this accuracy collapse because its planner explicitly reasons over obstacle geometry at test time, making generalization primarily dependent on model fidelity rather than exploration coverage.

Together, the above results expose a key limitation of model-free agents in constrained parking tasks: policies overfit to training layouts, and increased exploration does not reliably produce robust collision-avoidance behaviors. Additional PPO trials in dynamic-obstacle settings confirmed the trend, with success rates remaining near 2% after extensive training.

6.5 Practical Insights and Limitations

Both approaches faced challenges imposed by the vehicle's non-holonomic geometry, particularly the tight clearance requirement of parallel parking. This occasionally produced small yaw errors even when terminal position accuracy was high; a short demonstration is available here. For MFRL, limited exploration often resulted in inefficient reversing policies and several near-collision maneuvers. For MBRL, suboptimal horizon tuning or minor transition-model bias produced over-corrections and jitter during final alignment.

Despite these limitations, our results show that incorporating an explicit predictive structure (through learned dynamics and MPC) produces more reliable behavior for safety-critical, precise parking tasks.

7 Conclusion

This work presented a controlled comparison between model-based and model-free RL for autonomous parking. MBRL, using a learned dynamics model and MPC, demonstrated substantially higher data efficiency and stronger generalization to unseen layouts, while MFRL (SAC+HER) produced smoother trajectories but required significantly more interaction to reach consistent performance.

The study remains limited by simplified reward design and environment-specific training. Future directions include integrating comfort-aware objectives, adopting curriculum-based scenario scaling, and evaluating sim-to-real transfer on physical platforms. Overall, the results highlight model-based reinforcement learning as a strong foundation for reliable, precision-oriented automated parking.

8 Appendix

8.1 Literature Survey

Source	Model-Based RL?	Simulation Used?	Full Maneuver Covered?	Dynamic Obstacles?	Maneuver Optimization?
[1]	Yes (simulated vehicle dynamics; iterative data generation)	Yes (simulation-based training)	No (slot-entry phase only)	No (static vehicles only)	No (time minimization; penalizes large swings)
[2]	No (SAC-based DRL)	Yes (Gazebo + ROS)	Yes (parallel, perpendicular, complex parking)	Yes (static + occasional dynamic obstacles)	No (time-only cost)
[3]	No (PPO/SAC model-free RL)	Yes (simulation-only)	Yes (complete parking w/ alignment)	Not explicitly (mostly static)	Yes (reward shaping for smoothness + precision)
[4]	No (deep RL + optimization layer)	Yes (narrow-space simulations)	Yes (full trajectory from start to target pose)	No (static only)	Yes (smoothness + precision)
[5]	No (Transformer-based RL)	Yes (CARLA)	Yes (approach, alignment, entry, stop)	Yes (pedestrians + moving vehicles)	Yes (attention-based optimization)

Table 5: Summary of Related Work in Reinforcement Learning for Automated Parking

Source	Gaps/Limitations
[1]	Doesn't cover full maneuver (alignment + final pose) end-to-end. No dynamic obstacle handling (moving vehicles/pedestrians) included. Simulation models may not fully capture real-world dynamics or uncertainties; generalization to real vehicles may be limited.
[2]	Only validated in simulation (no real-world tests). Limited handling of dynamic obstacles. Focuses on time and accuracy, not comfort or maneuver efficiency. Lacks model-based planning or uncertainty modeling.
[3]	Relies on model-free RL, making it data-inefficient and slower to converge compared to model-based approaches. No explicit modeling or testing with dynamic obstacles (pedestrians or moving vehicles). Entirely simulation-based; lacks real-world validation or transfer experiments. Focused primarily on precision and reward shaping, not on planning or multi-objective optimization.
[4]	No dynamic obstacle handling - only static environments tested. Not model-based; lacks a predictive or search-driven RL framework. Only simulation results; no real-world validation. Federated learning adds complexity without solving dynamic obstacle challenges.
[5]	End-to-end learning without a model-based planning stage may reduce interpretability and sample efficiency. Heavy reliance on simulation (CARLA) data may not transfer well to real-world deployment. Focused more on control policy performance than multi-objective optimization (e.g., comfort or efficiency metrics).

Table 6: Gaps and Limitations of Related Work

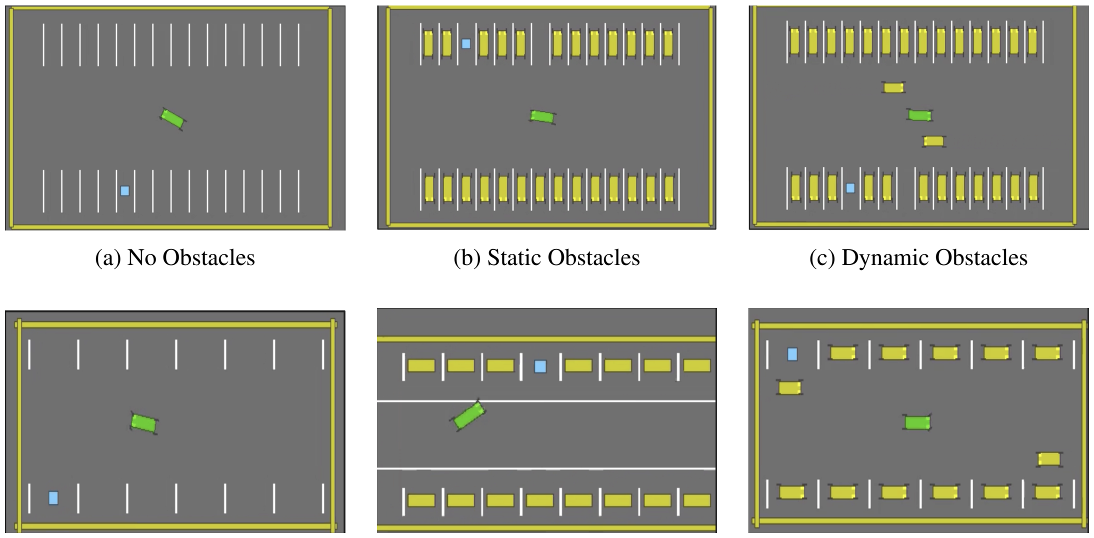
8.2 MDP Details

State Space ($\mathcal{S}$): Let $s_t \in \mathcal{S}$ represent the vehicle configuration and environment state at time t, defined as $s_t = [x_t, y_t, \theta_t, v_t, \mathbf{O}_t]^T$, where (x, y) is position, θ is heading, v is velocity, and $\mathbf{O}_t$ represents obstacle states.

Action Space ($\mathcal{A}$): Let $a_t \in \mathcal{A}$ represent the control inputs $a_t = [\dot{v}_t, \delta_t]^T$, corresponding to acceleration and steering angle.

Dynamics ($\mathcal{T}$): The vehicle follows non-holonomic bicycle kinematics, governed by the transition function $s_{t+1} = f(s_t, a_t)$.

8.3 Environment Visualizations


(a) No Obstacles (b) Static Obstacles (c) Dynamic Obstacles

(d) No Obstacles (e) Static Obstacles (f) Dynamic Obstacles

Figure 1: Reverse-parking scenarios (top row) and parallel-parking scenarios (bottom row) are shown under three obstacle conditions: no obstacles, static obstacles, and dynamic obstacles.

8.4 Experiment Scenarios

Sim. No.	Policy	Parking Type	Static Obstacles	Dynamic Obstacles
1	MBRL	Reverse	0	0
2	MBRL	Reverse	26	0
3	MBRL	Reverse	26	2
4	MBRL	Parallel	0	0
5	MBRL	Parallel	12	0
6	MBRL	Parallel	12	2
7	MFRL	Reverse	0	0
8	MFRL	Reverse	26	0
9	MFRL	Reverse	26	2
10	MFRL	Parallel	0	0
11	MFRL	Parallel	12	0
12	MFRL	Parallel	12	2

Table 7: Simulation Scenarios for MBRL and MFRL

8.5 Dynamics Model Training Results

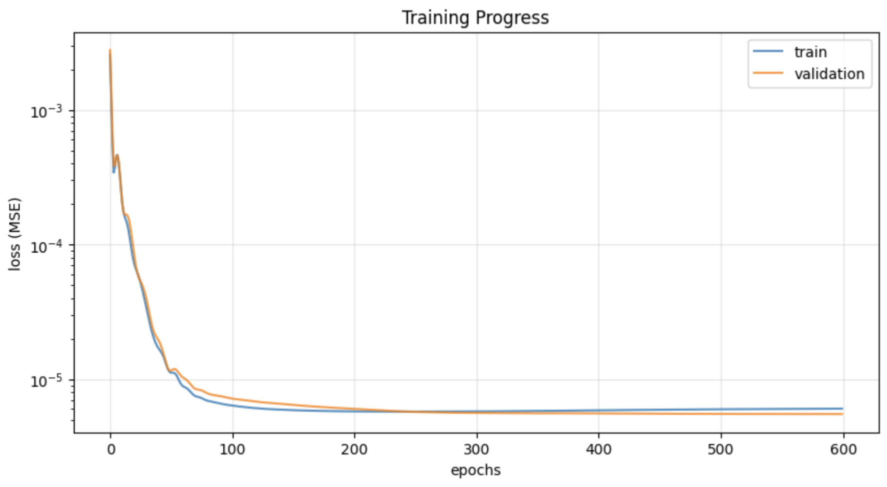


Figure 2: Training results of the vehicle dynamics model used in the parking tasks.

8.6 Reward model and success rate

We will outline all details of the environment in the following subsections.

8.6.1 State Vector

The vehicle state is represented by a 6-dimensional observation vector:

$$\mathbf{s} = [x, y, v_x, v_y, \cos(\theta), \sin(\theta)]^T \tag{2}$$

where:

- $x, y \in \mathbb{R}$ are the vehicle's position coordinates
- $v_x, v_y \in \mathbb{R}$ are the velocity components
- $\theta \in [0, 2\pi)$ is the heading angle (represented via $\cos(\theta)$ and $\sin(\theta)$ to avoid discontinuity)

8.6.2 Observation Scaling

To improve numerical stability and learning efficiency, the raw observations are normalized by scaling factors:

$$\mathbf{s}_{scaled} = \mathbf{s} \oslash \sigma \tag{3}$$

where $\oslash$ denotes element-wise division and the scaling vector is:

$$\sigma = [100, 100, 5, 5, 1, 1]^T \tag{4}$$

This means:

- Position coordinates are divided by 100 (converting meters to normalized units)
- Velocities are divided by 5 (converting m/s to normalized units)
- Angular components remain unchanged (already in $[-1, 1]$)

8.6.3 Goal State

The desired parking position is represented by a goal vector:

$$\mathbf{g} = [x_g, y_g, 0, 0, \cos(\theta_g), \sin(\theta_g)]^T \tag{5}$$

where (x_g, y_g) is the target position and θ_g is the target heading. The velocity components are zero, indicating the vehicle should be stationary at the goal.

8.6.4 Weighted Distance Metric

The reward function measures the proximity of the achieved state to the desired goal using a *weighted p-norm*. Given a state $\mathbf{s}$ and goal $\mathbf{g}$, the state error is:

$$\epsilon = |\mathbf{s} - \mathbf{g}| \tag{6}$$

where $|\cdot|$ denotes element-wise absolute value.

8.6.5 Feature Importance Weighting

Different state dimensions have varying importance for parking success. We apply importance weights:

$$\mathbf{w} = [w_x, w_y, w_{v_x}, w_{v_y}, w_{\cos}, w_{\sin}]^T = [1, 0.3, 0, 0, 0.2, 0.2]^T \tag{7}$$

The weighted error is computed as:

$$\epsilon_w = \epsilon \odot \mathbf{w} \tag{8}$$

where $\odot$ denotes element-wise multiplication (Hadamard product).

Interpretation of weights:

- $w_x = 1.0$: Longitudinal position is most critical
- $w_y = 0.3$: Lateral position has moderate importance
- $w_{v_x} = w_{v_y} = 0.0$: Velocities are not directly penalized (implicitly minimized by action costs)
- $w_{\cos} = w_{\sin} = 0.2$: Heading alignment has lower priority than position

8.6.6 Reward Function Formulation

The instantaneous reward at state $\mathbf{s}$ is defined as:

$$r(\mathbf{s}, \mathbf{g}) = -\left(\|\epsilon_w\|_1\right)^p \tag{9}$$

where:

- $\|\cdot\|_1$ is the L^1 norm: $\|\epsilon_w\|_1 = \sum_{i=1}^{6} |\epsilon_{w,i}|$
- $p = 0.5$ is the power parameter (chosen to provide high kurtosis)

Expanded form:

$$r(\mathbf{s}, \mathbf{g}) = -\sqrt{w_x|x - x_g| + w_y|y - y_g| + w_{\cos}|\cos\theta - \cos\theta_g| + w_{\sin}|\sin\theta - \sin\theta_g|} \tag{10}$$

8.6.7 Properties of the Reward Function

1. **Non-positive**: $r(\mathbf{s}, \mathbf{g}) \leq 0$ for all $\mathbf{s}, \mathbf{g}$
2. **Maximum at goal**: $r(\mathbf{g}, \mathbf{g}) = 0$ (perfect parking)
3. **Continuous**: The reward is differentiable almost everywhere, enabling gradient-based optimization
4. **Sub-linear growth**: Using $p = 0.5 < 1$ makes the reward less sensitive to large errors, providing better shaping for sparse reward scenarios

8.6.8 Terminal State Emphasis

In Model Predictive Control, we often emphasize the terminal state (final position in the planning horizon) to encourage convergence to the goal. The augmented reward with terminal weighting is:

$$r_{terminal}(\mathbf{s}_T, \mathbf{g}) = \lambda_T \cdot r(\mathbf{s}_T, \mathbf{g}) \tag{11}$$

where:

- $\mathbf{s}_T$ is the state at the end of the horizon
- $\lambda_T > 1$ is the terminal weight (typically $\lambda_T \in [2, 20]$)

The total MPC cost over a horizon of length H becomes:

$$J = \sum_{t=0}^{H-1} r(\mathbf{s}_t, \mathbf{g}) + (\lambda_T - 1) \cdot r(\mathbf{s}_H, \mathbf{g}) \tag{12}$$

8.6.9 Implementation

The reward model in PyTorch (from the code):

```
def reward_model(states, goal, gamma=None, terminal_weight=1.0):
    goal = goal.expand(states.shape)
    reward_weights = torch.Tensor([1, 0.3, 0, 0, 0.2, 0.2])
    rewards = -torch.pow(
        torch.norm((states - goal) * reward_weights, p=1, dim=-1),
        0.5
    )

    if terminal_weight != 1.0:
        terminal_mask = torch.zeros_like(rewards)
        terminal_mask[-1] = terminal_weight - 1.0
        rewards = rewards * (1.0 + terminal_mask)

    return rewards
```

8.6.10 Success Criterion: Definition

An episode is considered **successful** if the final reward exceeds a predefined threshold:

$$Success \iff r(\mathbf{s}_{final}, \mathbf{g}) > -\tau_s \tag{13}$$

where τ_s is the success threshold. In our implementation:

$$\tau_s = 0.12 \tag{14}$$

8.6.11 Success criterion: Equivalent Formulation

The success condition can be rewritten in terms of the weighted error:

$$Success \iff -\sqrt{\|\epsilon_w\|_1} > -0.12 \tag{15}$$

Taking the square of both sides (valid since both sides are negative):

$$Success \iff \|\epsilon_w\|_1 < (0.12)^2 = 0.0144 \tag{16}$$

Explicit form:

$$Success \iff w_x|x - x_g| + w_y|y - y_g| + w_{\cos}|\cos\theta - \cos\theta_g| + w_{\sin}|\sin\theta - \sin\theta_g| < 0.0144 \tag{17}$$

8.6.12 Success Criterion: Physical Interpretation

To understand what this threshold means in physical units, we need to account for the observation scaling. After unscaling:

$$Success \iff 1 \cdot \frac{|\Delta x|}{100} + 0.3 \cdot \frac{|\Delta y|}{100} + 0.2 \cdot |\Delta \cos\theta| + 0.2 \cdot |\Delta \sin\theta| < 0.0144 \tag{18}$$

where $\Delta x, \Delta y$ are in meters, and $\Delta \cos\theta, \Delta \sin\theta$ represent heading error.

8.6.13 Example Success Cases

Case 1: Position error only

If the vehicle is perfectly aligned ($\theta = \theta_g$) but displaced:

$$\frac{|\Delta x|}{100} + 0.3 \cdot \frac{|\Delta y|}{100} < 0.0144 \tag{19}$$

For pure longitudinal error ($\Delta y = 0$):

$$|\Delta x| < 1.44 meters \tag{20}$$

For pure lateral error ($\Delta x = 0$):

$$|\Delta y| < 4.8 meters \tag{21}$$

Case 2: Heading error only

If the vehicle is perfectly positioned ($\Delta x = \Delta y = 0$) but misaligned:

For small angles, $|\Delta \cos\theta| \approx \frac{\theta^2}{2}$ and $|\Delta \sin\theta| \approx |\theta|$, giving:

$$0.2 \cdot \frac{\theta^2}{2} + 0.2 \cdot |\theta| < 0.0144 \tag{22}$$

This allows heading errors up to approximately $\theta \approx$ 5-10.

Case 3: Realistic combined error

A typical successful parking might have:

- $\Delta x = 0.5$ m
- $\Delta y = 0.3$ m
- $\Delta\theta = 5$

Weighted error: $\|\epsilon_w\|_1 = 1 \cdot \frac{0.5}{100} + 0.3 \cdot \frac{0.3}{100} + 0.2 \cdot |1 - \cos(5)| + 0.2 \cdot |\sin(5)|$
$\approx 0.005 + 0.0009 + 0.0004 + 0.0175$
$\approx 0.0238 > 0.0144 \quad (FAIL)$

This would *not* be considered successful, indicating the threshold is quite strict.

8.6.14 Success Rate Metric

During training with Hindsight Experience Replay (HER) and Soft Actor-Critic (SAC), the **success rate** is tracked as:

$$SuccessRate = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{r(\mathbf{s}_i^{final}, \mathbf{g}_i) > -0.12\} \tag{23}$$

where:

- N is the number of recent episodes in the evaluation buffer
- $\mathbf{1}\{\cdot\}$ is the indicator function (1 if condition is true, 0 otherwise)
- $\mathbf{s}_i^{final}$ is the final state of episode i
- $\mathbf{g}_i$ is the goal for episode i

This metric provides a direct measure of the policy's effectiveness at completing the parking task.

8.6.15 Design Rationale: Why Weighted L^1 Norm with $p = 0.5$?

1. **Weighted**: Different dimensions have different physical units and importance. Weights ensure proper scaling and prioritization.
2. L^1 **norm**: More robust to outliers than L^2 norm. A large error in one dimension doesn't dominate the total cost as much.

3. **Power** $p = 0.5$: Creates high kurtosis in the reward distribution:
   - Small errors (near goal): Reward is close to 0, providing strong signal
   - Large errors (far from goal): Reward doesn't become extremely negative, avoiding numerical issues and providing better gradient information

8.6.16 Relationship to MPC Optimization

In MPC, we solve:

$$\mathbf{a}^* = \arg\max_{\mathbf{a}_{0:H-1}} \sum_{t=0}^{H} r(\mathbf{s}_t, \mathbf{g}) + \mathcal{R}_{reg}(\mathbf{a}_{0:H-1}) \tag{24}$$

where:

- $\mathbf{a}_{0:H-1}$ is the action sequence
- $\mathbf{s}_t$ evolves according to learned dynamics: $\mathbf{s}_{t+1} = f_\theta(\mathbf{s}_t, \mathbf{a}_t)$
- $\mathcal{R}_{reg}$ includes action regularization and smoothness terms

The differentiable reward function enables gradient-based optimization (Adam, L-BFGS) to find optimal action sequences.

8.7 SAC + HER Training Logs

```
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 53.4     |
|    ep_rew_mean     | -16.9    |
|    success_rate    | 0.37     |
| time/              |          |
|    episodes        | 144      |
|    fps             | 4        |
|    time_elapsed    | 2250     |
|    total_timesteps | 10225    |
| train/             |          |
|    actor_loss      | 2        |
|    critic_loss     | 0.00703  |
|    ent_coef        | 0.00562  |
|    ent_coef_loss   | 0.119    |
|    learning_rate   | 0.001    |
|    n_updates       | 10124    |
---------------------------------
```

(a) Training log for SAC + HER training in empty parking lot, 0 obstacles (Link to full log)

```
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 43.6     |
|    ep_rew_mean     | -15.6    |
|    success_rate    | 0.45     |
| time/              |          |
|    episodes        | 1144     |
|    fps             | 3        |
|    time_elapsed    | 20508    |
|    total_timesteps | 78197    |
| train/             |          |
|    actor_loss      | 1.53     |
|    critic_loss     | 0.00553  |
|    ent_coef        | 0.00306  |
|    ent_coef_loss   | -0.581   |
|    learning_rate   | 0.001    |
|    n_updates       | 78096    |
---------------------------------
```

(b) Training log showing 45% success rate for SAC + HER with 78,197 training time steps in parking lot with 6 obstacles

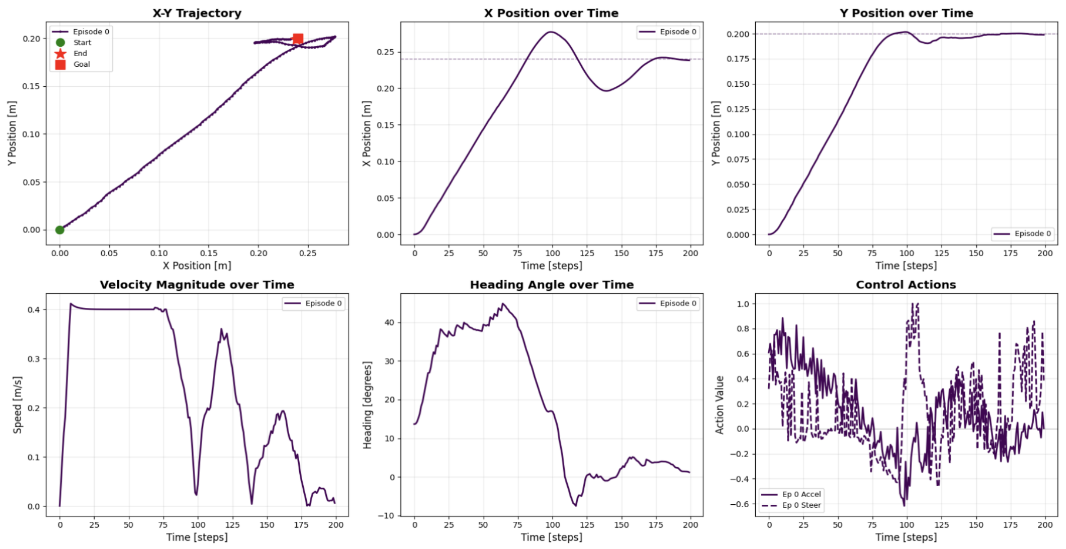
```
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 64.1     |
|    ep_rew_mean     | -26.5    |
|    success_rate    | 0.01     |
| time/              |          |
|    episodes        | 152      |
|    fps             | 4        |
|    time_elapsed    | 2135     |
|    total_timesteps | 10442    |
| train/             |          |
|    actor_loss      | 2.36     |
|    critic_loss     | 0.009    |
|    ent_coef        | 0.00528  |
|    ent_coef_loss   | -0.164   |
|    learning_rate   | 0.001    |
|    n_updates       | 10341    |
---------------------------------
```

(c) Training log showing 1% success rate for SAC + HER with 10,000 training time steps in parking lot with 6 obstacles

Figure 3: Training logs for SAC+HER under different experiment configurations at different steps. The complete log can be found here.

8.8 Trajectory Plots

8.8.1 Parallel parking with static obstacles using MPC planner



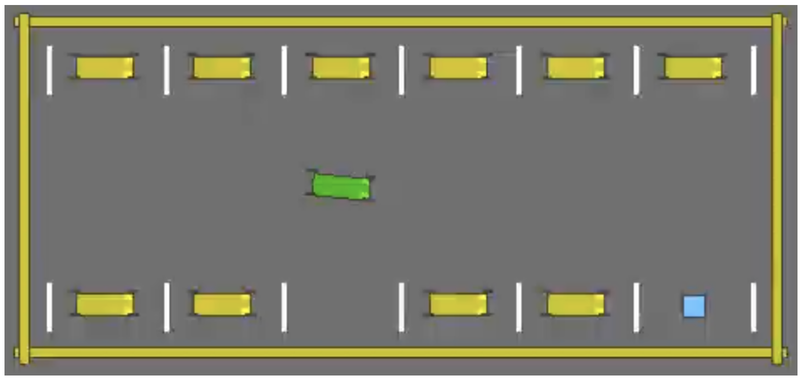


Figure 4: MPC parallel parking with static obstacles – *trajectory (top) and environment (bottom)*.

8.8.2 Reverse parking in empty parking lot with MPC planner

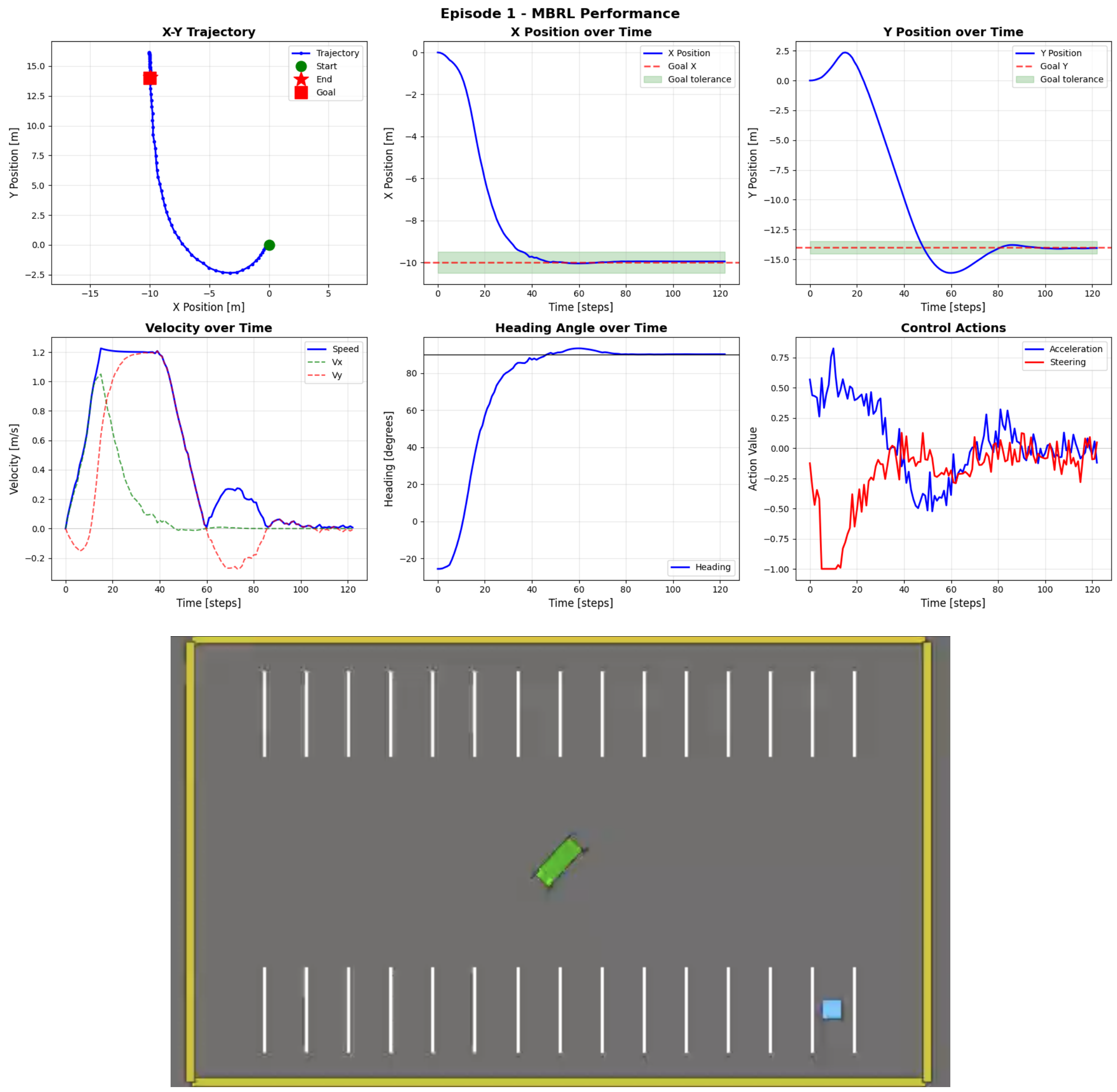
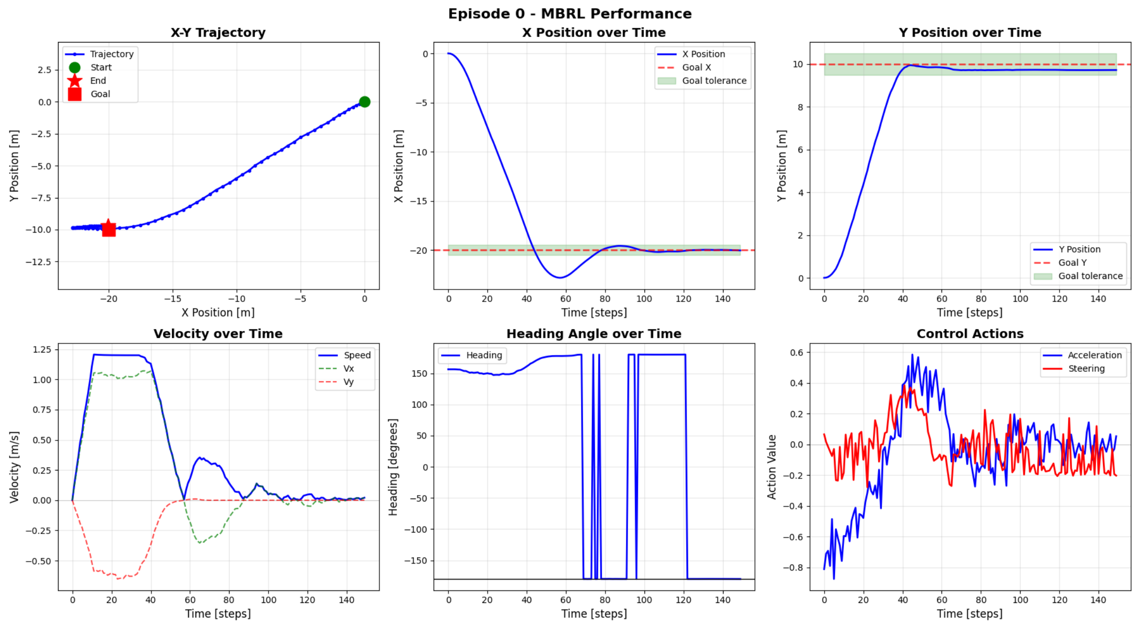


Figure 5: MPC reverse parking in an empty lot – *trajectory (top) and environment (bottom)*.

8.8.3 Parallel parking in empty parking lot with MPC planner



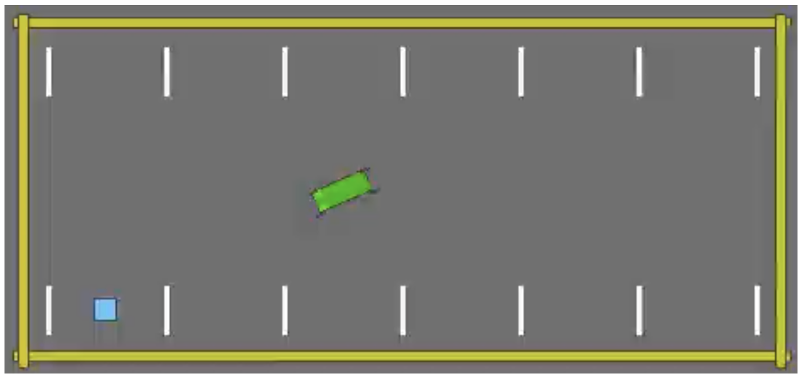


Figure 6: MPC parallel parking in an empty lot – *trajectory (top) and environment (bottom)*.

8.8.4 Reverse parking with static obstacles using SAC+HER

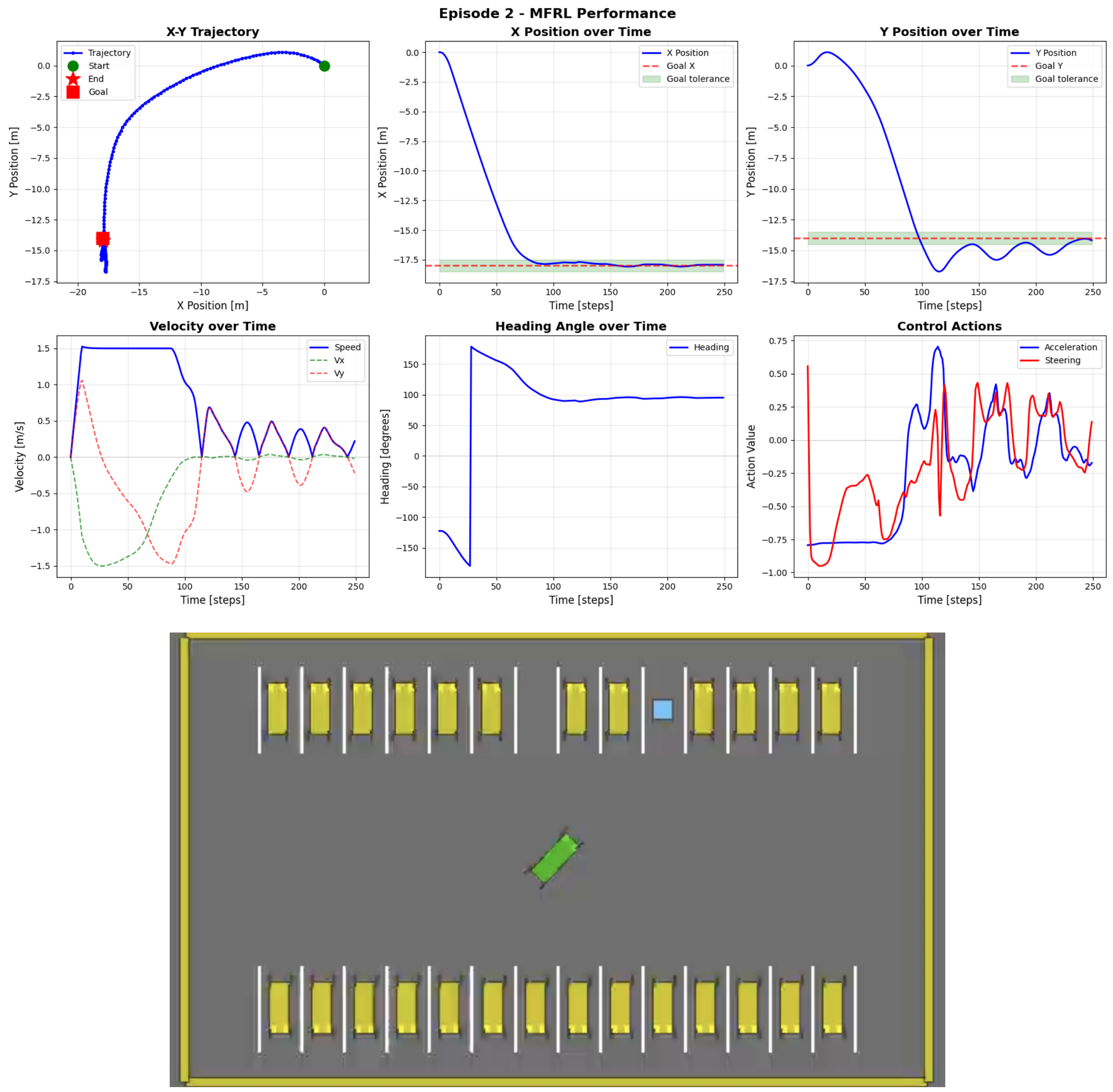
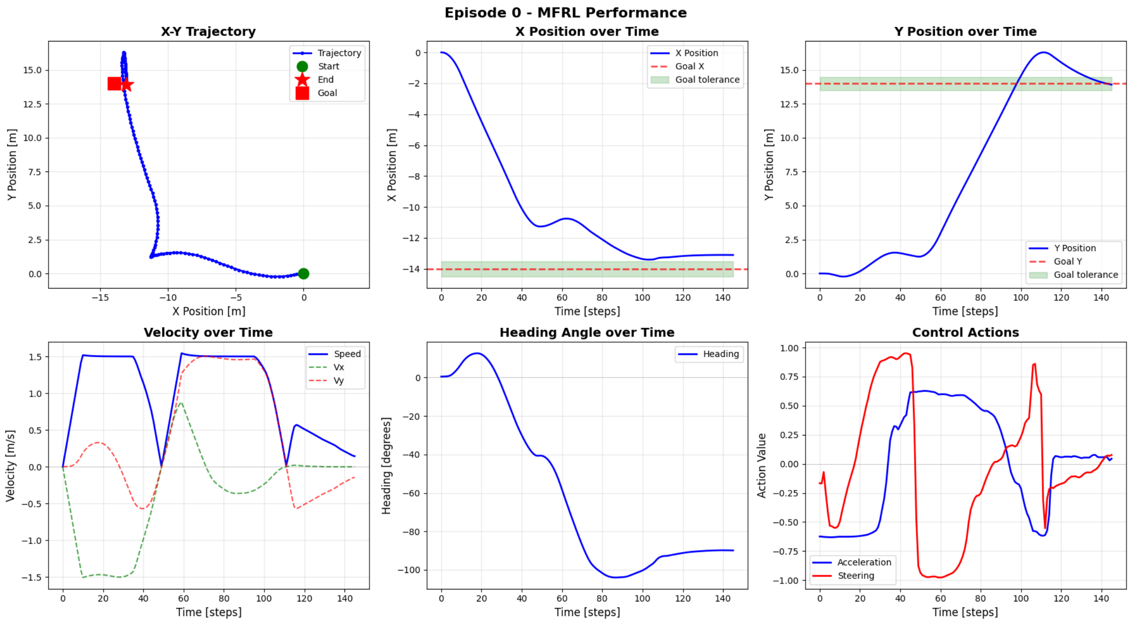


Figure 7: HER reverse parking with static obstacles – *trajectory (top) and environment (bottom)*.

8.8.5 Reverse parking in empty parking lot using SAC+HER



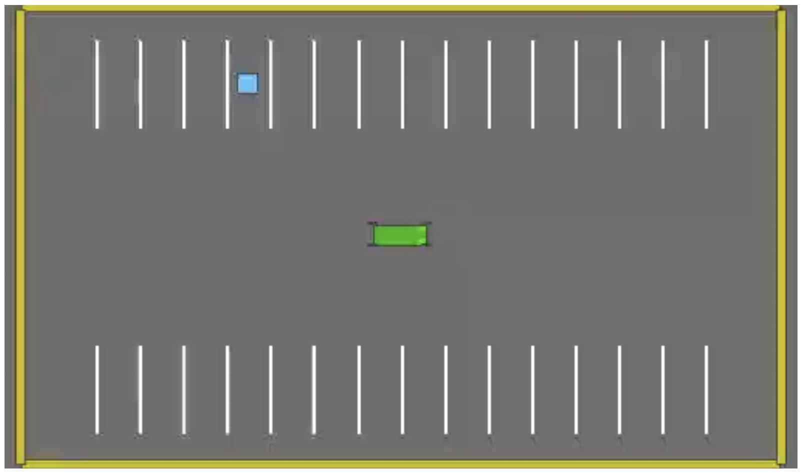


Figure 8: HER reverse parking in an empty lot – *trajectory (top) and environment (bottom)*.

8.8.6 Parallel parking with dynamic obstacles using MPC planner

We integrated MPC with general collision avoidance capability. These include avoiding collision with static vehicles, moving vehicles, and walls. The following is an image of the scene we use for simulating the MPC planner with the aforementioned obstacles in the environment.

Full video can be watched here: Parallel-parking-dynamic-obstacle-MPC-planner

Notice that the vehicle nearly avoided collision with the moving vehicle. The collision avoidance manoeuvre is not aggressive here because the **COLLISION AVOIDANCE WEIGHTS** and **SAFETY MARGINS** are set to low values. Setting these to higher values will make the collision avoidance manoeuvre aggressive.

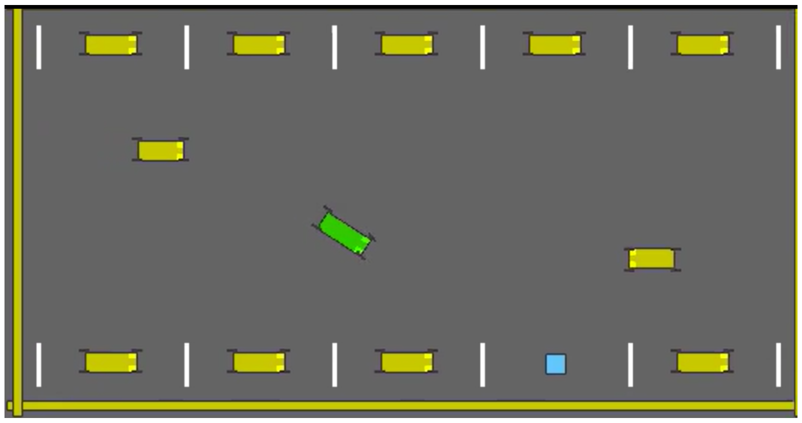


Figure 9: HER reverse parking in an empty lot – *trajectory (top) and environment (bottom)*.

8.9 Parameters for Experiments

Time	**Result**
Episodes	144
FPS	4
Training Time	2250
Total Timesteps	**10000**
Train	**Result**
Actor Loss	2
Critic Loss	0.00703
Entropy Coefficient	0.00562
Entropy Coefficient Loss	0.119
Learning Rate	0.001

Table 8: Parameters for RQ1 Experiments

Time	**Result**
FPS	4
Target Success Rate	**100%**
Train	**Result**
Actor Loss	1.66
Critic Loss	0.00496
Entropy Coefficient	0.00575
Entropy Coefficient Loss	-0.43
Learning Rate	0.001
N Updates	49845

Table 9: Parameters for RQ2 Experiments

9 Team Contributions

Team Member	**Contributions**
Ayush	Implemented the MPC planner for the learned dynamics model; generated trajectory plots; ran curriculum training in the static-obstacle environment. Conditioned the environments and tuned the SAC+HER and MPC planners to generate results.
Anmol	Created parallel parking environment with both static and dynamic obstacles; ran and tuned SAC+HER and MPC notebooks to generate results.
Mehak	Added dynamic obstacles with planned trajectories; wrote the *Problem Foundation* and *Methods* sections of the report.
Rachita	Optimized the CEM planner; wrote the *Experiments*, *Results and Discussion*, and *Conclusion* sections.
Saharsh	Ran pre-training for the PPO policy; wrote the *Introduction* and *Related Work* sections of the report.

Table 10: Summary of Individual Contributions

Discrepancies: Our project evolved significantly after the initial proposal, so the team contract may not perfectly match the individual contributions listed above. Although individual responsibilities shifted during development, all team members contributed equally to the final project.

References

[1] J. Zhang, H. Chen, S. Song, and F. Hu. Reinforcement learning-based motion planning for automatic parking system. *IEEE Access*, 8:154485–154501, 2020. doi:10.1109/ACCESS.2020.3017770.

[2] Z. Zhang, Y. Luo, Y. Chen, H. Zhao, Z. Ma, and H. Liu. Automated parking trajectory generation using deep reinforcement learning, 2025. URL `https://arxiv.org/abs/2504.21071`.

[3] A. Suleman, M. U. Khan, Z. Kaleem, A. H. Alenezi, I. Shabbir, S. Coleri, and C. Yuen. Reward-augmented reinforcement learning for continuous control in precision autonomous parking via policy optimization methods, 2025. URL `https://arxiv.org/abs/2507.19642`.

[4] Z. Yuan, Z. Wang, X. Li, L. Li, and L. Zhang. Hierarchical trajectory planning for narrow-space automated parking with deep reinforcement learning: A federated learning scheme. *Sensors (Basel)*, 23(8):4087, 2023. doi:10.3390/s23084087.

[5] J. Fu, B. Tian, H. Chen, S. Meng, and T. Yao. Parkformer: A transformer-based parking policy with goal embedding and pedestrian-aware control, 2025. URL `https://arxiv.org/abs/2506.16856`.