Progress Report for Autonomous Parking with DRL

Zheyuan Wu, Yuchen Wu

November 16, 2025

Abstract

The main problem of the project is to train an agent that controls a vehicle to park in the desired spot while avoiding collision with obstacles. This problem arises from the rapid development of autonomous vehicles. Compared to parking manually, autonomous parking not only saves time, but also achieves more compact parking spaces: it can significantly boost the operation efficiency of both private cars and commercial trucks.

1 Project Overview

We aim to train an agent that is able to park the vehicle in the target spot from any feasible starting position, under time or space constraints. We approach the problem by modeling it as an environment with continuous state space (position, angle, etc.) and discrete action space (steering, direction, etc.); Then we will attempt to train the agent with various Reinforcement learning algorithms. While traditional studies often focus on the parking of cars, we will attempt to park trailer trucks, which have more complex mechanics than cars and are more common in industrial contexts. A real-world application of the problem would be the autonomous parking of trailer trucks to facilitate efficient cargo loading and unloading. To start with, we aim to train the agent to back a trailer truck straight into a parking spot that is directly behind it (this is easy for cars, but tricky for trailer trucks).

Different from these existing works, we will attempt to solve the parking problem with tighter time and space constraints, in order to maximize the time and space efficiency of parking.

For the first stage of the experiments, we tested several RL algorithms to train the agent to back a trailer truck straight into a parking spot with constrained open spaces.

2 Team Member Roles/Tasks

2.1 Yuchen Wu

1. Build the environment via the gymnasium.
2. Test basic DQN algorithm and write discrete action space adapter
3. Write the paper.

2.2 Zheyuan Wu

1. Provides supporting functions and code reviews for the environment
2. Test PPO, DDPG algorithm, and write box action space adapter
3. Maintain documentation and a consistent environment
4. Write the paper and creating visualizations

3 Initial Results

We have implemented the environment for trailer trucks using pygame and gymnasium[5]. We have tested the DQN algorithm to train the agent to back a trailer truck straight into a parking spot with constrained open spaces. Github repository: `https://github.com/Trance-0/ATTPDRL`.

3.1 Environment Design

We use a discrete integer value $\{-numSteerAngle, \dots, 0, 1, 2, numSteerAngle\}$ (default numer of steering angles is 3) as a representation for steering angles. And use $\{1, 0\}$ as a representation for directions. In pygame simulation, we add some delays between action and observation pairs in order to simulate real trailer truck behavior.

We defined parking lots using three coordinates: the lower left corner, the upper left corner, and the lower right corner. Due to time constraints, we failed to implement the geometry intersection algorithm via the line sweep algorithm on time and decided to use the third-party module Shapely`https://github.com/shapely/shapely` to detect truck intersection with the parking lot and collision with obstacles in our environment.

The truck object is defined as the *Truck* and *TrailerTruck* class. The Truck object is responsible for truck shape movement in the environment, and the TrailerTruck object simulates the behavior of the trailer, which has an additional collision detector if the angle between the truck and trailer is too large.

We use pygame to create a visualization of the environment and implement *humanAction* and *modelAction* for interactive control of the environment.

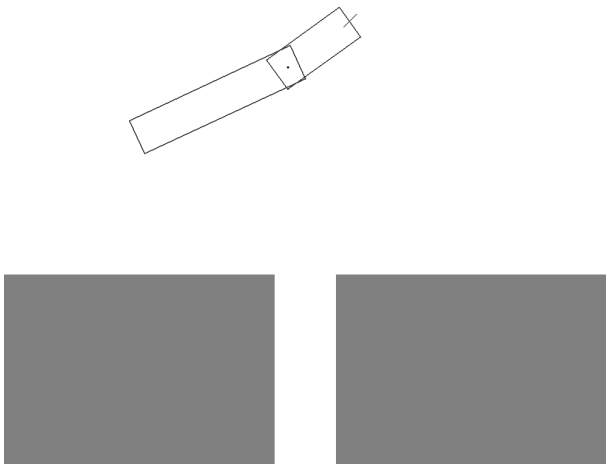

Figure 1: Pygame visualization of the environment, including the truck, obstacles, and parking lot.

Reward function for the environment consists of three components:

- Time penalty, to avoid the model taking too much action (default is -0.01)
- Collision penalty, when collision happens (default is -100)
- Success Reward, when the truck is in the parking lot defined by our environment (default is 100)

At current stage, no extra sensor or hints is provided to the model, only position and experience is recorded. Terminal states are defined when the truck is in the parking lot defined by our environment or collision happens.

3.2 Algorithms testing

We use stable-baselines3 [3] as our main framework for deep reinforcement learning. At current stage, we tested three algorithms (DQN[2], PPO[4] and DDPG[1]) with multi-layer perceptron to train the agent to back a trailer truck straight into a parking spot with constrained open spaces.

For PPO and DQN, we use default 64 units (per layer), 2 layer fully connected net as our policy network. For DDPG, we use default [400, 300] units for TD3/DDPG (values are taken from the original TD3 paper).

However, during the test stage, we observed that for the general parking problem, the model failed to gain valuable experience in exploration stages and was incapable of learning the correct strategy for moving the trailer truck. To resolve this problem, we created a simple environment and less constrained open spaces for trailer truck to park.

- **Simple**: the truck starts 1m in front of the parking spot, aligned with the target orientation, with no obstacles. The parking lot width is X m and the depth is Y m.
- **Standard**: the truck starts at a random position in region R with random orientation in $[-\theta_{\max}, \theta_{\max}]$, obstacles are placed immediately on the left and right of the parking lot, and the parking lot has width X', depth Y'.

Algorithm	Environment	total timesteps	Mean Reward (+/- Std)
DQN	simple	3000	116.58 +/- 0.0
PPO	simple	3000	107.41 +/- 0.0
PPO	standard	3000	-189.20 +/- 7.43
PPO	standard	30000	-187.31 +/- 16.53
DDPG	simple	3000	116.18 +/- 0.0
DDPG	standard	30000	-175.63 +/- 45.80

Figure 2: Initial results for different algorithms

4 Current Concerns and Questions

Currently, our RL algorithms have failed to gain valuable experience in open parking spaces. Compared with simple CartPole problem, our problem is more complex and less trivial to solve via random exploration. At current stage, we are using the simplified version, where the truck can achieve maximal reward by moving forward 1m to the parking lot, to test the validity of RL algorithms.

We suspect the following issues are causing the problem:

- **Sparse rewards:** success is rare and requires a long sequence of precise actions in standard environment
- **High-dimensional action space:** the combination of steering angles and directions yields many ineffective actions during early training.
- **Reward scaling:** the large magnitude of collision and success rewards relative to the time penalty may destabilize learning.

- **Model complexity:** the policy network is too simple to learn the correct strategy for moving the trailer truck.

During development, we observed that the stable-baselines3 library, for some unknown reasons, the environment must be saved after training and reloaded with the model to have reproducible results.

5 Tentative Plan

- Week 1: Train the agent with different algorithms and test hyperparameters on a home computer for general cases for parking, refine the reward function.
- Week 2: Evaluate the performance of the agent and compare it with the existing literature.
- Week 3: Try to use the cloud resources to train the agent with more complicated algorithms or add more constraints to the environment to make it more robust and realistic for real-world applications.
- Week 4: Deliver the report and the video demonstration of the agent's performance.

References

[1] Timothy P. Lillicrap et al. *Continuous control with deep reinforcement learning*. 2019. arXiv: 1509.02971 [cs.LG]. URL: https://arxiv.org/abs/1509.02971.

[2] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning*. 2013. arXiv: 1312.5602 [cs.LG]. URL: https://arxiv.org/abs/1312.5602.

[3] Antonin Raffin et al. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.

[4] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.

[5] Mark Towers et al. "Gymnasium: A Standard Interface for Reinforcement Learning Environments". In: *arXiv preprint arXiv:2407.17032* (2024).