Automated Parking in CARLA: a Deep Reinforcement Learning-based Approach

Luca Lazzaroni[1[0000-0001-8092-5473]], Alessandro Pighetti[1[0009-0001-7166-5750]], Francesco Bellotti[1], Alessio Capello[1[0000-0003-4277-7283]], Marianna Cossu[1[0009-0008-4548-7648]], and Riccardo Berta[1[0000-0003-1937-3969]]

[1] Department of Electrical, Electronic and Telecommunication Engineering (DITEN), University of Genoa, Via Opera Pia 11a, 16145 Genoa, Italy
luca.lazzaroni, alessandro.pighetti, alessio.capello, marianna.cossu}@edu.unige.it, {francesco.bellotti, riccardo.berta}@unige.it

Abstract. This paper focuses on developing a Deep Reinforcement Learning (DRL) - based agent for real-time trajectory planning and tracking in a simulated parking environment, specifically low-speed maneuvers in a parking area with comb-shaped spaces and a random distribution of non-player vehicles. We rely on CARLA as a virtual driving simulator due to its realistic graphics and physics simulation capabilities, and on the Gymnasium and Stable-Baselines3 toolkits for training the agent. We show that the agent is able to achieve a success rate of 97% in reaching the target parking lot without collisions. However, integrating CARLA with DRL frameworks poses challenges, such as determining suitable environment and neural network update frequencies. Despite these issues, the results demonstrate the potential of DRL agents in developing automated driving functions.

Keywords: automated driving functions, CARLA driving simulator, deep reinforcement learning, Gymnasium, automated parking, proximal-policy optimization.

1 Introduction

A growing trend in the development of automated driving functions (ADFs) is the use of DRL agents that learn specific tasks by interacting with the target environment following a trial-and-error mechanism and supported by a deep neural network brain [1]. Due to the inherent nature of this approach, it is useful to train them within virtual driving simulators that are as true to reality as possible. This allows for safe operating environments where the agent can learn the desired behavior, shaped through the awarding of rewards and penalties based on the status of the agent within the environment. Indeed, the objective of a DRL agent is to maximize the sum of expected rewards (i.e., the cumulative reward) over its lifetime. By exploiting the information learned about the expected utility (i.e., the sum of expected future rewards discounted by a factor), the agent can increase its long-term reward. This implies that the contributions

that make up the reward function (RF) have to be specifically designed to enable the agent to achieve the target policy.

We are interested in developing a DRL agent able to perform real-time trajectory planning and tracking in a high-fidelity simulated parking environment, taking as input sensors data and consequently acting on the steering wheel and the throttle and brake pedals, emulating the actions of a human driver. The focus of our research is low-speed maneuvers in a parking area with comb parking spaces and random distribution of the number of parked non-player vehicles (NPVs).

The remainder of the paper is structured as follows. Section 2 presents the environment, then Section 3 shows the performed experiment, while Section 4 the obtained results. Finally, conclusions are drawn in Section 5.

2 Environment

To train our model, we took a step toward more realistic simulations than previously done in [1] and [2] for parking (using Unity) and highway driving (using highway-env [3]), respectively. To increase the level of realism in terms of graphics and physics simulation, we opted for CARLA as a driving simulator [4], an open-source software created to support ADFs development. It includes several town maps and different vehicle models to which sensors such as radar, lidar, camera, etc. can be attached. In addition, CARLA exposes an API that allows users to control all aspects related to the simulation, including traffic, pedestrian behaviors, weather, sensors, etc. The official DRL-related section lacks recent updates but some recent works have recently been proposed [5, 6], showing promising results in trajectory tracking applications.

Starting from the built-in Town 5, where a parking area is already available, we removed all the unnecessary surrounding entities from the scene, so to obtain a lightweight scenario to reduce the computational cost of rendering and thus training time. The parking area, visible in Fig. 1, measures 50 × 50 m and is composed of 60 comb parking spaces with a 90° layout, 5 × 2.5 m each. Brick walls cordon off the area, preventing the agent from falling during training. The weather includes 15 different conditions of sun, rain, fog, and clouds and 41 vehicle models are available, from motorbikes to trucks, with customizable colors.

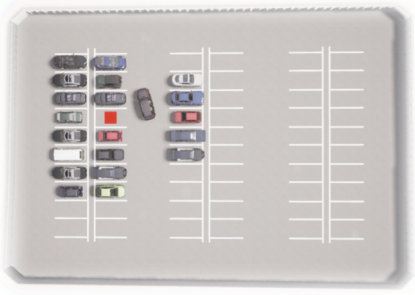

Fig. 1. Bird view of the parking area. The red square marks the target parking lot.

3 Experiment

To implement our DRL agent, we opted for Gymnasium (formerly known as OpenAI Gym [7]), an open-source Python toolkit that exposes an API to facilitate the design and implementation of DRL environments which has become a de facto standard for the communication between agents and simulation environments. In addition, Gymnasium offers the ability to define custom environments, so, we wrapped the CARLA parking setting into a Gymnasium environment to get a DRL-compatible interface. Furthermore, Gymnasium is compatible with Stable-Baselines3 (SB3) [8], an open-source Python library that offers a variety of DRL algorithms to set up the training algorithm and the neural network architecture and configuration quickly and intuitively. The schematic of the tools used and their interfacing are shown in Fig. 2.

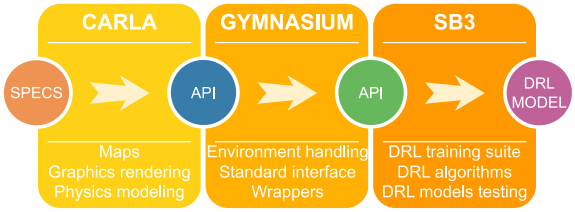

Fig. 2. Outline of the tools used for the implementation of the DRL agent.

At each episode, the vehicle starting point and orientation are chosen randomly within the drivable area. The target to be reached is one of the 60 lots, randomly chosen at each episode. To train our agent, we used the Audi e-tron vehicle model, available in CARLA. The car measures 4.90 × 1.93 × 1.62 m, and has a total mass of 2490 kg. The vehicle dynamics are modeled by CARLA with NVIDIA PhysX [9], one of the most popular physics simulation engines. The ego vehicle (EV) is equipped with a lidar sensor placed in the center of the vehicle with a 360° horizontal field of view.

The key part in the development of a DRL agent is RF. We split it into several contributions, some dense (given at each simulation step) and some sparse (awarded only when some events occur). Table 1 offers an overview the components of the RF. Distance and heading are the two dense contributions to induce EV to approach the target. Collision is sparse, needed to teach the agent to avoid other NPVs and the walls delimiting the area. Two contributions are related to reaching the target parking lot, goal and alignment. The goal reward is awarded when EV reaches the lot, while alignment rewards the agent if it parks aligned well.

Table 1. RF components.

Name	Description	Range	Type
Distance	Euclidean distance from EV to target	[-0.1, 0]	dense
Heading	Angle between EV forward vector and EV-target vector	[-0.1, 0]	dense
Collision	Collision with walls or NPVs event	[-1, 0]	sparse

Goal	Target achieved event	[0, 15]	sparse
Alignment	Angle between EV forward vector and parking lot orientation	[0, 10]	sparse

For the training of the model, we used the PPO algorithm [10]. PPO aims to optimize policies for sequential decision-making tasks. It balances exploration and exploitation by iteratively updating policy parameters based on the clipped surrogate objective, ensuring stability and reliable performance. As the backbone neural network, we opted for a multi-layer perceptron configuration, composed of 2 hidden layers of 512 neurons, whose input and output values are summed up in Table 2. To give the agent the ability to better perceive its motion, at each network update we stacked the last four input values on top of the current one. This allows the agent to capture the dynamics and changes in the environment over time.

Table 2. PPO network input and output.

Type	Quantity	Dimension	Resulting size
Input	Lidar data	61	340 (5 stacked inputs)
	Distance from target (x, y)	2	
	Speed (x, y)	2	
	Acceleration (x, y)	2	
	Angular velocity (yaw)	1	
Output	Throttle	1	4
	Steering	1	
	Brake	1	
	Reverse	1	

Given the results previously obtained in an environment similar to ours, although significantly less complex, such as [1], we adopted a curriculum learning (CL) strategy for training our agent [11]. In this case, we split the training into three distinct phases. The first phase involves only EV, no NPV is present and an episode terminates if the target is reached or if it goes over 240 steps. Collision is not penalized to incentivize the agent to explore the environment. In the second phase, the number of NPVs is chosen randomly between 0 and 20 at each episode and, again, only the goal reaching or the timeout can end the episode. Collisions are now slightly penalized with a weight equal to -0.1. Finally, in the third phase, conditions are similar to phase #2 but now a collision (penalty weight -1) terminates the episode.

4 Results

We trained our PPO agent for about 60M steps (Fig. 3). The first CL phase took 25M steps, after which the agent was able to reach the goal 100% of the time (Fig. 3a). Once this is achieved with no NPVs present in the scene, the second phase started, lasting

about 15M steps in which the agent learned to park with NPVs parked near the target parking lot. The last phase, in which collisions represented a terminal state, lasted 20M steps, achieving a final success rate of 97%.

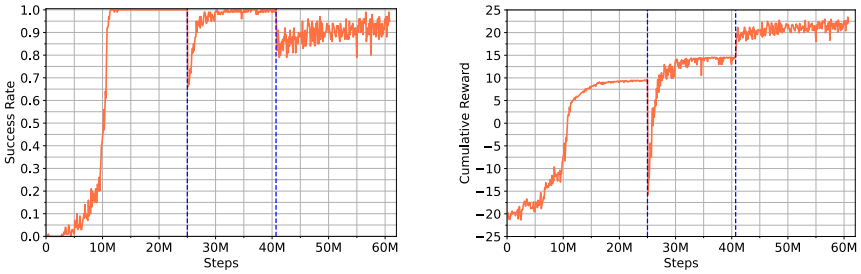

Fig. 3. The PPO agent training over ~60M steps. (a) is the success rate and (b) the cumulative reward. Dashed blue lines distinguish the three CL phases.

Although the success rate is satisfactory, some problems related to the use of CARLA arose during code development. In particular, the lack of direct support for DRL made the implementation of the environment and synchronization with Gymnasium and SB3 particularly complex. Too high values of the CARLA environment update frequency (greater than 20 Hz) greatly lengthened the training time due to the higher computational load but, conversely, frequencies that were too low (lower than 10 Hz) did not guarantee proper observation of the environment by the agent (e.g., it happened that the agent stepped on the goal without being detected). In addition, it was necessary to adopt the decision period (i.e., the frequency at which the agent takes an action) since too high network update frequencies (greater than 10 Hz) did not allow the agent to learn to move, since too short a press on the accelerator pedal is not sufficient to allow movement. A good trade-off we found is to use a CARLA environment update rate of 20 Hz and a network update rate of 5 Hz (equivalent to a decision period of 4) but this required a lot of trial and error.

5 Conclusions

This paper presented a DRL agent for real-time trajectory planning and tracking in a CARLA-simulated environment where low-speed maneuvers are performed in a parking area with comb-shaped spaces. The agent is able to achieve a 97% success rate in reaching the target parking lot without collisions. The training process involved a CL strategy with three distinct phases that gradually increased the complexity of the environment by introducing NPVs and penalizing collisions. However, the use of CARLA presented challenges in integrating DRL into the environment. Issues related to the environment update frequency and decision period had to be carefully handled to ensure proper learning and observation by the agent but the overall success rate and performance of the trained agent confirm the capability of DRL approaches for the

development of ADFs. Future work will focus on different sensor configurations and extend the range of action to more complex parking scenarios or other ADFs.

Acknowledgements

The authors would like to thank all partners within the Hi-Drive project for their cooperation and valuable contribution. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101006664. The sole responsibility of this publication lies with the authors. Neither the European Commission nor CINEA – in its capacity of Granting Authority – can be made responsible for any use that may be made of the information this document contains.

References

1. Lazzaroni, L., Bellotti, F., Capello, A., Cossu, M., De Gloria, A., Berta, R.: Deep Reinforcement Learning for Automated Car Parking. In: Berta, R. and De Gloria, A. (eds.) Applications in Electronics Pervading Industry, Environment and Society. pp. 125–130. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-30333-3_16.
2. Capello, A., Forneris, L., Pighetti, A., Bellotti, F., Lazzaroni, L., Cossu, M., De Gloria, A., Berta, R.: Investigating High-Level Decision Making for Automated Driving. In: Berta, R. and De Gloria, A. (eds.) Applications in Electronics Pervading Industry, Environment and Society. pp. 307–311. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-30333-3_41.
3. Leurent, E.: An Environment for Autonomous Driving Decision-Making, https://github.com/eleurent/highway-env, (2018).
4. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An Open Urban Driving Simulator, http://arxiv.org/abs/1711.03938, (2017). https://doi.org/10.48550/arXiv.1711.03938.
5. Pérez-Gil, Ó., Barea, R., López-Guillén, E., Bergasa, L.M., Gómez-Huélamo, C., Gutiérrez, R., Díaz-Díaz, A.: Deep reinforcement learning based control for Autonomous Vehicles in CARLA. Multimed Tools Appl. 81, 3553–3576 (2022). https://doi.org/10.1007/s11042-021-11437-3.
6. Gutiérrez-Moreno, R., Barea, R., López-Guillén, E., Araluce, J., Bergasa, L.M.: Reinforcement Learning-Based Autonomous Driving at Intersections in CARLA Simulator. Sensors. 22, 8373 (2022). https://doi.org/10.3390/s22218373.
7. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI Gym, http://arxiv.org/abs/1606.01540, (2016). https://doi.org/10.48550/arXiv.1606.01540.
8. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-Baselines3: Reliable Reinforcement Learning Implementations. Journal of Machine Learning Research. 22, 1–8 (2021).
9. NVIDIA PhysX, https://github.com/NVIDIA-Omniverse/PhysX, (2023).
10. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms, http://arxiv.org/abs/1707.06347, (2017).