EE245 Project Final Report

A Study on Reinforcement Learning for Parking: Vision/Radar/LiDAR Sensing

Kunyi Yu
Student ID: 862548836
University of California, Riverside
Riverside, CA 92521
kyu135@ucr.edu

Abstract

Autonomous parking is a critical yet challenging task in self-driving systems. This project investigates the effectiveness of different sensing modalities—Vision, Radar, and LiDAR—for autonomous parking using reinforcement learning (RL). This project horizontally compare the performance of three widely-used RL algorithms (TD3, SAC, and PPO) in a simulated parking environment with varying vehicle densities. Experiments show that the choice of sensing modality and algorithm is crucial for success. The results demonstrate that Radar-based observation (Kinematics) is the most effective input, enabling off-policy agents like SAC and TD3 to achieve high success rates, with SAC reaching up to 90.9%. In contrast, agents using Vision or LiDAR inputs, as well as the on-policy PPO algorithm, failed to learn effective parking policies under the tested configurations. These findings highlight the significant impact of observation representation on the performance and sample efficiency of RL agents in autonomous parking tasks.

1 Introduction 15%

Traffic accidents caused by human judgment errors remain a leading cause of fatalities worldwide. However, the rapid development of autonomous driving technologies holds promise for significantly reducing such incidents. In December 2024, the author visited the Bay Area and observed a growing presence of autonomous vehicles (AVs) on the road, either undergoing testing or already in commercial operation. Waymo, one of the industry pioneers, has been operating a fleet of AVs in San Francisco since August 2021. On the other hand, unlike Waymo's radar-based approach, Tesla's Full Self-Driving (FSD) system relies primarily on vision-based sensing. Tesla's commercial success demonstrates that a camera-only solution can be viable for autonomous driving.

Sensing modality, like vision, Radar or LiDAR, are widely researched and adopted in the industry, each with its own advantages and limitations. A 2024 news report [1] highlighted an incident where Waymo's AVs were excessively honking in a San Francisco parking lot, disturbing nearby residents multiple times at night. The issue was reportedly caused by interference from other vehicles, leading to a deadlock scenario—a common challenge in multi-agent systems. This also underscores the complexity and importance of autonomous parking as a research topic.

Thus, the modality of sensing is crucial for autonomous driving, especially in complex environments like parking lots where dense traffic and pedestrians could lead to problems such as deadlocks and security concerns. Reasonable solutions could be combined with different sensing modalities or dynamically switching between them to enhance the robustness of the system. Therefore, this project aims to explore the effectiveness of different sensing modalities in various parking scenarios by horizontally comparing the performance of several reinforcement learning (RL) algorithms.

Preprint. Under review.

In the rest of this report, Section 2 reviews related work in the the application of RL algorithms in autonomous parking and some common sensing modalities. Section 3 describes the methodology, including task definition, environment setup, RL algorithms, observation modalities, reward function and training settings. Section 4 presents the experimental results, including the performance comparison and findings. Finally, Section 5 includes summary, contributions, and future work.

2 Related Work 15%

This section reviews related work in the application of reinforcement learning (RL) algorithms in autonomous parking and common sensing modalities in autonomous driving. Most papers in this section are based on the `highway-env` environment [2], which is the platform used in this project and also widely adopted in the autonomous driving research community.

2.1 Reinforcement Learning in Autonomous Parking

The `highway-env` environment [2] is a widely used platform (2.9k star on GitHub) for simulating autonomous driving scenarios, including parking tasks. The well written documentation and abided by OpenAI Gym API make it easy to use and extend. The environment provides multiple sensing modalities (Kinematics, GrayScale Image, LiDAR, etc.), continuous/discrete action spaces, and predefined scenarios with reward functions.

There are several papers study the autonomous parking problem directly in the `highway-env` environment. Kapoor et al. [3] introduces a model-based RL approach that integrates neural dynamics prediction with Signal Temporal Logic (STL) guided model predictive control, applied to robotics and autonomous driving. This approach is demonstratesd on toy robotics tasks including a parking-lot scenario. The key strenth is its use of formal task specifications to helps avoid reward-shaping issues. A weakness is that it relies on a learned model and uses computationally expensive planning.

Moreira [4]'s master's thesis proposes a deep reinforcement learning method with SAC, DDPG, and TD3 algorithms to teach a wheeled vehicle to park in confined spaces. The agents are trained in a simulated environment to follow a predefined parking trajectory. The study finds that TD3 converges fastest and achieves the most reliable policy, while SAC also learns satisfactorily but DDPG is less stable and efficient. Strengths of this work include a thorough comparison and careful designed reward function. However, the method needs to predefined the parking path manually and leaves out perception or sensing issues.

Lazzaroni et al. [5] presents a DRL-based agent trained in different environments (a Unity-based, highway-env, and CARLA) for low-speed parking maneuvers, achieving a 97% success rate. The paper also uses Stable-Baselines3 [6] toolkits for training the agents, which provides a set of reliable RL algorithm implementations. The ego-vehicle is equipped with a 360-degree LiDAR sensor, and the agent observation also includes relative target position and self velocity. Training uses PPO algorithm with MLP. Although the high success rate is impressive, the long training time (over 60M timesteps) and needs for hyperparameter tuning are drawbacks.

Note that Stable-Baselines3 [6] is library of reliable implementations of standard deep reinforcement learning algorithms. It is not specifically designed for autonomous driving, but provides high-quality code for many algorithms (PPO, DDPG, SAC, TD3, A2C, etc.) with consistent API. There also exist some YouTube videos [7, 8] visulizing the training process of reverse parking and parallel parking. The former video uses a version of Rainbow DQN on raw camera images, showing that even a complex 3D task like reverse parking can be learned but missing performance metrics and robustness analysis due to the pattern of videos. The latter one utilizes a PPO-based policy using Unity3D environment. However, details on sensors (maybe also vision-based), reward function, or evaluation are not provided.

A survey paper [9] provides a comprehensive overview of RL applications in autonomous driving, not limited to only parking scenarios. With in this survey, some instances parking tasks are discussed, including a RL-guided Monte Carlo Tree Search (MCTS) algorithm research. Strengths is its comprehensive coverage of perception, planing, and control. As it is a high-level review, however, sensor modalities for parking are not included.

2.2 Sensing Modalities in Autonomous Driving

The operational logic of autonomous system, from a simple housekeeping robot to a complex self-driving car, can be summaryized into a foundamental loop: See, Think, Act. In this paradigm, the "See" component-perception-serves as the foundation layer upon which all subsequent capabilities are built. To meet the stringent safety and reliability demands of autonomous driving, the academic and industrial have invested significant sensing strategies: cameras (Vision), Radio Detection and Ranging (Radar), and Light Detection and Ranging (LiDAR). As no signle sensor can provide a complete and infallible representation of the environment, each sensing modality provide unique advantages and inherent limitations.

Yurtsever et al. [10] provides a comprehensive survey on common practices and emerging trends, including serveral sensoring technologies: vision-based sensors (monocular cameras, omnidirectional cameras, and event cameras), radar, and LiDAR (Point clouds). Firstly, vision-based sensors have advantages in color sensing, passive sensing, and low cost due to established technology. However, their illumination sensitivity and difficulty in depth perception are significant drawbacks. Radar sensors, on the other hand, have better long range detection, robustness to bad weather, and also low cost. However, they have lower resolution and their field of view is limited. Lastly, LiDAR have pros in high accuracy, accuracy in depth perception, and robustness to illumination changes. However, they are expensive, heavy, and require large-scale data processing. If combing these sensors, the advantages of each sensor can be utilized. The figure 1 illustrates the pros and cons of these three sensing modalities.

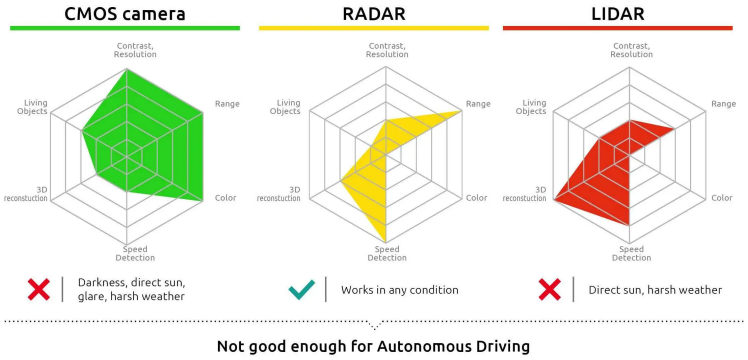


Figure 1: Pros and cons of common sensing modalities in autonomous driving. (from the internet)

Some studies also explore the integration of multiple sensing modalities. Pederiva et al. [11] alleviates the computationally demands of tranditional detecting objects by integrating Camera and LiDAR data. Their method utilizes cutting-edge Deep Learning techniques into the feature extraction process achieving a 2.39% accuracy improvement and 10% parameter reduction within 10 ms inference time. The lightweight and powerful model is suitable for real-time applications.

3 Methodology 30%

This section describes the methodology of this project, including task definition, environment setup, reinforcement learning (RL) algorithms, observation, reward function and training settings.

3.1 Task Definition and Environment Setup

The ultimate goal of teaching a vehicle to park is to enable it to autonomously navigate and park in any real-world parking lot fast and precisely while avoiding collisions with other vehicles, obstacles, and pedestrians. For each epoch, the ego-vehicle is spawned at a random position and orientation in the parking lot, and the goal is to park it in a randomly selected parking spot. The objective function is to maximize a return $G_t = \sum_{t=0}^{T} \gamma^t r_t$, where r_t is the reward at time step t and γ is the discount factor.

To abstract the task and make it suitable for RL research, the project used a simulated environment based on the `highway-env` platform [2] with varing other vechicles denseness to change the difficulty.

In details, there will be `empty`, `normal`, and `almost-full` parking lots, which have 0, 3, and 10 other vehicles, respectively. The parking lot is a 2D continuous space with 15 spots for both north and south side as the figure 2 shows. The ego-vehicle is a 4-wheeled car can be controlled by fixed steering angles and acceleration. The success of parking is defined as the ego-vehicle being within a certain distance to the parking spot without angle requirement (but will be penalized in the reward function).

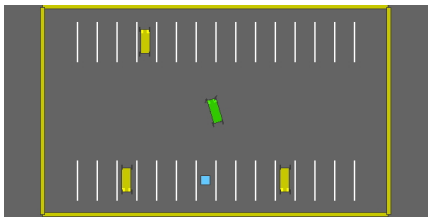


Figure 2: The parking lot environment `empty` with 0 other vehicles. The ego-vehicle is the green one, other vehicles are yellow, and the parking spot is the blue rectangle. There are walls on each side of the parking lot.

Note that the `highway-env` environment follows the OpenAI Gym API and configurations are stored in a YAML file where we can specify the action, observation, reward, vehicle number, has walls or not, and other settings. To make a gif output, `offscreen-rendering` should be unenabled.

3.2 Reinforcement Learning algorithms

The reason for adopting reinforcement learning (RL) algorithms in this project is that the autonomous parking task is a sequential decision-making problem with a continuous action space, where optimal strategies must be learned through interactions with the environment. RL algorithms are well-suited for such problems due to their ability to handle complex dynamics, perform long-term planning, and optimize policies without requiring expert demonstrations. This project utilizes the Stable-Baselines3 [6] library to implement three widely used RL algorithms: Twin Delayed DDPG (TD3) [12], Soft Actor-Critic (SAC) [13], and Proximal Policy Optimization (PPO) [14]. These algorithms are particularly effective for continuous control tasks. A brief overview of each is provided below.

TD3 is and off-policy actor-critic algorithm that builds on the DDPG algorithm by introducing several improvements: target policy smoothing, delayed policy updates, and twin Q-networks to reduce the problem of overestimation bias. These enhancements make TD3 more stable, data-efficient, and accurate than DDPG, especially in environments with high-dimensional action spaces. However, TD3 requires careful hyperparameter tuning and deterministic policies could be less robust in stochastic settings.

SAC is an off-policy algorithm that improves a stochastic policy by maximizing entropy in addition to the expected return (i.e., exploration). This encourages a strong sample efficiency, more balanced exploration and exploitation, and higher tolerance to environment noise. Meanwhile, SAC has a relative high computational cost and also requires careful hyperparameter tuning on entropy coefficient and learning rates.

PPO is an on-policy policy gradient method that uses a clipped surrogate objective to optimize the policy. It is a simple yet stable and effective algorithm in many continuous control tasks. However, the on-policy nature makes it less sample-efficient and may perform suboptimally in high dimensional action spaces.

3.3 Observation, Reward Function and Training Settings

Usually, the training settings of RL algorithms are highly influence the performance of the agent and transferability to real-world scenarios. This subsection first describes the observation modalities, followed by the reward function and other training settings.

Observation Modalities. The `highway-env` environment naturely supports all the three sensing modalities we need: Vision (GrayScale Image), Radar (Kinematics), and LiDAR, where in the parentheses are the names used by the `highway-env`.

1. **Vision (GrayScale Image):** The observation is a $W \times H$ image, where W and H are configurable parameters. Several images can be stacked to enable the agent to perceive dynamic changes without explicit memory.
2. **Radar (Kinematics):** The observation is a $V \times F$ array, where V is the number of vehicles in the environment and F is the number of features (e.g., position, velocity, acceleration, etc.). Options include normalization and using absolute positions.
3. **LiDAR:** The observation is a $N \times 2$ array, where N is the number of divided angular sectors around the ego-vehicle. For each sector, 1) the distance to the nearest collidable object and 2) the relative velocity along that direction are recorded. Configurable parameters include normalization, number of sectors, and maximum detectable distance. Figure 3 illustrates the LiDAR observation.

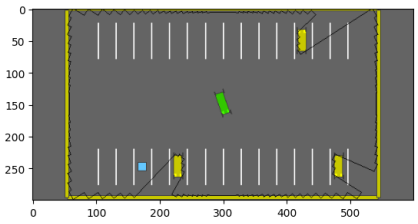


Figure 3: The LiDAR observation.

Reward Function. The reward function is crucial for guiding the agent to learn the desired behavior. In this project, the reward function is designed to encourage parking while penalizing collisions and deadlocks, which remains the default setting as environment provided. The reward function is defined as follows:

$$\begin{aligned} \mathbf{s} &= [x, y, v_x, v_y, \cos(h), \sin(h)]^T \\ \mathbf{w} &= [1, 0.3, 0, 0, 0.02, 0.02]^T \\ R_{\text{total}} &= (\mathbf{w}^T \mathbf{s}) - 5 \cdot \mathbb{I}(\text{collision}) + 0.12 \cdot \mathbb{I}(\text{success}) \end{aligned}$$

where $\mathbf{w}^T \mathbf{s}$ reward the agent for moving towards the parking spot in a good direction, $-5 \cdot \mathbb{I}(\text{collision})$ penalizes the agent for colliding, and $0.12 \cdot \mathbb{I}(\text{success})$ rewards the agent for successfully parking.

Training Settings. The training settings are summarized in Table 1.

Parameter	**Value**
Action Space	Continuous (steering angle, acceleration)
Observation Space	Varying by sensing modality (see above)
Discount Factor (γ)	0.99
Training Timesteps	500,000
Batch Size	256
Learning Rate	0.001
Target Network Update Frequency (TD3/SAC)	1,000
Policy Update Frequency (TD3)	2
Entropy Coefficient (SAC)	Auto-tuned
Number of Vehicles in Environment	Varies by difficulty (0, 3, or 10)
Parking Spot Size	Configurable (default: 2m x 5m)
Maximum Detectable Distance (LiDAR)	Configurable (default: 50m)
Number of LiDAR Sectors	Configurable (default: 36)
Render Mode	RGB array for visualization and GIF

Table 1: Summary of training settings.

4 Experiments 30%

This section presents the experimental results of the project, where all the conducted experiments are summarized in Table 2. Then, performance demonstrations, findings, and analysis are provided in the following subsections.

vision input

algo	empty	normal	almost-full
TD3	x		
SAC	x		
PPo	x	x	x

radar input

algo	empty	normal	almost-full
TD3	73.3%	65.3%	
SAC	90.9%	81.4%	
PPo	x	x	x

LiDAR input

algo	empty	normal	almost-full
TD3	x	x	
SAC	x	x	
PPo	x	x	x

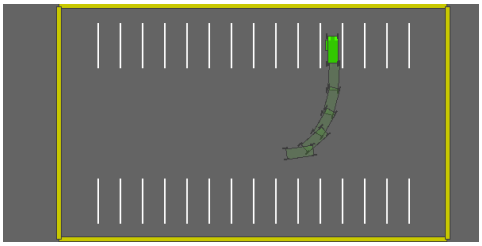
**Notes:* All experiments are run for 500k time-steps.

`Blank block`	Experiment to-do.
`[x]%`	Success rate of the experiment.
`x`	Success rate is less than 10%.

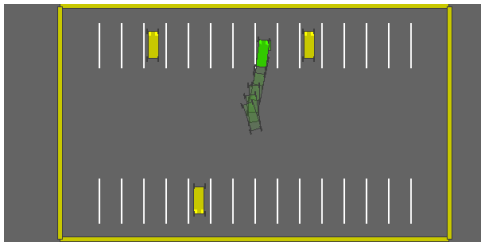
Table 2: Summary of experimental results.

4.1 Performance Demonstrations

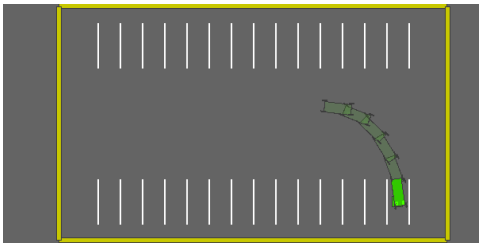
As the results in Table 2 show, the TD3 and SAC algorithms can successfully learn to park in the `empty` and `normal` (maybe also `almost-full`) parking lots with Radar observation, while other combinations yield poor performance. The Figure 4 shows the trajectories of those good experiments, where agents shows a good navigation behavior and successfully park in the parking spot without colliding with other vehicles. There are some rare cases where the agent can even learn to reverse park instead of always forward driving into the parking spot, which is a more efficient way to park in real-world scenarios.



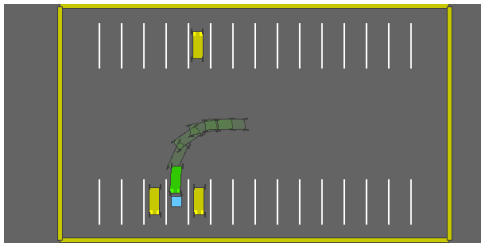
SAC with Radar in the empty parking lot.



SAC with Radar in the normal parking lot.



TD3 with Radar in the empty parking lot.



TD3 with Radar in the normal parking lot.

Figure 4: Performance demonstrations of the successful experiments. The trajectories only show the last 5 time-steps of the agent. From gifs, we can see that the agents learned complex behaviors such as reversing and long-distance parking.

For those failed experiments, the agents either always collide with other vehicles or walls but still can reduce epoch time by lossing the game. This might be the relative small reward for successful parking (0.12) compared to the collision penalty (-5) and the sparse reward problem lead to the failure of trend-off between exploration and exploitation. A demonstration of the failed experiments is shown in Figure 5, and a proof of successful reward coverage and total epoch time reduction is shown in Figure 6.

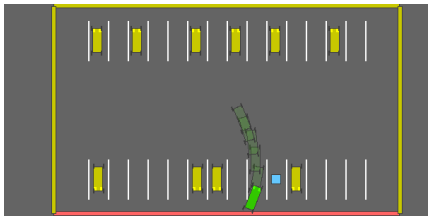


Figure 5: Performance demonstration of the failed experiment (PPO with Radar in the almost-full parking lot.). The agent always collides with other vehicles or walls, but can still reduce epoch time by lossing the game.

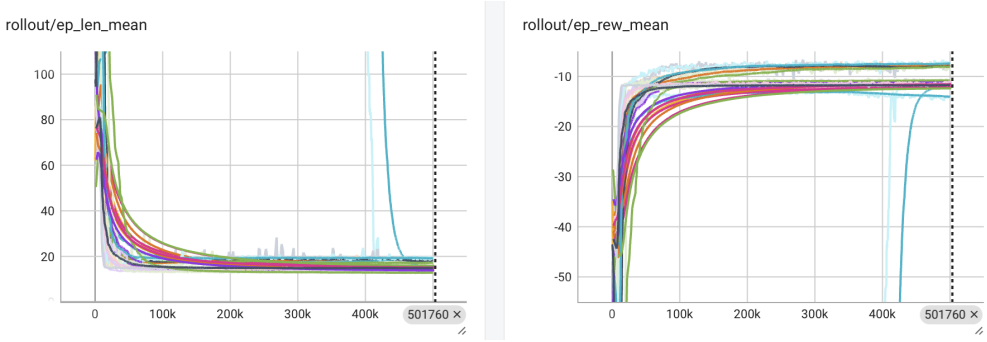


Figure 6: Proof of successful reward coverage and total epoch time reduction. The left figure is the average epoch length, and the right figure is the average reward per epoch.

4.2 Findings and Analysis

Training Time and Sample Efficiency. The training time for each algorithm varies significantly, with PPO fastest, while TD3 and SAC require more time to converge. Another factor is the observation modality. For example, Vision-based agents has a training speed of 5 fps (frames per second) without GPU acceleration, while Radar-based and LiDAR-based agents can achieve more than 100 fps. This is because the Vision-based agents need to process high-dimensional images, which are computationally expensive, even if I tried to use a small CNN model.

Algorithm Performance. Although in some observations, all three algorithms can not learn to park, but PPO still fails to learn a good policy using a Radar observation where TD3 and SAC can achieve a high success rate. This is because PPO is an on-policy algorithm, which requires more samples to learn a good policy. In contrast, TD3 and SAC are off-policy algorithms, which can reuse past experiences from a replay buffer, making them more sample-efficient.

Sensing Modalities. The results show that Radar is the most effective sensing modality for parking tasks, achieving the highest success rates and fastest training times. However, it should admit that the environment is a 2D continuous simulated environment where Vision input is the up-to-down view instead of the front view or 360-degree view and the LiDAR input is a 2D array instead of a 3D point cloud. This difference may lead to the poor performance of Vision and LiDAR in this project. In real-world scenarios, Vision-based and LiDAR-based systems are more commonly used, and they can provide rich information about the environment. Therefore, it is important to further explore the performance of these modalities in more complex environments.

5 Conclusion 5%

This project aimed to evaluate the performance of different reinforcement learning algorithms (TD3, SAC, PPO) combined with various sensing modalities (Vision, Radar, LiDAR) for the autonomous parking problem. Through a series of experiments in a simulated environment, we have drawn several key conclusions.

Summary and Contributions: Our primary finding is that the combination of off-policy algorithms (SAC and TD3) with Radar-based kinematic observations yields the best performance, successfully learning to park in empty and moderately occupied lots. The Soft Actor-Critic (SAC) algorithm, in particular, proved to be the most robust and sample-efficient, achieving the highest success rate. In contrast, the on-policy PPO algorithm and agents relying on high-dimensional Vision or LiDAR data failed to converge to a successful policy within the 500,000-timestep training budget. This suggests that for this simulated task, the abstract and direct state representation from Radar is far more conducive to learning than the raw sensory data from the other modalities, and that off-policy methods are better suited for this continuous control problem due to their superior sample efficiency.

Future Work: While this study provides valuable insights, there are several avenues for future research. First, improving the performance of Vision and LiDAR-based agents is a key priority. This could be achieved by employing more sophisticated neural network architectures, utilizing pre-training, or transitioning to more realistic 3D simulation environments like CARLA, which better represent the richness of these sensors. Second, future work could explore sensor fusion techniques, combining the strengths of different modalities to create a more robust and reliable parking system. Finally, applying more advanced reward shaping techniques or curriculum learning could help agents tackle more complex and densely populated parking scenarios, which proved challenging in our current experiments.

Acknowledgments

The final project was independently conducted by the author Kunyi Yu. Meanwhile, the author would like to thank Professor Jiachen Li and TAs for their patient guidance and help.

References

[1] Tim Johns. Waymo cars honk at each other throughout the night, disturbing sf neighbors, 2024. URL `https://abc7ne.ws/3Ai47Ci`.

[2] Edouard Leurent. An environment for autonomous driving decision-making. `https://github.com/eleurent/highway-env`, 2018.

[3] Parv Kapoor, Anand Balakrishnan, and Jyotirmoy V Deshmukh. Model-based reinforcement learning from signal temporal logic specifications. *arXiv preprint arXiv:2011.04950*, 2020.

[4] Dinis Moreira. Deep reinforcement learning for automated parking. Master's thesis, Universidade do Porto (Portugal), 2021.

[5] Luca Lazzaroni, Alessandro Pighetti, Francesco Bellotti, Alessio Capello, Marianna Cossu, and Riccardo Berta. Automated parking in carla: A deep reinforcement learning-based approach. In *International Conference on Applications in Electronics Pervading Industry, Environment and Society*, pages 352–357. Springer, 2023.

[6] 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. URL `http://jmlr.org/papers/v22/20-1364.html`.

[7] Samuel Arzt. Ai learns to park - deep reinforcement learning, 2019. URL `https://www.youtube.com/watch?v=VMp6pq6_QjI`.

[8] Matthew Euliano. Parallel parking using reinforcement learning - unity3d and ml-agents, 2022. URL `https://www.youtube.com/watch?v=QDVwSAgY6cA`.

[9] Badr Ben Elallid, Nabil Benamar, Abdelhakim Senhaji Hafid, Tajjeeddine Rachidi, and Nabil Mrani. A comprehensive survey on the application of deep and reinforcement learning approaches in autonomous driving. *Journal of King Saud University-Computer and Information Sciences*, 34(9):7366–7390, 2022.

[10] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020.

[11] Marcelo Eduardo Pederiva, José Mario De Martino, and Alessandro Zimmer. A light perspective for 3d object detection. In *Seventeenth International Conference on Machine Vision (ICMV 2024)*, volume 13517, pages 136–143. SPIE, 2025.

[12] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[13] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

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

Appendix

How to Find the Code?

The GitHub public repository for this project is at `https://github.com/Rock3Yu/EE245_Advanced_Robotics`. Please feel free to explore the code and data files. The structure of the repository is as follows:

```
./project
  |-- final
  |   |-- main.pdf
  |   |-- main.tex
  |   |-- ...
  |-- proposal
  |   |-- main.pdf
  |   |-- main.tex
  |   |-- ...
  |-- src
      |-- HighwayEnv
      |-- make_gif.py
      |-- models
      |-- parking_config.py
      |-- smallCNN.py
      |-- tensorboard_logs
      |-- train_0_kinematics.py
      |-- train_1_vision.py
      |-- train_2_lidar.py
      |-- ...
# The purpose of each dictory/file is as their name suggests.
```