iPark: Intelligent Parking

Members:

Kushagra (2000271530029)
Abhinav Sajjan (2000271530006)
Himanshu Mishra (2000271530024)
Shreyashi Jaiswal (2000271530056)

Guide:

Mr. Samender Singh

Project VIVA Presentation

Outline

Contents of the presentation

- Introduction
- Tools and Technology Utilised
- System Overview & Architecture
- Results
- Conclusion
- Demonstration

Introduction

Introduction

- Project Goal: Develop an autonomous parking system using Reinforcement Learning (RL) within a Unity simulation environment.

- Background: The rapid development in AI and ML technologies has significant applications in the automotive industry, especially in autonomous driving and parking.

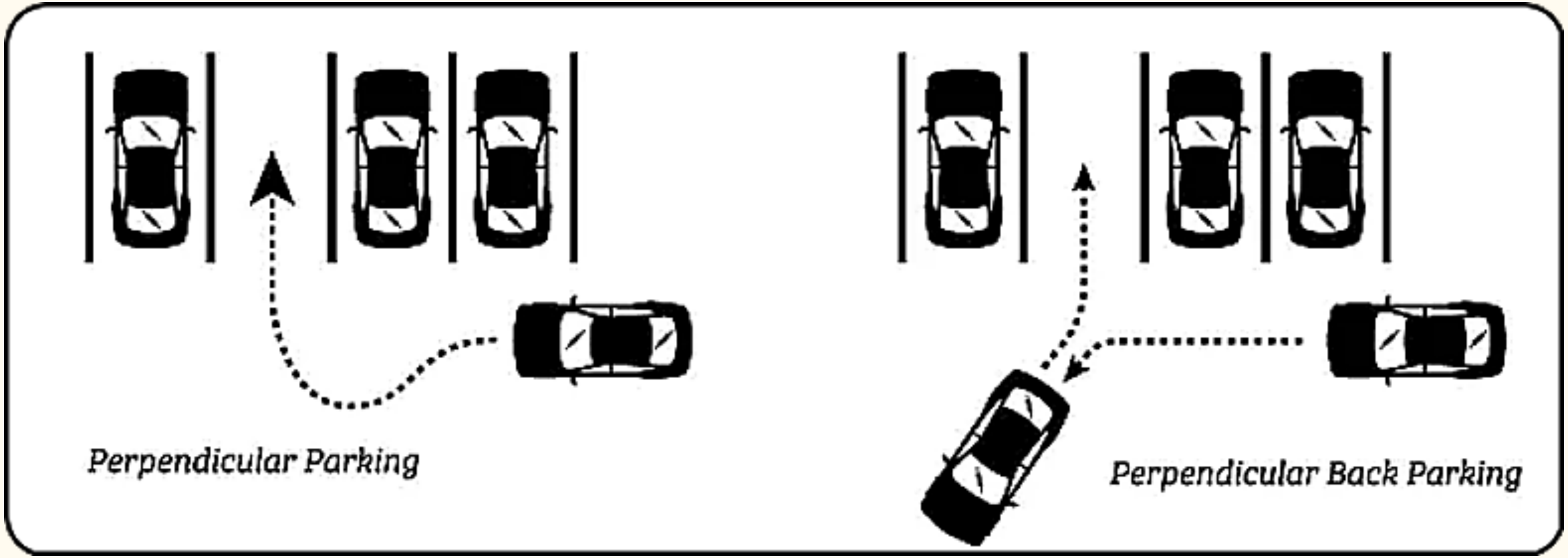
- Identification of the Problem:
    - Traditional parking methods are time-consuming and inefficient.
    - Increasing urbanization demands better parking solutions.

Introduction Contd.

- Significance:
  - Reduces time spent on parking.
  - Enhances user convenience and traffic flow.
  - Contributes to the development of fully autonomous vehicles.

- Objectives:
  - Create an RL model capable of autonomous parking.
  - Implement diverse and realistic parking scenarios in Unity.
  - Train and evaluate the RL agent's performance.
  - Provide a scalable solution for real-world applications.

- Note: For more details, refer to the report PDF file Chapter 1.


Perpendicular Parking
Perpendicular Back Parking

Tools and Technologies Utilised

Tools and Technologies Utilised

Reinforcement Learning Algorithms:

- Proximal Policy Optimization (PPO):
  - Advantages:
    - Stability: PPO is designed to maintain stability and reliability during training by limiting the update step size.
    - Simplicity: Easier to implement compared to other RL algorithms.
    - Performance: Often performs well across a range of tasks due to its robust training mechanism.
  - Mechanism:
    - Uses a clipped objective function to ensure the new policy does not deviate significantly from the old policy.
    - Balances exploration and exploitation effectively.
  - Suitability for iPark: Effective in environments with discrete and continuous actions, making it versatile for different parking scenarios.

Tools and Technologies Utilised Contd.

Soft Actor-Critic (SAC):

- Advantages:
  - Sample Efficiency: SAC is known for its high sample efficiency, making it suitable for environments with continuous action spaces.
  - Entropy Regularization: Encourages exploration by adding an entropy term to the reward, which helps in learning diverse behaviors.
  - Performance: Generally achieves state-of-the-art results in continuous control tasks.
- Mechanism:
  - Uses both value and policy networks, optimizing them simultaneously.
  - Incorporates a stochastic policy that improves exploration and robustness.

Tools and Technologies Utilised Contd.

- Simulation Environment:
    - Unity3D Game Engine: A powerful cross-platform engine used for developing simulations and games.
    - Unity Editor: A component of Unity for designing and developing interactive 3D content.
    - Unity ML-Agents Framework: A toolkit for creating intelligent agents using RL within the Unity platform.

- Note: For more details, refer to the report PDF file Chapter 3.

System Overview & Architecture

System Overview & Architecture

System Components:

- RL Training Module: Implements RL algorithms for training the parking agent.
- Simulation Environment: Uses Unity to create realistic parking scenarios for training and testing.
- User Interface (UI) Component: Provides an interactive UI for users to interact with the system and visualize results.
- Performance Metrics Component: Tracks and analyzes the performance of the parking agent.

System Overview & Architecture Contd.

Project Environment:

- Unity Engine: Central platform for creating and running the simulations.
- C-Sharp (C#) Programming Language: Used for scripting and developing components within Unity.
- Visual Studio: Integrated development environment (IDE) used for coding and debugging.
- GitHub: Version control platform for managing code and collaboration.

System Overview & Architecture Contd.

- Project Concept:
    - Working Concept: The RL agent learns to navigate and park in various scenarios by interacting with the environment and receiving feedback.
    - Design & Development of Components: Includes the creation of simulation environments, agent behaviors, and performance tracking systems.
    - Amalgamation of Components: Integration of all system components to create a cohesive and functional autonomous parking system.

- Note: For more details, refer to the report PDF file chapter 4.

Interaction Between Components

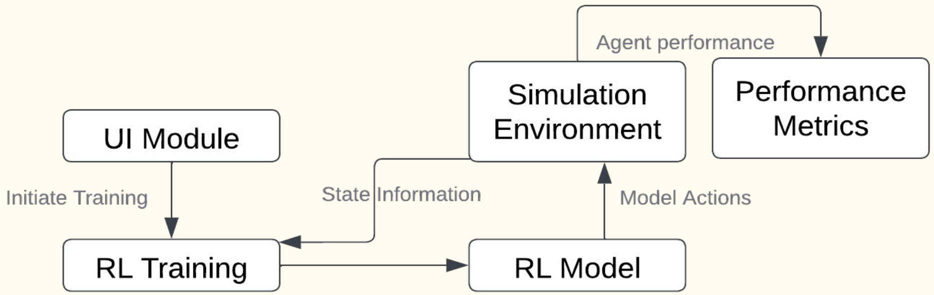


Fig 3.2 Interaction between components

Data Flow

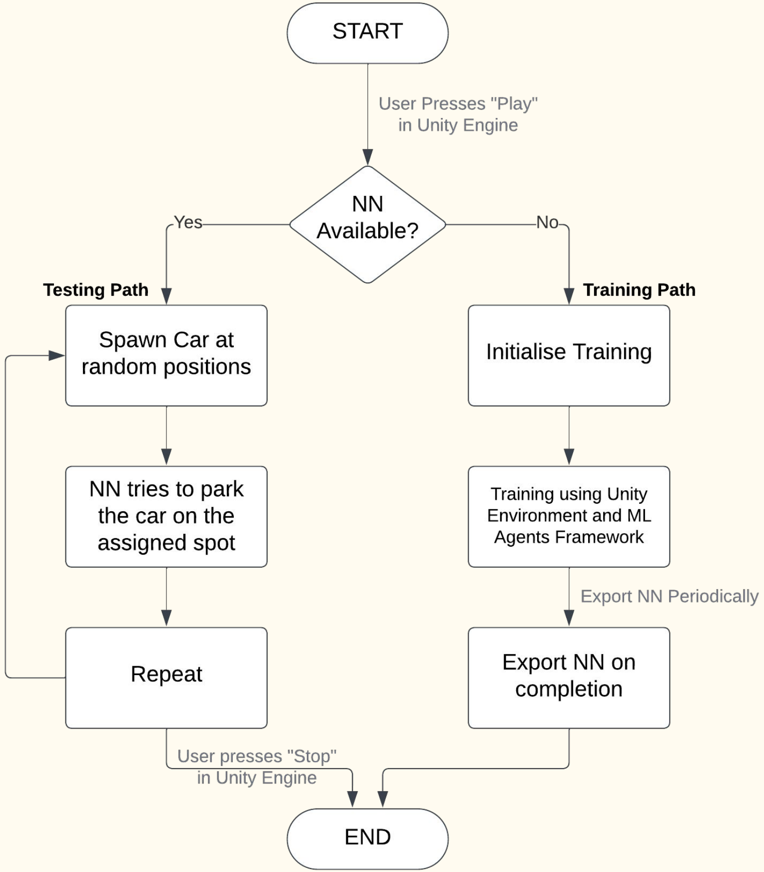


Fig 4.4 Data Flow Diagram

Results

Results

- Evaluation Tests: Conducted multiple tests to evaluate the efficiency and effectiveness of the parking agent.

- Key Metrics:
  - Success Rate: The percentage of successful parking attempts.

- Performance:
  - Efficiency ranged from 78.57% to 89.38%.
  - Highest efficiency model achieved 89.38%.

Results Contd.

- Graphs & Analysis: Included detailed graphs and analysis to illustrate the agent's performance over time.

- Note: For more details, refer to the report PDF file chapter 5.

Conclusion

Conclusion

- Achievements: Successfully developed a robust RL-based autonomous parking system.

- Future Enhancements:
  - Integration with Real-World Vehicles: Implementing the system in actual cars to test and improve real-world performance.
  - Advanced Driver Assistance Systems (ADAS): Enhancing the system to work in conjunction with ADAS for better safety and efficiency.
  - More Diverse Scenarios: Including more complex parking scenarios such as multi-level parking structures.
  - Emerging Technologies: Leveraging technologies like Vehicle-to-Infrastructure (V2I) communication for better decision-making.

Conclusion Contd.

- Project Impact: Demonstrates significant advancements in autonomous parking technology.
- Potential: Paves the way for safer, more efficient, and more convenient parking experiences.
- Contributions: Adds to the body of knowledge in machine learning applications for autonomous vehicles.
- Note: For more details, refer to the report PDF file chapter 6.

Demonstration

Video present on the pendrive.

References

References

- Joy Zhang (2021). A hands-on introduction to deep reinforcement learning using Unity ML-Agents. Coder One. https://www.gocoder.one/blog/hands-on-introduction-to-deep-reinforcement-learning

- Code Monkey (2021). Machine Learning AI in Unity (ML-Agents). YouTube. https://tinyurl.com/ykjpwqdk

- U. T. (2022). Unity-Technologies/ml-agents. GitHub. https://github.com/Unity-Technologies/ml-agents

- U. T. (2022). ML-Agents Toolkit Overview https://unity-technologies.github.io/ml-agents/ML-Agents-Overview/

References Contd.

- Andres Leonardo Bayona (2023). Comparative Study of SAC and PPO in Multi-Agent Reinforcement Learning Using Unity ML-Agents (Universidad de los Andes). https://repositorio.uniandes.edu.co/server/api/core/bitstreams/cadff679-f3f3-43fa-a543-d6313c0a4932/content

- Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., Lange, D. (2020). Unity: A General Platform for Intelligent Agents (arXiv:1809.02627). arXiv. http://arxiv.org/abs/1809.02627

- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. (2017). Proximal Policy Optimization Algorithms (arXiv:1707.06347). arXiv. http://arxiv.org/abs/1707.0634735

- ABL. (2023, May 30). PPO vs SAC [Video]. YouTube. https://www.youtube.com/watch?v=ZtdtpRmoFSE

References Contd.

- Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. (2019b). Quantifying generalization in reinforcement learning. (arXiv:1812.02341). arXiv. https://arxiv.org/abs/1812.02341

- Lample, G. and Chaplot, D. S. (2017). Playing fps games with deep reinforcement learning. AAAI. https://ojs.aaai.org/index.php/AAAI/article/view/1082736

- Note: For more details, refer to the references section of the report PDF file.

Thank You!

Questions?