Reinforcement Learning-Based Motion Planning

For Automatic Parking System

ID :- 202201239
Name :- Aryan Solanki

Course ID :- IE415 (CAS)
Prof. :- Sujay Kadam

ABSTRACT

This study addresses the challenges of automatic parking motion planning by developing a model-based reinforcement learning approach that eliminates the heavy reliance on human expertise and prior knowledge. Traditional methods depend on expert drivers' data, which is often scarce and does not guarantee optimal parking performance across safety, comfort, efficiency, and accuracy. Our method learns parking strategies autonomously by iteratively generating data, evaluating it, and training a neural network. The trained network improves the data generation process in subsequent cycles.

We create a simulated environment to accelerate learning through interactions between the agent and the model, bypassing the need for expert input. A data generation algorithm, combining Monte Carlo Tree Search (MCTS) with longitudinal and lateral policies, generates diverse parking scenarios. A reward function ensures that only high-quality data contributing to multi-objective optimization is used for training.

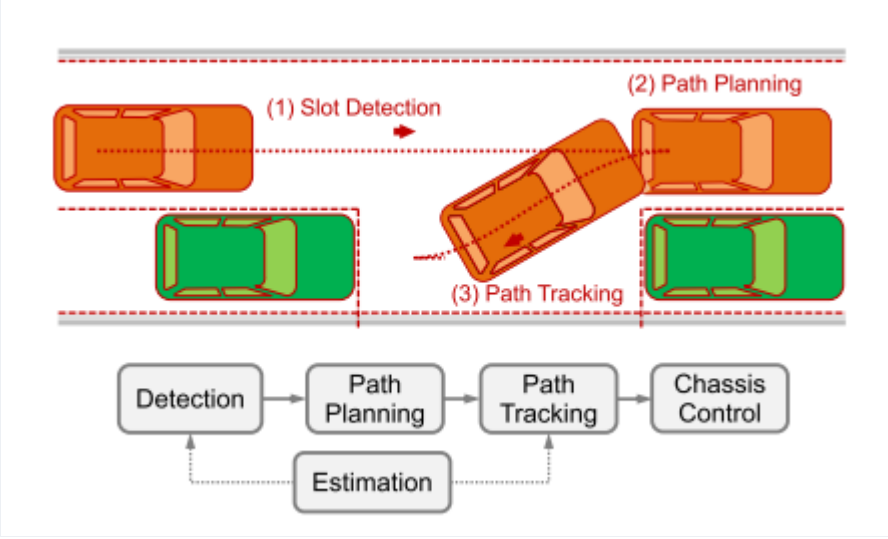
The neural network learns parking strategies from this refined data, ensuring optimal performance. Real vehicle tests show that our approach achieves better parking efficiency, accommodates a wider range of initial parking conditions, and outperforms traditional systems, demonstrating the effectiveness of the proposed method.

INTRODUCTION

The increasing number of vehicles has led to serious traffic congestion and environmental issues, such as higher fuel consumption and emissions. Connected automated vehicles offer a solution by reducing these problems and improving traffic flow. Additionally, the demand for time-sharing electric vehicles is growing due to their efficiency and eco-friendliness. However, challenges like tight parking spaces, unskilled driving, and the need for precise parking for wireless charging remain. Drivers also require comfort during parking, with minimal sudden movements and smooth steering operations.

To address these issues, achieving safe, comfortable, efficient, and accurate parking is critical for the future of shared vehicles. Automatic parking systems typically include several key components: detecting parking spaces, motion planning, tracking the vehicle's path, estimating the vehicle's posture, and controlling the vehicle’s movement. Motion planning acts as the link between understanding the environment and controlling the vehicle, determining how the vehicle moves based on real-time data about the parking space and its surroundings.

Research in parking motion planning focuses on various approaches, including methods that rely on expert drivers’ data, human knowledge, and reinforcement learning. These methods aim to improve parking performance while addressing safety, efficiency, and comfort.



Reinforcement Learning-Based Method :

Traditional parking methods rely heavily on expert driver data or human knowledge, which require large amounts of high-quality data. Even with expert data, the system's performance may be limited by human expertise. In contrast, reinforcement learning (RL) allows systems to learn from their own experience, potentially surpassing human capabilities.

RL involves the agent (vehicle) interacting with its environment to learn strategies. RL methods can be classified into two types based on whether an environment model is used:

1. **Model-Based Methods:** These first create a model of the environment using a state transition probability matrix and a reward function, and then determine the best actions to maximize cumulative rewards.

2. **Model-Free Methods:** These do not rely on an environment model. Instead, they directly estimate the value of taking specific actions in different states and select actions with the highest estimated rewards.

Model-free RL methods, like Deep Deterministic Policy Gradient (DDPG), Q-learning, and Deep Q-Networks (DQN), have been used in scenarios like lane changes and lane-keeping assistance (LKA). However, model-free methods are slow to learn because they require direct interaction with the environment, which is inefficient and less safe.

On the other hand, model-based RL methods are faster and safer as they can predict the outcomes of actions without requiring real-world vehicle interactions. Monte Carlo Tree Search (MCTS), a well-known model-based method, has been successfully used in games like AlphaGo and AlphaGo Zero, where it exceeded human capabilities by combining MCTS with neural networks.

Applying MCTS to parking problems is more complex than using it in games like Go because:

- Parking requires building an environment model to estimate the vehicle's state and evaluating performance using a reward function.
- Parking is a single-agent task, unlike the two-player nature of Go.
- The search space for parking actions is much larger than that in Go, and parking systems need to operate in real time.

Given these challenges, this study explores model-based reinforcement learning, particularly MCTS, to address the complex requirements of parking motion planning efficiently and safely.

Objectives and Contributions :

The primary objective of this study is to develop an autonomous parking strategy that does not rely on human experience or prior knowledge. The goal is to achieve multi-objective optimization, ensuring safety, comfort, parking efficiency, and an optimal final parking posture. The main contributions of this work are as follows:

1. **Proposing a Reinforcement Learning-Based Parking Strategy:** A reinforcement learning method is introduced that iteratively improves the parking strategy. The process involves generating parking data, evaluating its quality, and using it to train a neural network. This iterative approach ensures continuous enhancement of the parking strategy until it converges to an optimal solution.

2. **Developing a Vehicle Model for Realistic Simulation:** A vehicle model is constructed for simulation purposes, combining a transfer function and a kinematic vehicle model to closely approximate the behavior of a real vehicle.

3. **Introducing a Data Generation Method for Agent Training:** A method is proposed for generating training data based on a

longitudinal policy and a lateral policy algorithm. The lateral policy leverages P-MCTS, a modified Monte Carlo Tree Search algorithm designed for parking scenarios.

4. **Designing a Reward Function for Multi-Objective Optimization:** A reward function is constructed to evaluate parking performance, considering safety, comfort, efficiency, and the final parking posture. This function helps guide the agent toward achieving optimal results.

Algorithm Framework :

The proposed algorithm framework consists of three main stages: **Data Generation**, **Data Evaluation**, and **Network Training**. It is primarily focused on the first stage of parallel parking, where the vehicle enters the parking slot, as this significantly impacts the overall parking success.

A. Data Generation

- The motion planning algorithm generates parking data by controlling the vehicle model in simulations under various scenarios.
- The algorithm includes components such as longitudinal vehicle speed policy, lateral policy network, Monte Carlo Tree Search (MCTS), and the vehicle model.
- Inputs: Parking space size and real-time vehicle state.
- Outputs: Steering wheel angle and vehicle speed commands.
- The expected speed is directly obtained using the speed policy, while the steering angle is calculated through simulations combining the policy network, MCTS, and the vehicle model.
- MCTS focuses computations on high-probability branches, improving performance.

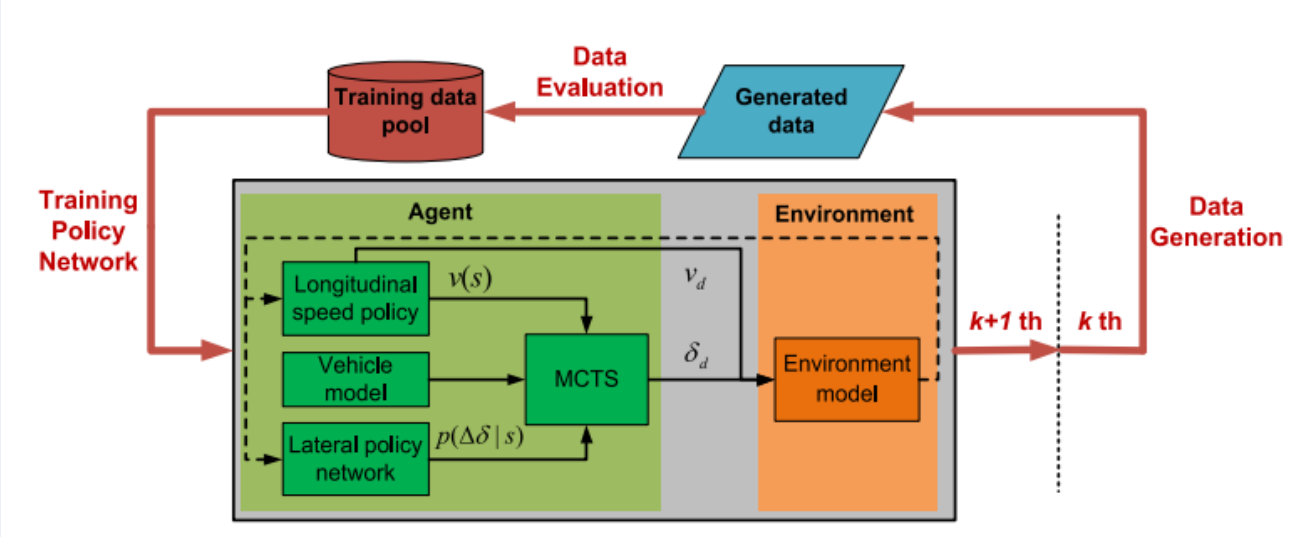
B. Data Evaluation

- A reward function evaluates parking data based on safety, comfort, efficiency, and final posture.
- The best-quality data for each scenario are selected for further training.

C. Network Training

- The network is updated with the best-quality data.
- Inputs: Vehicle status (position, attitude, real-time steering angle) and parking slot information.
- Outputs: Probability distribution of steering wheel angles.
- Initial training uses random strategies in MCTS to avoid reliance on human experience.
- The updated network guides the next iteration of data generation, improving data quality and parking strategies.

This iterative process continues until convergence, resulting in an optimal reinforcement learning agent for parking motion planning, which is deployed on the vehicle as the parking controller.

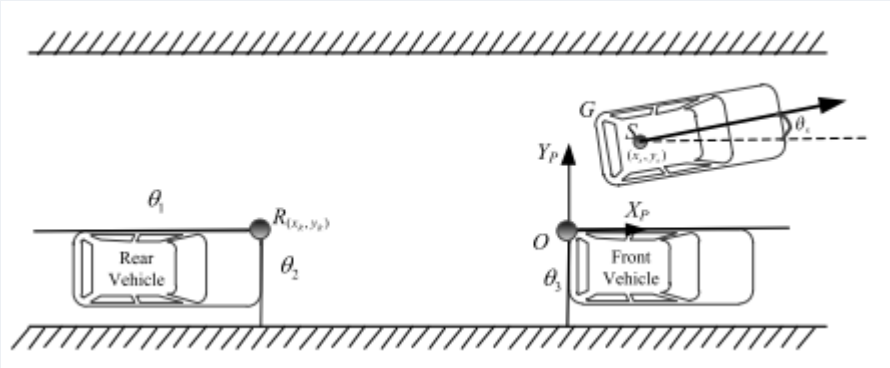


ENVIRONMENT MODEL

In model-based reinforcement learning, an environment model predicts future states and expected rewards to select optimal actions. For automatic parking, the parking planning algorithm is the agent, and the ego-vehicle along with the parking slot (front and rear obstacles) represents the environment.

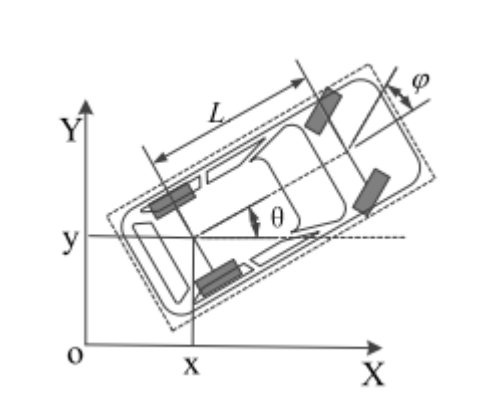
A. Parking Slot Model

The research focuses on parallel parking on the right side, defined by two parked cars (front and rear obstacles). The parking slot is represented using a coordinate system, where the origin is the left rear corner of the front vehicle. The positive X-axis points toward the vehicle's front. The ego-vehicle's position is represented as S(xs,ys,θs) with motion planning based on this coordinate system.



B. Vehicle Model

1. **Kinematic Vehicle Model**
   Due to low parking speeds, a kinematic model is used to predict vehicle posture:



$$\dot{x} = v\cos(\theta)$$

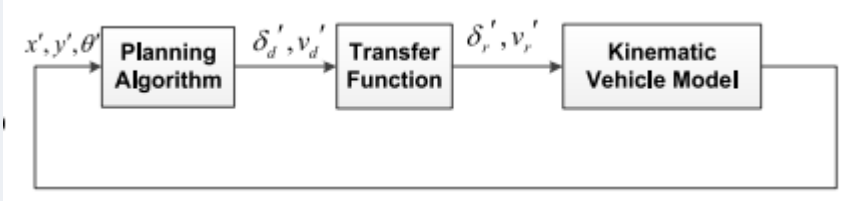
$$\dot{y} = v\sin(\theta)$$

$$\dot{\theta} = v\tan(\varphi)\ L$$

where (x, y, θ) represents the posture of the vehicle in parking coordinate, φ denotes steering angle of the front wheel, v is the

velocity at the center of the rear axle. Based on the vehicle kinematic model, the actual steering wheel of the front wheel and actual vehicle speed can be used to predict the vehicle's posture

2. **Chassis Control Model**
   The chassis control system addresses discrepancies between the desired and actual steering wheel angles. Transfer functions model the relationship between input commands and observed vehicle responses, ensuring accurate posture predictions.



   The lateral and longitudinal transfer functions as follows:

   δr = Gδ(s) · δd

   vr = Gv(s) · vd

   where δd and vd are the command values of the steering wheel angle and vehicle speed, respectively, δr and vr are the actual values of the steering wheel and vehicle speed, respectively. The Gδ(s) and Gv(s) are the identified lateral and longitudinal transfer functions, respectively.

$$G_\delta(s) = \frac{8.57s^3 + 26.90s^2 + 78.34s + 248.60}{s^4 + 8.33s^3 + 36.60s^2 + 74.92s + 248.2}$$

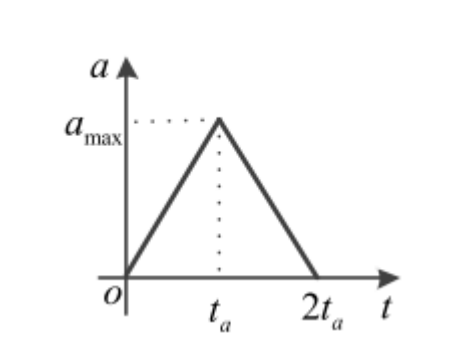
$$G_v(s) = \frac{25.75s + 47.85}{s^4 + 4.03s^3 + 27.09s^2 + 46.48s + 52.80}$$

By combining the kinematic model and these transfer functions, the constructed vehicle model is used for reinforcement learning-based parking strategies. Verification results show minimal deviations between simulated and actual parking, confirming model accuracy.

PARKING DATA GENERATION :

A. Longitudinal Motion Planning

1. **Speed Policy**:
   - Designed to ensure safety, comfort, and optimal parking time**.**
   - Divided into three phases**:**
     1. **Acceleration Phase**:
        - Simulates realistic parking with an acceleration profile that increases and then decreases.



$$v_{acc}(t) = v_0 - 0.5a_{\max}t^2 \Big/ t_a$$

$$v_{acc}(t) = v_0 - 0.5a_{\max}t_a - 0.5\left(a_{\max} + \left(a_{\max} - a_{\max}\left(t - t_a\right)/t_a\right)\right) \times \left(t - t_a\right)$$

     2. **Stable Phase**:

$$v_{stable}(t) = v_0 - a_{\max}t_a$$

     3. **Deceleration Phase**:
        - Controls the distance between the vehicle and obstacles.
        - Stops at a safe distance (Dsafe=0.20 m) before a parked vehicle using ultrasonic sensor feedback.
        - Velocity policy:

$$v_{dec}(t) = \begin{cases} kd + b & 0.25 \le d \le 1.50 \\ 0 & d < 0.25 \end{cases}$$

B. Lateral Motion Planning

1. **Approach**:
   - Combines **P-MCTS (Prioritized Monte Carlo Tree Search)** with neural networks and vehicle speed policies.
   - Focuses on **single-agent decision-making** with lateral (steering) and longitudinal (speed) policies.

2. **P-MCTS**:
   - Adapts MCTS for parking tasks by prioritizing high-probability actions during selection.
   - Tracks both **average** and **maximum simulation results** for better decision-making.

3. **Algorithm Steps**:
   - **Selection**:
     - Balances exploration (testing less promising actions) and exploitation (choosing high-reward actions).
     - Selection strategy:

$$a_t = \text{argmax}_a(Q(s_t, a) + u(s_t, a))$$

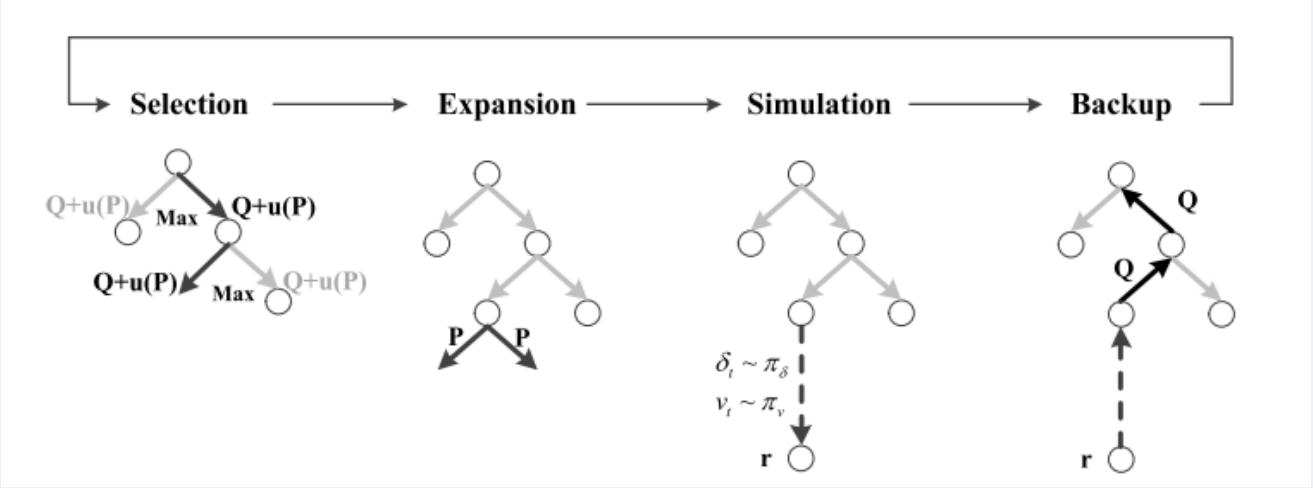
$$u(s, a) = c_{P-MCTS} P(s, a) \frac{\sum_b N(s, b)}{1 + N(s, a)}$$

   - **Expansion**:
     - Adds new nodes (sL) to the search tree with initialized statistics:
     - {N(sL, a) = 0, W(sL, a) = 0, Q(sL, a) = 0, P(sL, a) = pa}
   - **Simulation**:
     - Samples actions using neural network probabilities.

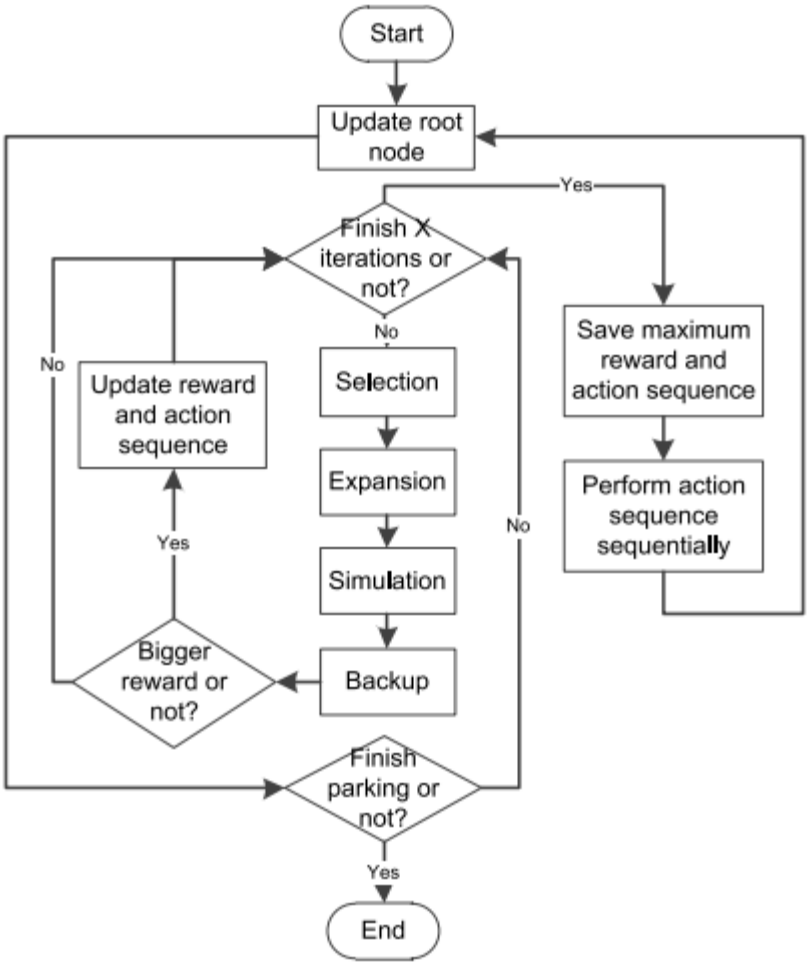
        - Simulates parking motion until stop conditions are met (e.g., zero velocity).
    - **Final Action Selection**:
        - Chooses the best action based on aggregated statistics.
4. **Advantages**:
    - Efficiently narrows search space by focusing computational resources on likely actions.
    - Handles uncertainty and improves decision quality for parking in complex environments.

.

One iteration of P-MCTS :



P-MCTS running pipeline :



Reward Function Construction :

The total reward r is expressed as:

$$r = r_{safe} + r_{pos} + r_{com} + r_{eff}$$

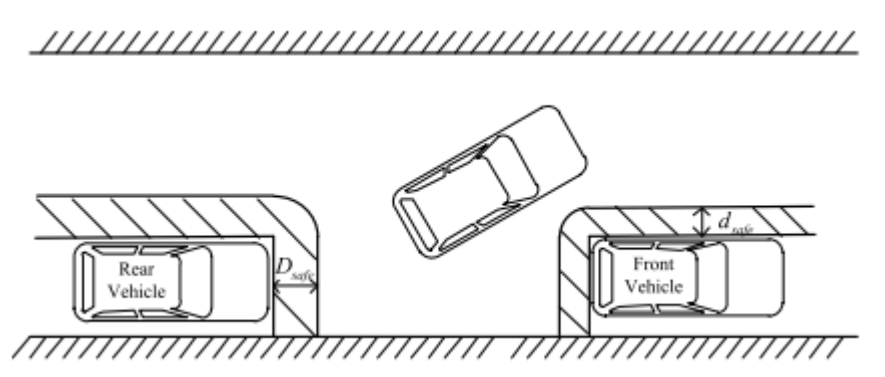
Where each term represents:

- rsafe: Safety reward.
- rpos: Final parking posture reward.
- rcom: Comfort reward.
- reff: Parking efficiency reward.

A. Safety(r_{safe}) :

Safety is ensured by avoiding collisions in hazardous areas defined within a distance d_{safe} =0.15m of surrounding vehicles. A large penalty is imposed for collisions:

$$r_{safe} = \begin{cases} -20000 & if\ collision \\ 0 & else \end{cases}$$



Parking collision risk area

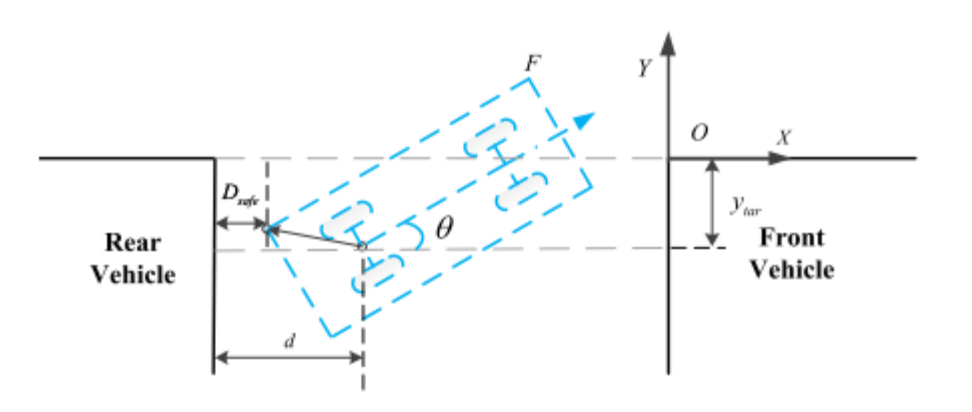
Hazardous Areas:

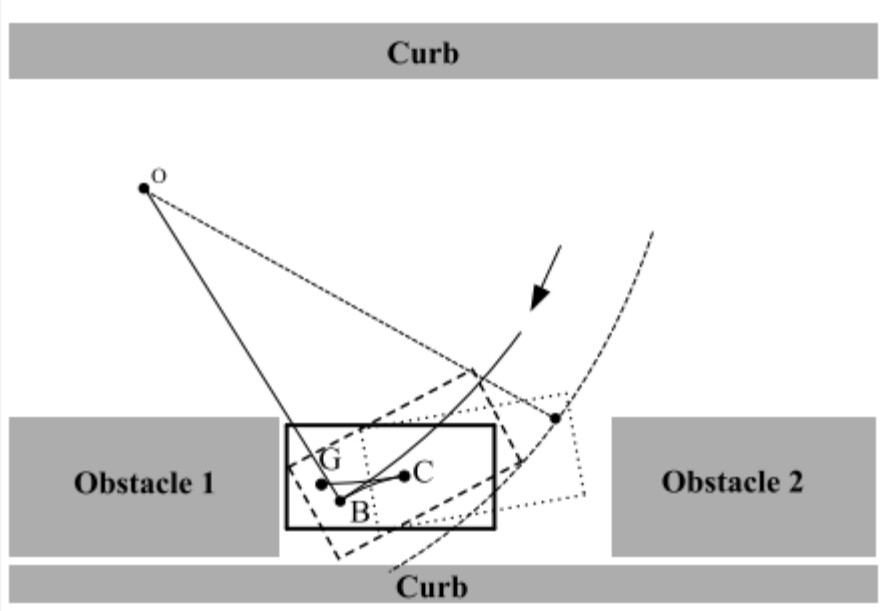
Front vehicle: Left and rear sides.

Rear vehicle: Left and front sides.

B. Final Parking Posture (r_{pos}) :

This term rewards alignment with the target parking position and orientation, incorporating both ordinate (y) and heading angle (θ).





Posture alignment process in parallel parking spaces

$$r_{pos} = \left(h_y + k_y \left|y - y_{tar}\right|\right) + \left(h_\theta + k_\theta \left|\theta - \theta_{tar}\right|\right)$$

where hy and hθ are the upper limit reward of y and θ, ky and kθ are reward weights in y and θ, ytar and θtar are target values of y and θ.

C. Comfort (r_{com}) :

Comfort is assessed based on the smoothness of steering wheel changes to minimize sudden adjustments:

$$r_{com} = k_\delta \sum_{1}^{n-1} |\delta_{k+1} - \delta_k|$$

where δk and δk+1 are the steering wheel angles for time steps k and k + 1, respectively, kδ is the weight of comfort.

D. Parking Efficiency (r_{eff}) :

Efficiency is indirectly accounted for in r_{pos} and r_{com}:

- Better alignment (smaller heading angle adjustment) improves efficiency.
- Smooth steering minimizes unnecessary movement, reducing parking time.

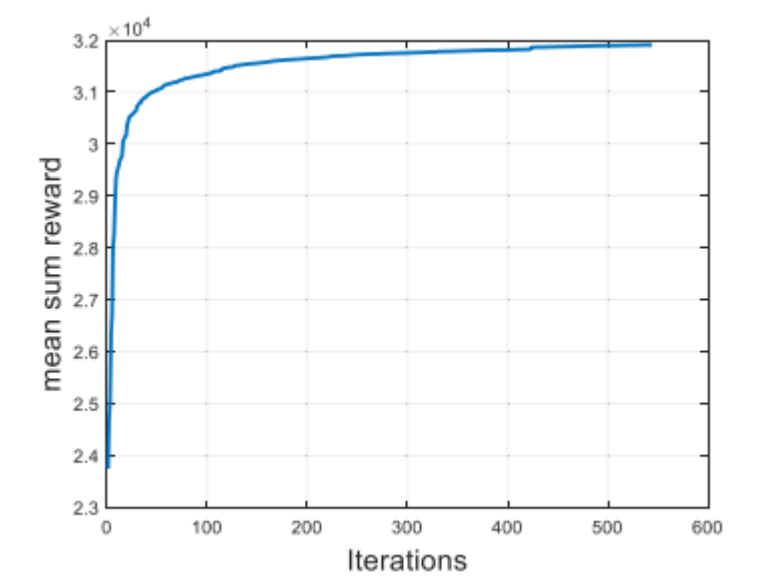
The final reward function combines all components:

$$r = \left(h_y + k_y |y - y_{tar}|\right) + \left(h_\theta + k_\theta |\theta - \theta_{tar}|\right) + k_\delta \sum_{1}^{n-1} |\delta_{k+1} - \delta_k| + r_{safe}$$

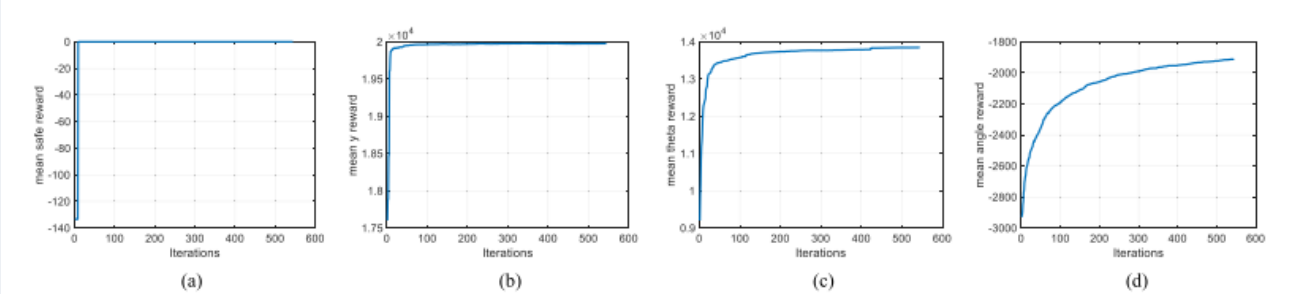
Key Takeaways :

- **Safety** is prioritized through collision avoidance with significant penalties.
- **Final posture** ensures precise parking alignment with defined target parameters.
- **Comfort** evaluates smoothness for an enhanced user experience.
- **Efficiency** is indirectly optimized through posture and comfort rewards.

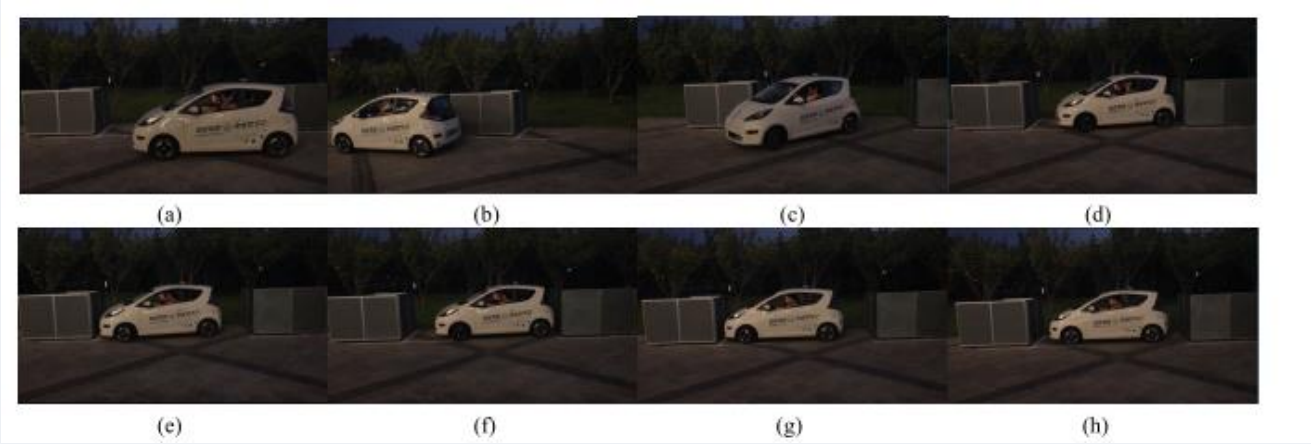
VARIATION AND ANALYSIS OF PARKING PERFORMANCE DURING REINFORCEMENT LEARNING



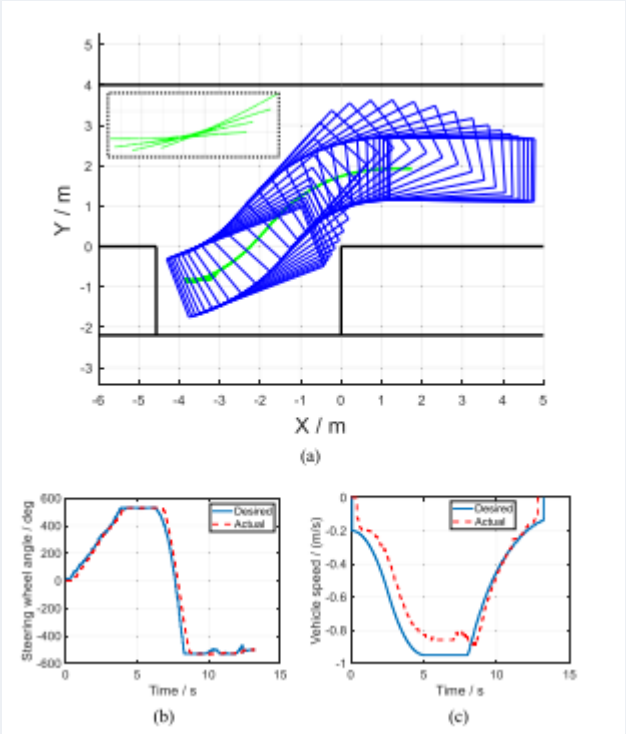
Mean sum reward of training data in the reinforcement learning process.



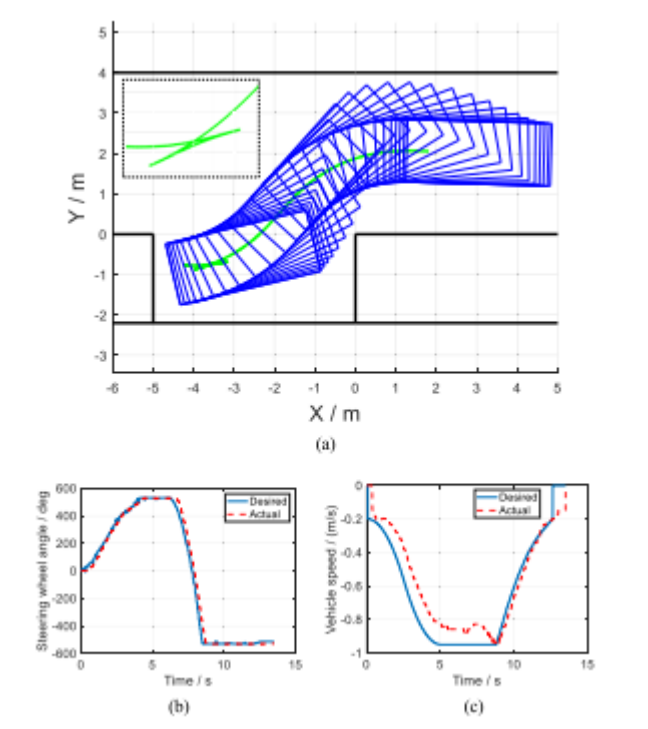
Mean reward of each item in the reward function of training data in the reinforcement learning process. (a) Safety. (b) Y in the posture. (c) θ in the posture. (d) Comfort.



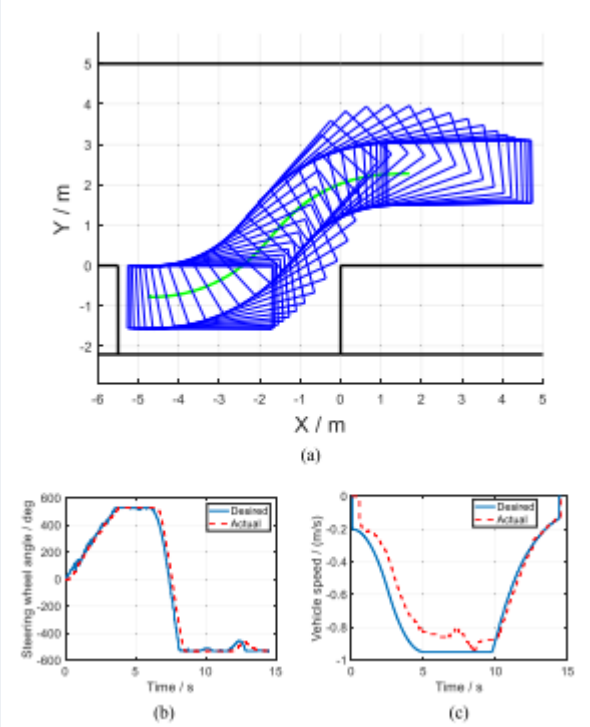
Real vehicle parking process. (a) Slot detection. (b) The parking slot is detected, and the vehicle starts to reverse. (c) The entering parking slot stage. (d) Finish the entering parking slot stage. (e)-(h) Adjusting the heading angle of the vehicle in the parking slot.



Experimental results of 4.57 m parking slot. (a) Parking path. (b) Desired and actual steering wheel angle. (c) Desired and actual vehicle speed.



Experimental results of 5 m parking slot. (a) Parking path. (b) Desired and actual steering wheel angle. (c) Desired and actual vehicle speed.



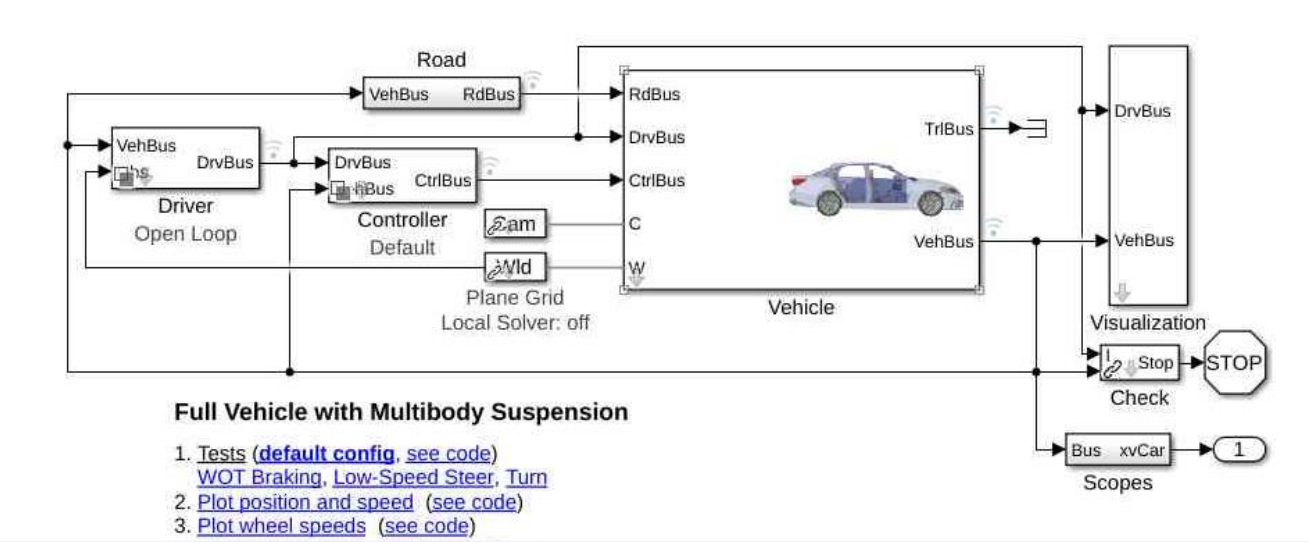
Experimental results of 5.5 m parking slot. (a) Parking path. (b) Desired and actual steering wheel angle. (c) Desired and actual vehicle speed

This result was taken from the resource :

Reinforcement_Learning-Based_Motion_Planning_for_Automatic_Parking_System

https://drive.google.com/file/d/1djdxD4HOHjazeaMQpm6bK7YIricA6Uoo/view?usp=sharing

Simscape Vehicle Templates : (System Model)



1. Open-Loop Driver Control :

- **Purpose:** The Driver subsystem provides open-loop control signals (throttle, brake, and steering) to simulate driver inputs.
- **Control Signals:**
  - **Throttle Input:** Controls the vehicle's acceleration.
  - **Brake Input:** Simulates braking force to decelerate or stop the vehicle.
  - **Steering Input:** Defines the steering angle for turning.
- **Usage:** Open-loop control is used for testing specific scenarios like wide-open throttle (WOT) braking, low-speed steering, or controlled turns.

2. Controller Subsystem :

- **Purpose:** Implements a control logic layer to process driver inputs and provide signals to the vehicle subsystems.
- **Components:**
  - **Longitudinal Control:** Manages speed by balancing throttle and brake commands.
  - **Lateral Control:** Adjusts steering angle for lane keeping or turning maneuvers.
  - **Control Signals:**

        - CtrlBus: Communicates control commands to the vehicle subsystems.
- **Strategy:** Default control strategy often includes Proportional-Integral-Derivative (PID) or feedforward-feedback control to maintain desired performance.

3. Vehicle Dynamics (Multibody Simulation) :

- **Purpose:** Models the dynamics of the full vehicle, including suspension, body, wheels, and powertrain.
- **Inputs:** Control signals from the controller and external forces (like road conditions or aerodynamic drag).
- **Outputs:**
    - VehBus: Provides vehicle state information, such as position, speed, and acceleration.
    - DrvBus: Updates driver-related data (e.g., feedback on vehicle behavior).

4. Road and Environment Interactions :

- **Purpose:** Models the interaction between the tires and the road surface.
- **Components:**
    - Tire forces (longitudinal and lateral) based on slip conditions.
    - Road slope or roughness affecting the vehicle's performance.
- **Inputs:** Road data (RdBus) to simulate real-world conditions like friction or terrain.

5. Visualization and Monitoring :

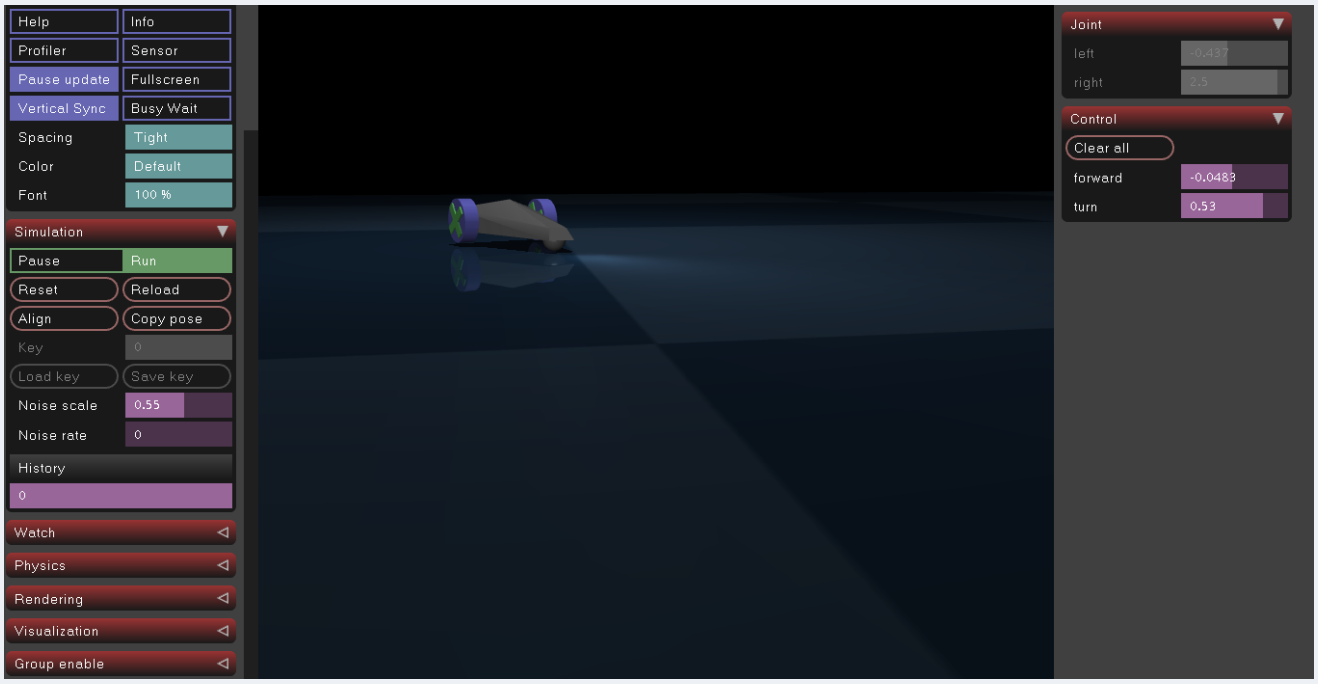
- **Purpose:** Includes tools for visualizing and analyzing vehicle motion and performance.
- **Features:**
    - Plotting position, speed, and wheel speed for analyzing test results.

- Scope blocks to capture and display signals for debugging or validation.

6. Test Scenarios :

- **Predefined Tests:**
  - **WOT Braking:** Tests vehicle response to maximum acceleration and subsequent braking.
  - **Low-Speed Steering:** Evaluates steering performance at low speeds.
  - **Turns:** Simulates turning maneuvers to test lateral stability.
- **Customization:** You can modify test inputs or parameters to simulate additional scenarios.

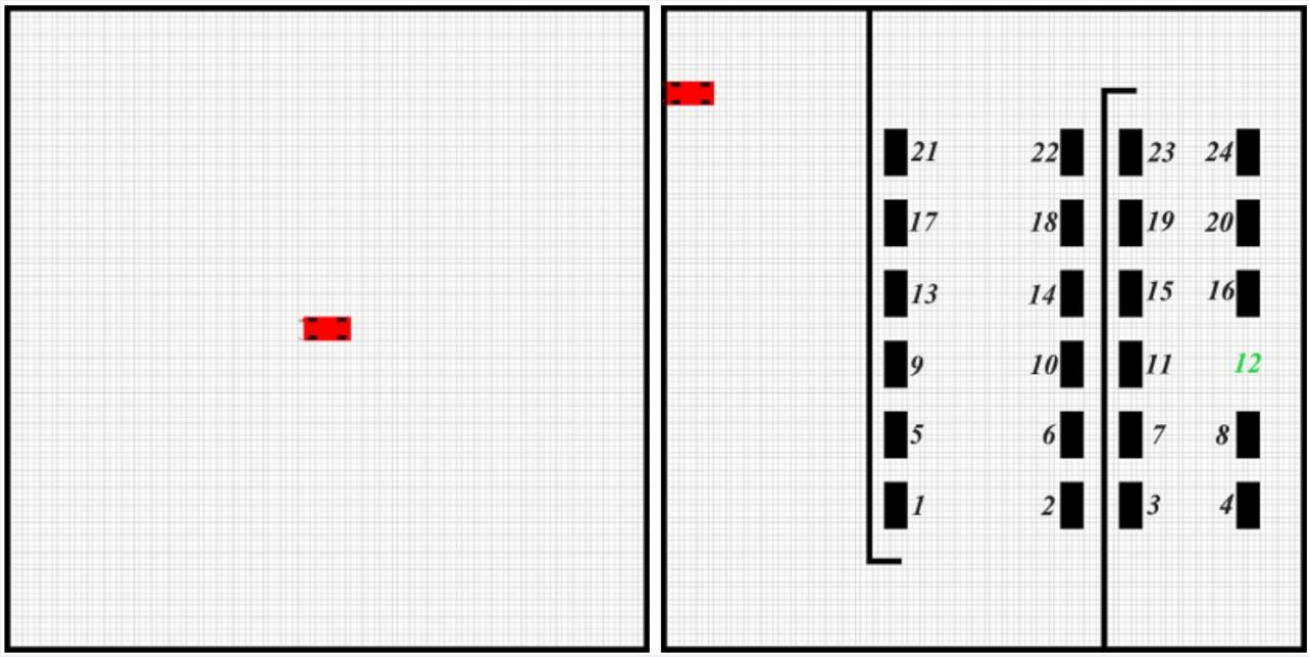
I am using the system model from the Simscape Vehicle Model, but it is very complex to understand and produces many errors. As a result, I decided to use a new model from MuJoCo, but I have not been able to fully integrate it with my code.



Simulation Results :

Environment :

The first step is to develop the auto parallel parking environment. For this, I am using the OpenCV library.



Path Planning :

A* Algorithm **:**
The agent uses the A* algorithm to find a path from the start to the goal, considering obstacles and the robot's size.

Path Smoothing with B-spline :
After finding a rough path in a 100x100 grid, the path is smoothed and scaled to fit a 1000x1000 environment using a B-spline. This gives a smooth set of points to guide the agent.

Path Tracking :

Kinematic Model of the Car
The equations describing the car's motion are:

- $\dot{x} = v \cdot \cos(\psi)$
- $\dot{y} = v \cdot \sin(\psi)$
- $\dot{v} = a$
- $\dot{\psi} = (v \cdot \tan(\delta)) / L$

Where:

- **a**: Acceleration
- **δ**: Steering angle
- **ψ**: Yaw angle
- **L**: Wheelbase
- **x**: X-position
- **y**: Y-position
- **v**: Velocity

State Vector

The state of the car is represented as:
z = [x, y, v, ψ]

- **x**: X-position
- **y**: Y-position
- **v**: Velocity
- **ψ**: Yaw angle

Input Vector

The inputs to the car are:
u = [a, δ]

- **a**: Acceleration
- **δ**: Steering angle

Control

The **MPC controller** (Model Predictive Control) adjusts the vehicle's speed and steering to follow the desired path.

Parallel Parking :

Parking Strategy

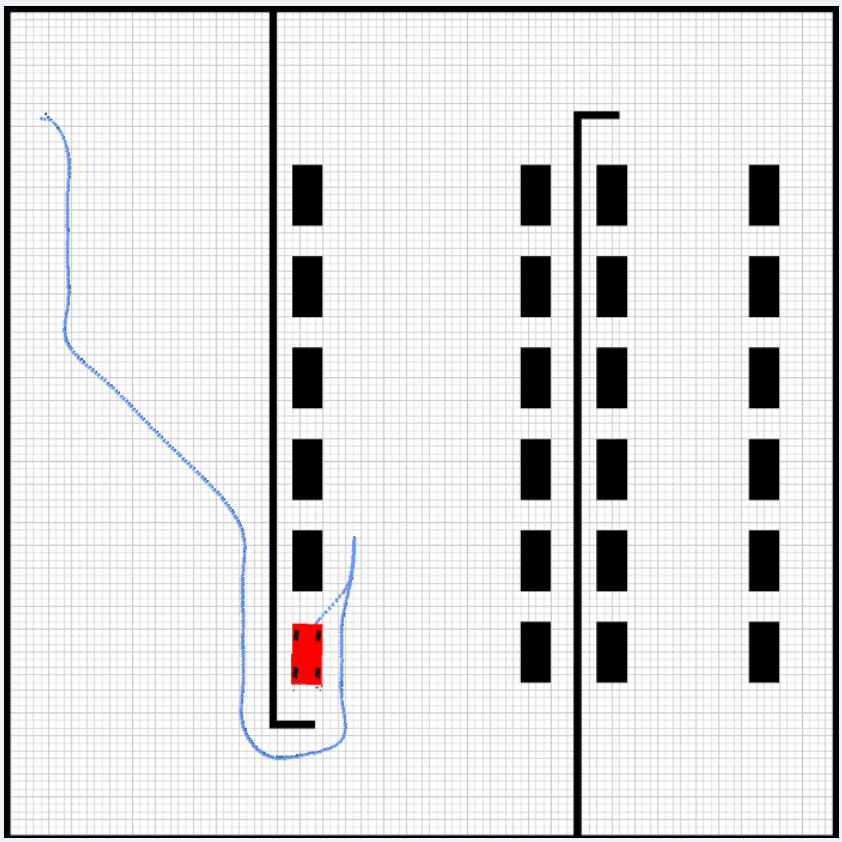
The process has 4 rules based on the parking position:

1. The agent finds a path to the parking position and calculates the arriving angle.
2. Based on this angle, the agent selects a coordinate called **ensure1**.
3. The parking path is planned from **ensure1** to **ensure2** using two circle equations.
4. The **MPC controller** guides the car to park at the **ensure2** coordinate.

Results :

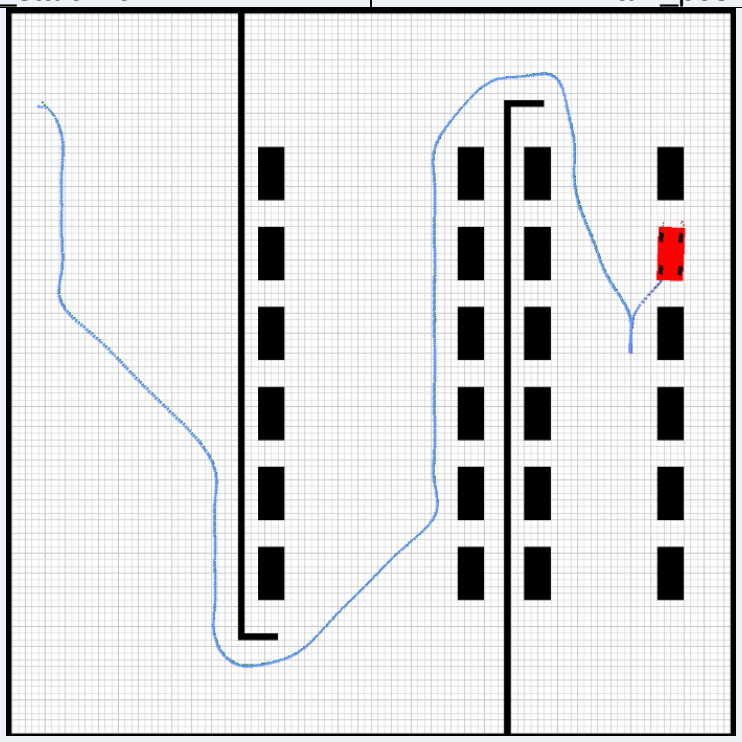
1.

X_start = 0	X_end = 90
Y_start = 90	Y_end = 80
Psi_start = 0	Park_pos = 1



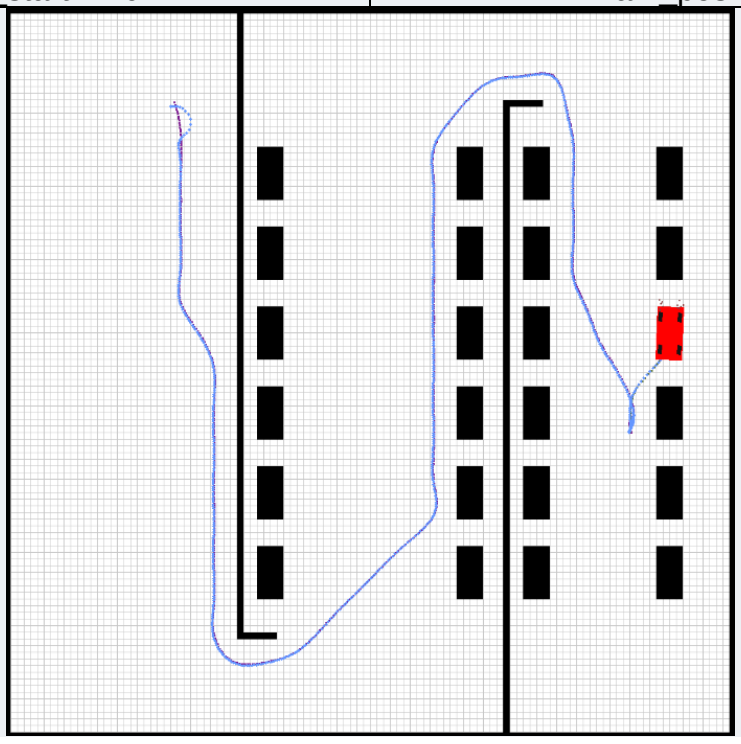
2.

X_start = 0	X_end = 90
Y_start = 90	Y_end = 80
Psi_start = 0	Park_pos = 20



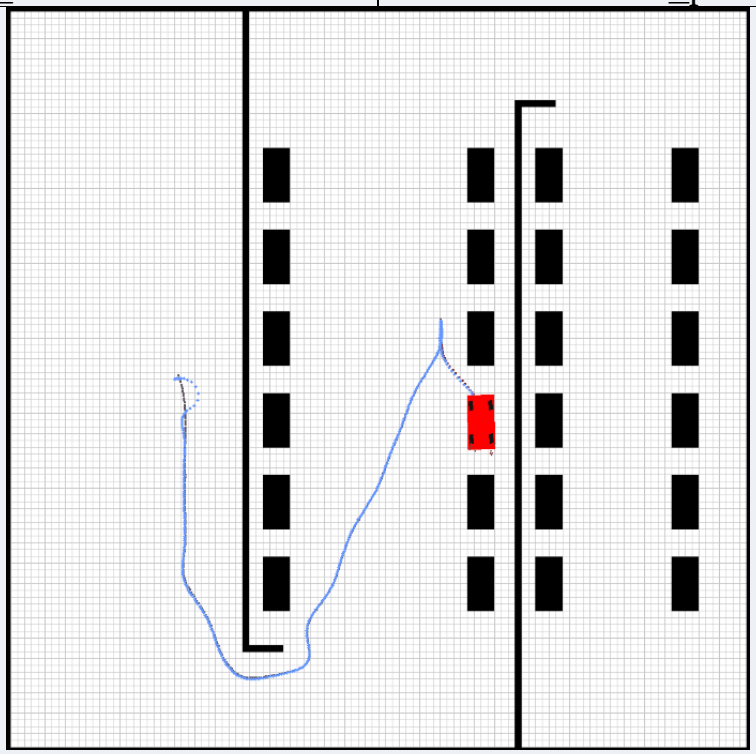
3.

X_start = 20	X_end = 90
Y_start = 90	Y_end = 80
Psi_start = 10	Park_pos = 16



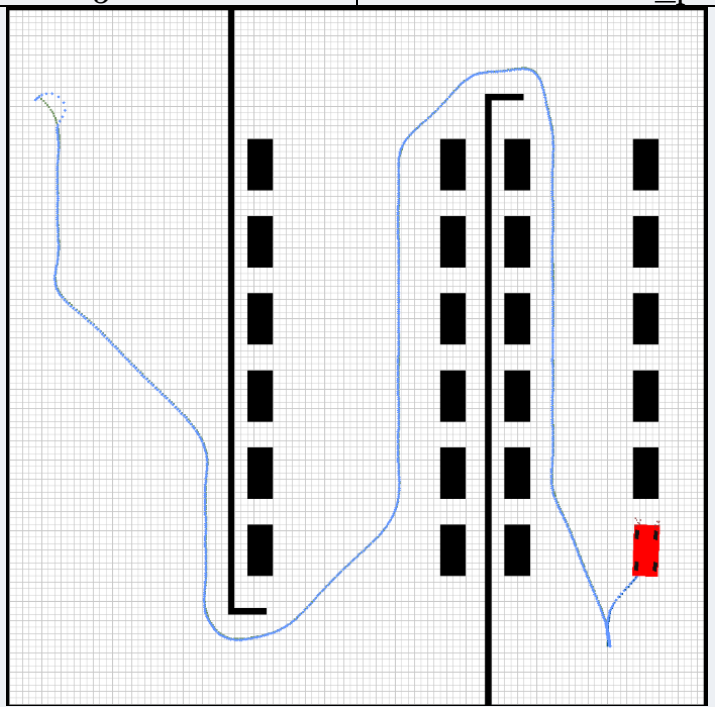
4.

X_start = 0	X_end = 90
Y_start = 50	Y_end = 80
Psi_start = 20	Park_pos = 10



5.

X_start = 0	X_end = 90
Y_start = 90	Y_end = 0
Psi_start = 50	Park_pos = 4



Conclusion :

This research would introduces a new model-based reinforcement learning approach for automatic parking motion planning. The method minimizes reliance on human input, allowing the system to learn parking strategies quickly and autonomously. These strategies achieve multiple goals, including safety, comfort, parking efficiency, and final parking accuracy.

The proposed framework integrates data generation, evaluation, and training in an iterative cycle. A vehicle model combining transfer functions and kinematic principles simulates the interaction between the vehicle and its environment, enabling rapid learning. A data generation algorithm produces extensive parking data, while a reward function evaluates the data based on performance criteria. A neural network is then trained using the best-selected data.

Future research could incorporate additional environmental factors, like road conditions and perception systems, to improve model accuracy. Expanding the reinforcement learning algorithms to include longitudinal control could further reduce reliance on human input and enhance parking system performance.

Github Link :-

https://github.com/202201239/Reinforcement-Learning-Based-Motion-Planning-For-Automatic-Parking-System.git

THANK YOU!