C2SMARTER
CONNECTED COMMUNITIES WITH SMART MOBILITY TO EQUITABLY REDUCE CONGESTION

A U.S. DOT University Transportation Center

New York University
NYC College of Technology
North Carolina A&T University
Rutgers University
Texas Southern University
University of Texas at El Paso
University of Washington

Control of Connected and Autonomous Vehicles for Congestion Reduction in Mixed Traffic: A Learning-Based Approach

September 2024

Control of Connected and Autonomous Vehicles for Congestion Reduction in Mixed Traffic: A Learning-Based Approach

Zhong-Ping Jiang
New York University
0000-0002-4868-9359

Kaan Ozbay
New York University
0000-0001-7909-6532

Xianning Li
New York University
0009-0005-0959-0996

Disclaimer

Acknowledgements

This project is supported in part by the USDOT Tier-1 C2SMART Center grant, in part by the US NSF, and in part by the Mitsubishi Electric Research Laboratories (MERL).

Executive Summary

This project investigates control strategies for automated lane-changing and combined longitudinal-lateral vehicle following in mixed traffic scenarios, with the goal of enhancing traffic safety and efficiency for Connected and Autonomous Vehicles (CAVs). The study addresses the challenges posed by real-world uncertainties, complex vehicle interactions, and the presence of both human-driven and autonomous vehicles.

In the lane-changing scenario, we designed and validated a data-driven control algorithm to achieve safe and efficient lane-change maneuvers. The proposed approach utilizes both model-based policy iteration and data-driven adaptive dynamic programming to accommodate different levels of state observability (full-state feedback and output feedback). The lane-changing control strategy comprises three main components: dynamic modeling, real-time decision-making, and adaptive learning modules. The decision-making layer ensures safe execution by incorporating safety constraints based on the relative positions and velocities of surrounding vehicles. The real-world experiments, conducted with a scaled car model, demonstrate the ability of the proposed controller to adapt to dynamic traffic conditions, maintaining stability and successfully performing lane changes even in the presence of measurement noise and model uncertainties. The safety of the lane change maneuver is continuously evaluated, and if the surrounding conditions become unsafe, the controller aborts the maneuver and returns to the original lane.

For the car-following problem, a data-driven combined longitudinal and lateral control framework is developed to ensure stability and optimal performance under varying road geometries and traffic conditions. The proposed method utilizes a two-phase data-driven policy iteration approach. In the first phase, a feedforward controller is designed using input-state data to mitigate steady-state errors. The second phase refines the control policy iteratively using a model-free policy iteration algorithm to achieve near-optimal feedback control. Extensive simulation experiments using SUMO and CommonRoad demonstrate that the designed controller effectively manages interactions between the leading and following vehicles, ensuring smooth trajectory tracking and minimizing tracking errors on roads with varying curvatures and during sudden speed changes. Additionally, the robustness of the control strategy was validated under disturbances, such as fluctuating leading vehicle speeds and changing road friction coefficients.

The project highlights the potential of using data-driven learning algorithms to tackle complex control challenges in autonomous driving, providing a scalable solution that balances safety, stability, and computational efficiency in real-world scenarios. The integration of learning-based control with robust

safety constraints enables CAVs to perform safe and efficient maneuvers, even in highly dynamic and uncertain mixed traffic environments.

Table of Contents

List of Figures

List of Tables

Introduction

Urban areas are increasingly plagued by traffic congestion due to the rising number of vehicles, causing commuters to spend more time idling in traffic. According to researchers at the Texas A&M Transportation Institute, the average American commuter loses 56 extra hours and 21 gallons of fuel annually due to traffic jams [1]. As commuters spend more and more time stuck in traffic, finding effective traffic management solutions to alleviate congestion has become both urgent and critically important.

One significant cause of traffic congestion in urban regions is the stop-and-go waves occurring at signalized intersections. Vehicles decelerating at red lights and accelerating during green lights disrupt the flow of traffic, causing these stop-and-go patterns to propagate backward. This leads to reduced traffic velocity and decreased throughput, resulting in excessive congestion and delays. The introduction of vehicle-to-vehicle communication technology and autonomous vehicles presents new opportunities to smooth traffic flow and mitigate the adverse effects of stop-and-go waves. By combining these technologies, connected and autonomous vehicles (CAVs) can coordinate with each other, adjusting their speeds based on predicted signal timings to pass through intersections more efficiently. This requires accurate prediction of traffic signals, real-time optimization of CAV trajectories, and precise control of autonomous vehicles to ensure safe interaction with other vehicles and pedestrians.

1.1 Traffic Light Prediction

The Speed Advisory System (SAS) installed in vehicles can leverage the Signal Phase and Timing (SPaT) message, which includes the current traffic light phase and the remaining phase time, to recommend optimal speeds. This helps in reducing fuel consumption [2], alleviating traffic congestion, and minimizing drivers' waiting time [3]. However, SPaT messages are often unavailable and need to be predicted frequently. In [4], the authors employed a Kalman Filter (KF) to predict the frequency distributions of green signals using historical data, achieving only 71% accuracy in future signal predictions. In their subsequent research [5], they computed the empirical frequency distribution using historical green light data, noting that green times tend to follow daily patterns. Another study [6] assumed uniform vehicle arrivals and used the current phase along with average red and green times from historical data to compute the conditional probability of the future phase. Although this method could be improved with more accurate vehicle arrival distributions, it struggles with out-of-distribution traffic demands. The strategy presented in [7] integrates real-time measurements with historical data to enhance prediction accuracy. However, this approach is challenging to implement in SAS due to the constant variation in predictions.

All the previously mentioned models are probabilistic in nature. Recently, there has been a growing interest in using machine learning (ML) models for traffic light prediction. In studies [8, 9], the authors compared a Random Survival Forest (RSF) model, a long short-term memory (LSTM) neural network model, and a Linear Regression (LR) model using historical data. They found that both the RSF and LSTM models outperformed the LR model on a small dataset, with the RSF model proving to be the most accurate on a larger dataset. However, the specifics of the training process and model configurations were not provided. Given the complexity of deep learning models, there is significant potential for further improvement in these results. Additionally, these studies [8, 9] only focused on a single intersection.

1.2 Trajectory Planning of CAVs for Optimum Traffic Performance

The analysis of traffic conditions often relies on macroscopic fundamental diagrams, which describe the relationships among space-mean flow, density, and speed within traffic networks [10]. With the integration of vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication technologies, connected and autonomous vehicles (CAVs) hold significant potential for enhancing traffic efficiency through a new approach to fundamental diagrams [11, 12]. A central challenge in reducing traffic congestion is determining the optimal movements of multiple CAVs at both unsignalized and signalized intersections. This involves optimizing vehicular trajectories while adhering to physical and safety constraints. The authors in [13] introduced a parsimonious shooting heuristic algorithm to solve the trajectory optimization problem for CAVs, focusing on minimizing average travel time, system throughput, and fuel consumption on highways, with extensions to signalized intersections and the inclusion of Human-Driven Vehicles (HDVs) in [14]. In [15], a smart re-routing mechanism was proposed to manage the approach of CAVs to intersections, aiming for traffic load balancing and congestion avoidance. In another study [16], a group-based trajectory planning scheme was developed to coordinate CAV movements in alignment with traffic signal timings. The research in [17] focused on designing safe trajectories for CAVs, using algorithms to eliminate trajectory overlaps at intersections. Coordination of vehicle trajectories was further explored in [18], where a mixed-integer linear program was formulated to optimize CAV trajectories along a corridor, thereby reducing total vehicle delay. Similarly, graph-based methods were introduced in [19] to optimize the longitudinal trajectories of CAVs, taking into account traffic lights and preventing vehicle collisions.

However, these studies typically rely on simplified system models, which can lead to significant tracking errors when using feedback controllers, thereby negatively affecting overall system performance. Additionally, heuristic algorithms are commonly used, but they often fail to find the optimal solution for the original problem. Therefore, there is a pressing need for further exploration in the field of traffic reduction.

1.3 Safety-oriented Motion Control of CAVs

Inaccuracies in trajectory tracking can significantly affect motion control in autonomous systems, leading to various undesirable outcomes. Deviations from the intended path can cause erratic or jittery movements, disrupting navigation. These inaccuracies impede optimal system performance, compromise control stability, and may result in energy inefficiency. Furthermore, safety hazards arise as inaccuracies can lead to collisions, endangering pedestrians or other vehicles. Numerous trajectory-tracking controllers have been utilized in the past, including pure pursuit [21, 22], Stanley [23], and model predictive control (MPC) [24, 25]. The pure pursuit controller, while effective in many scenarios, has notable drawbacks. It struggles with well-defined paths or trajectories that include sharp turns. The Stanley controller, although advantageous, is sensitive to changes in vehicle dynamics and road conditions, affecting performance in various environments. Its reliance on lateral position measurements makes it susceptible to sensor inaccuracies and noise, potentially causing erratic behavior. Additionally, complex or non-smooth paths challenge accurate trajectory tracking in intricate road geometries or challenging terrains. One significant drawback of MPC is its computational complexity, as it involves solving optimization problems at each control step. This can lead to an increased computational load and slower control loop execution, limiting its real-time applicability, especially in fast-paced scenarios. Moreover, modeling inaccuracies and uncertainties can impact performance, as MPC relies on precise models for prediction.

In addition to errors in trajectory tracking, vehicle collisions can significantly degrade system performance. In congestion control scenarios, individual vehicles or groups of vehicles may need to decide whether to maintain their lane or change lanes. Ensuring safety among neighboring vehicles is crucial when implementing controllers to follow computed trajectories. Thus, in congestion control, there is an interplay between achieving control objectives (such as trajectory tracking, stability, and disturbance rejection) and ensuring safety. Achieving both simultaneously is challenging. A formal approach to explore this coupling involves using control Lyapunov functions (CLFs) and control barrier functions (CBFs). CLFs express stability, while CBFs represent safety conditions. Barrier functions ensure the forward invariance of safety sets, eliminating the need to compute complex reachable sets. Combining CLFs and CBFs through quadratic programming, known as CLF-CBF-QP, has been applied to various areas including driver intention prediction [26, 27], adaptive cruise control, lane keeping [28, 29], obstacle avoidance [30], and multi-agent interaction [31, 32]. However, these approaches typically assume an accurate system model for CLF-CBF-QP, which can be problematic in safety-critical contexts due to potential inaccuracies. Recent efforts have been directed towards exploring data-driven methods for learning control barrier functions to address these limitations.

Methods like those in [33, 34, 35] synthesize control barrier functions (CBFs) from expert demonstration data in locally safe regions. However, the localized nature of these learned CBFs may not guarantee real-time safety across broader scenarios. In [36], the authors demonstrate a method for learning quadratic

CBFs for nonlinear polynomial systems, but this approach is limited to systems that can be represented quadratically. Another method, proposed in [37], involves learning high relative degree CBFs using neural networks, which requires accurate computation of high-order derivatives—a challenging task. An alternative approach described in [38] utilizes discrete-time Koopman operators to synthesize CBFs, assuming linear dynamics and requiring substantial data for effective learning.

1.4 Data-Driven Learning Control of CAVs

Recent advancements in sensor technologies and computational power have paved the way for applying purely data-driven machine learning (ML) techniques to Connected and Autonomous Vehicles (CAVs). Such methods leverage vast quantities of real-world driving data to train models capable of making lane change decisions and determining optimal policies. Hoel et al. proposed a decision-making framework that integrated planning principles with deep reinforcement learning (DRL) [39]. In a similar vein, Li et al. presented a transformer network-based DRL architecture to generate safe lane change policies [40], while Xie et al. utilized a deep belief network to perform lane-changing maneuvers based on camera views and trajectory data [41]. The fundamental advantage of these approaches lies in their capacity to handle complex interactions and adapt to dynamic driving scenarios without needing a detailed vehicle dynamics model. However, the data-driven nature of these ML methods also introduces significant drawbacks, such as the need for extensive and diverse datasets to capture all potential driving conditions [42]. Training such models to convergence often requires substantial time, making their real-time applicability for safety-critical scenarios challenging. This reliance on empirical data, combined with the difficulty of ensuring convergence and generalizability, limits their practicality for autonomous driving applications in CAVs.

To address some of the limitations associated with purely data-driven methods, non-model based learning techniques, particularly Adaptive Dynamic Programming (ADP), have gained attention. ADP methods, rooted in reinforcement learning principles, offer a robust solution by directly learning the optimal control policies using real-time data. Unlike traditional ML approaches, ADP can handle unknown dynamics and parameter variations in complex systems, making it particularly suitable for autonomous driving scenarios where system dynamics might be uncertain or change over time [43-47]. For instance, Jiang et al. introduced a framework for ADP-based control, demonstrating its ability to stabilize and track complex dynamical systems using fewer data samples compared to purely data-driven techniques [48]. In similar works, Gao and Jiang [49] and Vamvoudakis et al. [50] showed that ADP-based approaches can efficiently learn optimal policies and guarantee the closed-loop stability of the system under uncertain conditions. In the context of CAVs, ADP enables learning-based controllers to adapt and optimize lane change decisions without a complete system model, thus reducing reliance on predefined dynamics and overcoming the drawbacks of model-based methods. Furthermore, these techniques can guarantee the stability and robustness of learned policies, making them a promising alternative for real-world autonomous vehicle control applications in mixed traffic scenarios [51-57].s

1.5 Contributions

In this project, automated lane changing control in mixed traffic with safety consideration is designed to address the critical issue of ensuring safe lane change maneuvers for autonomous vehicles (AVs) operating alongside human-driven vehicles. A decision-making module based on safety constraints is employed to avoid potential collisions with surrounding vehicles. With the assumption of the linear system, the proposed framework integrates state/output feedback and model-free adaptive dynamic programming to handle the varying traffic conditions and uncertainties associated with mixed traffic scenarios. Experiments using scaled AV prototypes validate the approach, demonstrating that the AV can execute safe lane changes. By incorporating safety into both decision-making and control, this strategy ensures that the AV can effectively navigate complex driving environments, contributing to improved traffic flow and passenger safety.

In this project, we also develop a data-driven combined longitudinal and lateral control for the car following problem by developing a robust controller to manage the interactions between a leading vehicle and a following autonomous vehicle where vehicle dynamics are assumed to be nonlinear. This control strategy tackles the challenges of maintaining safe inter-vehicle distances and precise trajectory tracking under varying road conditions. The framework introduces a two-phase policy iteration algorithm that first designs a feedforward controller using input-state data and then iteratively learns an optimal feedback controller using real-time measurements. This data-driven approach eliminates the need for precise system models, making it highly adaptable to different vehicle dynamics and road geometries. Through extensive simulations on circular and mixed road segments, the proposed controller successfully minimizes tracking errors and ensures smooth vehicle following, even when the lead vehicle undergoes abrupt speed changes or the road surface properties vary. The results demonstrate the effectiveness of the combined control strategy in achieving stability and safety in vehicle platooning scenarios.

The main contributions of this work are summarized as follows:

- Automated lane changing control in mixed traffic with safety consideration:
    - Proposed a data-driven adaptive dynamic programming method for the vehicle system with full-state feedback scenarios.
    - Developed a data-driven adaptive dynamic programming method to handle unknown states with output feedback scenarios.
    - Conducted real-vehicle experiments with scaled prototypes, demonstrating that the control approach ensures safe and efficient lane changes in dynamic traffic conditions.

- Data-driven combined longitudinal and lateral control for the car following problem:
  - Designed a nonlinear controller to address the coupling effects between longitudinal and lateral dynamics.
  - Developed a data-driven solution using ADP techniques to iteratively learn a near-optimal controller from motion data, eliminating the need for precise vehicle dynamics modeling.
  - Validated the proposed control method through extensive simulations under various road and traffic conditions.

Notations:

- $\otimes$ indicates the Kronecker product.
- $\text{vec}(\mathbf{T}) = [t_1^{\mathrm{T}}, t_2^{\mathrm{T}}, \cdots, t_m^{\mathrm{T}}]^{\mathrm{T}}$ with $t_i \in \mathbb{R}^r$ being the columns of $\mathbf{T} \in \mathbb{R}^{r \times m}$.
- For a symmetric matrix $\mathbf{P} \in \mathbb{R}^{m \times m}$, $\text{vecs}(\mathbf{P}) = [p_{11}, 2p_{12}, \cdots, 2p_{1m}, p_{22}, 2p_{23}, \cdots, 2p_{(m-1)m}, p_{mm}]^{\mathrm{T}} \in \mathbb{R}^{(1/2)m(m+1)}$
- For a column vector $\mathbf{v} \in \mathbb{R}^n$, $\text{vecv}(\mathbf{v}) = [v_1^2, v_1 v_2, \cdots, v_1 v_n, v_2^2, v_2 v_3, \cdots, v_{n-1} v_n, v_n^2]^{\mathrm{T}} \in \mathbb{R}^{(1/2)n(n+1)}$.
- For any two sequences of vectors $\mathbf{a} = \{a_{k_i}\}_{i=0}^{s}$ and $\mathbf{b} = \{b_{k_i}\}_{i=0}^{s}$, define $\boldsymbol{\Delta}_{\mathbf{a}} = [\text{vecv}(a_{k_0+1}) - \text{vecv}(a_{k_0}), \cdots, \text{vecv}(a_{k_s}) - \text{vecv}(a_{k_s-1})]^{\mathrm{T}}$, $\mathbf{I}_{\mathbf{a},\mathbf{b}} = [a_{k_0} \otimes b_{k_0}, \cdots, a_{k_s} \otimes b_{k_s}]^{\mathrm{T}}$, $\mathbf{I}_{\mathbf{a}} = [\text{vecv}(a_{k_0}), \cdots, \text{vecv}(a_{k_s})]^{\mathrm{T}}$.
- $\mathbf{I}_n (0_n)$ is the identity (zero) matrix of dimension $n \times n$.

Automated Lane Changing Control in Mixed Traffic with Safety Consideration

This part mainly presents the design of the autonomous lane-changing control scheme and real-vehicle experiments in mixed traffic scenarios, considering both full-state feedback and output feedback conditions.

2.1 Dynamic Model and Problem Formulation

2.1.1 Dynamic Model

In this section, we assume that the vehicle speed remains constant during the lane change and the vehicles travel on a straight road. The lateral dynamic model is based on the position and orientation error variables as shown in Fig. 1. Let (T_x, T_y) be the coordinates of the target point, ψ be the orientation of the vehicle, e_1 be the error between the distance of the center of gravity of the vehicle and the center line of the target lane, and e_2 be the orientation error of the vehicle with respect to the road.

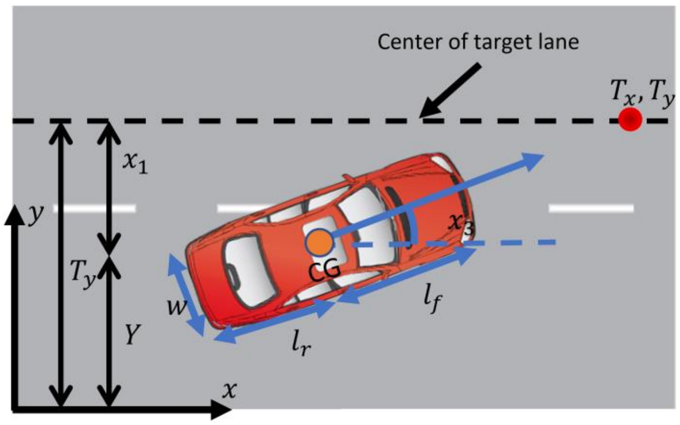

Fig. 1: Position and orientation error of AV

After that, the lateral dynamics model is defined as follows:

$$\mathbf{x}_{la,k+1} = \mathbf{A}_{la}\mathbf{x}_{la,k} + \mathbf{B}_{la}u_{la,k}, \tag{1}$$

where u_{la} denotes the front wheel steering angle, $\mathbf{x}_{la}=[x_1,x_2,x_3,x_4]^T$ with x_2 and x_4 representing the rate of change of x_1 and x_3, respectively, and k representing the current discrete time index,

$$
\mathbf{A}_{la}=\begin{bmatrix} 1 & t_s & 0 & 0 \\ 0 & 1-\dfrac{2C_{\alpha f}+2C_{\alpha r}}{mV_x}\cdot t_s & \dfrac{2C_{\alpha f}+2C_{\alpha r}}{m}\cdot t_s & \dfrac{-2C_{\alpha f}l_f+2C_{\alpha r}l_r}{mV_x}\cdot t_s \\ 0 & 0 & 1 & t_s \\ 0 & -\dfrac{2C_{\alpha f}l_f-2C_{\alpha r}l_r}{I_zV_x}\cdot t_s & \dfrac{2C_{\alpha f}l_f-2C_{\alpha r}l_r}{I_z}\cdot t_s & 1-\dfrac{2C_{\alpha f}{l_f}^2+2C_{\alpha r}{l_r}^2}{I_zV_x}\cdot t_s \end{bmatrix},
\quad
\mathbf{B}_{la}=\begin{bmatrix} 0 \\ \dfrac{2C_{\alpha f}}{m}\cdot t_s \\ 0 \\ \dfrac{2C_{\alpha f}l_f}{I_z}\cdot t_s \end{bmatrix}, \tag{2}
$$

with t_s representing the time step of the discrete-time system.

Since the vehicle travels on a straight road, the desired orientation of the vehicle is considered as $\psi_{des}=0$. Then from Fig. 1, the lateral position Y_k and yaw angle ψ_k can be obtained as:

$$
\begin{aligned} Y_k &= T_y - x_{1,k}, \\ \psi_k &= x_{3,k}. \end{aligned} \tag{3}
$$

2.1.2 Lane Change Decision Making with Safety Consideration

To ensure safety during the lane-changing process, the decision-making layer considers the safety of the system in mixed traffic scenarios. Lane-change maneuvers are executed only when the surrounding environment is deemed suitable; otherwise, the vehicle remains in its current lane.

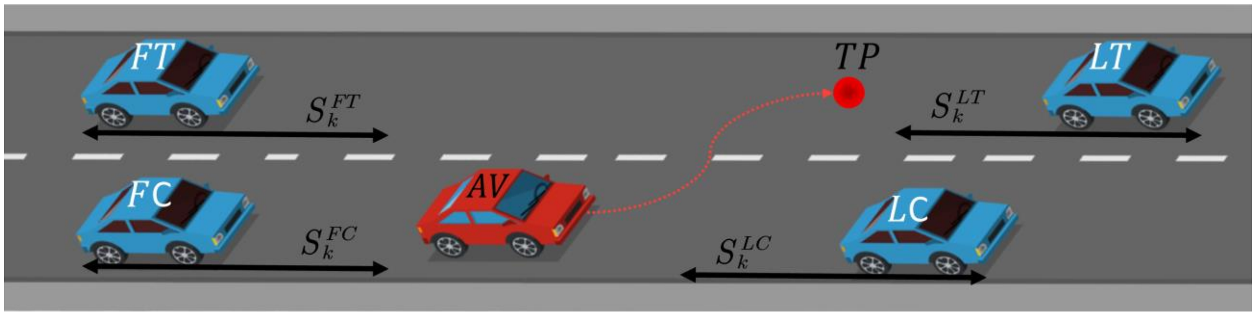

Fig. 2: Lane change Scenario

In Fig. 2, without loss of generality, AV denotes the autonomous vehicle, LC denotes the lead vehicle in the current lane, FC denotes the following vehicle in the current lane, LT denotes the lead vehicle in the target lane, FT denotes the following vehicle in the target lane, $S_k^i = L + hv_k^i, \quad i \in \{LT, FT, LC, FC\}$, h = headway time, L = length of vehicle, and v^i = velocity of the i^{th} vehicle, TP is the target point. In this work, the lane change decision making is introduced for a single lane change maneuver. As shown in Fig. 2, four vehicles are involved in a lane change maneuver. The AV performs a maneuver to change the lane and places itself in the target point. Let x_k^{AV}, x_k^{LT}, x_k^{FT}, x_k^{LC}, x_k^{FC} be the longitudinal positions of the vehicles involved in the lane changing process. Then, the following conditions must hold true for a safe lane change:

$$\begin{aligned} x_k^{AV} &\le x_k^{LC} - S_k^{LC}, \\ x_k^{AV} &\ge x_k^{FC} + S_k^{FC}, \\ x_k^{AV} &\le x_k^{LT} - S_k^{LT}, \\ x_k^{AV} &\ge x_k^{FT} + S_k^{FT}. \end{aligned} \tag{4}$$

The safe distances $S_i(t)$ are evaluated continuously. If the above inequalities are violated at any time instant during the lane changing, the AV maneuvers back to the original lane. This maneuver is done based on the change of leader vehicle. Thus, when the safe conditions violate, the target point TP is chosen at a safe distance from the leader in the original lane.

2.1.3 Problem Formulation

Given that the dynamics of the AV is unknown and TP in the target lane is defined, the following problem is addressed in this work:

Problem: Design a lane changing algorithm for the AV that incorporates the following:

(i) An optimal model-free controller for the AV's lateral maneuver such that $x_{1,k} \to 0$, $x_{3,k} \to 0$.

(ii) A lane change decision making mechanism based on inequalities in (4).

(iii) An optimal model-free controller for post lane change platooning.

2.2 Real-World Experimental Validation with Full-State Feedback

2.2.1 Model-Based Policy Iteration

For the following discrete-time linear system:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \tag{5}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state, $\mathbf{u}_k \in \mathbb{R}^m$ is the control input, $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times m}$. To minimize state deviations and control effort, we aim to develop a linear optimal control law expressed as:

$$\mathbf{u}_k = -\mathbf{K}\mathbf{x}_k. \tag{6}$$

The minimization of state deviation problem can then be formulated as minimizing the following cost function:

$$\min_{\mathbf{u}} \quad J = \sum_{k=0}^{\infty} (\mathbf{x}_k^{\mathrm{T}}\mathbf{Q}\mathbf{x}_k + \mathbf{u}_k^{\mathrm{T}}\mathbf{R}\mathbf{u}_k), \tag{7}$$

where $\mathbf{Q} = \mathbf{Q}^T \geq 0$, $\mathbf{R} = \mathbf{R}^T > 0$, and $(\mathbf{A}, \sqrt{\mathbf{Q}})$ is observable. If $\mathbf{A}$ and $\mathbf{B}$ are fully known, the solution to the above problem is well-established and can be obtained by solving the following discrete-time algebraic Riccati equation:

$$\mathbf{A}^T\mathbf{P}\mathbf{A} - \mathbf{P} + \mathbf{Q} - \mathbf{A}^T\mathbf{P}\mathbf{B}(\mathbf{R} + \mathbf{B}^T\mathbf{P}\mathbf{B})^{-1}\mathbf{B}^T\mathbf{P}\mathbf{A} = \mathbf{0}. \tag{8}$$

Assume the system is stabilizable, then Eq. (8) has a unique solution $\mathbf{P}^* = \mathbf{P}^{*T} > \mathbf{0}$. The optimal feedback gain $\mathbf{K}^*$ is then given as follows:

$$\mathbf{K}^* = (\mathbf{R} + \mathbf{B}^T\mathbf{P}^*\mathbf{B})^{-1}\mathbf{B}^T\mathbf{P}^*\mathbf{A}. \tag{9}$$

It should be noted that Eq. (8) is nonlinear with respect to $\mathbf{P}$. Therefore, directly solving Eq. (8) is generally challenging, particularly for high-dimensional systems. A model-based policy iteration (PI) method for solving Eq. (8) is detailed in Algorithm 1 [58]. In Algorithm 1, $\mathbf{A}_j = \mathbf{A} - \mathbf{B}\mathbf{K}_j$ is defined.

Algorithm 1 Model-Based PI

```
1: Select a stabilizing control policy $\mathbf{K}_0$ such that $\mathbf{A} - \mathbf{B}\mathbf{K}_0$ is a Schur matrix. Initialize $j \leftarrow 0$. Select
   a sufficiently small constant $\epsilon > 0$.
2: **repeat**
3: Policy Evaluation (Solve for $\mathbf{P}_j$ from):
       $\mathbf{A}_j^T\mathbf{P}_j\mathbf{A}_j - \mathbf{P}_j + \mathbf{Q} + \mathbf{K}_j^T\mathbf{R}\mathbf{K}_j = \mathbf{0}.$   (10)
4: Policy Improvement:
       $\mathbf{K}_{j+1} = (\mathbf{R} + \mathbf{B}^T\mathbf{P}_j\mathbf{B})^{-1}\mathbf{B}^T\mathbf{P}_j\mathbf{A}.$   (11)
5: $j \leftarrow j + 1$.
```

6: **until** $\|\mathbf{P}_j - \mathbf{P}_{j-1}\| < \epsilon$.

2.2.2 Data-driven Policy Iteration

In this section, we propose an online data-driven learning-based controller design approach, adaptive dynamic programming (ADP), that does not require precise knowledge of the system matrices $\mathbf{A}$ and $\mathbf{B}$. The modified system equation is defined as follows:

$$\mathbf{x}_{k+1} = \mathbf{A}_j \mathbf{x}_k + \mathbf{B}(\mathbf{u}_k + \mathbf{K}_j \mathbf{x}_k). \tag{12}$$

Along the trajectories of (12), one can obtain that

$$\mathbf{x}_{k+1}^T \mathbf{P}_j \mathbf{x}_{k+1} - \mathbf{x}_k^T \mathbf{P}_j \mathbf{x}_k = [\mathbf{A}_j \mathbf{x}_k + \mathbf{B}(\mathbf{u}_k + \mathbf{K}_j \mathbf{x}_k)]^T \mathbf{P}_j [\mathbf{A}_j \mathbf{x}_k + \mathbf{B}(\mathbf{u}_k + \mathbf{K}_j \mathbf{x}_k)] - \mathbf{x}_k^T \mathbf{P}_j \mathbf{x}_k. \tag{13}$$

Then, using (10) can get

$$\begin{aligned} &\mathbf{x}_{k+1}^T \mathbf{P}_j \mathbf{x}_{k+1} - \mathbf{x}_k^T \mathbf{P}_j \mathbf{x}_k + \mathbf{x}_k^T \mathbf{Q}_j \mathbf{x}_k \\ &= 2\mathbf{x}_k^T \mathbf{A}^T \mathbf{P}_j \mathbf{B} \mathbf{u}_k + 2\mathbf{x}_k^T \mathbf{A}^T \mathbf{P}_j \mathbf{B} \mathbf{K}_j \mathbf{x}_k - \mathbf{x}_k^T \mathbf{K}_j^T \mathbf{B}^T \mathbf{P}_j \mathbf{B} \mathbf{K}_j \mathbf{x}_k + \mathbf{u}_k^T \mathbf{B}^T \mathbf{P}_j \mathbf{B} \mathbf{u}_k \end{aligned} \tag{14}$$

where $\mathbf{Q}_j = \mathbf{Q} + \mathbf{K}_j^T \mathbf{R} \mathbf{K}_j$. By the property of Kronecker product that $\mathrm{vec}(\mathbf{XYZ}) = (\mathbf{Z}^T \otimes \mathbf{X})\mathrm{vec}(\mathbf{Y})$, we have:

$$\begin{aligned} &[(\mathbf{x}_{k+1}^T \otimes \mathbf{x}_{k+1}^T) - (\mathbf{x}_k^T \otimes \mathbf{x}_k^T)]\mathrm{vec}(\mathbf{P}_j) + (\mathbf{x}_k^T \otimes \mathbf{x}_k^T)\mathrm{vec}(\mathbf{Q}_j) \\ &= [2(\mathbf{x}_k^T \otimes \mathbf{u}_k^T) + 2(\mathbf{x}_k^T \otimes \mathbf{x}_k^T)(\mathbf{I}_n \otimes \mathbf{K}_j^T)]\mathrm{vec}(\mathbf{B}^T \mathbf{P}_j \mathbf{A}) \\ &+ [-(\mathbf{K}_j \mathbf{x}_k)^T \otimes (\mathbf{K}_j \mathbf{x}_k)^T + (\mathbf{u}_k^T \otimes \mathbf{u}_k^T)]\mathrm{vec}(\mathbf{B}^T \mathbf{P}_j \mathbf{B}). \end{aligned} \tag{15}$$

Collecting the data for the time sequence $k_0 < k_1 < \cdots < k_s$, we get

$$\mathbf{\Psi}_j \boldsymbol{\theta}_j = -\mathbf{I}_{\mathbf{x},\mathbf{x}} \mathrm{vec}(\mathbf{Q}_j), \tag{16}$$

where

$\mathbf{\Psi}_j = [\mathbf{\Delta}_{\mathbf{x},\mathbf{x}}, -2\mathbf{I}_{\mathbf{x},\mathbf{u}} - 2\mathbf{I}_{\mathbf{x},\mathbf{x}}(\mathbf{I}_n \otimes \mathbf{K}_j^T), \tilde{\mathbf{I}}_{\mathbf{x},\mathbf{x}} - \mathbf{I}_{\mathbf{u},\mathbf{u}}]$, $\boldsymbol{\theta}_j = [\mathrm{vecs}(\mathbf{P}_j)^T, \mathrm{vec}(\mathbf{B}^T \mathbf{P}_j \mathbf{A})^T, \mathrm{vecs}(\mathbf{B}^T \mathbf{P}_j \mathbf{B})^T]^T$, $\mathbf{\Delta}_{\mathbf{x}} \in \mathbb{R}^{s \times n(n+1)/2}$, $\mathbf{I}_{\mathbf{x},\mathbf{x}} \in \mathbb{R}^{s \times n^2}$, $\tilde{\mathbf{I}}_{\mathbf{x},\mathbf{x}} = [\mathrm{vecv}(\mathbf{K}_j \mathbf{x}_{k_0}), \cdots, \mathrm{vecv}(\mathbf{K}_j \mathbf{x}_{k_s})]^T \in \mathbb{R}^{s \times m(m+1)/2}$, $\mathbf{I}_{\mathbf{x},\mathbf{u}} \in \mathbb{R}^{s \times mn}$, $\mathbf{I}_{\mathbf{u},\mathbf{u}} \in \mathbb{R}^{s \times m(m+1)/2}$.

Assumption: There exists a $s^* \in \mathbb{Z}_+$ such that for all $s > s^*$:

$$\operatorname{rank}([\mathbf{I}_{\mathbf{x},\mathbf{x}},\mathbf{I}_{\mathbf{x},\mathbf{u}},\mathbf{I}_{\mathbf{u},\mathbf{u}}]) = \frac{n(n+1)}{2} + nm + \frac{m(m+1)}{2}. \tag{17}$$

The above data-driven policy iteration method is summarized in Algorithm 2.

Algorithm 2 Model-Free PI

```
1: Employ $\mathbf{u}_k = -\mathbf{K}_0\mathbf{x}_k + \boldsymbol{\eta}_k$ as the input on the time interval $[k_0, k_s]$, where $\mathbf{K}_0$ is an initial
   stabilizing control gain and $\boldsymbol{\eta}_k$ is the exploration/probing noise.
2: Compute $\boldsymbol{\Delta}_{\mathbf{x}}, \mathbf{I}_{\mathbf{x},\mathbf{x}}, \mathbf{I}_{\mathbf{x},\mathbf{u}}, \mathbf{I}_{\mathbf{u},\mathbf{u}}$ until the rank condition (17) is satisfied.
3: Solve for $\boldsymbol{\theta}_j$ from (16). Then $\mathbf{K}_{j+1} = (\mathbf{R} + \mathbf{B}^T\mathbf{P}_j\mathbf{B})^{-1}\mathbf{B}^T\mathbf{P}_j\mathbf{A}$.
4: Let $j \leftarrow j+1$ and repeat Step 3 until $\|\mathbf{P}_j - \mathbf{P}_{j-1}\| \leq \epsilon_0$ for $j \geqslant 1$, where the constant $\epsilon_0 > 0$ is a
   predefined small threshold.
```

2.2.3 Automated Lane-Changing Controller Architecture with Full-State Feedback

The autonomous lane-changing controller architecture shown in Fig. 3 comprises three main modules: (1) Decision Making Module, (2) Data Storing Module, and (3) Learning Module. The input-state data is collected from the AV's sensing unit.

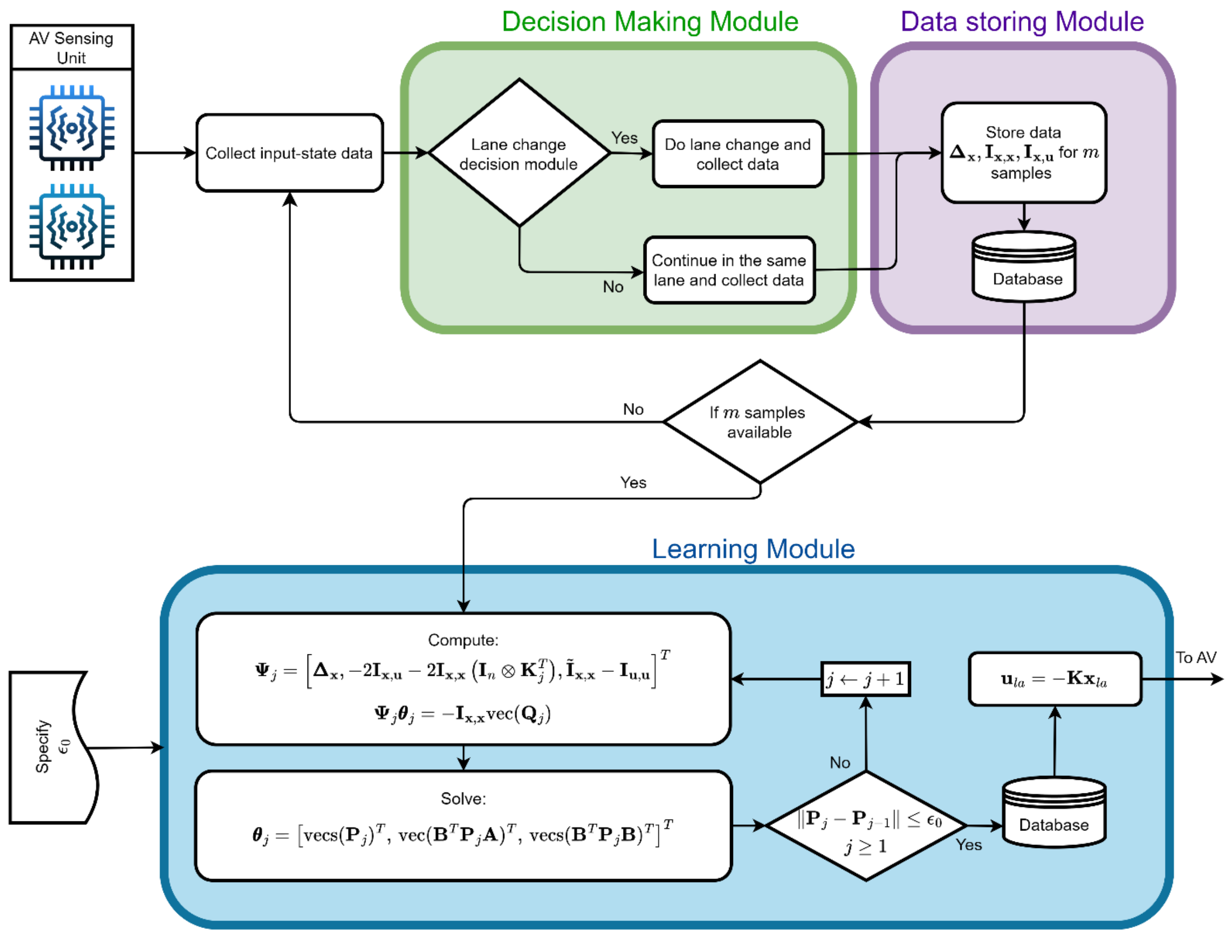
AV Sensing Unit
Collect input-state data
Decision Making Module
Lane change decision module
Yes
Do lane change and collect data
No
Continue in the same lane and collect data
Data storing Module
Store data $\mathbf{\Delta}_{\mathbf{x}}, \mathbf{I}_{\mathbf{x},\mathbf{x}}, \mathbf{I}_{\mathbf{x},\mathbf{u}}$ for m samples
Database
No
If m samples available
Yes
Learning Module
Specify ϵ_0
Compute:
$\mathbf{\Psi}_j = \left[\mathbf{\Delta}_{\mathbf{x}}, -2\mathbf{I}_{\mathbf{x},\mathbf{u}} - 2\mathbf{I}_{\mathbf{x},\mathbf{x}}\left(\mathbf{I}_n \otimes \mathbf{K}_j^T\right), \tilde{\mathbf{I}}_{\mathbf{x},\mathbf{x}} - \mathbf{I}_{\mathbf{u},\mathbf{u}}\right]^T$
$\mathbf{\Psi}_j\boldsymbol{\theta}_j = -\mathbf{I}_{\mathbf{x},\mathbf{x}}\mathrm{vec}(\mathbf{Q}_j)$
$j \leftarrow j+1$
$\mathbf{u}_{la} = -\mathbf{K}\mathbf{x}_{la}$
To AV
No
Solve:
$\boldsymbol{\theta}_j = \left[\mathrm{vecs}(\mathbf{P}_j)^T, \mathrm{vec}(\mathbf{B}^T\mathbf{P}_j\mathbf{A})^T, \mathrm{vecs}(\mathbf{B}^T\mathbf{P}_j\mathbf{B})^T\right]^T$
$\|\mathbf{P}_j - \mathbf{P}_{j-1}\| \leq \epsilon_0$
$j \geq 1$
Yes
Database

Fig. 3: Automated lane-changing controller architecture with full-state feedback

- **Decision Making Module:** This module uses input data to perform a safety check. If conditions are met and the vehicle is not in the target lane, a lane-change maneuver is initiated. Otherwise, the vehicle stays in its current lane to ensure safety.

- **Data Storing Module:** Since the controller needs to handle lane-change scenarios at various speeds and requires sufficient samples that meet the conditions for online policy iteration at each speed, it is necessary to store and manage these data effectively.

- **Learning Module:** Once enough data is collected for a particular speed, it is passed to the learning module to update the control gains. When the optimal gain is learned, it replaces the initial gain and is used to generate the control signal for the vehicle.

2.2.4 Experiments

2.2.4.1 Car Model

We designed a compact car model to validate a lane-change algorithm, which is also suitable for lane-following tasks. The car is equipped with GPS, IMU, and a camera to collect data, processed in real-time by an Nvidia Jetson AGX Xavier board. The sensors and computational modules are carefully selected for compatibility with the experimental platform. The base is a Traxxas TRX-4 RC vehicle, known for its high power and precise steering, equipped with additional components like differential gears to closely replicate real car behavior. Using PWM signals, the TRX-4 can control all attached motors. The total weight, including sensors (0.21 kg), Jetson board (0.27 kg), batteries (1.1 kg), and other parts (0.53 kg), exceeds 2.1 kg. Despite the added weight, the TRX-4 can still perform precise maneuvers, generating realistic trajectories.

The car model hardware is connected as shown in Fig. 4. It uses Marvelmind's Indoor GPS with IMU sensors to collect real-time position and orientation data. However, the Indoor GPS has an error of ±2 cm and is sensitive to nearby obstructions. To improve accuracy, two beacons and a Kalman filter are used to minimize GPS errors and reduce IMU noise. The IMU's performance depends on beacon frequency and location, so filtering is applied to reduce high-pitching errors. A wide-angle camera is also used for lane detection in lane-following tasks. The Nvidia Jetson AGX Xavier board is responsible for processing all sensor data via UDP and discards unnecessary information. All sensors send data to the Jetson board continuously, ensuring data integrity. PWM signals control the TRX-4's steering and speed, but due to insufficient power from the Jetson's internal PWM generator, an external PWM generator powered by an external battery is used to operate the steering motor.

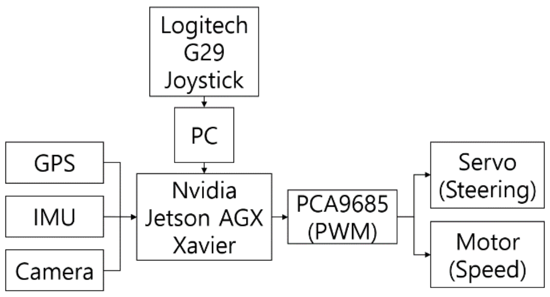

Fig. 4: Hardware connection of the car model through the flow of data.

The PCA9685 board is used to address the power issue by connecting to the Jetson board via I2C and to external batteries (5V, 0.5A) to power the servomotor. The acceleration motor, already powered by its external battery, only requires PWM signals to control speed. Using both PCA9685 and Jetson's PWM generators simultaneously can cause data collisions, so only PCA9685 is used for the acceleration motor, simplifying control. Additionally, a Logitech G29 Driving Force Racing Wheel is wirelessly connected as a joystick for manual control, offering functionality similar to real driving. With programmable buttons for actions like starting autonomous driving, reversing, and more, it enhances the convenience of the experiment.

The central processing unit of the car model is the Nvidia Jetson AGX Xavier, a high-performance compact computer. To minimize sensor latency and improve scalability, sensors are directly connected to the Jetson via serial ports, as shown in Fig. 5. The Jetson board runs three programs simultaneously. The Motor-Servo-Controller, written in C++ and compatible with the PCA9685 driver, controls the acceleration motor and servomotor using parameters like steering angle, velocity, and driving type, which are received as UDP packets. This controller operates independently, sharing no data with the lane-changing or lane-following modules.

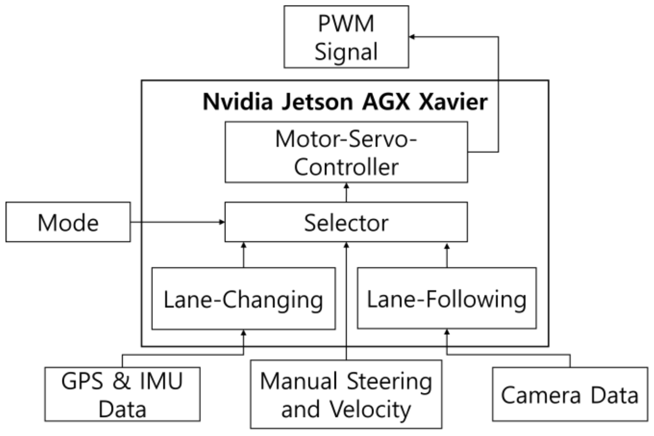

Fig. 5: Hardware connection of the car model through the flow of data

The Lane-Changing and Lane-Following programs implement algorithms that determine the car's direction and speed. Each program receives sensor data as input parameters through serial ports. After processing, the resulting steering angle, speed, and driving mode are sent to the Motor-Servo-Controller via the localhost address using specific port numbers. The driving mode identifier is used to differentiate between outputs from different programs; for example, a driving mode of 1 represents Lane-Changing, while 2 represents Lane-Following. The Motor-Servo-Controller then executes the corresponding commands based on the driving mode. Detailed driving modes are outlined in Table 1.

Table 1: Driving Type Specification

Driving Type	Purpose	Steering Range	Speed Range
0	No data	90	0 (m/sec)
1	Lane changing	60~120	-0.3~0.3
2	Lane following	60~120	0.5

Fig. 6 shows the car model used in the experiment. The body dimensions are 50 cm in width, 24 cm in length, and 26 cm in height. The maximum steering angle is ±35 degrees, but for safety, it is limited to 30 degrees to prevent the wheels from contacting the body frame. The motor has a maximum output of 46 W and a top speed of 15 km/h, which can be maintained until the battery drops to around 80%. Thus, experiments require the battery to maintain a charge above 80%.

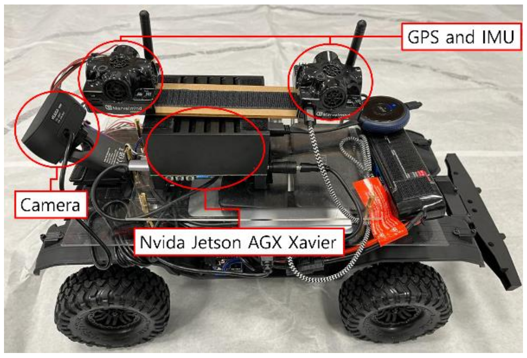

Fig. 6: The car model with all the hardware for all-state feedback case

The lane-changing algorithm collects data for the first 100 time steps (1 time step = 0.083 seconds) to find a near-optimal gain K. After lane changing is completed, the car switches to the lane-following algorithm to ensure continuous movement. The overall data flow is illustrated in Fig. 7.

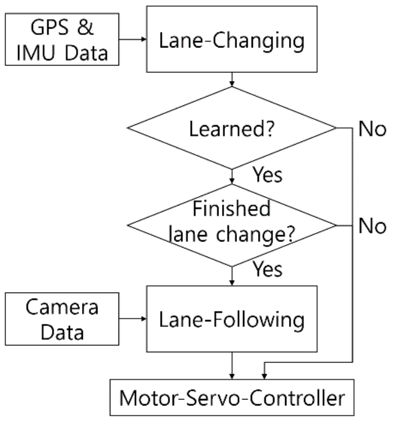

Fig. 7: Basic algorithmic structure of autonomous driving for full-state feedback case

2.2.4.2 Experiments Environment

We developed a two-lane road section in the lab for experiments, with a total length of 4.5 meters and a lane width of 0.3 meters. As shown in Fig. 8, there are four vehicles involved: the RC car (remote-controlled car), LT (leader in the target lane), LC (leader in the current lane), and FT (follower in the target lane). $S_{FT}(t)$, $S_{LT}(t)$ and $S_{LC}(t)$ present safety distances. In Fig. 8, x_1 is the lateral displacement of the RC car's center of gravity from the centerline of the target lane, and x_3 represents the orientation error of the RC car relative to the road. The objective is for the RC car to switch lanes safely, avoiding collisions. The lane-change decision-making algorithm is based on the state vector $\mathbf{x} = [x_1, x_2, x_3, x_4]$, where x_2 is the change rate of x_1 , and x_4 is the change rate of x_3 . The control input u_k is defined as the steering wheel angle δ_k.

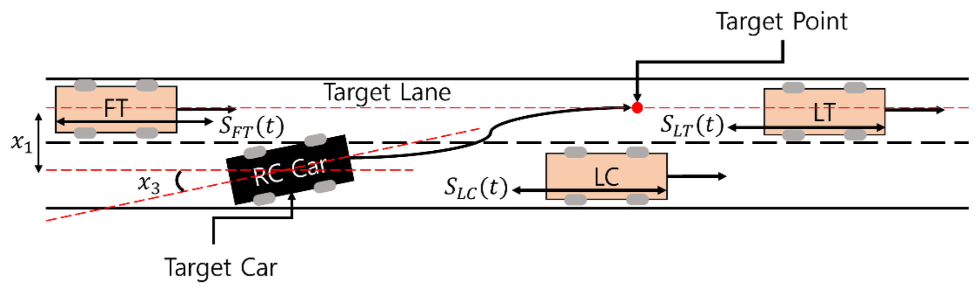

Fig. 8: The desirable movement of the experiments for full-state feedback case

Our goal is to develop a data-driven optimal controller to compute the steering angle δ_k for the RC car to successfully execute lane changes from the current lane to the target lane while ensuring the safety of surrounding vehicles. We use Algorithm 2 to find the optimal δ_k, based on real-time data matrices formed by collecting the RC car's state $\mathbf{x}_k$ and input δ_k. The initial stabilizing gain is defined as $\mathbf{K}_0 = [0.0047, -0.0447, 2.0002, 0.0002]$, with weight matrices $\mathbf{Q} = \mathbf{I}_4$ and $\mathbf{R} = \mathbf{I}_1$. The exploration noise $\boldsymbol{\eta}_k$ is introduced as a combination of sinusoidal waves to enhance learning.

2.2.4.3 Result and Analysis

This section presents the experimental results for the RC car's lane-change performance. During the learning phase, we applied the initial stabilizing controller gain $\mathbf{K}_0$ to compute the steering angle $\delta_{0k} = -\mathbf{K}_0\mathbf{x}_k + \eta_k$. This steering angle was used as the control input for the RC car to collect data over 100 time steps, ensuring that the rank condition specified in equation (17) was satisfied. The training data $\{\mathbf{x}_k, \delta_k\}_{k=0}^{100}$ were gathered using GPS sensors mounted on the RC car.

This section presents and analyzes the experimental results. We conducted 10 experiments to validate the proposed methodology, and the mean values of x_1, x_2, x_3 and x_4 are shown as red plots in Fig. 9. For each experiment, 100 data samples were collected to learn the optimal control gain specific to that run. The training phase concluded once the optimal gain was obtained. After training, the RC car returned to the initial testing position. As shown in Fig. 9, during testing, using the trained gain $\mathbf{K}$, the car successfully performed the lane change, with its states converging to zero, indicating successful execution. This confirms that the RC car learned the optimal control policy for lane changing in real-time using sensor data.

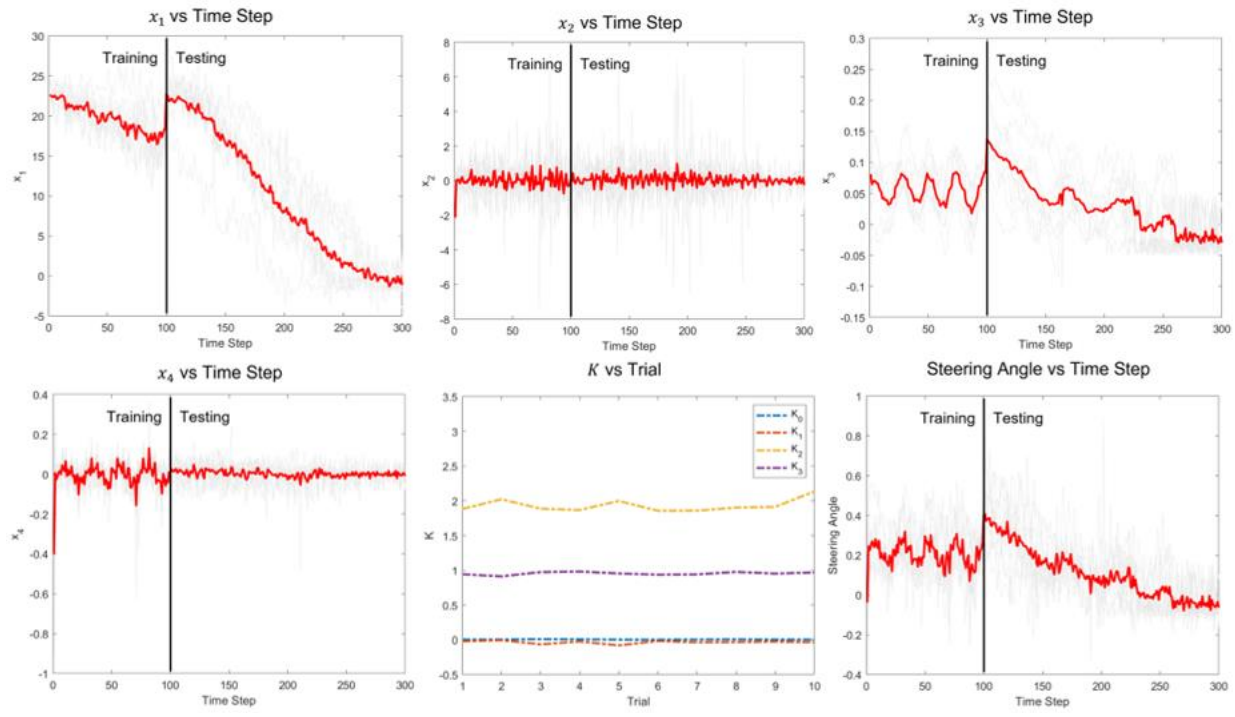

Fig. 9: Training and testing experiments are conducted in the same environment.

As shown in Fig. 9, the trained control gain $\mathbf{K}$ is consistent across experiments, with only minor variations due to sensor noise, highlighting the algorithm's robustness. Each experiment showed smooth lane-changing using the learned $\mathbf{K}$, indicating that the car reliably estimates the optimal control policy through repeated trials, as summarized in Table 2.

Table 2: Initial and Trained $\mathbf{K}$ Comparison for Full-State Feedback Case

	K_0	K_1	K_2	K_3
Initial $\mathbf{K}_0$	0.0047	-0.0447	2.0002	0.0002
Average Trained $\mathbf{K}$	0.0053	-0.0339	1.9330	0.9558

2.3 Real-World Experimental Validation with Output Feedback

2.3.1 Model-Based Policy Iteration

When the system states cannot be fully observed directly, an observation equation needs to be added to system (5) to obtain the complete discrete system equation as follows:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k. \end{aligned} \tag{18}$$

where $y_k \in \mathbb{R}^r$ is the system output, $C \in \mathbb{R}^{r\times n}$ is constant matrix. And assume that (A, B) is controllable and (A, C) is observable. And we still design a linear optimal control law of the form:

$$\mathbf{u}_k = -\mathbf{K}^* \mathbf{x}_k, \tag{19}$$

that minimizes the following cost function:

$$\min_{\mathbf{u}} \quad J = \sum_{k=0}^{\infty} (\mathbf{y}_k^{\mathrm{T}} \mathbf{Q} \mathbf{y}_k + \mathbf{u}_k^{\mathrm{T}} \mathbf{R} \mathbf{u}_k), \tag{20}$$

where $\mathbf{Q} = \mathbf{Q}^T \geq 0$, $\mathbf{R} = \mathbf{R}^T > 0$, and $(\mathbf{A}, \sqrt{\mathbf{Q}})$ is observable. If $\mathbf{A}$ and $\mathbf{B}$ are fully known, the solution to the above problem is well-established and can be obtained by solving the following discrete-time algebraic Riccati equation:

$$\mathbf{A}^T\mathbf{P}\mathbf{A} - \mathbf{P} + \mathbf{C}^T\mathbf{Q}\mathbf{C} - \mathbf{A}^T\mathbf{P}\mathbf{B}(\mathbf{R} + \mathbf{B}^T\mathbf{P}\mathbf{B})^{-1}\mathbf{B}^T\mathbf{P}\mathbf{A} = \mathbf{0}. \tag{21}$$

Assume the system is stabilizable, then Eq. (21) has a unique solution $\mathbf{P}^* = \mathbf{P}^{*T} > \mathbf{0}$. The optimal feedback gain $\mathbf{K}^*$ is then given as follows:

$$\mathbf{K}^* = (\mathbf{R} + \mathbf{B}^T\mathbf{P}^*\mathbf{B})^{-1}\mathbf{B}^T\mathbf{P}^*\mathbf{A}. \tag{22}$$

Similar to the full-state feedback case, the model-based policy iteration algorithm is presented in Algorithm 3.

Algorithm 3 Model-Based PI

```
1: Select an admissible control policy $\mathbf{K}_0$ such that $\mathbf{A} - \mathbf{B}\mathbf{K}_0$ is a Schur matrix. Initialize $j \leftarrow 0$.
   Select a sufficiently small constant $\bar{\epsilon} > 0$.
2: repeat
3: Policy Evaluation (Solve for $\mathbf{P}_j$ from):
```

$$\mathbf{A}_j^T\mathbf{P}_j\mathbf{A}_j-\mathbf{P}_j+\mathbf{C}^T\mathbf{Q}\mathbf{C}+\mathbf{K}_j^T\mathbf{R}\mathbf{K}_j=\mathbf{0}. \tag{23}$$

```
4: Policy Improvement:
        $\mathbf{K}_{j+1}=(\mathbf{R}+\mathbf{B}^T\mathbf{P}_j\mathbf{B})^{-1}\mathbf{B}^T\mathbf{P}_j\mathbf{A}.$   (24)
5: $j\leftarrow j+1$.
6: until $\|\mathbf{P}_j-\mathbf{P}_{j-1}\|<\epsilon$ .
```

2.3.2 Data-driven Policy Iteration

In this section, we present a non-model-based approach for state reconstruction using only the control input and system output. The system matrices $\mathbf{A}$ and $\mathbf{B}$ are assumed to be unknown, and the state is considered unmeasurable. To address this, we introduce a policy iteration-based online output-feedback algorithm to solve the optimal control problem. By leveraging the input-output data and system equations (18), the state can be reconstructed as shown as follow [59]:

$$\mathbf{x}_k=\mathbf{M}_\mathbf{u}\tilde{\mathbf{u}}_{k-1,k-N}+\mathbf{M}_\mathbf{y}\tilde{\mathbf{y}}_{k-1,k-N}=\mathbf{\Theta}\mathbf{z}_k \tag{25}$$

where N is the observability index,

$\mathbf{z}_k=[\tilde{\mathbf{u}}_{k-1,k-N}^T,\tilde{\mathbf{y}}_{k-1,k-N}^T]^T$, $\mathbf{\Theta}=[\mathbf{M}_\mathbf{u},\mathbf{M}_\mathbf{y}]$, $\mathbf{M}_\mathbf{y}=\mathbf{A}^N\mathcal{O}_N^+$, $\mathbf{M}_\mathbf{u}=\mathcal{C}_N-\mathbf{M}_\mathbf{y}\mathcal{T}_N$,

$\tilde{\mathbf{u}}_{k-1,k-N}=[\mathbf{u}_{k-1}^T,\mathbf{u}_{k-2}^T,\cdots,\mathbf{u}_{k-N}^T]^T$, $\tilde{\mathbf{y}}_{k-1,k-N}=[\mathbf{y}_{k-1}^T,\mathbf{y}_{k-2}^T,\cdots,\mathbf{y}_{k-N}^T]^T$,

$\mathcal{C}_N=[\mathbf{B}\ \ \mathbf{AB}\ \ \mathbf{A}^2\mathbf{B}\ \ \cdots\ \ \mathbf{A}^{N-1}\mathbf{B}]$,

$$\mathcal{O}_N=\begin{bmatrix}\mathbf{CA}^{N-1}\\ \vdots\\ \mathbf{CA}\\ \mathbf{C}\end{bmatrix},\mathcal{T}_N=\begin{bmatrix}0 & \mathbf{CB} & \mathbf{CAB} & \cdots & \mathbf{CA}^{N-2}\mathbf{B}\\ 0 & 0 & \mathbf{CB} & \cdots & \mathbf{CA}^{N-3}\mathbf{B}\\ \vdots & \vdots & \ddots & \ddots & \vdots\\ 0 & \cdots & \cdots & 0 & \mathbf{CB}\\ 0 & 0 & 0 & 0 & 0\end{bmatrix}.$$

Letting $\mathbf{A}_j=\mathbf{A}-\mathbf{B}\mathbf{K}_j$, (18) can be rewritten as

$$\mathbf{x}_{k+1}=\mathbf{A}_j\mathbf{x}_k+\mathbf{B}(\mathbf{u}_k+\mathbf{K}_j\mathbf{x}_k). \tag{26}$$

Using (23), the following can obtain that

$$\begin{aligned}\mathbf{x}_{k+1}^T\mathbf{P}\mathbf{x}_{k+1}-\mathbf{x}_k^T\mathbf{P}\mathbf{x}_k=&-\mathbf{x}_k^T\mathbf{C}^T\mathbf{Q}\mathbf{C}\mathbf{x}_k-\mathbf{x}_k^T\mathbf{K}_j^T\mathbf{R}\mathbf{K}_j\mathbf{x}_k+2\mathbf{x}_k^T\mathbf{A}^T\mathbf{P}_j\mathbf{B}\mathbf{u}_k+2\mathbf{x}_k^T\mathbf{A}^T\mathbf{P}_j\mathbf{B}\mathbf{K}_j\mathbf{x}_k\\ &-\mathbf{x}_k^T\mathbf{K}_j^T\mathbf{B}^T\mathbf{P}_j\mathbf{B}\mathbf{K}_j\mathbf{x}_k+\mathbf{u}_k^T\mathbf{B}^T\mathbf{P}_j\mathbf{B}\mathbf{u}_k\end{aligned} \tag{27}$$

Then, using (25) can get

$$\begin{aligned}\mathbf{z}_{k+1}^T\tilde{\mathbf{P}}\mathbf{z}_{k+1}-\mathbf{z}_k^T\tilde{\mathbf{P}}\mathbf{z}_k=&-\mathbf{y}_k^T\mathbf{Q}\mathbf{y}_k-\mathbf{z}_k^T\tilde{\mathbf{K}}_j^T\mathbf{R}\tilde{\mathbf{K}}_j\mathbf{z}_k+2\mathbf{z}_k^T\mathbf{\Gamma}_{1j}^T\mathbf{u}_k+2\mathbf{z}_k^T\mathbf{\Gamma}_{1j}^T\tilde{\mathbf{K}}_j\mathbf{z}_k\\ &-\mathbf{z}_k^T\tilde{\mathbf{K}}_j^T\mathbf{\Gamma}_{2j}\tilde{\mathbf{K}}_j\mathbf{z}_k+\mathbf{u}_k^T\mathbf{\Gamma}_{2j}\mathbf{u}_k\end{aligned} \tag{28}$$

where $\mathbf{\Gamma}_{1j} = \mathbf{B}^T \mathbf{P}_j \mathbf{A}\mathbf{\Theta}$, $\mathbf{\Gamma}_{2j} = \mathbf{B}^T \mathbf{P}_j \mathbf{B}$, $\bar{\mathbf{P}}_j = \mathbf{\Theta}^T \mathbf{P}\mathbf{\Theta}$, $\tilde{\mathbf{K}}_j = \mathbf{K}_j \mathbf{\Theta}$. Note that (28) does not require the full-state measurements.

By collecting the data for the time sequence $k_0 < k_1 < \cdots < k_s$, we get

$$\mathbf{\Phi}_j \boldsymbol{\gamma}_j = -\mathbf{I}_{\mathbf{y}} \operatorname{vec}(\mathbf{Q}) - \mathbf{I}_{\mathbf{z}} \operatorname{vec}(\tilde{\mathbf{K}}_j^T \mathbf{R} \tilde{\mathbf{K}}_j) \tag{29}$$

where

$\mathbf{\Phi}_j = [\mathbf{\Delta}_{\mathbf{z}}, -2\mathbf{I}_{\mathbf{z},\mathbf{u}} - 2\mathbf{I}_{\mathbf{z},\mathbf{z}}(\mathbf{I}_n \otimes \tilde{\mathbf{K}}_j^T), \mathbf{I}_{\tilde{\mathbf{K}}_j \mathbf{z}} - \mathbf{I}_{\mathbf{u}}]$, $\boldsymbol{\gamma}_j = [\operatorname{vecs}(\tilde{\mathbf{P}}_j)^T, \operatorname{vec}(\mathbf{\Gamma}_{1j})^T, \operatorname{vecs}(\mathbf{\Gamma}_{2j})^T]^T$.

Assumption: There exists a $s^* \in \mathbb{Z}_+$ such that for all $s > s^*$:

$$\operatorname{rank}([\mathbf{I}_{\mathbf{z}}, \mathbf{I}_{\mathbf{z},\mathbf{u}}, \mathbf{I}_{\mathbf{u}}]) = \frac{p_1(p_1+1)}{2} + p_1 m + \frac{m(m+1)}{2}. \tag{30}$$

where $p_1 = N(m+r)$.

The above data-driven policy iteration method is summarized in Algorithm 4.

Algorithm 4 Model-Free PI

```
1: Employ $\mathbf{u}_k = -\widetilde{\mathbf{K}}_0 \mathbf{z}_k + \boldsymbol{\eta}_k$ as the input on the time interval $[k_0, k_s]$, where $\widetilde{\mathbf{K}}_0$ is an initial
   stabilizing control gain and $\boldsymbol{\eta}_k$ is the exploration/probing noise.
2: Compute $\mathbf{\Psi}_j$ until the rank condition (30) is satisfied.
3: Solve for $\boldsymbol{\theta}_j$ from (29). Then $\tilde{\mathbf{K}}_{j+1} = (\mathbf{R} + \mathbf{\Gamma}_{2j})^{-1} \mathbf{\Gamma}_{1j}$.
4: Let $j \leftarrow j+1$ and repeat Step 3 until $\|\widetilde{\mathbf{P}}_j - \widetilde{\mathbf{P}}_{j-1}\| \le \bar{\epsilon}$ for $j \geq 1$, where the constant $\bar{\epsilon} > 0$ is a
   predefined small threshold.
```

2.3.4 Experiments

Similar to the full-state feedback case, the architecture of the autonomous lane-changing controller with output feedback is shown in Fig. 10. The main difference from the full-state feedback scenario lies in the variations of the policy iteration equations.

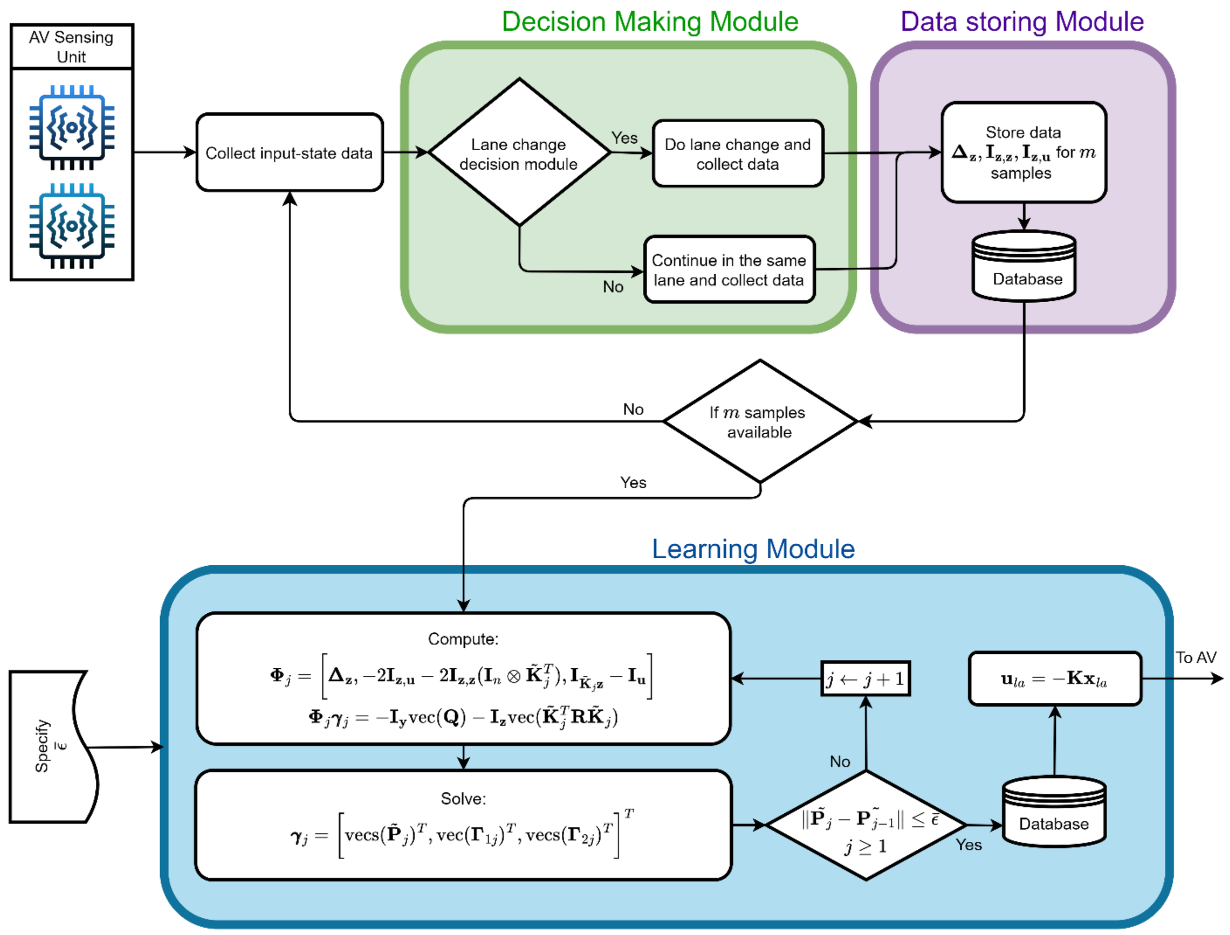
AV Sensing Unit
Collect input-state data
Decision Making Module
Lane change decision module
Yes
Do lane change and collect data
No
Continue in the same lane and collect data
Data storing Module
Store data $\mathbf{\Delta_z}, \mathbf{I_{z,z}}, \mathbf{I_{z,u}}$ for m samples
Database
If m samples available
No
Yes
Learning Module
Compute:
$\mathbf{\Phi}_j = \left[\mathbf{\Delta_z}, -2\mathbf{I_{z,u}} - 2\mathbf{I_{z,z}}(\mathbf{I}_n \otimes \tilde{\mathbf{K}}_j^T), \mathbf{I}_{\tilde{\mathbf{K}}_j\mathbf{z}} - \mathbf{I_u}\right]$
$\mathbf{\Phi}_j \boldsymbol{\gamma}_j = -\mathbf{I_y}\mathrm{vec}(\mathbf{Q}) - \mathbf{I_z}\mathrm{vec}(\tilde{\mathbf{K}}_j^T \mathbf{R} \tilde{\mathbf{K}}_j)$
Solve:
$\boldsymbol{\gamma}_j = \left[\mathrm{vecs}(\tilde{\mathbf{P}}_j)^T, \mathrm{vec}(\mathbf{\Gamma}_{1j})^T, \mathrm{vecs}(\mathbf{\Gamma}_{2j})^T\right]^T$
$j \leftarrow j+1$
No
$\|\tilde{\mathbf{P}}_j - \tilde{\mathbf{P}}_{j-1}\| \leq \bar{\epsilon}$
$j \geq 1$
Yes
Database
$\mathbf{u}_{la} = -\mathbf{K}\mathbf{x}_{la}$
To AV
Specify $\bar{\epsilon}$

Fig. 10: Automated lane-changing controller architecture with output feedback

2.3.4.1 Hardware Setup

We developed a scaled car model equipped with GPS, IMU, and a camera for lane-changing and lane-following tests. The Nvidia Jetson AGX Xavier board processes data in real-time. The setup emulates real-world conditions to evaluate the algorithm's performance under various scenarios, repeated with slight variations to test robustness. The car is based on the TRX-4 RC from Traxxas, known for its power and precise steering, and includes components that simulate real vehicle dynamics. The total weight, including sensors, computing hardware, and batteries, is over 2.1 kg, allowing the TRX-4 to replicate real car behavior effectively.

The experimental platform has been upgraded, as shown in Fig. 11, to include Marvelmind Indoor GPS and the RealSense D435 camera for improved accuracy and versatility. The Marvelmind GPS, with ±2 cm accuracy, uses two beacons to enhance precision and reduce measurement errors, while a Kalman filter mitigates IMU noise. A wide-angle camera is used for lane detection. The RealSense D435 camera, combining high-resolution RGB and depth sensing, provides accurate spatial perception with an 85.2°

horizontal and 58° vertical field of view, operating at 90 FPS. Its infrared stereo technology improves depth accuracy, making it reliable even in low-light conditions, such as night driving. Additionally, the experimental car model measures 50 cm in width, 24 cm in length, and 26 cm in height. To prevent wheel contact with the body, the maximum steering angle of ±35 degrees is limited to 30 degrees. The motor, with a peak output of 46 W, enables speeds up to 15 km/h. The vehicle maintains stable performance as long as the battery charge remains above 80%.

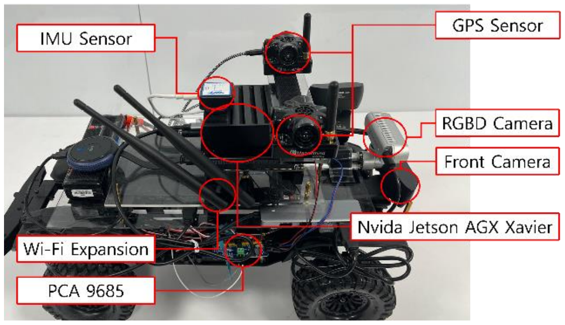

Fig. 11: The upgraded car model with all the hardware for state output feedback

The experimental setup integrates a RealSense D435 camera to create a detailed 3D map of the surroundings using RGB and depth data, facilitating obstacle detection and lane-change decisions by tracking static and dynamic objects such as vehicles, pedestrians, and barriers. This information feeds into a path-planning algorithm to determine the safest and most efficient trajectory, enabling the vehicle to perform lane changes safely while continuously updating based on environmental changes. The Nvidia Jetson AGX Xavier handles all computational tasks and sensor data collection via UDP packets, ensuring complete and real-time data processing. The vehicle's steering and acceleration are controlled by PWM signals, with an external PWM generator powering the TRX-4's steering mechanism. Additionally, a wireless Logitech G29 Driving Force Racing Wheel joystick is used for manual control, offering flexibility and convenience with programmable buttons for commands like starting autonomous driving and reversing, thus eliminating the need for a keyboard and enhancing user experience.

At the core of our car model is the Nvidia Jetson AGX Xavier Board, a powerful and compact computing module. Sensors are directly connected to the Jetson to reduce latency and improve scalability through serial connections. As shown in Fig. 12, three separate programs run simultaneously on the Jetson board. The Motor-Servo-Controller program handles all motor activities, such as acceleration and servo control, using parameters like steering angle, speed, and driving mode, which are communicated through UDP packets. This program operates independently from the Lane-Changing and Lane-Following programs. Written in C++, it ensures compatibility with drivers for smooth operation using the PCA9685

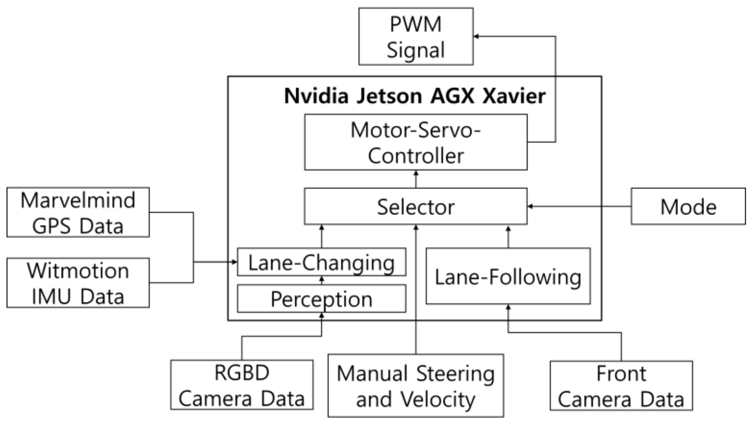

Fig. 12: Hardware connection of the upgraded car model through the flow of data

The Lane-Changing and Lane-Following programs control the vehicle's steering angle and speed based on inputs from dedicated sensors received through serial connections. After processing, the computed steering angle, speed, and driving mode are sent to the Motor-Servo-Controller over the local network using designated port numbers. The driving mode distinguishes data from each program, with Lane-Changing labeled as mode 1 and Lane-Following as mode 2, allowing the controller to identify and execute the appropriate driving commands accordingly.

In our car model, the lane-changing algorithm collects data over the first 100 seconds, with a time step of 0.083 seconds, to compute a near-optimal gain $\bar{K}$. After completing a lane change, the vehicle seamlessly transitions to the lane-following algorithm to maintain smooth driving. This ensures that the experimental data flow follows the schematic illustrated in Fig. 13.

2.3.4.2 Software

We developed a model-free control algorithm that uses input-output data to estimate the system's unmeasurable states, eliminating the need for parameters like $\mathbf{A}$ and $\mathbf{B}$. This online output-feedback algorithm, improved through policy iteration, relies entirely on observed outputs and control inputs. It adapts to system changes by continuously updating itself with new data, optimizing a cost function to balance performance and control effort. This approach ensures high efficiency and robustness, making it suitable for real-world applications such as autonomous driving.

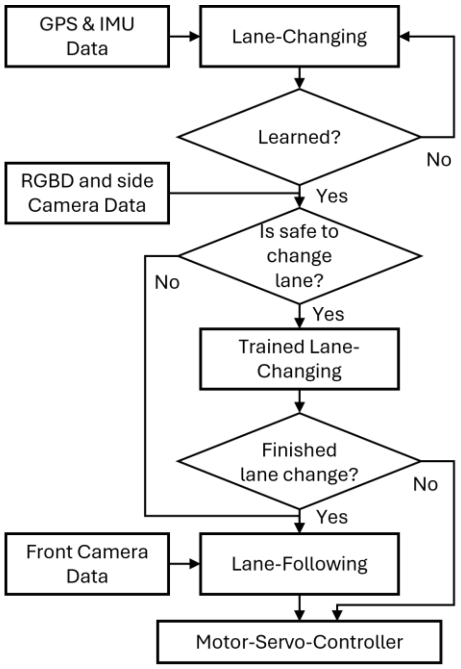

Fig. 13: Basic framework of autonomous driving algorithms for output feedback case

2.3.4.3 Experimental Results

We conducted three experiments to evaluate our learning-based optimal control algorithm, which utilizes a data-driven approach to adjust the feedback gain $\mathbf{K}$ based on real-time state $\mathbf{x}_k$ and steering angle input δ_k. The algorithm ensures safe lane changes by optimizing $\mathbf{K}$ through the ADP algorithm. The initial feedback gain is set to $\bar{\mathbf{K}}_0 = [-0.057, 0.187, -0.182, 3.016, 0.276, -10.646]$ with identity matrices as weights $\mathbf{Q}$ and penalties for control effort $\mathbf{R}$. To enhance exploration, noise η_k is added during the optimization of δ_k. A two-lane test track, 4.5 m long and 0.3 m wide, was constructed, as shown in Fig. 14, involving the RC car and three other vehicles: the current lane leader (LC), the target lane leader (LT), and the target lane follower (FT), representing potential hazards. Safe distances from these vehicles are defined as $S_{FT}(t)$, $S_{LT}(t)$, and $S_{LC}(t)$. The state vector $\mathbf{x} = [x_1, x_2]$ represents the RC car's lateral displacement from the target lane and its orientation error. The initial and trained gains are summarized in Table 3.

Table 3: Initial and Trained $\mathbf{K}$ Comparison for Output Feedback Case

	K_0	K_1	K_2	K_3
Initial $\mathbf{K}_0$	0.0047	-0.0447	2.0002	0.0002

Average Trained $\mathbf{K}$	0.0053	-0.0339	1.9330	0.9558

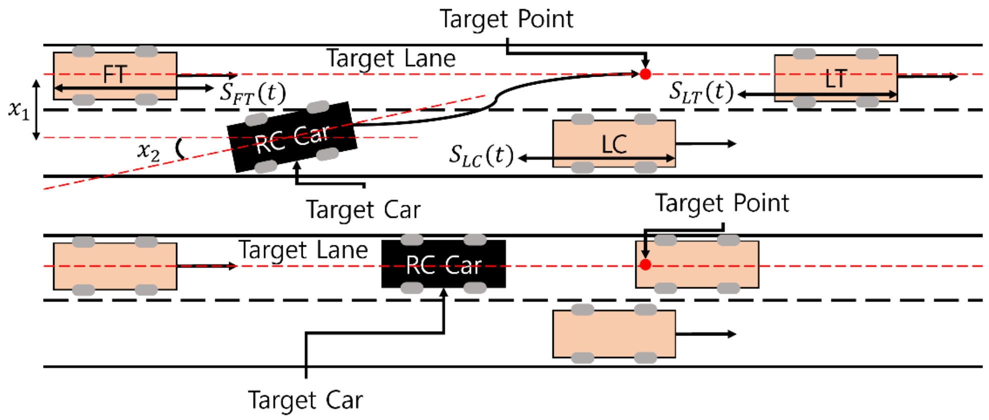

Fig. 14: Desirable movement of the experiments for output feedback case

2.3.4.4 Analysis and Discussion

As shown in Fig. 15, the control gain vector $\mathbf{K}$, consisting of components K_1 to K_6, remains stable across multiple trials, with only minor variations due to sensor noise, highlighting the robustness of the approach. The RC car consistently performed smoother lane changes using the trained gains $\bar{\mathbf{K}}$, demonstrating its capability to effectively learn and apply the optimal control strategy iteratively.

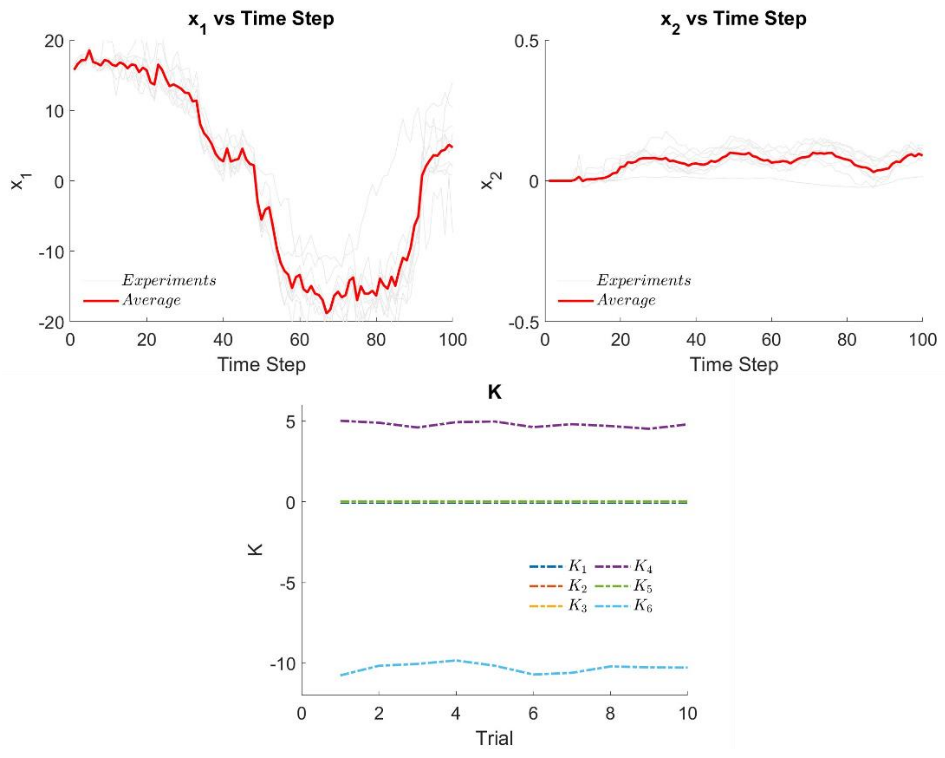

Fig. 15: Experimental results of training and testing

Testing in two scenarios demonstrated the RC car's ability to adapt to varying conditions by maintaining its path, performing lane changes, or stopping as required. These results emphasize the effectiveness of the control algorithms in handling diverse traffic situations.

Scenario 1: Lane Change to an Empty Lane. As illustrated in Fig. 16, if the depth reading at the center is less than 1 meter, indicating an obstacle ahead, and the left-side depth exceeds 1.5 meters, signifying a clear adjacent lane, the system initiates a lane change. A UDP signal is then sent to the vehicle's control system to execute the lane change maneuver safely.

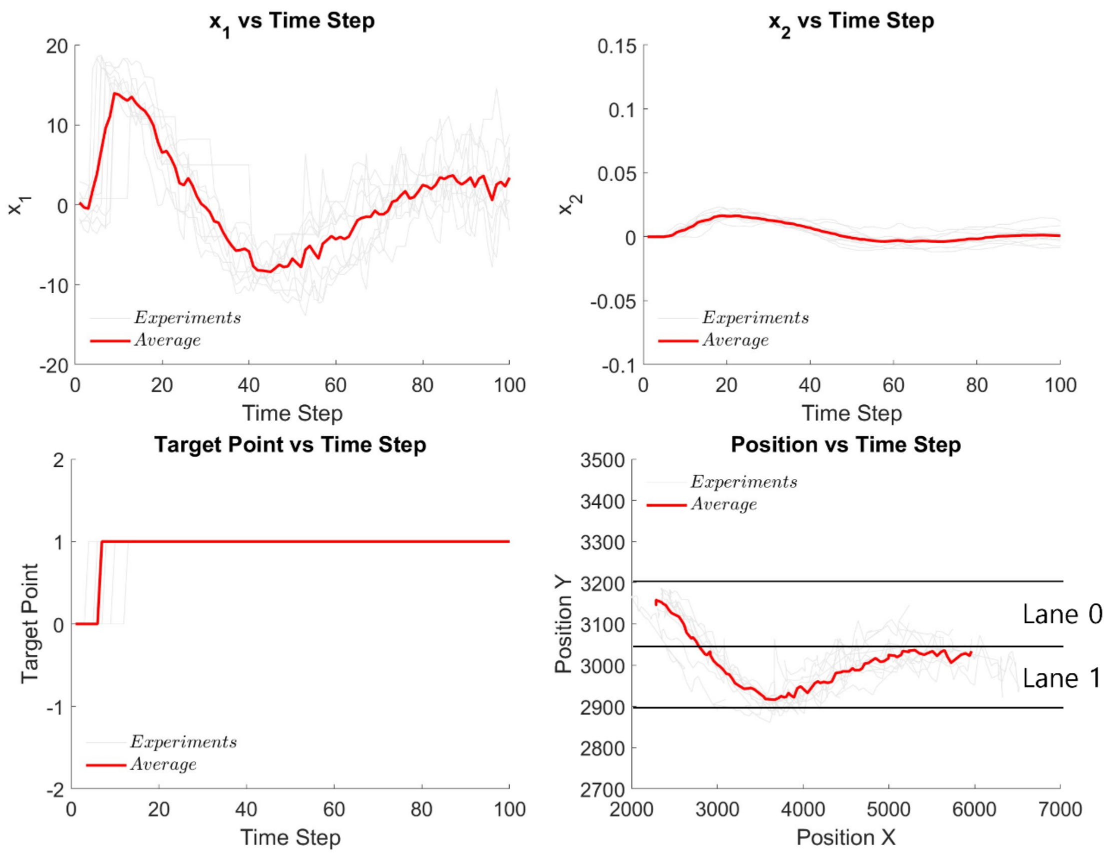

Fig. 16: Outcome of scenario 1

Scenario 2: Emergency Stop. As shown in Fig. 17, the RC car initially detects an obstacle in its current lane, triggering a lane change to the target lane, indicated by a spike in x_1. However, upon entering the new lane, it encounters additional obstructions and decides to stop. This is represented by a sharp drop in x_1 and the stabilization of the target lane signal at -1, indicating the stop command.

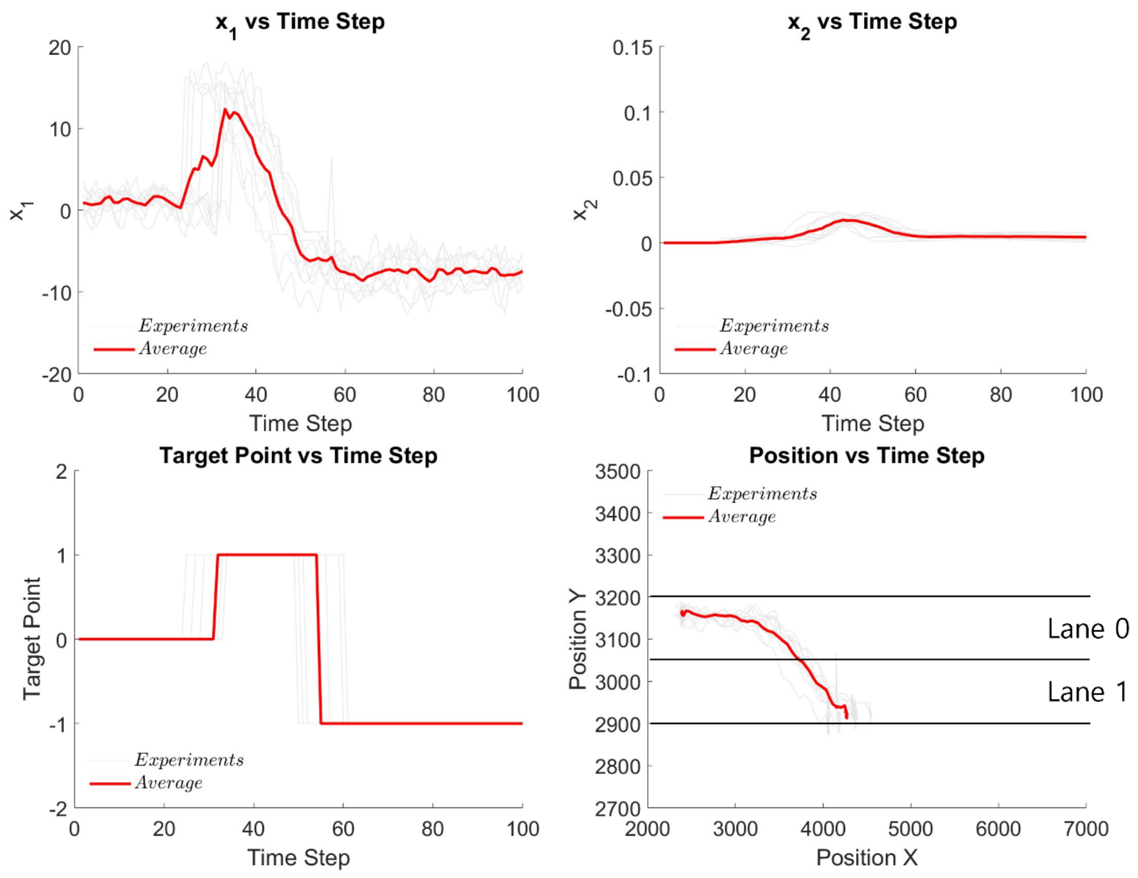

Fig. 17: Outcome of scenario 2

Data-Driven Combined Longitudinal and Lateral Control for the Car Following Problem

This part presents the detailed formulation of the dynamic model and tracking error analysis for autonomous vehicles (AVs), followed by the development of model-based policy iteration techniques and the proposed two-phase data-driven policy iteration approach. Finally, the effectiveness of the designed control strategies is validated through simulation experiments using SUMO and CommonRoad environments.

2.1 Model Description and Problem Formulation

2.1.1 Model Description of the Leading Vehicle

In Fig. 18, vehicle F represents the following AV, while vehicle L denotes the preceding/leading vehicle. The path's constant radius is denoted as r, and the curvature is given by $\kappa = 1/r$. In Fig. 19, the variables $x_w, y_w,$ and ϕ_w ($w \in \{L, F\}$) correspond to the x-coordinate, y-coordinate, and yaw angle of the respective vehicles. The slip angle is denoted by β_w, v_w, ω_w, and $v_{\beta,w}$ represent the velocity, yaw angular velocity, and slip angular velocity. Accelerations along v_w and ω_w are denoted by a_w and $a_{\phi,w}$, respectively. The motion of the leading vehicle L is described by the following differential equations

$$\begin{aligned} \dot{x}_L &= v_L \cos(\varphi_L + \beta_L), \\ \dot{y}_L &= v_L \sin(\varphi_L + \beta_L), \\ \dot{\varphi}_L &= \omega_L, \\ \dot{v}_L &= a_L, \\ \dot{\omega}_L &= a_{\varphi,L}, \\ \dot{\beta}_L &= v_{\beta,L}. \end{aligned} \tag{31}$$

To make the equation more compact, we define $p_L = [x_L, y_L, \varphi_L]^T$, $q_L = [v_L, \omega_L, \beta_L,]^T$ and $b_L = [a_L, a_{\varphi,L}, v_{\beta,L}]^T$. Define the state of the leading vehicle as $\chi_L = [p_L^T, q_L^T]^T$. Then, the motion of the leading vehicle is

$$\dot{\chi}_L = f_L(\chi_L) + g_L b_L, \tag{32}$$

where $f_L(\cdot)$ is a function from $\mathbb{R}^6$ to $\mathbb{R}^6$, and $g_L \in \mathbb{R}^{6\times 3}$ is a constant matrix. They can be obtained from (31).

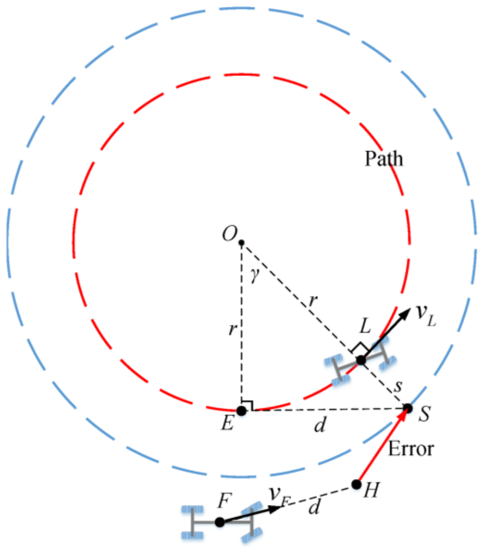

Fig. 18: The leading vehicle and the following vehicle (AV)

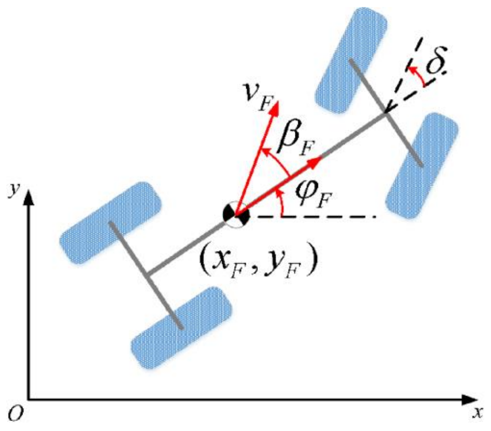

Fig. 19: Model description of the following vehicle.

2.1.2 Model Description of the Following Vehicle

The kinematics of the following vehicle is

$$\begin{aligned}\dot{x}_F &= v_F\cos(\varphi_F+\beta_F),\\ \dot{y}_F &= v_F\sin(\varphi_F+\beta_F),\\ \dot{\varphi}_F &= \omega_F.\end{aligned} \tag{33}$$

According to [60] and [61], the bicycle model of the vehicle is

$$\begin{aligned}\dot{v}_F &= u_1,\\ \dot{\omega}_F &= \lambda_1 \frac{\omega_F}{v_F} + \lambda_2 \beta_F + \lambda_3 u_2,\\ \dot{\beta}_F &= -\omega_F + \theta_1 \frac{\omega_F}{v_F^2} + \theta_2 \frac{\beta_F}{v_F} + \theta_3 \frac{1}{v_F} u_2,\end{aligned} \tag{34}$$

where λ_i and θ_i $(i=1\sim3)$ are determined by the physical parameters listed in Table 4, and their detailed expressions are presented in Appendix A. The first equation is for longitudinal motion; the second and third one are for lateral motion. The control inputs of the following vehicle are defined as

$$u_1 = a_F,\ u_2 = \delta. \tag{35}$$

The lateral dynamics can be expressed as the following compact form

$$\begin{bmatrix}\dot{\omega}_F\\ \dot{\beta}_F\end{bmatrix} = f_l(v_F, \omega_F) + g_l(v_F)\begin{bmatrix}u_2\\ \beta_F\end{bmatrix}, \tag{36}$$

where $f_l(v_F, \omega_F): \mathbb{R}^2 \to \mathbb{R}^2$ and $g_l(v_F): \mathbb{R} \to \mathbb{R}^{2\times2}$.

2.1.2 Problem Formulation

As illustrated in Fig. 18, the leading vehicle follows a defined path. By utilizing vehicle-to-vehicle (V2V) communication, the following vehicle receives the position and velocity of the leading vehicle to maintain a safe distance and remain in the same lane. Traditional approaches to solving the combined longitudinal and lateral control problem are primarily model-based, which rely on having an accurate dynamic model, as shown in equation (34). The effectiveness of these controllers heavily depends on precise knowledge of vehicle dynamics. However, in practice, physical parameters vary significantly across different autonomous vehicles (AVs), making accurate models challenging to obtain. Consequently, conventional model-based controllers may lack practical applicability. Motivated by these limitations, this paper explores a data-driven control approach to address these challenges.

Problem: Without precise knowledge of the following vehicle's dynamics, the objective is to develop a sub-optimal controller directly from data. This controller should enable vehicle F to maintain a safe distance from the leading vehicle L while following the path indicated by the red line in Fig. 18.

2.2 Tracking Error of the Following Vehicle

When the leading vehicle moves at a constant speed and the following vehicle tracks it accurately, the conditions $v_F = v_L$, $\omega_F = \omega_L$, $a_F = a_L = 0$, $a_{\phi,F} = a_{\phi,L} = 0$, and $\beta_F = v_{\beta,L} = 0$ hold. Under these circumstances, it can be seen from (34) and Fig. 18 that the following vehicle's steering angle u_2 and slip angle β_F are non-zero when driving along a curved path. Thus, a feedforward controller u_d is necessary to compute the driving input for vehicle F to maintain accurate tracking of the leading vehicle L and eliminate steady-state tracking errors. From equation (34), u_d is derived as follows:

$$\begin{aligned} u_{1,d} &= 0, \\ 0 &= f_l(v_L, \omega_L) + g_l(v_L)[u_{2,d}, \beta_{F,d}]^T, \end{aligned} \tag{37}$$

or equivalently

$$\begin{aligned} u_{1,d} &= 0, \\ [u_{2,d}(v_L, \omega_L), \beta_{F,d}(v_L, \omega_L)]^T &= -g_l^{-1}(v_L) f_l(v_L, \omega_L). \end{aligned} \tag{38}$$

It can be checked that the determinant of g_l is nonzero. Hence, g_l is invertible.

As illustrated in Fig. 18, the look-ahead point H aligns with the direction of the velocity v_F. To ensure a safe following distance, a speed-dependent inter-vehicle spacing strategy is implemented as the headway policy

$$d = d_s + v_F t_s, \tag{39}$$

where d_s represents the standstill distance and t_s is the time headway, a velocity-dependent spacing policy is used to ensure string stability of the vehicle platoon. If vehicle F directly follows vehicle L, it may cut the corner of the path. To avoid this issue, a virtual point S is defined along the line OL, and vehicle F follows S instead of L. Here, E is the desired position of vehicle F, where $ES = d$, which is tangent to the path. The triangle OES is a right triangle, and L is located on line OS such that $LS = s = \sqrt{d^2 + r^2} - r$. For vehicle F to follow L accurately along the path, it must move to point E. Once vehicle F reaches E, point S will align with the look-ahead point H. Based on the geometry of triangle OES, the angle γ and its trigonometric relations can be derived as follows

$$\begin{aligned} \gamma &= \arctan(\kappa d), \\ \sin(\gamma) &= \frac{\kappa d}{\sqrt{1+\kappa^2 d^2}}, \\ \cos(\gamma) &= \frac{1}{\sqrt{1+\kappa^2 d^2}}. \end{aligned} \tag{40}$$

The positions of the virtual point and look-ahead point in Fig. 18 are

$$
\begin{aligned}
S &= \begin{bmatrix} x_L \\ y_L \end{bmatrix} + s\begin{bmatrix} \sin(\varphi_L+\beta_L) \\ -\cos(\varphi_L+\beta_L) \end{bmatrix}, \\
H &= \begin{bmatrix} x_F \\ y_F \end{bmatrix} + d\begin{bmatrix} \cos(\varphi_F+\beta_F) \\ \sin(\varphi_F+\beta_F) \end{bmatrix}.
\end{aligned} \tag{41}
$$

To ensure that the look-ahead point H coincides with the virtual point S, it is necessary to minimize the error variables that describe the deviation between H and S. These error variables are defined as follows

$$
\begin{aligned}
e_1 &= x_L + s\sin(\varphi_L+\beta_L) - x_F - d\cos(\varphi_F+\beta_F), \\
e_2 &= y_L - s\cos(\varphi_L+\beta_L) - y_F - d\sin(\varphi_F+\beta_F), \\
e_3 &= (\varphi_L+\beta_L-\gamma) - (\varphi_F+\beta_F), \\
e_4 &= v_L - v_F, \\
e_5 &= \omega_L - \omega_F, \\
e_6 &= \beta_{F,d} - \beta_F,
\end{aligned} \tag{42}
$$

The error variables include e_1 and e_2, representing the position tracking errors along the x- and y-axes, respectively. The term e_3 is the traveling angle error, which reduces to zero when the heading direction of vehicle F aligns with ES. The variable e_4 indicates the velocity tracking error, e_5 corresponds to the angular velocity tracking error, and e_6 represents the slip angle tracking error. Additionally, e_1 and e_2 are transformed from the world frame to a local frame rotated by $\phi_F+\beta_F+\gamma$, as shown below

$$
\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} \cos(\varphi_F+\beta_F+\gamma) & \sin(\varphi_F+\beta_F+\gamma) \\ -\sin(\varphi_F+\beta_F+\gamma) & \cos(\varphi_F+\beta_F+\gamma) \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}. \tag{43}
$$

Subtracting (37) from (34) and substituting (42) into it, we have

$$
\begin{aligned}
\dot{v}_F &= u_{1,e}, \\
\dot{\omega}_F &= \lambda_1 \frac{\omega_L e_4 - v_L e_5}{v_L(v_L-e_4)} - \lambda_2 e_6 + \lambda_3 u_{2,e}, \\
\dot{\beta}_F &= e_5 + \theta_1 \frac{-v_L^2 e_5 + 2v_L\omega_L e_4 - e_4^2\omega_L}{v_L^2(v_L-e_4)^2} \\
&\quad + \theta_2 \frac{\beta_{F,d}e_4 - v_L e_6}{v_L(v_L-e_4)} + \frac{\theta_3 u_{2,d} e_4}{v_L(v_L-e_4)} + \frac{\theta_3}{v_L-e_4} u_{2,e}.
\end{aligned} \tag{44}
$$

where

$$u_e = u - u_d. \tag{45}$$

The control input $u = u_d + u_e$ consists of two components: a feedforward controller u_d, defined in equation (38), and a feedback controller u_e, which will be designed later. Let the system's error state be defined as $e = [z_1, z_2, e_3, e_4, e_5, e_6]^T$. Consequently, the error dynamics of the system can be expressed as follows

$$\dot{e} = f_e(\chi_L, e) + g_e(\chi_L, e)u_e + h_e(\chi_L, e)b_L. \tag{46}$$

Here, $f_e(\cdot,\cdot)$ is a function from $\mathbb{R}^6 \times \mathbb{R}^6$ to $\mathbb{R}^6$, $g_e(\cdot,\cdot)$ is a function from $\mathbb{R}^6 \times \mathbb{R}^6$ to $\mathbb{R}^{6\times 2}$, and $h_e(\cdot,\cdot)$ is a function from $\mathbb{R}^6 \times \mathbb{R}^6$ to $\mathbb{R}^{6\times 3}$.

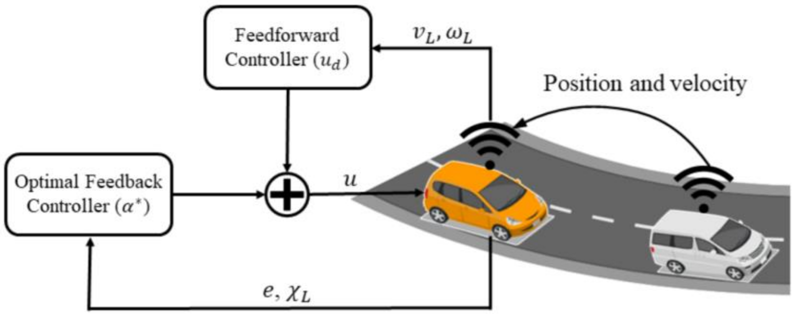
Feedforward Controller (u_d)
v_L, ω_L
Position and velocity
Optimal Feedback Controller (α^*)
u
e, χ_L

Fig. 20: The control framework of the following vehicle.

2.3 Model-Based Policy Iteration for Feedback Controller Design

As shown in Fig. 20, the control framework for the following vehicle comprises a feedforward controller and a feedback controller. The design of the feedforward controller is detailed in equation (38). Here, a model-based policy iteration method is employed to derive the optimal feedback controller $u_e^*(t) = \alpha^*(\chi_L(t), e(t))$. When the leading vehicle maintains a steady state—keeping the lane at a constant speed with $a_L = a_{\phi,L} = v_{\beta,L} = 0$—the goal is to ensure accurate tracking by the following vehicle while minimizing energy consumption. Under these conditions, the optimal control problem is defined to design the feedback controller as follows:

$$\begin{aligned} &\min_{u_e} J(\chi_L(0), e(0), u_e) = \int_0^\infty e^T Q e + u_e^T R u_e \mathrm{d}t, \\ &\text{s.t.} \quad \dot{\chi}_L = f_L(\chi_L), \\ &\qquad \dot{e} = f_e(\chi_L, e) + g_e(\chi_L, e)u_e, \end{aligned} \tag{47}$$

Here, Q and R are constant real symmetric and positive definite matrices of appropriate dimensions. Equations constraints of (47) are derived from equations (32) and (46) when $b_L = 0$. The initial conditions $\chi_L(0)$ and $e(0)$ represent the starting states of the leading vehicle and the error system, respectively. Let $\mathbb{X} \subset \mathbb{R}^6$ be a compact set that includes all the possible initial states of the leading vehicle. The set of signals $\mathcal{I}(\mathbb{X})$ represents all the trajectories of the leading vehicle, i.e., solutions $\chi_L(t) = \chi_L(t, \chi_L(0))$ of equation starting from $\chi_L(0) \in \mathbb{X}$.

The solution of the optimal control problem (47) depends on the solvability of the following HJB equation

$$\frac{\partial V^{*T}}{\partial \chi_L} f_L + \frac{\partial V^{*T}}{\partial e} f_e - \frac{1}{4}\left(\frac{\partial V^{*T}}{\partial e} g_e\right) R^{-1} \left(\frac{\partial V^{*T}}{\partial e} g_e\right)^T + e^T Q e = 0, \tag{48}$$

The optimal value function, denoted as $V^*(\chi_L, e)$, represents the minimized cost. The feedback controller that achieves the minimum of equation (47) can be derived using

$$\alpha^*(\chi_L, e) = -\frac{1}{2} R^{-1} g_e(\chi_L, e)^T \frac{\partial V^*(\chi_L, e)}{\partial e}. \tag{49}$$

From equation (49), it is evident that the optimal controller can be determined once the Hamilton-Jacobi-Bellman (HJB) equation is solved. However, the HJB equation is a nonlinear partial differential equation, making it challenging to solve directly. Moreover, it relies on an accurate dynamic model (f_e, g_e), adding to its complexity. To address this issue, a model-based policy iteration algorithm is proposed to approximate the solution iteratively. Before delving into the details of the algorithm, the following definition is introduced.

Definition 1: A controller $\alpha(\chi_L, e): \mathbb{X} \times \mathbb{R}^6 \to \mathbb{R}^3$ is considered admissible with respect to the cost J in equation (17a) if the following conditions are met:

1) α is continuous on $\mathbb{X} \times \mathbb{R}^6$.

2) For $u_e(t) = \alpha(\chi_L(t), e(t))$ and any $\chi_L \in \mathcal{I}(\mathbb{X})$, the error system defined in (17c) is globally asymptotically stable at the origin.

3) The cost function $J(\chi_L(0), e(0), u_e) < \infty$ holds for all $(\chi_L(0), e(0)) \in \mathbb{X} \times \mathbb{R}^6$.

In general, as shown in Fig. 18, an admissible controller ensures that the following vehicle converges asymptotically to the desired position E. Policy iteration starts by selecting an initial (non-optimal) admissible policy and then evaluates the value function for each state, which is referred to as *policy evaluation*. The policy is subsequently adjusted to reduce the cost, forming a sequence of improved policies. The detailed procedure is outlined in Algorithm 5. When $u_{e,i}(\chi_L, e)$ and $V_i(\chi_L, e)$ are updated using Algorithm 5, the following theorem guarantees that the resulting controller remains admissible and converges to the optimal solution.

Algorithm 5 Model-based policy iteration

```
1: Let $\alpha_1(\chi_L, e)$ be any admissible controller. Choose a sufficiently small threshold $\zeta > 0$. $i \leftarrow 1$.
2: repeat
3:     Policy evaluation: solve the following partial differential equation to obtain $V_i(\chi_L, e)$
           $\frac{\partial V_i^T}{\partial \chi_L} f_L + \frac{\partial V_i^T}{\partial e}(f_e + g_e \alpha_i) + e^T Q e + \alpha_i^T R \alpha_i = 0.$     (50)
4:     Policy improvement: update the control law to $\alpha_{i+1}(\chi_L, e)$ by
           $\alpha_{i+1}(\chi_L, e) = -\frac{1}{2} R^{-1} g_e^T \frac{\partial V_i}{\partial e}.$     (51)
5: $i \leftarrow i+1$
6: until $\mathrm{norm}\, \alpha_{i+1} - \alpha_{i\infty} \leq \zeta$
```

Lemma 1: Given that $\alpha_1(\chi_L, e)$ is admissible, and $V_i(\chi_L, e)$ and $\alpha_i(\chi_L, e)$ are obtained by solving equations (20) and (21), the following properties are satisfied:

1) $V^*(\chi_L, e) \leq V_{i+1}(\chi_L, e) \leq V_i(\chi_L, e)$ for all $(\chi_L, e) \in \mathbb{X} \times \mathbb{R}^n$.

2) $\alpha_i(\chi_L, e)$ is admissible for all $i \geq 1$.

3) $\lim_{i \to \infty} V_i(\chi_L, e) = V^*(\chi_L, e)$ and $\lim_{i \to \infty} \alpha_i(\chi_L, e) = \alpha^*(\chi_L, e)$.

As stated in Lemma 1, the performance of the updated controller $\alpha_i(\chi_L, e)$ consistently improves. Moreover, each iteration produces a stabilizing controller that converges to the optimal solution as the number of iterations approaches infinity. These attributes are crucial for ensuring the stability and effectiveness of the AV's control system. However, implementing equations (50) and (51) still relies on an accurate dynamic model (f_e, g_e). To address this limitation, the next section introduces a data-driven policy iteration algorithm that eliminates the need for an exact dynamic model.

2.4 Two-Phase Data-Driven Policy Iteration

The controller $u = u_d + u_e$ for the following vehicle consists of two components: a feedforward controller u_d and a feedback controller u_e. The feedforward controller u_d is obtained using the model parameters of the following vehicle, as defined in Equation (38), while the feedback controller u_e is designed using a policy iteration method (Algorithm 5). However, both methods depend on accurate system models for f_i and g_i (related to the system dynamics). In practice, these models are often difficult to acquire due to variations in vehicle dynamics caused by changing road conditions or environmental factors. To overcome this limitation, a two-phase data-driven approach is proposed. In the first phase, a feedforward controller $\hat{u}_d$ is designed using input-state data collected from the leading and following vehicles. The second phase then iteratively learns the optimal feedback controller $\alpha^*(\chi_L, e)$ using real-time data, bypassing the need for precise system models. The complete learning framework is illustrated in Figure 21.

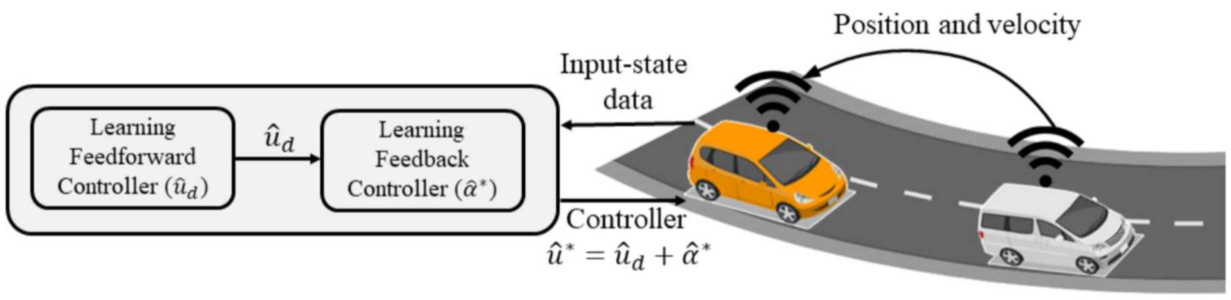
Position and velocity
Input-state data
Learning Feedforward Controller ($\hat{u}_d$)
$\hat{u}_d$
Learning Feedback Controller ($\hat{\alpha}^*$)
Controller
$\hat{u}^* = \hat{u}_d + \hat{\alpha}^*$

Fig. 21: The two-phase learning framework

2.4.1 Phase One: Design the Feedforward Controller Using Input-State Data

To obtain the feedforward controller as defined in Equation (38), it is essential to approximate the nonlinear functions f_i and g_i in Equation (36), where $f_i : \mathbb{R}^2 \to \mathbb{R}^2$ and $g_i : \mathbb{R}^2 \to \mathbb{R}^{2\times 2}$. Let $t_1 < t_2 < \ldots < t_{K+1}$ be the boundaries of the sampling intervals. By integrating Equation (36) over the interval $[t_k, t_{k+1}]$, the following expression is derived:

$$\begin{aligned}[\omega_F(t), \beta_F(t)]_{t=t_k}^{t_{k+1}} = &\int_{t_k}^{t_{k+1}} f_l(v_F(t), \omega_F(t))\mathrm{d}t \\ &+ \int_{t_k}^{t_{k+1}} [u_2(t), \beta_F(t)] \otimes I_2 \mathrm{vect}(g_l(v_F(t)))\mathrm{d}t,\end{aligned} \tag{52}$$

According to the approximation theory, over any compact set $[v_F, \omega_F]^T \in \mathcal{V} \subset \mathbb{R}^2$, f_l and g_l can be expressed as

$$\begin{aligned} f_l(v_F, \omega_F) &= \sigma_f^T \Phi_f(v_F, \omega_F) + e_f(v_F, \omega_F), \\ \mathrm{vect}(g_l(v_F)) &= \sigma_g^T \Phi_g(v_F) + e_g(v_F), \end{aligned} \tag{53}$$

where $\sigma_f \in \mathbb{R}^{2\times N_f}$ and $\sigma_g \in \mathbb{R}^{4\times N_g}$ are the weighting matrices; $\Phi_f(v_F, \omega_F)$ and $\Phi_f(v_F)$ are N_f-dimensional and N_g-dimensional vectors of linearly independent basis functions; $e_f(v_F, \omega_F)$ and $e_g(v_F)$ are the truncation errors, which uniformly converge to zero over $\mathcal{V}$ as N_f and N_g tend to infinity. Plugging (53) into (52) yields

$$\begin{aligned} [\omega_F(t), \beta_F(t)]^T|_{t=t_k}^{t_{k+1}} = &\int_{t_k}^{t_{k+1}} \Phi_f(v_F(t), \omega_F(t))^T \otimes I_2 \mathrm{d}t \,\mathrm{vect}(\sigma_f^T) \\ &+ \int_{t_k}^{t_{k+1}} \Phi_g(v_F(t))^T \otimes [u_2(t), \beta_F(t)] \otimes I_2 \mathrm{d}t \,\mathrm{vect}(\sigma_g^T) \\ &+ \int_{t_k}^{t_{k+1}} e_f(v_F(t), \omega_F(t)) + [u_2(t), \beta_F(t)] \otimes I_2 e_g(v_F(t)) \mathrm{d}t. \end{aligned} \tag{54}$$

Stacking the equations (54) for $k = 1, \cdots, K$ yields

$$\Lambda[\mathrm{vect}(\sigma_f^T)^T, \mathrm{vect}(\sigma_g^T)^T]^T + \varepsilon = \Xi, \tag{55}$$

where

$$\begin{aligned} \Lambda &= \begin{bmatrix} \int_{t_1}^{t_2} \Phi_f^T \otimes I_2 \mathrm{d}t & \int_{t_1}^{t_2} \Phi_g^T \otimes [u_2(t), \beta_F(t)] \otimes I_2 \mathrm{d}t \\ \dots & \dots \\ \int_{t_K}^{t_{K+1}} \Phi_f^T \otimes I_2 \mathrm{d}t & \int_{t_K}^{t_{K+1}} \Phi_g^T \otimes [u_2(t), \beta_F(t)] \otimes I_2 \mathrm{d}t \end{bmatrix}, \\ \Xi &= \left[[\omega_F(t), \beta_F(t)]_{t=t_1}^{t_2}, \dots, [\omega_F(t), \beta_F(t)]_{t_K}^{t_{K+1}} \right]^T, \\ \varepsilon &= \left[\int_{t_1}^{t_2} (e_f + [u_2(t), \beta_F(t)] \otimes I_2 e_g)^T \mathrm{d}t, \cdots, \int_{t_K}^{t_{K+1}} (e_f + [u_2(t), \beta_F(t)] \otimes I_2 e_g)^T \mathrm{d}t \right]^T. \end{aligned} \tag{56}$$

Assumption 1: There exist $K > 0$ and $\eta > 0$, such that the following inequality holds:

$$\frac{1}{K} \Lambda^T \Lambda \geq \eta I_{2N_f + 4N_g}. \tag{57}$$

With Assumption 1, $\hat{\sigma}_f$ and $\hat{\sigma}_g$, the estimations of σ_f and σ_g, can be obtained as the least-squares solution to (55), i.e.

$$[\mathrm{vect}(\hat{\sigma}_f^T)^T, \mathrm{vect}(\hat{\sigma}_g^T)^T]^T = (\Lambda^T \Lambda)^{-1} \Lambda^T \Xi. \tag{58}$$

Hence, the functions $f_l(v_F, \omega_F)$ and $g_l(v_F)$ over the compact set $\mathcal{V}$ can be approximated by

$$\begin{aligned} \hat{f}_l(v_F, \omega_F) &= \hat{\sigma}_f^T \Phi_f(v_F, \omega_F), \\ \mathrm{vect}(\hat{g}_l(v_F)) &= \hat{\sigma}_g^T \Phi_g(v_F). \end{aligned} \tag{59}$$

The following proposition ensures that the approximation errors $(f_l(v_F, \omega_F) - \hat{f}_l(v_F, \omega_F))$ and $(g_l(v_F) - \hat{g}_l(v_F))$ uniformly converge to zero as $N_f, N_g \to \infty$.

Proposition 1: Under Assumption 1, when $\hat{\sigma}_f$ and $\hat{\sigma}_g$ are calculated by (58), the following limits hold over the compact set $[v_F, \omega_F]^T \in \mathcal{V}$:

$$\begin{aligned} \lim_{N_f, N_g \to \infty} \mathrm{norm}\, f_l(v_F, \omega_F) - \hat{f}_l(v_F, \omega_F)_\infty &= 0, \\ \lim_{N_f, N_g \to \infty} \mathrm{norm}\, g_l(v_F) - \hat{g}_l(v_F)_\infty &= 0. \end{aligned} \tag{60}$$

Therefore, with the obtained estimations $\hat{f}_l$ and $\hat{g}_l$, according to (38), the feedforward controller can be approximated as

$$\begin{aligned} & u_{1,d} = 0, \\ & [\hat{u}_{2,d}(v_L, \omega_L), \hat{\beta}_{F,d}(v_L, \omega_L)]^T = -\hat{g}_l^{-1}(v_L) \hat{f}_l(v_L, \omega_L) \end{aligned} \tag{61}$$

Proposition 2: Under Assumption 1, when $\hat{\sigma}_F$ and $\hat{\sigma}_g$ are calculated by (58) and $[\hat{u}_{2,d}(v_L, \omega_L), \hat{\beta}_{F,d}(v_L, \omega_L)]^T$ is calculated by (61), the following limits hold over $[v_L, \omega_L]^T \in \mathcal{V}$:

$$\begin{aligned} \lim_{N_f, N_g \to \infty} \mathrm{norm}\, \hat{u}_{2,d}(v_L, \omega_L) - u_{2,d}(v_L, \omega_L)_\infty &= 0, \\ \lim_{N_f, N_g \to \infty} \mathrm{norm}\, \hat{\beta}_{F,d}(v_L, \omega_L) - \beta_{F,d}(v_L, \omega_L)_\infty &= 0. \end{aligned} \tag{62}$$

2.4.2 Phase Two: Design the Optimal Feedback Controller Using Input-State Data

Building on the model-based policy iteration approach presented in Algorithm 5, we propose a data-driven method to iteratively learn the optimal controller using motion data from both the leading and following vehicles. Define $v_i(t) = u_e(t) - \alpha_i(\chi_L(t), e(t))$, which represents the difference between the exploratory policy and the policy $\alpha_i(\chi_L, e)$ obtained at the i-th iteration of Algorithm 5. Consequently, Equation (47) can be reformulated as follows:

$$\dot{e} = f_e(\chi_L, e) + g_e(\chi_L, e)\alpha_i + g_e(\chi_L, e)\nu_i. \tag{63}$$

Along the trajectories of (63), the derivative of $V_i(\chi_L, e)$ can be expressed as

$$\dot{V}_i = \frac{\partial V_i^T}{\partial \chi_L} f_L + \frac{\partial V_i^T}{\partial e}(f_e + g_e\alpha_i + g_e\nu_i). \tag{64}$$

Recall that $V_i(\chi_L, e)$ is the value function at the i th iteration of Algorithm 5. Substituting (50) and (51) into (64), we obtain

$$\dot{V}_i = -e^T Q e - \alpha_i^T R \alpha_i - 2\alpha_{i+1}^T R \nu_i. \tag{65}$$

Integrating both sides of (65) from t_p to t_{p+1}, we have

$$V_i(t_{p+1}) - V_i(t_p) + 2\int_{t_p}^{t_{p+1}} \alpha_{i+1}^T R \hat{\nu}_i \mathrm{d}t = \int_{t_p}^{t_{p+1}} -e^T Q e - \hat{\alpha}_i^T R \hat{\alpha}_i \mathrm{d}t + \Delta_{i,p}, \tag{66}$$

where $t_1 < \cdots < t_p < t_{p+1} < \cdots < t_{P+1}$ are the boundaries of the sampling intervals. Notice that $V_i(t) = V_i(\chi_L(t), e(t))$. In addition, $\hat{\nu}_i = u - \hat{u}_d - \hat{\alpha}_i$, and

$$\Delta_{i,p} = \int_{t_p}^{t_{p+1}} 2\alpha_{i+1}^T R(u_d - \hat{u}_d + \alpha_i - \hat{\alpha}_i) + (\hat{\alpha}_i - \alpha_i)^T R(\alpha_i + \hat{\alpha}_i)\mathrm{d}t, \tag{67}$$

where $\hat{\alpha}_i$ is the approximation of α_i. By the approximation theory, V_i and α_i can be approximated by the linear combinations of a set of linearly independent basis functions, i.e.

$$\begin{aligned} V_i(\chi_L, e) &= \rho_i^T \Phi_V(\chi_L, e) + e_{V,i}(\chi_L, e), \\ \alpha_i(\chi_L, e) &= \mu_i^T \Phi_\alpha(\chi_L, e) + e_{\alpha,i}(\chi_L, e), \end{aligned} \tag{68}$$

where $\Phi_V(\chi_L, e)$ is an N_V-dimensional vector of linearly independent basis functions; $\Phi_\alpha(\chi_L, e)$ is an N_α-dimensional vector of linearly independent basis functions; $\rho_i \in \mathbb{R}^{N_V}$ and $\mu_i \in \mathbb{R}^{N_\alpha \times 2}$ are weighting matrices; $e_{V,i}(\chi_L, e) \in \mathbb{R}$ and $e_{\alpha,i}(\chi_L, e) \in \mathbb{R}^2$ are approximation truncation errors that converge to zero uniformly over any compact set $(\chi_L, e) \in \mathcal{S} \subset \mathbb{R}^6 \times \mathbb{R}^6$ as $N_V, N_\alpha \to \infty$. Plugging (68) into (66), we have

$$\rho_i^T(\Phi_V(t_{p+1}) - \Phi_V(t_p)) + 2\int_{t_p}^{t_{p+1}} \Phi_\alpha^T \mu_{i+1} R\hat{\nu}_i \mathrm{d}t = \int_{t_p}^{t_{p+1}} -e^T Q e - \hat{\alpha}_i^T R \hat{\alpha}_i \mathrm{d}t + \Delta_{i,p} + \xi_{i,p}, \quad (69)$$

where

$$\xi_{i,p} = e_{V,i}(t_p) - e_{V,i}(t_{p+1}) - 2\int_{t_p}^{t_{p+1}} e_{\alpha,i+1}^T R\hat{\nu}_i \mathrm{d}t. \quad (70)$$

Let $\Delta_i = [\Delta_{i,1}, ..., \Delta_{i,p}, ..., \Delta_{i,P}]^T$ and $\xi_i = [\xi_{i,1}, ..., \xi_{i,p}, ..., \xi_{i,P}]^T$. Then, stacking the equations (69) for $p = 1, \cdots, P$, we have

$$\Theta_i \begin{bmatrix} \rho_i \\ \mathrm{vect}(\mu_{i+1}) \end{bmatrix} = \Gamma_i + \Delta_i + \xi_i, \quad (71)$$

where

$$\Theta_i = \begin{bmatrix} (\Phi_V(t_2) - \Phi_V(t_1))^T & 2\int_{t_1}^{t_2} (\hat{\nu}_i^T R) \otimes \Phi_\alpha^T \mathrm{d}t \\ \cdots & \cdots \\ (\Phi_V(t_{P+1}) - \Phi_V(t_P))^T & 2\int_{t_P}^{t_{P+1}} (\hat{\nu}_i^T R) \otimes \Phi_\alpha^T \mathrm{d}t \end{bmatrix},$$
$$\Gamma_i = \left[-\int_{t_1}^{t_2} e^T Q e + \hat{\alpha}_i^T R \hat{\alpha}_i \mathrm{d}t, \cdots, -\int_{t_P}^{t_{P+1}} e^T Q e + \hat{\alpha}_i^T R \hat{\alpha}_i \mathrm{d}t \right]^T. \quad (72)$$

In Equation (71), we apply the property of the Kronecker product, which states that for any matrices X, Y, and Z with compatible dimensions, the expression $\mathrm{vec}(XYZ)$ can be rewritten as $(Z^T \otimes X)\mathrm{vec}(Y)$.

Assumption 2: There exist $P > 0$, and $\eta > 0$, such that for any $i \geq 1$, the following inequality holds:

$$\frac{1}{P}\Theta_i^T \Theta_i \geq \eta I_{N_V + 2N_\alpha}. \quad (73)$$

Under Assumption 2, by the least-squares method and (71), the approximations of ρ_i and μ_{i+1} are

$$\begin{bmatrix} \hat{\rho}_i \\ \text{vect}(\hat{\mu}_{i+1}) \end{bmatrix} = (\Theta_i^T \Theta_i)^{-1} \Theta_i^T \Gamma_i. \tag{74}$$

With the approximated weighting matrices $\hat{\rho}_i$ and $\text{vect}(\hat{\mu}_{i+1})$, the value function and the updated controller are

$$\begin{aligned} \hat{V}_i(\chi_L, e) &= \hat{\rho}_i^T \Phi_V(\chi_L, e), \\ \hat{\alpha}_{i+1}(\chi_L, e) &= \hat{\mu}_{i+1}^T \Phi_\alpha(\chi_L, e). \end{aligned} \tag{75}$$

The following proposition guarantees that at each iteration, the approximated value function and feedback controller achieve sufficient accuracy, provided that the number of basis functions is adequately large.

Proposition 3: For each step of data-driven policy iteration, the following properties hold:

$$\begin{aligned} \lim_{N_f, N_g, N_V, N_\alpha \to \infty} \hat{V}_i(\chi_L, e) &= V_i(\chi_L, e), \\ \lim_{N_f, N_g, N_V, N_\alpha \to \infty} \hat{\alpha}_{i+1}(\chi_L, e) &= \alpha_{i+1}(\chi_L, e), \end{aligned} \tag{76}$$

where V_i and α_{i+1} are the solutions to (50) and (51).

According to Propositions 1 and 3, the optimal feedback control policy $\alpha^*(\chi_L, e)$ and the value function $V^*(\chi_L, e)$ can be approximated to a desired accuracy by increasing the number of linearly independent basis functions and iterations. The complete data-driven policy iteration process is detailed in Algorithm 6.

Algorithm 6 Two-Phase Data-Driven Policy Iteration

```
 1: Phase-one: learning the feedforward controller
 2: Set the sampling intervals $t_1, \cdots, t_k, t_{k+1}, \cdots, t_{K+1}$.
 3: Choose driving inputs $u_{exp,1}(t)$ to explore system (36) and collect input-state data.
 4: Construct the matrices $\Lambda$ and $\Xi$ by (56).
 5: Solve (58) and get $\hat{f}_l$ and $\hat{g}_l$ by (59).
 6: Get the expression of $\hat{u}_d$ and $\hat{\beta}_{F,d}$ by (61).
 7: Phase-two: learning the optimal feedback controller
 8: Choose an initial admissible controller $\alpha_1(\chi_L, \omega_L)$.
 9: Set the sampling intervals $t_1, \cdots, t_p, t_{p+1}, \cdots, t_{P+1}$.
10: Set the small threshold $\zeta > 0$.
```

```
11: Set the exploration signals $\psi(t)$.
12: Using $u_{exp,2}(t)=\hat{u}_d(v_L(t),\omega_L(t))+\alpha_1(\chi_L(t),e(t))+\psi(t)$ to explore the system and collect
    input-state data.
13: repeat
14:     Construct the matrix $\Theta_i$ and $\Gamma_i$ by (72).
15:     Solve (74) and get the updated controller $\hat{\alpha}_{i+1}$ by (75).
16:     $i \leftarrow i+1$
17: until $|\hat{\mu}_{i+1}-\hat{\mu}_i|\leq\zeta$
18: Update controller $\hat{u}_i(\chi_L,e)=\hat{\alpha}_i(\chi_L,e)+\hat{u}_d(v_L,\omega_L)$.
```

2.5 Sumo Testing

In this section, the efficiency of the two-phase data-driven policy iteration method for combined longitudinal and lateral control is evaluated using the integrated simulation platforms SUMO and CommonRoad. The single-track model for vehicle 1 in CommonRoad is used to represent the dynamic model of the following vehicle. Table 4 lists the parameters of the following vehicle and the inter-vehicle spacing policy. The inter-vehicle spacing policy is based on the guidelines outlined in a previous work.

Table 4: Model Parameters

Parameters	Description	Value
$C_f\ [N/rad]$	Front cornering stiffness	20.90
$C_r\ [N/rad]$	Rear cornering stiffness	20.90
$M\ [kg]$	Mass of the vehicle	1226
$I_z\ [m/sec]$	Yaw inertia of the vehicle	1539
$l_f\ [m]$	Length from front axle to CoM	0.88
$l_r\ [m]$	Length from back axle to CoM	1.51
$g_a\ [m/sec^2]$	Gravitational acceleration	9.81
c	Friction coefficient	1.04
$t_s\ [sec]$	Time headway	1
$d_s\ [m]$	Standstill distance	5

2.5.1 Training Process

The initial controller α_1 for the policy iteration algorithm is constructed using a feedback linearization method, leveraging an estimated value of the vehicle's dynamic parameters. In the first phase of learning, the exploratory driving input $u_{exp,1}$ is defined as a combination of sinusoidal signals with varying frequencies, formulated as

$$[u_{exp,1}(t)]_j = c_j \cdot \sum_{l=1}^{500} \sin(w_{j,l}t), \quad j = 1,2, \tag{77}$$

where $c_1 = 0.1$ and $c_2 = 0.001$. The element $[u_{exp,1}]_j$ represents the j-th component of the input vector $u_{exp,1}(t)$, where $w_{j,l}$ is randomly selected from a uniform distribution over $[-250, 250]$. After obtaining the feedforward controller $\hat{u}_d(v_L, \omega_L)$, the exploratory input for the second phase of learning is set as

$$[u_{exp,2}(t)]_j = [\alpha_1(\chi_L(t), e(t)) + \hat{u}_d(v_L(t))]_j + m_j \cdot \sum_{l=1}^{500} \sin(w_{j,l}{}'t), \quad j = 1,2,3, \tag{78}$$

where $m_1 = 0.5$, $m_2 = 0.02$, and $w_{j,l}{}'$ is randomly sampled from the uniform distribution over $[-250, 250]$.

The performance index for the optimal feedback control design is defined as $J = \int_0^\infty e^T Q e + u_e^T R u_e \, dt$, where $Q = 10 \times I_6$ and $R = I_2$. During data collection, the leading vehicle moves in a circular path with a radius of $r = 100\,m$ and a constant speed of $v_L = 20\,m/s$. The integration interval is defined as $t_{k+1} - t_k = t_{p+1} - t_p = 0.01\,s$. The basis functions for the approximation of f_L are given by $\Phi_f(v_F, \omega_F) = [\omega_F/v_F, \omega_F, \omega_F/v_F^2]^T$. For g_L, the basis functions are $\Phi_g(v_F) = [1, 1/v_F]^T$. Define $Poly_m(x_1, \ldots, x_n)$ as the vector that includes all polynomial terms of $(x_1 + \ldots + x_n)^m$. The cosine and sine components CS_1 are represented as $[1 - \cos e_3, \sin e_3]$, and the error vector e_{456} includes $[e_4, e_5, e_6]$. The vector of basis functions for $\hat{V}_i(\chi_L, e)$ is

$$\Phi_V(\chi_L, e) = [\text{Poly}_3(e), \text{Poly}_2(e), \text{Poly}_2(e) \otimes \text{CS}_1, \text{Poly}_1(e), \text{Poly}_1(e) \otimes \text{CS}_1, e_{456}/(v_L - e_4), 1]^T.$$

The vector of basis functions for $\hat{\alpha}_i(\chi_L, e)$ is

$$\Phi_\alpha(\chi_L, e) = [\text{Poly}_2(e), \text{Poly}_1(e), \text{Poly}_1(e) \otimes \text{CS}_1, e_{456}/(v_L - e_4), 1]^T.$$

The choice of basis functions is based on the Weierstrass approximation theorem, which states that any continuous function over a compact set can be uniformly approximated by polynomials. We employ third-order polynomials of the error states e to approximate the value functions, and second-order polynomials of e to approximate the control policies. Additional terms such as $[e_4/(v_L - e_4), e_5/(v_L - e_4), e_6/(v_L - e_4)]$ and $\text{Poly}_1(e) \otimes [1 - \cos(e_3), \sin(e_3)]$ are included, inspired by the error dynamics.

The input-state data includes p_L, q_L, p_F, q_F, and u. For phase-one learning, data collected from 0 to 0.5 seconds is used, while data from 0.5 to 10.5 seconds is used for phase-two learning. Figure 22 shows the trajectories of the leading and following vehicles during data collection. Figure 23 illustrates the evolution of the control policy weights and value function. After 20 iterations in Algorithm 6, the weight update converges to the stopping criterion $|\hat{\mu}_{i+1} - \hat{\mu}_i| < 10^{-3}$. The resulting ADP controller $u_{20} = \hat{u}_d + \hat{u}_{e,20}$ is then tested across different traffic scenarios. Compared to traditional deep reinforcement learning approaches, which require extensive datasets, this proposed data-driven method achieves comparable performance using only 10.5 seconds of motion data, highlighting its data efficiency.

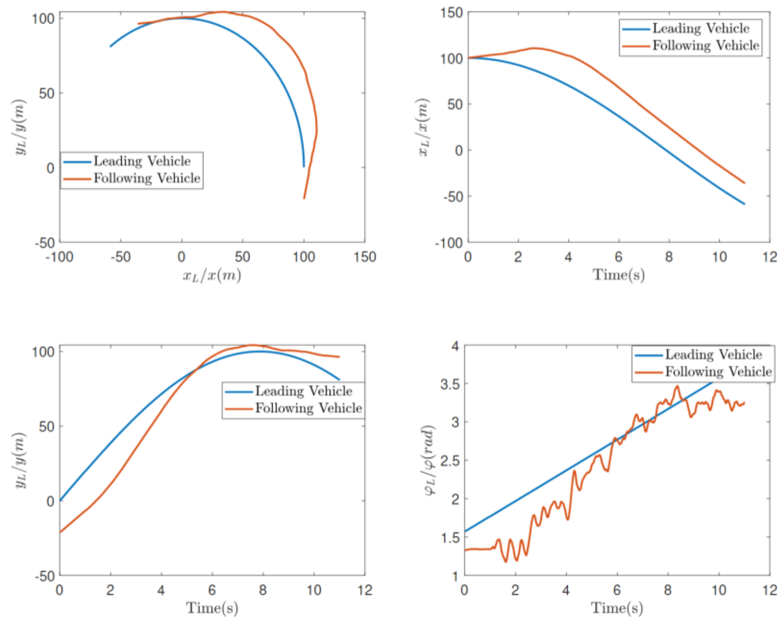

Fig. 22: The trajectory of the vehicles during data collection

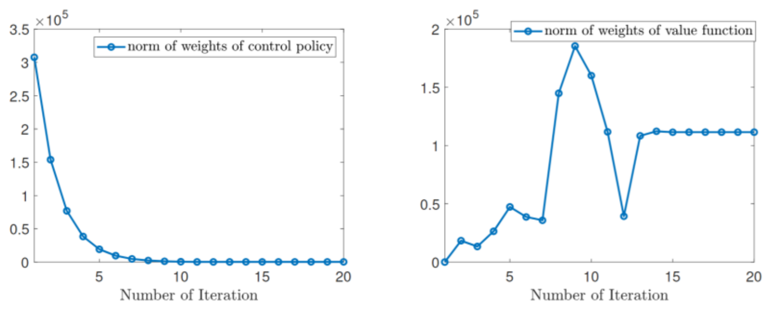

Fig. 23: Evolution of the weights ($\|\mu_i\|$ and $\|\rho_i\|$) of the control policy and value function

2.5.2 Performance Evaluation for Roads of Different Shapes

The ADP controller is initially evaluated on a circular road and compared to the baseline controller used to initialize Algorithm 6. The circular road has a center at $(50m, 50m)$ and a radius of 51.6 m. The leading vehicle maintains a constant speed of $v_L = 20\,m/s$. The trajectories and error states of both vehicles are illustrated in Fig. 24. At the start of the simulation, the following vehicle is in a separate lane from the leading vehicle. Within 5 seconds, the following vehicle aligns with the leading vehicle's lane using the ADP controller, and all tracking errors reduce to zero as seen in Fig. 24. When comparing the ADP controller to the initial controller in Fig. 24 over the time interval $t = 0\,sec$ to $t = 10\,sec$, the ADP controller reduces tracking errors faster. The performance index for the ADP controller is $J = 43.6067$, significantly lower than that of the initial controller at $J = 622.6262$. This demonstrates that Algorithm 6 yields substantial improvement in performance based on the designed quadratic performance index.

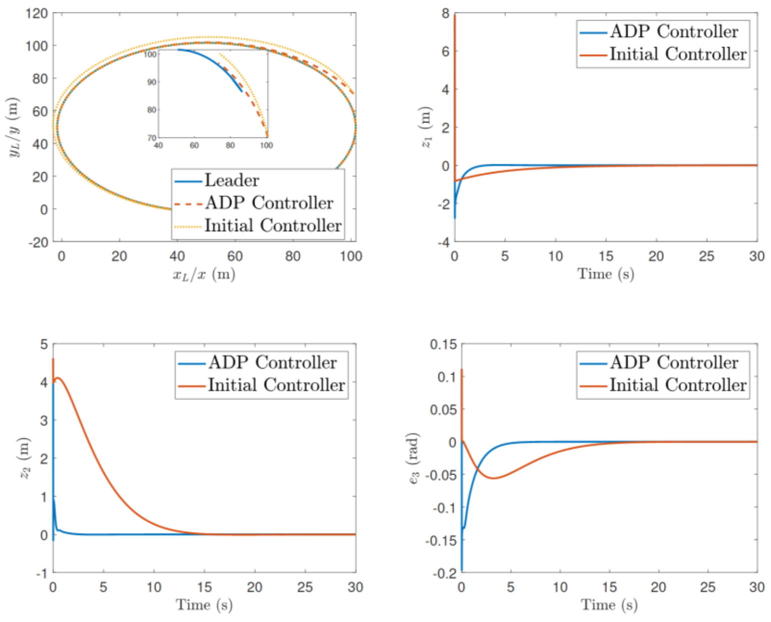

Fig. 24: The trajectories and tracking errors of the following vehicle for a circular road

The ADP controller is further tested on a complex mixed road combining straight segments and circular paths, as illustrated in Fig. 25. The larger circle has a radius of 101.6 m, while the smaller circle's radius is $51.6m$. These two circles are connected by straight tangent lines, with the bottom line extending for $300m$. The leading vehicle travels along the entire route at a constant speed of $20m/s$, starting from the left end of the bottom straight section and moving in an anticlockwise direction. The corresponding vehicle trajectories and tracking errors are displayed in Fig. 25. During the initial straight segment, the following vehicle quickly catches up to the leading vehicle and accurately maintains the lane using the ADP controller. After about 10 seconds, the leading vehicle transitions onto the smaller circle. At around 20 seconds, the leading vehicle moves from the smaller circle onto the straight section, and by 35 seconds, it reaches the larger circle. Due to sudden curvature changes at these junctions, the tracking errors momentarily spike but are rapidly corrected with the ADP controller. In contrast, the initial controller struggles to maintain zero tracking errors during these transitions due to frequent road geometry changes, resulting in gradual and inconsistent convergence. In conclusion, the proposed two-phase data-driven algorithm successfully generates a near-optimal controller for combined longitudinal and lateral control. The learned controller not only ensures stability but also significantly enhances the tracking performance of the following vehicle based on the designed performance metrics.

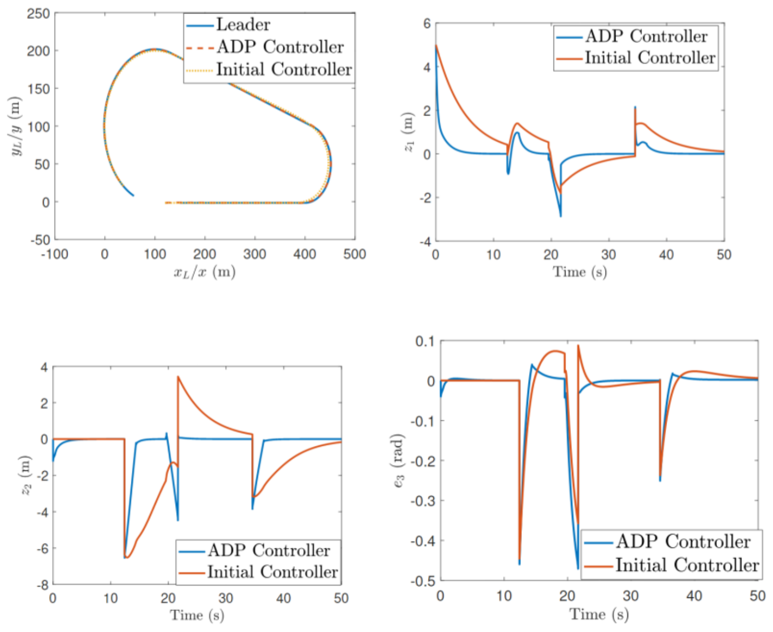

Fig. 25: The trajectories and tracking errors of the following vehicle for a mixed road

2.5.3 Analysis of String Stability on a Circle Road

In the previous subsection, the ADP controller was tested in a simplified scenario where a single autonomous vehicle (AV) followed a leading vehicle, ensuring stability as guaranteed by Lemma 1 and Proposition 3. However, when multiple AVs form a platoon, string stability is crucial to mitigate the impact of disturbances. String stability ensures that the disturbance magnitude decreases as it propagates along the vehicle string within the platoon. In this section, we numerically analyze the string stability of a homogeneous connected autonomous vehicle (CAV) platoon shown in Fig. 26. Specifically, the effect of the weighting matrices of the performance index defined in (17) on the ℓ_2-norm string stability is investigated. All AVs in Fig. 26 have identical physical parameters as listed in Table 4 and use the same controller generated by Algorithm 6. Based on [4], the definition of ℓ_2-norm string stability for two-dimensional combined longitudinal and lateral control is given as follows.

Definition 2: For a platoon consisting of N CAVs, it is strictly ℓ_2 string stable if

$$\|z_{l+1}\|_{\ell_2} \leq \|z_l\|_{\ell_2}, \quad \forall l \in \{1, 2, \cdots, N\}, \tag{79}$$

where $z_l(t) = [z_{l,1}(t), z_{l,2}(t)]^T$ is the two-dimensional tracking error of the l th CAV defined in (43), and

$$\|z_l\|_{\ell_2} = \int_0^\infty \sqrt{z_l^T(t) z_l(t)}\, \mathrm{d}t. \tag{80}$$

For the platoon shown in Fig. 26, starting from the same non-equilibrium state, the string stability is evaluated under different ADP controllers, each corresponding to a distinct weighting matrix Q. The matrices tested are $Q = 0.1 I_6$, $Q = 1 I_6$, $Q = 10 I_6$, and $Q = 100 I_6$. As illustrated in Fig. 27, for all tested Q values, the ℓ_2-norm of the system response continuously decreases, indicating that condition (79) is satisfied. This result confirms that the ADP controllers effectively ensure ℓ_2-string stability for the platoon.

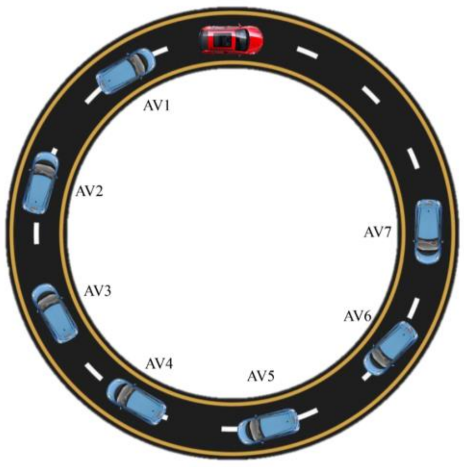

Fig. 26: A platoon consists of seven AVs and a leading vehicle on a circle road

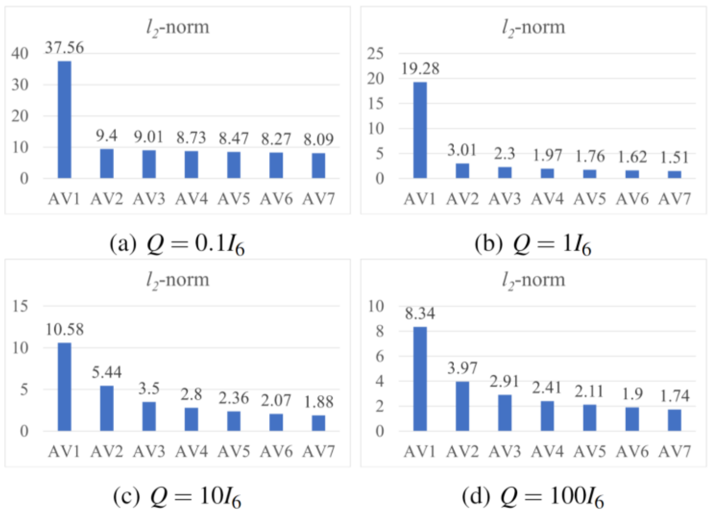

Fig. 27: The ℓ_2 norm of each AV ($\|z_l\|_{\ell_2}$, $l = 1, \cdots, 7$) under the ADP controllers

2.5.4 Robustness Evaluation

In this simulation, the robustness of the learned controller is assessed against external disturbances from the leading vehicle. Two ADP controllers with different Q matrices are evaluated to compare their robustness. Specifically, the evaluation considers $Q = 0.5I_6$ and $Q = 10I_6$ while keeping $R = I_2$ fixed. The two controllers are generated using the same procedure outlined in Subsection VI-A and evaluated on a circular road with a radius of $101.6m$. During the test, the leading vehicle's velocity oscillates as $v_L(t) = 10 + 10\sin(2t)\ m/s$. To avoid singularities in the controllers' calculation, the measured value of $v_L(t)$ is set as $0.1 \cdot \text{sign}(v_L(t))$ when $v_L(t) < 0.1\,m/s$. The resulting tracking errors and control inputs u_e are illustrated in Fig. 28. For $Q = 10I_6$, the tracking errors are reduced more effectively compared to $Q = 0.5I_6$, but this improvement comes at the cost of increased control input magnitude, as seen in Fig. 28, leading to higher energy consumption. According to the performance index, Q is associated with tracking errors and R relates to energy consumption. Thus, increasing Q prioritizes minimizing tracking errors but necessitates a trade-off with higher energy consumption.

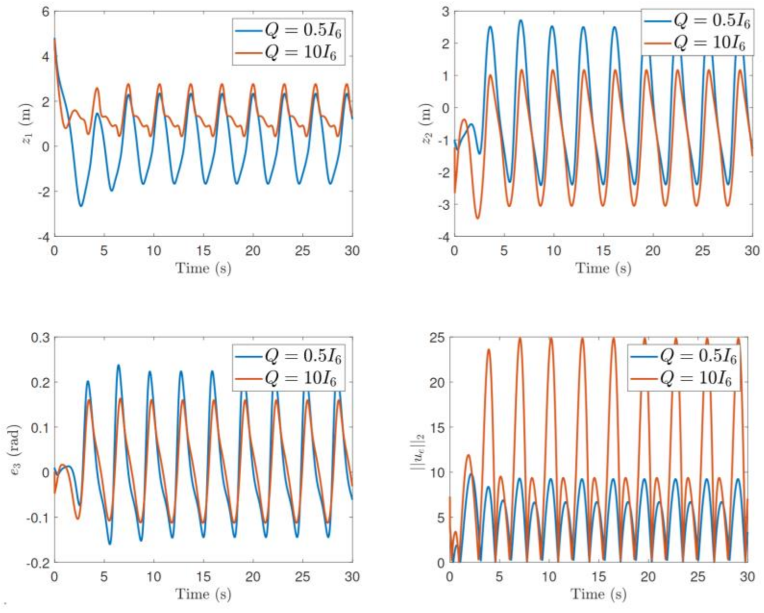

Fig. 28: The tracking errors of the following vehicle when the velocity of the leading vehicle is oscillating

2.5.5 Adaption to Different Road Conditions

This simulation evaluates the adaptiveness of the proposed data-driven control strategy under varying road conditions with different friction coefficients. The leading vehicle moves at a constant speed of $v_L(t) = 20\,m/s$ along a circular path with a radius of 100 m. Initially, the friction coefficient is set to $c = 0.9 \times 1.04$ from $t = 0\,sec$ to $t = 30\,sec$. Subsequently, the friction coefficient is reduced to $c = 0.7 \times 1.04$ for $t = 30\,sec$ to $t = 60\,sec$. Finally, the friction coefficient is further decreased to $c = 0.5 \times 1.04$. After the leading vehicle travels on the modified road conditions for 10 seconds, the data is collected, and the controller is updated using Algorithm 6. As observed from Fig. 29, with the updated controller, tracking errors converge rapidly to zero under various friction coefficients. This demonstrates that the proposed method is effective in adapting to changes in road conditions.

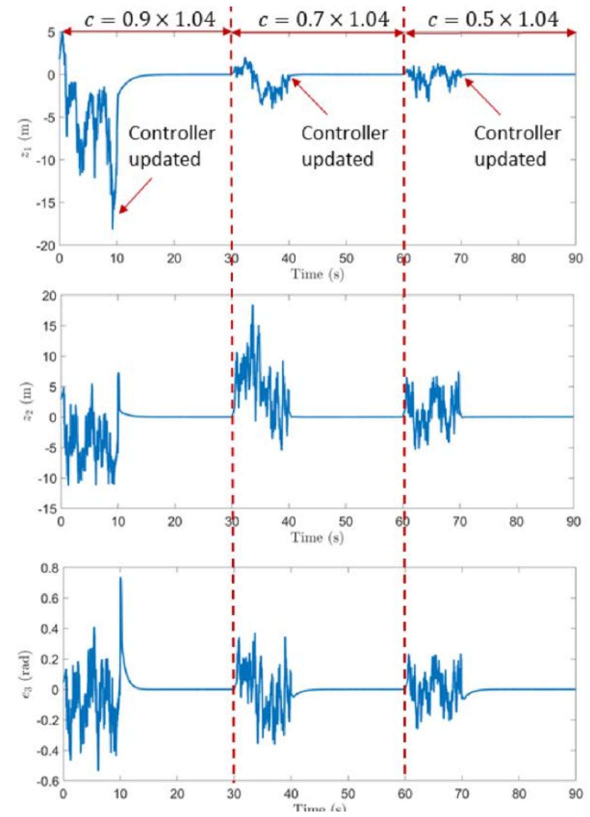

Fig. 29: The tracking errors of the following vehicle when the road condition is changing

Conclusions

This study proposed an adaptive lane-changing control strategy for automated vehicles operating in mixed traffic, focusing on safety through a combination of model-based policy iteration and data-driven adaptive dynamic programming (ADP). The approach integrates real-time state/output feedback and safety constraints to ensure safe execution of lane changes even in complex and uncertain driving environments. Experimental results using scaled car models demonstrated the method's effectiveness in maintaining stability and successfully performing lane changes while continuously evaluating safety conditions. The robustness of the proposed state feedback and output feedback schemes in achieving safe and efficient lane changes is demonstrated experimentally under varying traffic scenarios.

The research also addressed the car-following problem through a combined longitudinal and lateral control framework that utilizes a two-phase data-driven policy iteration approach. This method is designed to ensure smooth trajectory tracking and safety by using input-state data for feedforward control and a model-free policy iteration algorithm for optimal feedback control. Extensive simulations in SUMO and CommonRoad validated the controller's ability to adapt to varying road geometries and traffic conditions, demonstrating superior tracking accuracy and string stability compared to traditional approaches. The framework's adaptability to complex dynamics makes it a promising solution for vehicle platooning and other car-following scenarios.

The data-driven adaptive dynamic programming (ADP) methods used in this research provide significant advantages over traditional model-based approaches, particularly for handling both linear and nonlinear systems with model uncertainties. ADP is highly adaptable, making it suitable for complex control problems where model inaccuracies can degrade performance. By iteratively learning control policies from real-world data, the proposed framework can achieve near-optimal control performance with reduced computational complexity. This flexibility allows ADP-based controllers to maintain high control accuracy and robustness, ensuring effective performance across a range of dynamic driving scenarios.

References

[1] D. Schrank, B. Eisele, and T. Lomas, "2019 urban mobility report," Texax A&M Transportation Institute, 2019.

[2] W. Nianfeng, Ardalan Vahidi, and Andre Luckow. "Optimal speed advisory for connected vehicles in arterial roads and the impact on mixed traffic." Transportation Research Part C: Emerging Technologies 69 (2016): 548-563.

[3] B. Mikhail, M. Arcak, and A. Kurzhanskiy. "Speed Advisory System Using Real-Time Actuated Traffic Light Phase Length Prediction." arXiv preprint arXiv:2107.10372 (2021).

[4] P. Valentin, K. Wiesner, and S. Feit. "Adaptive traffic light prediction via Kalman filtering." In 2014 IEEE Intelligent Vehicles Symposium Proceedings, pp. 151-157. IEEE, 2014.

[5] P. Valentin, S. Feit, and C. Linnhoff-Popien. "Extensive traffic light prediction under real-world conditions." In 2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall), pp. 1-5. IEEE, 2014.

[6] M. Grant, and A. Vahidi. "An optimal velocity-planning scheme for vehicle energy efficiency through probabilistic prediction of traffic-signal timing." IEEE Transactions on Intelligent Transportation Systems 15, no. 6 (2014): 2516-2523.

[7] I. Shahana, D. Kalathil, R. O. Sanchez, and Pravin Varaiya. "Estimating phase duration for SPaT messages." IEEE Transactions on Intelligent Transportation Systems 20, no. 7 (2018): 2668-2676.

[8] G. Alexander, L. Ambühl, K. Yang, M. Menendez, and A. Kouvelas. "Time-to-Green predictions: A framework to enhance SPaT messages using machine learning." In 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), pp. 1-6. IEEE, 2020.

[9] G. Alexander, M. A. Makridis, K. Yang, L. Ambühl, M. Menendez, and A. Kouvelas. "Time-to-Green predictions for fully-actuated signal control systems with supervised learning." arXiv preprint arXiv:2208.11344 (2022).

[10] Geroliminis, Nikolas, and Carlos F. Daganzo. "Existence of urban-scale macroscopic fundamental diagrams: Some experimental findings." Transportation Research Part B: Methodological 42.9 (2008): 759-770.

[11] Mannering F L, Washburn S S. Principles of highway engineering and traffic analysis[M]. John Wiley & Sons, 2020.

[12] Vahidi A, Sciarretta A. Energy saving potentials of connected and automated vehicles[J]. Transportation Research Part C: Emerging Technologies, 2018, 95: 822-843.

[13] Ma, Jiaqi, et al. "Parsimonious shooting heuristic for trajectory design of connected automated traffic part II: Computational issues and optimization." Transportation Research Part B: Methodological 95 (2017): 421-441.

[14] Guo, Yi, et al. "Joint optimization of vehicle trajectories and intersection controllers with connected automated vehicles: Combined dynamic programming and shooting heuristic approach." Transportation research part C: emerging technologies 98 (2019): 54-72.

[15] Mushtaq A, Haq I U, Imtiaz M U, et al. Traffic flow management of autonomous vehicles using deep reinforcement learning and smart rerouting[J]. IEEE Access, 2021, 9: 51005-51019.

[16] Nguyen C H P, Hoang N H, Vu H L. A Joint Trajectory Planning and Signal Control Framework for a Network of Connected and Autonomous Vehicles[J]. IEEE Transactions on Intelligent Transportation Systems, 2023. DOI: 10.1109/TITS.2023.3241281.

[17] Lee J, Park B. Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment[J]. IEEE Transactions on Intelligent Transportation Systems, 2012, 13(1): 81-90.

[18] Yu C, Feng Y, Liu H X, et al. Corridor level cooperative trajectory optimization with connected and automated vehicles[J]. Transportation Research Part C: Emerging Technologies, 2019, 105: 405-421.

[19] Krajewski R, Themann P, Eckstein L. Decoupled cooperative trajectory optimization for connected highly automated vehicles at urban intersections[C]//2016 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2016: 741-746.

[20] De Persis C, Tesi P. Formulas for data-driven control: Stabilization, optimality, and robustness[J]. IEEE Transactions on Automatic Control, 2019, 65(3): 909-924.

[21] Amidi, Omead, and Chuck E. Thorpe. "Integrated mobile robot control." Mobile Robots V. Vol. 1388. SPIE, 1991.

[22] Samuel, Moveh, Mohamed Hussein, and Maziah Binti Mohamad. "A review of some pure-pursuit based path tracking techniques for control of autonomous vehicle." International Journal of Computer Applications 135.1 (2016): 35-38.

[23] Snider, Jarrod M. "Automatic steering methods for autonomous automobile path tracking." Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08 (2009).

[24] Ming, Tingyou, et al. MPC-based trajectory tracking control for intelligent vehicles. No. 2016-01-0452. SAE Technical Paper, 2016.

[25] Wang, Hengyang, et al. "Path tracking control for autonomous vehicles based on an improved MPC." IEEE Access 7 (2019): 161064-161073.

[26] Y. Rahman, M. Jankovic, and M. Santillo, “Driver intent prediction with barrier functions,” in 2021 American Control Conference (ACC). IEEE, 2021, pp. 224–230.

[27] Y. Rahman, A. Sharma, M. Jankovic, M. Santillo, and M. Hafner, “Driver intent prediction and collision avoidance with barrier functions,” IEEE/CAA Journal of Automatica Sinica, vol. 10, no. 2, pp. 365–375, 2023.

[28] A. D. Ames, J. W. Grizzle, and P. Tabuada, “Control barrier function based quadratic programs with application to adaptive cruise control,” in 53rd IEEE Conference on Decision and Control. IEEE, 2014, pp. 6271–6278.

[29] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, “Control barrier function based quadratic programs for safety critical systems,” IEEE Transactions on Automatic Control, vol. 62, no. 8, pp. 3861–3876, 2016.

[30] Y. Chen, H. Peng, and J. Grizzle, “Obstacle avoidance for low-speed autonomous vehicles with barrier function,” IEEE Transactions on Control Systems Technology, vol. 26, no. 1, pp. 194–206, 2017.

[31] M. Jankovic and M. Santillo, “Collision avoidance and liveness of multi-agent systems with cbf-based controllers,” in 2021 60th IEEE Conference on Decision and Control (CDC). IEEE, 2021, pp. 6822– 6828.

[32] M. Jankovic, M. Santillo, and Y. Wang, “Multi-agent systems with cbf-based controllers–collision avoidance and liveness from instability,” arXiv preprint arXiv:2207.04915, 2022.

[33] L. Lindemann, H. Hu, A. Robey, H. Zhang, D. Dimarogonas, S. Tu, and N. Matni, “Learning hybrid control barrier functions from data,” in Conference on Robot Learning. PMLR, 2021, pp. 1351–1370.

[34] A. Robey, L. Lindemann, S. Tu, and N. Matni, “Learning robust hybrid control barrier functions for uncertain systems,” IFAC-PapersOnLine, vol. 54, no. 5, pp. 1–6, 2021.

[35] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni, “Learning control barrier functions from expert demonstrations,” in 2020 59th IEEE Conference on Decision and Control (CDC). IEEE, 2020, pp. 3717–3724.

[36] A. Nejati, B. Zhong, M. Caccamo, and M. Zamani, “Data-driven controller synthesis of unknown nonlinear polynomial systems via control barrier certificates,” in Learning for Dynamics and Control Conference. PMLR, 2022, pp. 763–776.

[37] C. Wang, Y. Meng, Y. Li, S. L. Smith, and J. Liu, “Learning control barrier functions with high relative degree for safety-critical control,” in 2021 European Control Conference (ECC). IEEE, 2021, pp. 1459–1464.

[38] C. Folkestad, Y. Chen, A. D. Ames, and J. W. Burdick, “Data-driven safety-critical control: Synthesizing control barrier functions with Koopman operators,” IEEE Control Systems Letters, vol. 5, no. 6, pp. 2012–2017, 2020.

[39] Hoel, Carl-Johan, et al. "Combining planning and deep reinforcement learning in tactical decision making for autonomous driving." IEEE transactions on intelligent vehicles 5.2 (2019): 294-305.

[40] Li, Guofa, et al. "Lane change strategies for autonomous vehicles: A deep reinforcement learning approach based on transformer." IEEE Transactions on Intelligent Vehicles 8.3 (2022): 2197-2211.

[41] Xie, Dong-Fan, et al. "A data-driven lane-changing model based on deep learning." Transportation research part C: emerging technologies 106 (2019): 41-60.

[42] Kiran, B. Ravi, et al. "Deep reinforcement learning for autonomous driving: A survey." IEEE Transactions on Intelligent Transportation Systems 23.6 (2021): 4909-4926.

[43] Bellman, Richard. "Dynamic programming." science 153.3731 (1966): 34-37.

[44] Powell, Warren B. Approximate Dynamic Programming: Solving the curses of dimensionality. Vol. 703. John Wiley & Sons, 2007.

[45] Lewis, Frank L., and Draguna Vrabie. "Reinforcement learning and adaptive dynamic programming for feedback control." IEEE circuits and systems magazine 9.3 (2009): 32-50.

[46] Bertsekas, Dimitri. Dynamic programming and optimal control: Volume I. Vol. 4. Athena scientific, 2012.

[47] Jiang, Zhong-Ping, Tao Bian, and Weinan Gao. "Learning-based control: A tutorial and some recent results." Foundations and Trends® in Systems and Control 8.3 (2020): 176-284.

[48] Jiang, Yu, and Zhong-Ping Jiang. "Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics." Automatica 48.10 (2012): 2699-2704.

[49] Gao, Weinan, Zhong-Ping Jiang, and Kaan Ozbay. "Data-driven adaptive optimal control of connected vehicles." IEEE Transactions on Intelligent Transportation Systems 18.5 (2016): 1122-1133.

[50] Vamvoudakis, Kyriakos G., and Nick-Marios T. Kokolakis. "Synchronous reinforcement learning-based control for cognitive autonomy." Foundations and Trends® in Systems and Control 8.1–2 (2020): 1-175.

[51] Jiang, Yu, and Zhong-Ping Jiang. "Robust adaptive dynamic programming and feedback stabilization of nonlinear systems." IEEE Transactions on Neural Networks and Learning Systems 25.5 (2014): 882-893.

[52] Gao, Weinan, Zhong-Ping Jiang, and Kaan Ozbay. "Adaptive optimal control of connected vehicles." 2015 10th International Workshop on Robot Motion and Control (RoMoCo). IEEE, 2015.

[53] Gao, Weinan, et al. "Reinforcement-learning-based cooperative adaptive cruise control of buses in the Lincoln tunnel corridor with time-varying topology." IEEE Transactions on Intelligent Transportation Systems 20.10 (2019): 3796-3805.

[54] Cui, Leilei, Kaan Ozbay, and Zhong-Ping Jiang. "Combined longitudinal and lateral control of autonomous vehicles based on reinforcement learning." 2021 American control conference (ACC). IEEE, 2021.

[55] Ha, Won Yong, et al. "Automated Lane Changing Through Learning-Based Control: An Experimental Study." 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2023.

[56] Chakraborty, Sayan, et al. "Automated lane changing control in mixed traffic: An adaptive dynamic programming approach." Transportation Research Part B: Methodological 187 (2024): 103026.

[57] Won Yong Ha, Chakraborty, S., Xiaoyi Lin, Kaan Ozbay & Jiang, Z. P., "Learning-Based State Estimation for Automated Lane-Changing", 27th IEEE International Conference on Intelligent Transportation Systems (ITSC). September 24-27, 2024, Edmonton, Canada.

[58] G. Hewer, "An iterative technique for the computation of the steady state gains for the discrete optimal regulator," IEEE Transactions on Automatic Control, vol. 16, no. 4, pp. 382–384, 1971.

[59] F. L. Lewis and K. G. Vamvoudakis, "Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 41, no. 1, pp. 14–25, 2011.

[60] M. Althoff, M. Koschi, and S. Manzinger, "Commonroad: Composable benchmarks for motion planning on roads," in 2017 IEEE Intelligent Vehicles Symposium (IV), pp. 719–726, 2017.

[61] R. Rajamani, Vehicle Dynamics and Control. NY, USA: Springer, 2nd ed., 2012.