2024 IEEE 27th International Conference on
Intelligent Transportation Systems (ITSC)
September 24- 27, 2024. Edmonton, Canada

Learning-Based State Estimation for Automated Lane-Changing

Won Yong Ha[1], Sayan Chakraborty[1], Xiaoyi Lin[1], Kaan Ozbay[2] and Zhong-Ping Jiang[1,2]

2024 IEEE 27th International Conference on Intelligent Transportation Systems (ITSC) | 979-8-3315-0592-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ITSC58415.2024.10919550

Abstract— **This paper introduces a learning-based optimal control strategy enhanced with nonmodel-based state estimation to manage the complexities of lane-changing maneuvers in autonomous vehicles. Traditional approaches often depend on comprehensive system state information, which may not always be accessible or accurate due to dynamic traffic environments and sensor limitations. Our methodology dynamically adapts to these uncertainties and sensor noise by iteratively refining its control policy based on real-time sensor data and reconstructed states. We implemented an experimental setup featuring a scaled vehicle equipped with GPS, IMUs, and cameras, all processed through an Nvidia Jetson AGX Xavier board. This approach is pivotal as it addresses the limitations of simulations, which often fail to capture the complexity of dynamic real-world conditions. The results from real-world experiments demonstrate that our learning-based control system achieves smoother and more consistent lane-changing behavior compared to traditional direct measurement approaches. This paper underscores the effectiveness of integrating Adaptive Dynamic Programming (ADP) with state estimation techniques, as demonstrated through small-scale experiments. These experiments are crucial as they provide a practical validation platform that simulates real-world complexities, representing a significant advancement in the control systems used for autonomous driving.**

I. INTRODUCTION

Autonomous driving is often recognized for its potential to transform transportation systems and decrease the incidence of traffic accidents attributed to human driving errors [1]. Autonomous vehicles (AV) typically employ an array of sensors to identify nearby vehicles, pedestrians, and obstacles [2], [3], which informs their movement decisions [4]. Furthermore, mastering lane-following and lane-changing maneuvers is crucial in autonomous driving, necessitating a dependable control strategy that is both theoretically sound and empirically proven [5], [6]. This work employs a small-scale car model to test lane-changing algorithms, providing unique insights often unattainable through computer simulations alone. Unlike full-scale cars, small-scale models allow for rapid prototyping and testing in controlled environments, enabling detailed observation of dynamics that computer models might overlook [7]. This approach follows methodologies in other works (see [5], [6]), where small-scale implementations have successfully predicted behaviors in larger systems, offering a cost-effective alternative to full-scale testing.

Learning-based optimal control is a powerful tool for handling complex tasks in autonomous vehicles [8], especially for sophisticated maneuvers such as auto lane-changing. This methodology allows for a practical and adaptive control system that learns and refines its strategy based on direct interaction with the environment rather than relying solely on predefined models [9]. Adaptive Dynamic Programming (ADP) is a widely used learning-based optimal control, significantly advancing in optimizing AV lane-changing by accommodating model inaccuracies and enhancing performance in real-time [10]. The authors of [11] presented a unified framework for data-driven optimal control for which prior knowledge of model parameters was unnecessary but required an initial stabilizing control law. However, the initial admissible control policy is difficult to find in practice.

The authors of [5] explore the data-driven optimal gain-scheduling control for vehicle-lateral stability via ADP, showcasing its potential to outperform non-adaptive and model-based optimal controllers. However, the applications of ADP in autonomous driving under mixed traffic conditions generally rely on the exact full-state information [12]. Similarly, the work by [6] introduces an automated lane-changing control using an ADP approach, illustrating AADP'sutility in managing the complexities of mixed traffic environments. However, this approach requires accurate and costly sensors, and the closed-loop system performance is vulnerable to unavoidable sensor noise.

State estimation is a virtual sensor technique to solve the problem caused by lacking direct full-state measurement [13]. Instead of measuring the entire state vector, states could be reconstructed based solely on the output from the system [14]. Recent research in state estimation for autonomous driving has made significant progress in addressing complex driving scenarios, such as navigating around intersections [15] and tracking maneuvering targets under various conditions. Traditional methods, like the widely employed Kalman Filter [16]. Its nonlinear counterparts—the Extended [17] and Unscented Kalman Filters [15] —primarily address Gaussian noise and involve assumptions about system linearity or lack of stability and robustness guarantees when the system is strongly nonlinear. Although robust, these techniques can fall short in handling highly dynamic or non-Gaussian environments typical in autonomous driving.

In contrast, our approach integrates state estimation deeply into the control process, establishing a robust foundation for real-time autonomous vehicle control without requiring complete knowledge of system dynamics. It combines input-

*This work has been supported in part by a C2SMARTER grant and in part by the NSF grants CNS-2148309 and CPS-2227153.

[1]Control and Networks Lab, Department of Electrical and Computer Engineering, New York University, 370 Jay Street, Brooklyn, NY 11201, U.S.A. Email: wh784@nyu.edu, sc8804@nyu.edu, xl4708@nyu.edu, zjiang@nyu.edu

[2]C2SMARTER Center, Department of Civil and Urban Engineering, New York University, 6 MetroTech Center, Brooklyn, NY 11201, U.S.A. Email: kaan.ozbay@nyu.edu

output data with nonmodel-based formulations to estimate states, indirectly addressing uncertainties and system dynamics through estimation rather than direct sensor readings.

This paper introduces a nonmodel-based control approach that leverages state estimation techniques and a policy iteration-based output-feedback algorithm to handle systems where full-state measurements are unavailable. It utilizes sensor data to estimate the system state and employs an iterative process to refine control policies based on the estimated states. It is built on the theoretical and practical insights provided by the preceding works [6]. To demonstrate the empirical advancement of our algorithm, we utilized a remote control (RC) car and implemented our algorithm on this platform to conduct laboratory experiments.

The remainder of this paper is organized as follows: Section II outlines the design of the data-driven optimal control with state estimation. Section III details the experimental setup and evaluates the performance of the learning-based control algorithm. Section IV concludes with a summary of our findings and observations.

Notations: $\otimes$ indicates the Kronecker product, $\mathrm{vec}(T) = \left[t_1^{\mathrm{T}}, t_2^{\mathrm{T}}, \cdots, t_m^{\mathrm{T}}\right]^{\mathrm{T}}$ with $t_i \in \mathbb{R}^r$ being the columns of $T \in \mathbb{R}^{r\times m}$. For a symmetric matrix $P \in \mathbb{R}^{m\times m}$, $\mathrm{vecs}(P) = [p_{11}, 2p_{12}, \cdots, 2p_{1m}, p_{22}, 2p_{23}, \cdots, 2p_{(m-1)m}, p_{mm}]^{\mathrm{T}} \in \mathbb{R}^{(1/2)m(m+1)}$, for a column vector $v \in \mathbb{R}^n$, $\mathrm{vecv}(v) = [v_1^2, v_1v_2, \cdots, v_1v_n, v_2^2, v_2v_3, \cdots, v_{n-1}v_n, v_n^2]^{\mathrm{T}} \in \mathbb{R}^{(1/2)n(n+1)}$. For any two vectors a, b, define $\Xi_a = \left[\mathrm{vecv}(a_{k_0+1}) - \mathrm{vecv}(a_{k_0}), \cdots, \mathrm{vecv}(a_{k_s}) - \mathrm{vecv}(a_{k_s-1})\right]^{\mathrm{T}}$, $J_{a,b} = \left[a_{k_0} \otimes b_{k_0}, \cdots, a_{k_s} \otimes b_{k_s}\right]^{\mathrm{T}}$, $J_a = \left[\mathrm{vecv}(a_{k_0}), \cdots, \mathrm{vecv}(a_{k_s})\right]^{\mathrm{T}}$. $I_n(0_n)$ is the identity (zero) matrix of dimension $n \times n$.

II. LEARNING-BASED CONTROLLER DESIGN

This section presents a methodology for designing a data-driven optimal controller using only input-output data.

A. Model-based formulation

Consider the following discrete-time linear system:

$$x_{k+1} = Ax_k + Bu_k, \tag{1}$$
$$y_k = Cx_k. \tag{2}$$

where $x_k \in \mathbb{R}^n$ is the state, $u_k \in \mathbb{R}^m$ is the control input, $y_k \in \mathbb{R}^r$ is the system output. $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{r\times n}$ are constant matrices. Throughout the paper, we assume that (A,B) is controllable and (A,C) is observable. Also, it is assumed that the state x_k is not measurable. Thus, we seek to develop a control strategy via output feedback in this paper. To reduce the state deviations and control effort, we seek to design a linear optimal control law of the form:

$$u_k = -K^\star x_k, \tag{3}$$

that minimizes the following cost function:

$$\min_u \quad J = \sum_{k=0}^{\infty} (y_k^{\mathrm{T}} Q y_k + u_k^{\mathrm{T}} R u_k), \tag{4}$$

Algorithm 1 Model-based PI

1: Select an admissible control policy K_0 such that $A - BK_0$ is a Schur matrix. Initialize $j \leftarrow 0$. Select a sufficiently small threshold $\bar{\varepsilon} > 0$.
2: **repeat**
3: Policy Evaluation (Solve for P_j from):

$$A_j^T P_j A_j - P_j + C^T Q C + K_j^T R K_j = 0. \tag{7}$$

4: Policy Improvement:

$$K_{j+1} = (R + B^T P_j B)^{-1} B^T P_j A. \tag{8}$$

5: $j \leftarrow j + 1$.
6: **until** $\|P_j - P_{j-1}\| < \bar{\varepsilon}$.

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

$$A^T PA - P + C^T QC - A^T PB(R + B^T PB)^{-1} B^T PA = 0. \tag{5}$$

By the assumptions mentioned above, Eq. (5) has a unique solution $P^* = P^{*T} > 0$. The optimal feedback gain K^* can be found as follows:

$$K^* = (R + B^T P^* B)^{-1} B^T P^* A. \tag{6}$$

It should be noted that (5) is nonlinear concerning P, making direct computation of P challenging, especially in high-dimensional systems. A model-based policy iteration (PI) technique to solve (5) was presented in [18] and is reproduced in Algorithm 1. Note that $A_j = A - BK_j$ in Algorithm 1.

B. Nonmodel-based formulation

In this section, we illustrate a nonmodel-based approach for state reconstruction in terms of only u_k and y_k. The system parameters A and B are considered unknown, and the state x_k is considered unmeasurable. This work proposes a policy iteration-based online output-feedback algorithm to solve the optimal control problem. Using (1)-(2), it is possible to reconstruct x_k using input-output data as follows [19]

$$x_k = M_u \tilde{u}_{k-1,k-N} + M_y \tilde{y}_{k-1,k-N} = \Theta z_k, \tag{9}$$

where N is the observability index [13],
$z_k = [\tilde{u}_{k-1,k-N}^T,\ \tilde{y}_{k-1,k-N}^T]^T$, $\Theta = [M_u,\ M_y]$,
$M_y = A^N \mathscr{O}_N^+$, $M_u = \mathscr{C}_N - M_y \mathscr{T}_N$,
$\tilde{u}_{k-1,k-N} = [u_{k-1}^T,\ u_{k-2}^T,\ \cdots,\ u_{k-N}^T]^T$
$\tilde{y}_{k-1,k-N} = [y_{k-1}^T,\ y_{k-2}^T,\ \cdots,\ y_{k-N}^T]^T$
$\mathscr{C}_N = \begin{bmatrix} B & AB & A^2B & \cdots & A^{N-1}B \end{bmatrix}$,

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

Letting $A_j = A - BK_j$, (1) can be rewritten as

$$x_{k+1} = A_j x_k + B(u_k + K_j x_k), \tag{10}$$

Using (7), the following can be obtained

$$\begin{aligned} x_{k+1}^T P x_{k+1} - x_k^T P x_k = &- x_k^T C^T Q C x_k - x_k^T K_j^T R K_j x_k \\ &+ 2x_k^T A^T P_j B u_k + 2x_k^T A^T P_j B K_j x_k \\ &- x_k^T K_j^T B^T P_j B K_j x_k + u_k^T B^T P_j B u_k \end{aligned} \tag{11}$$

Using (9), the following can be obtained from (11)

$$\begin{aligned} z_{k+1}^T \tilde{P} z_{k+1} - z_k^T \tilde{P} z_k = &- y_k^T Q y_k - z_k^T \tilde{K}_j^T R \tilde{K}_j z_k \\ &+ 2z_k^T \Gamma_{1j}^T u_k + 2z_k^T \Gamma_{1j}^T \tilde{K}_j z_k \\ &- z_k^T \tilde{K}_j^T \Gamma_{2j} \tilde{K}_j z_k + u_k^T \Gamma_{2j} u_k \end{aligned} \tag{12}$$

where $\Gamma_{1j} = B^T P_j A \Theta$, $\Gamma_{2j} = B^T P_j B$, $\bar{P}_j = \Theta^T P \Theta$, $\tilde{K}_j = K_j \Theta$. Note that (12) does not require the full-state measurements.

By collecting the data for the time sequence $k_0 < k_1 < \cdots < k_s$, the following can be obtained from (12)

$$\Psi_j \theta_j = -J_y \mathrm{vec}(Q) - J_z \mathrm{vec}(\tilde{K}_j^T R \tilde{K}_j), \tag{13}$$

where $\Psi_j = \left[\Xi_z, -2J_{z,u} - 2J_{z,z}(I_n \otimes \tilde{K}_j^T), J_{\tilde{K}_j z} - J_u\right]$,

$\theta_j = \left[\mathrm{vecs}(\tilde{P}_j)^T, \mathrm{vec}(\Gamma_{1j})^T, \mathrm{vecs}(\Gamma_{2j})^T\right]^T$.

Assumption 2.1: There exists a $s^* \in \mathbb{Z}_+$ such that for all $s > s^*$:

$$\mathrm{rank}([J_z, J_{z,u}, J_u]) = \frac{p_1(p_1+1)}{2} + p_1 m + \frac{m(m+1)}{2}, \tag{14}$$

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

Remark 1: Notice that $s^* \geq \frac{p_1(p_1+1)}{2} + p_1 m + \frac{m(m+1)}{2}$ to guarantee the feasibility of (14).

Remark 2: Under Assumption 2.1, Ψ_j has full column rank for all $j \in \mathbb{Z}_+$ [9].

Remark 3: Note that (14) is like the persistency of excitation condition in adaptive control, which is used in previous ADP algorithms by adding an exploration signal to the input to satisfy (14) [9].

Algorithm 2 Data-Driven PI

1: Employ $u_k = -\tilde{K}_0 z_k + \eta_k$ as the input on the time interval $[k_0, k_s]$, where $\tilde{K}_0$ is an initial stabilizing control gain and η_k is the exploration/probing noise.
2: Compute Ψ_j until the rank condition in (14) is satisfied. Let $j = 0$.
3: Solve for θ_j from (13). Then, $\tilde{K}_{j+1} = (R + \Gamma_{2j})^{-1} \Gamma_{1j}$.
4: Let $j \leftarrow j+1$ and repeat Step 3 until $\|\tilde{P}_j - \tilde{P}_{j-1}\| \leq \bar{\varepsilon}$ for $j \geq 1$, where the constant $\varepsilon_0 > 0$ is a predefined small threshold.

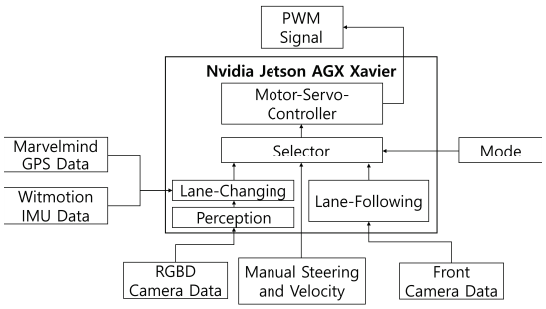

Fig. 1. The diagram illustrates the car model's software architecture, detailing data processing and flow. The Jetson board is the central hub for all computational tasks, enabling multiple data processing programs to operate simultaneously.

III. EXPERIMENTS

A. Hardware Setup[6]

We created a scaled car model for indoor testing of our lane-changing algorithm. This model performs lane changes and lane following using sensors like GPS, IMU, and a camera to collect data. Due to its compact size and efficiency, the Nvidia Jetson AGX Xavier board handles real-time data processing. Our setup aims to emulate real-world driving conditions within a controlled environment. We test the lane-changing algorithm under various scenarios to observe the vehicle's reactions to obstacles. Each experiment is repeated with slight variations to assess algorithm robustness.

The experimental car model is based on the TRX-4 RC from Traxxas Inc., known for its strong power output and precise steering. The TRX-4 chassis includes components that simulate actual vehicle mechanics and efficiently manage motor control through PWM signals. The total assembly, including sensors (0.21 kg), computing hardware (0.27 kg), batteries (1.1 kg), and accessories (0.53 kg), weighs slightly over 2.1 kg. Despite the added weight, the TRX-4 maintains maneuverability, closely replicating real car behavior.

Our car model's hardware configuration, shown in Figure 3, includes the Marvelmind Indoor GPS, which uses GPS and IMU sensors to track the car's position and orientation in real-time with an accuracy of ±2 cm. We use two beacons to enhance GPS data precision and reduce position measurement errors. The Kalman filter is applied to mitigate noise from the IMU sensor, influenced by the beacons' frequency and placement, effectively reducing high-frequency errors. A wide-angle camera is used for lane detection in our lane-following algorithm.

The RealSense D435 camera is a critical component in our self-driving vehicle setup for lane changes. It combines high-resolution RGB and depth information, making it suitable for various indoor and outdoor applications in autonomous driving. With a field of view of 85.2° horizontally, 58° vertically, and 94° diagonally, the RealSense D435 enhances spatial detection for precise depth perception and maneuvering in complex environments. It captures depth images at 1280×720 resolution at up to 90 frames per second, providing real-time environmental updates. The camera's active

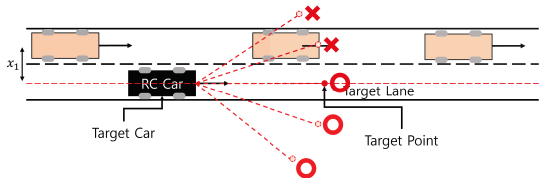

Fig. 2. Schematic of the obstacle detection and decision-making process for an autonomous vehicle (AV) during lane-changing maneuvers. The diagram illustrates the AV (RC Car) in the center lane with dotted red lines representing the perception range of the RGBD camera. Circles with checkmarks indicate clear paths, while red marks signify the presence of an obstacle.

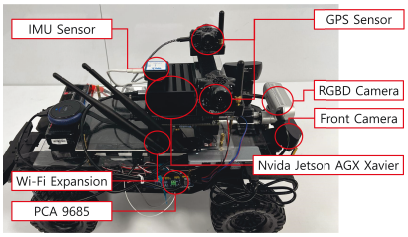

Fig. 3. The car model comes fully outfitted with diverse hardware components, all battery-powered. Careful consideration has been given to the placement of each component to guarantee maximum safety, with a particular focus on achieving optimal weight distribution.

infrared stereo technology improves depth accuracy, which is essential for reliable operation in low-light conditions, such as night driving.

Our experimental setup mounts a RealSense D435 camera on a vehicle to scan its surroundings, combining RGB and depth data to create a detailed 3D environment map. This map aids in obstacle detection and lane-change decisions by detecting and tracking static and dynamic objects like vehicles, pedestrians, and barriers. The visual and depth information classifies these entities and assesses their movement, feeding into a path-planning algorithm that determines the safest and most efficient trajectory. The vehicle proceeds with lane changes if deemed safe and necessary, constantly updating based on the environment. This approach enhances autonomous vehicle technology by enabling intelligent, proactive lane-changing decisions, improving traffic safety and fluidity.

The vehicle uses the Nvidia Jetson AGX Xavier for computational tasks and sensor data gathering, employing UDP packets to capture relevant data while ignoring the rest. Sensors continuously send data to the Jetson device, ensuring completeness and integrity. PWM signals control the vehicle's steering and acceleration, with an external PWM generator required to supply the necessary power for the TRX-4's Steering mechanism.

The car model includes a wireless joystick for manual control, enhancing experimental flexibility. We use the Logitech G29 Driving Force Racing Wheel, designed for car simulations with several programmable buttons for commands, including starting autonomous driving and reversing. This setup eliminates the need for a keyboard, making experiments more user-friendly.

At the heart of our car model lies the Nvidia Jetson

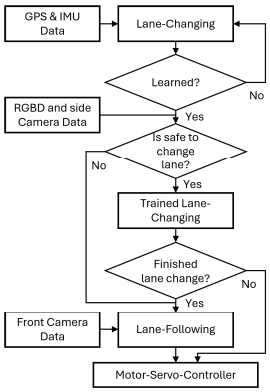

Fig. 4. This figure illustrates the basic framework of autonomous driving algorithms, each structured to activate its output to the motor solely under certain predefined scenarios.

AGX Xavier Board, a sophisticated, compact computing module. Sensors are directly hooked to the Jetson board to reduce sensor latency and improve scalability through serial connections. The integration process on the Jetson board is illustrated in Fig. 1, where three separate programs operate concurrently. Among these, the Motor-Servo-Controller program manages all motor activities, including acceleration and servo motors. It operates based on several parameters, such as steering angle, speed, and driving mode, all communicated via UDP packets. The Motor-Servo-Controller exclusively handles the processing of these parameters without interacting with the Lane-Changing or Lane-Following programs. Designed in C++, the Motor-Servo-Controller ensures smooth operation with drivers compatible with the PCA9685 module.

The Lane-Changing and Lane-Following programs feature algorithms that set the vehicle's steering angle and velocity, as outlined in prior research [20]. Inputs from dedicated sensors are fed into each program via serial connections, acting as algorithm parameters. Following data processing, the results—steering angle, speed, and the type of driving—are sent to the Motor-Servo-Controller using specific port numbers over the local network. The driving mode serves to distinguish between data sent from each program. For example, outputs from the Lane-Changing program are labeled with a driving mode of 1, whereas Lane-Following outputs are assigned a mode of 2. This labeling enables the motor-servo-controller to recognize the originating program based on the indicated driving mode.

Fig. 3 illustrates the car model featured in our experiments, with dimensions of 50cm in width, 24cm in length, and 26cm in height. To enhance safety, the steering angle, which can reach up to ± 35 degrees, is limited to 30 degrees to avoid any contact between the wheels and the car's body. The vehicle's motor, capable of a maximum output of 46W, enables speeds of up to 15km/h. Performance remains stable until the battery charge declines to below 80%.

In our car model, the lane-changing algorithm collects data over the first 100 seconds, with each time-step being 0.083 seconds, to calculate a near-optimal $\bar{K}$ value. Once

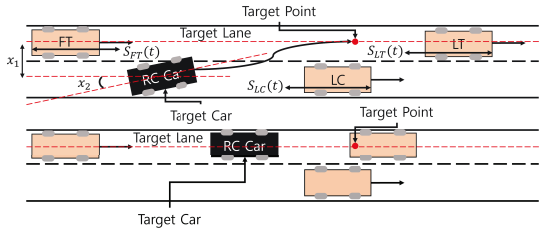

Fig. 5. The initial diagram depicts the optimal motion trajectory for our experiments. Our main aim is to attain consistent positioning within the designated lane. Therefore, our key focus lies on critical measurements: x_1, denoting the minimal distance between the target vehicle and the designated lane, and x_2, representing the orientation angle of the target vehicle. This diagram showcases two distinct scenarios during lane-changing: Change Lane to Left and Stop, arranged from top to bottom.

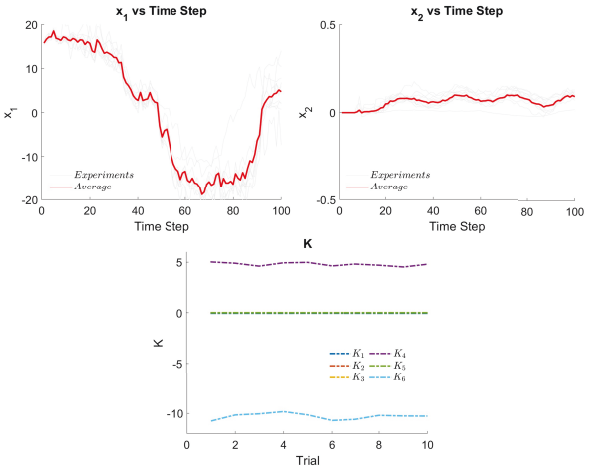

Fig. 6. Both training and testing experiments were conducted in the same environment. The state vector of the RC car, represented by the variables x_1 and x_2, measures the lateral displacement from the target lane and the orientation error in units of cm and radians, respectively. The gray line in the graph shows the raw data collected from each experiment, while the red line represents the average values across the 10 experiments. Additionally, the graph of **K** versus Trial presents the values of **K** following the training phase in each experiment.

a lane change is completed, the vehicle smoothly switches to the lane-following algorithm to maintain steady progress. This process ensures that the data flow within the experiment matches the schematic shown in Fig. 4.

B. Software

We designed a nonmodel-based control algorithm using input-output data to reconstruct the system's unmeasurable state, avoiding the need for system parameters A and B. This online output-feedback algorithm, refined through policy iteration techniques, relies solely on observed outputs and control inputs. It adapts to system dynamics by continuously updating the controller with new data. The algorithm optimizes a cost function that balances performance and control effort, ensuring efficiency and robustness for real-world applications like autonomous driving.

C. Experimental Results

TABLE I

INITIAL AND TRAINED $\bar{K}$ COMPARISON

	K_0	K_1	K_2	K_3	K_4	K_5
$\bar{\mathbf{K}}_0$	-0.057	0.187	-0.182	3.016	0.276	-10.646
$\bar{\mathbf{K}}^*$	$\mathbf{-0.050}$	$\mathbf{-.051}$	$\mathbf{-0.0011}$	**5.015**	0.018	-10.761

We conducted three experiments to evaluate our new learning-based optimal control algorithm. This method uses a data-driven approach, as detailed in Algorithm 2, leveraging both real-time state $\mathbf{x}k$ and the steering angle input δk. The safety of lane transitions for the RC car is maintained by optimizing the steering feedback gain $\mathbf{K}$ through the ADP algorithm [6]. The initial setting for the feedback gain is $\bar{\mathbf{K}}_0 = [-0.057, 0.187, -0.182, 3.016, 0.276, -10.646]$, with identity matrices serving as weights $\mathbf{Q}$ and penalties for control effort $\mathbf{R}$. The exploration noise $\boldsymbol{\eta}k$ enhances the optimization of δ_k. We constructed a two-lane road for these tests, measuring 4.5,m in length and 0.3,m in lane width. The experimental setup, depicted in Fig. 5, features an RC car and three other vehicles: the current lane leader (LC), the target lane leader (LT), and the target lane follower (FT), which act as potential hazards. Safe distances from each of these vehicles are marked as $S_{FT}(t)$, $S_{LT}(t)$, and $S_{LC}(t)$. The state vector $\mathbf{x} = [x_1, x_2]$ of the RC car indicates the lateral displacement from the target lane and orientation error.

D. Analysis and Discussion

The graph in Fig. 6 shows that the control gain vector $\mathbf{K}$, including components K_1 to K_6, remains stable across trials, with slight variations due to sensor noise, demonstrating the robustness of our method. The RC car consistently achieved smoother lane changes using the learned control gains $\bar{\mathbf{K}}$, confirming its ability to effectively learn and apply the optimal control strategy iteratively.

Testing in two scenarios showed the RC car's capability to adapt to changing conditions by maintaining its path, executing lane changes, or stopping as needed. These results highlight the effectiveness of the control algorithms in various traffic situations.

1) Scenario 1: Lane change to empty lane: If the depth at the center is less than $(1m)$, indicating an obstacle in the vehicle path, and the depth on the left exceeds $(1.5m)$, indicating a clear adjacent lane, the system activates a lane change. A UDP signal is sent to the vehicle's control system to commence a safe lane change maneuver.

2) Scenario 2: Emergency Stop: Fig. 8 presents data from a complex scenario in the final test. Initially, the RC car encounters an obstruction in its current lane, prompting a lane change to the target lane, as indicated by the spike in (x_1). However, after the lane change, the RC car finds obstructions and decides to stop. This decision is highlighted by the sharp decrease in (x_1) and the stabilization in the target point graph, where the target lane signal drops to -1, signaling the stop command.

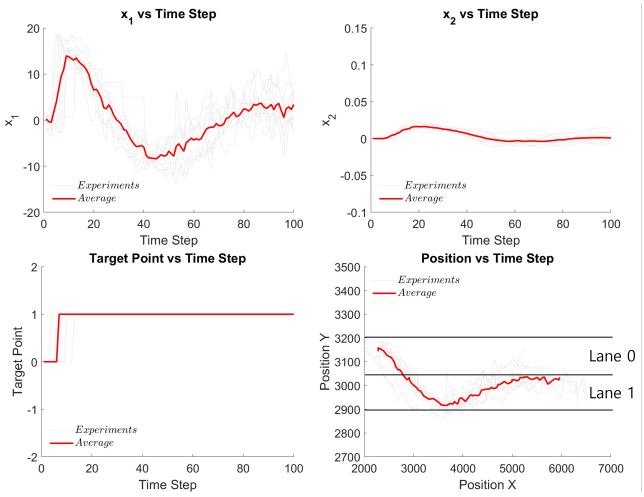

Fig. 7. The graph displays the outcome of scenario 1, illustrating the necessary lane change.

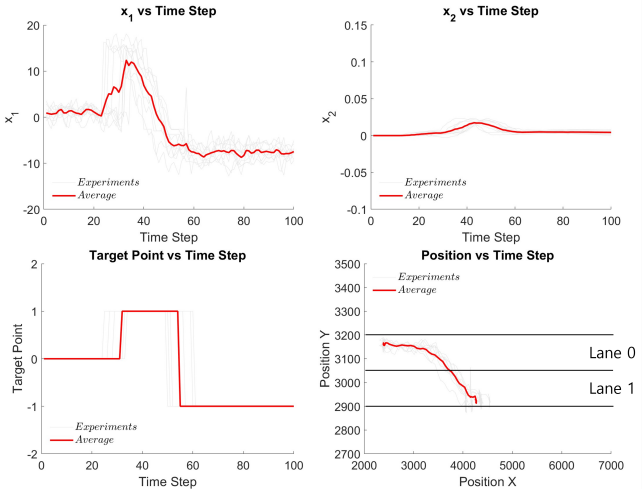

Fig. 8. The graph displays the outcome of scenario 2, illustrating the emergency stop.

IV. CONCLUSION

We presented a learning-based control system with nonmodel-based state estimation for autonomous lane-changing. This system adapts to uncertainties and sensor noise, enhancing precision and stability in complex traffic. Using a scaled vehicle equipped with GPS, IMU, and cameras processed by an Nvidia Jetson AGX Xavier board, we demonstrated the robustness of our approach. This small-scale model offers a flexible, safe, and cost-efficient method for testing in varied conditions, which are hard to simulate, particularly regarding sensor integration and latency issues. Our results show that the RC car exhibited smoother and more consistent lane-changing behavior than traditional methods, with better vehicle dynamics handling and reduced noise-induced errors. Future work will expand testing environments and refine algorithms, using a broader range of sensory data to improve decision-making accuracy.

REFERENCES

[1] J.-F. Bonnefon, *The Car That Knew Too Much: CAN A MACHINE BE MORAL?* The MIT Press, 2021.

[2] D. J. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh, "Sensor and sensor fusion technology in autonomous vehicles: A review," *Sensors*, vol. 21, no. 6, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/6/2140

[3] H. Min, X. Wu, C. Cheng, and X. Zhao, "Kinematic and dynamic vehicle model-assisted global positioning method for autonomous vehicles with low-cost gps/camera/in-vehicle sensors," *Sensors*, vol. 19, no. 24, p. 5430, 2019.

[4] M. Huang, M. Zhao, P. Parikh, Y. Wang, K. Ozbay, and Z.-P. Jiang, "Reinforcement learning for vision-based lateral control of a self-driving car," in *2019 IEEE 15th International Conference on Control and Automation (ICCA)*. IEEE, 2019, pp. 1126–1131.

[5] S. Chakraborty, L. Cui, K. Ozbay, and Z.-P. Jiang, "Automated lane changing control in mixed traffic: An adaptive dynamic programming approach," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022, pp. 1823–1828.

[6] W. Y. Ha, S. Chakraborty, Y. Yu, S. Ghasemi, and Z.-P. Jiang, "Automated lane changing through learning-based control: An experimental study," in *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2023, pp. 4215–4220.

[7] X. Xiao, J. Biswas, and P. Stone, "Learning inverse kinodynamics for accurate high-speed off-road navigation on unstructured terrain," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 6054–6060, 2021.

[8] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2022.

[9] Z.-P. Jiang, Bian, Tao, and W. Gao, "Learning-based control: A tutorial and some recent results," *Foundations and Trends® in Systems and Control*, vol. 8, no. 3, pp. 176–284, 2020.

[10] F.-Y. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: An introduction," *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, pp. 39–47, 2009.

[11] T. Liu, L. Cui, B. Pang, and Z.-P. Jiang, "A unified framework for data-driven optimal control of connected vehicles in mixed traffic," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 8, pp. 4131–4145, 2023.

[12] P. Ioannou and B. Fidan, *Adaptive Control Tutorial*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2006. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9780898718652

[13] 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.

[14] S. Agarwal, S. Mustavee, J. Contreras-Castillo, and J. Guerrero-Ibañez, "Chapter 20 - sensing and monitoring of smart transportation systems," in *The Rise of Smart Cities*, A. H. Alavi, M. Q. Feng, P. Jiao, and Z. Sharif-Khodaei, Eds. Butterworth-Heinemann, 2022, pp. 495–522.

[15] X. Li, L. Guvenc, and B. Aksun-Guvenc, "Vehicle state estimation and prediction for autonomous driving in a round intersection," *Vehicles*, vol. 5, no. 4, pp. 1328–1352, 2023.

[16] Y. Li and J. Ibanez-Guzman, "Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems," *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 50–61, 2020.

[17] B. Xia, C. Song, and B. Wu, "An integrated state space partition and optimal control method of multi-model for nonlinear systems with state estimation," in *The 27th Chinese Control and Decision Conference (2015 CCDC)*, 2015, pp. 1604–1609.

[18] 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.

[19] 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, 2010.

[20] T. Liu, L. Cui, B. Pang, and Z.-P. Jiang, "A unified framework for data-driven optimal control of connected vehicles in mixed traffic," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 8, pp. 4131–4145, 2023.