2023 IEEE 26th International Conference on
Intelligent Transportation Systems (ITSC)
24-28 September 2023. Bilbao, Bizkaia, Spain

Automated Lane Changing through Learning-Based Control: An Experimental Study*

Won Yong Ha[1], Sayan Chakraborty[1], Yujie Yu[1], Samin Ghasemi[1] and Zhong-Ping Jiang[1]

2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC) | 979-8-3503-9946-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/ITSC57777.2023.10422053

Abstract— **This paper presents a learning-based methodology for developing an optimal lane-changing control policy for a Remote Controlled (RC) car using real-time sensor data. The RC car is equipped with sensors including GPS, IMU devices, and a camera integrated in an Nvidia Jetson AGX Xavier board. By a novel Adaptive Dynamic Programming (ADP) algorithm, our RC car is capable of learning the optimal lane-changing strategies based on the real-time processed measurement from the sensors. The experimental outcomes show that our learning-based control algorithm can be effectively implemented, adapt to parameter changes, and complete the lane changing tasks in a short learning time with satisfactory performance.**

I. INTRODUCTION

Autonomous driving is widely regarded as having the potential to revolutionize transportation systems and reduce traffic accidents caused by improper human driving behaviors [1], [2], [3]. Typically, autonomous vehicles use a variety of sensors to detect other vehicles, pedestrians, and obstacles around them [4], [5] and make decisions on vehicles' movement [6]. Also, lane-following and lane-changing are important topics in the autonomous driving, and a reliable control strategy supported by theoretical analysis and experimental validation is needed.

The existing lane-changing methods can be classified into three categories: trajectory planning [7], [8], factory assessment-based methods [9], and learning-based methods [7], [10], [11], [12]. The classical approaches to trajectory planning rely on optimization methods to generate a nonlinear program, considering vehicle dynamics and obstacle avoidance requirements. Besides, the emerging technology of environmental sensing and vehicle-to-vehicle (V2V) communication, vehicle-to-infrastructure (V2I) communication enables the cooperative trajectory planning of lane changes for connected and automated vehicles (CAVs) to improve the efficiency and stability of traffic [7]. In [8], Wang et al. propose a Deep Q-Network (DQN) algorithm with rule-based constraints for lane change. Through the combination of high-level lateral control and low-level longitudinal trajectory planning, a safe and efficient lane change behavior can be achieved. With the appropriate setting of the states and rewards, the trained agent completes the lane changing tasks in a real-world-like simulator.

However, the uncertainties of vehicle dynamics are not considered in the above methods, which can cause the degradation of driving performance, for example, occurrence of unsafe driving distance, in practice. To overcome this issue, factory assessment-based methods usually adopt the following two steps to guarantee driving safety: 1) assess the benefit, safety, and tolerance of the current driving state, 2) find a sequential actions based on the factory assessment rules to maintain safety. Liu et al. of [9] have established an autonomous lane-changing method considering benefits, safety, and tolerance based on driver's habits. They also collect data using Next Generation SIMulation Fact Sheet (NGSIM) and demonstrate the performance of their algorithms through computer simulation experiments [13].

A limitation of the above methods is that only expert knowledge is utilized to generate rule-based decisions. Due to the complexity of real traffic and road conditions, the feasibility of the model needs to be further validated. Data-driven methods, for example, reinforcement learning, do not rely on the knowledge of prescribed models, but instead, try to learn the optimal decisions through the interactions with the environment. In [10], Xie et al. propose a data-driven lane-changing method based on deep learning, combining Deep Belief Network (DBN) and Long Short-Term Memory (LSTM) neural network to make lane-changing decisions based on the relative position of the neighboring vehicles in the target lane. The authors of [5] have proposed a hybrid neural network to predict lane changing behavior. The data-driven methods are usually lack of theoretical analysis due to the unknown learning mechanism, instead, adaptive dynamic programming, which is developed from the optimal control theory but requires no knowledge of models, is recently developed for different control tasks with rigorous theoretical guarantees [14].

Liu et al. and Cui et al have built controllers using ADP method to lane changing [15], [16]. While the efficacy of ADP algorithms has been validated via simulation results, there is still a gap between theory and practice. This is because there can be multiple sources of uncertainties in data collection and system modeling. The ADP algorithms will be more convincing when they can be implemented and validated in a high-fidelity semi-physical experiments.

To validate its effectiveness, we implement the optimal data-driven controller for lane-changing and conduct experiments by building an RC car that is similar to real car. Using an RC car allows us to collect data mimicking autonomous driving of a full-size vehicle, while dramatically lowering experimental costs, conducting short experiments, and more importantly reducing the safety concern of field experiments.

The rest of this paper is structured as follows: Section II

*This work has been supported in part by the NSF grants CNS- 2148309 and CNS-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

describes the data-driven control design. Section III presents our experimental preparation and performance analysis of the designed learning-based control algorithm for lane changing. Finally, section IV closes the paper with concluding remarks.

Notations: Throughout this paper, $\mathbb{Z}_+$ denotes the set of non-negative integers, $\|.\|$ represents the spectral norm of matrices, $\sigma(\mathbf{W})$ is the complex spectrum of $\mathbf{W}$, $\otimes$ indicates the Kronecker product, $\text{vec}(\mathbf{T}) = \left[t_1^T, t_2^T, \cdots, t_m^T\right]^T$ with $t_i \in \mathbb{R}^r$ being the columns of $\mathbf{T} \in \mathbb{R}^{r\times m}$. For a symmetric matrix $\mathbf{P} \in \mathbb{R}^{m\times m}$, $\text{vecs}(\mathbf{P}) = [p_{11}, 2p_{12}, \cdots, 2p_{1m}, p_{22}, 2p_{23}, \cdots, 2p_{(m-1)m}, p_{mm}]^T \in \mathbb{R}^{(1/2)m(m+1)}$, for a column vector $v \in \mathbb{R}^n$, $\text{vecv}(v) = [v_1^2, v_1v_2, \cdots, v_1v_n, v_2^2, v_2v_3, \cdots, v_{n-1}v_n, v_n^2]^T \in \mathbb{R}^{(1/2)n(n+1)}$. $\mathbf{I}_n$ is the identity matrix of dimension n.

II. Data-driven controller design

A. Preliminaries results

Consider the following discrete-time linear system:

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

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state, $\mathbf{u}_k \in \mathbb{R}^m$ is the control input, $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times m}$. It is assumed that that system is stabilizable. In order to reduce the state deviations and control effort, we seek to design a linear optimal control law of the form:

$$\mathbf{u}_k = -\mathbf{K}\mathbf{x}_k, \tag{2}$$

that can minimize the following cost function:

$$\min_{\mathbf{u}} \quad J = \sum_{k=0}^{\infty} (\mathbf{x}_k^{\mathrm{T}}\mathbf{Q}\mathbf{x}_k + \mathbf{u}_k^{\mathrm{T}}\mathbf{R}\mathbf{u}_k), \tag{3}$$

where $\mathbf{Q} = \mathbf{Q}^T \geq 0$, $\mathbf{R} = \mathbf{R}^T > 0$, and $(\mathbf{A}, \sqrt{\mathbf{Q}})$ is observable. If $\mathbf{A}$, $\mathbf{B}$ 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:

$$\mathbf{A}^T\mathbf{P}\mathbf{A} - \mathbf{P} + \mathbf{Q} - \mathbf{A}^T\mathbf{P}\mathbf{B}(\mathbf{R} + \mathbf{B}^T\mathbf{P}\mathbf{B})^{-1}\mathbf{B}^T\mathbf{P}\mathbf{A} = \mathbf{0}. \tag{4}$$

By the assumptions mentioned above, Eq. (4) has an unique solution $\mathbf{P}^* = \mathbf{P}^{*T} \geq \mathbf{0}$. Then, the optimal feedback gain $\mathbf{K}^*$ can be determined as follows:

$$\mathbf{K}^* = (\mathbf{R} + \mathbf{B}^T\mathbf{P}^*\mathbf{B})^{-1}\mathbf{B}^T\mathbf{P}^*\mathbf{A}. \tag{5}$$

Note that, (4) is nonlinear in $\mathbf{P}$. Thus, it is usually difficult to directly solve (4) especially for high-dimensional systems. A model-based policy interation (PI) technique to solve (4) was presented in [17] and is reproduced in Algorithm 1. Note that $\mathbf{A}_j = \mathbf{A} - \mathbf{B}\mathbf{K}_j$ in Algorithm 1.

B. Data-driven formulation

Here, we present an online data-driven learning-based controller design strategy that does not assume the exact knowledge of the system matrices $\mathbf{A}$ and $\mathbf{B}$. Consider the modified system equation as follows:

$$\mathbf{x}_{k+1} = \mathbf{A}_j\mathbf{x}_k + \mathbf{B}(\mathbf{u}_k + \mathbf{K}_j\mathbf{x}_k). \tag{8}$$

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 $\varepsilon > 0$.
2: **repeat**
3: Policy Evaluation (Solve for $\mathbf{P}_j$ from):

$$\mathbf{A}_j^T\mathbf{P}_j\mathbf{A}_j - \mathbf{P}_j + \mathbf{Q} + \mathbf{K}_j^T\mathbf{R}\mathbf{K}_j = \mathbf{0}. \tag{6}$$

4: Policy Improvement:

$$\mathbf{K}_{j+1} = (\mathbf{R} + \mathbf{B}^T\mathbf{P}_j\mathbf{B})^{-1}\mathbf{B}^T\mathbf{P}_j\mathbf{A}. \tag{7}$$

5: $j \leftarrow j + 1$.
6: **until** $\|\mathbf{P}_j - \mathbf{P}_{j-1}\| < \varepsilon$.

Along the trajectories of (8), one can obtain that

$$\begin{aligned}&\mathbf{x}_{k+1}^T\mathbf{P}_j\mathbf{x}_{k+1} - \mathbf{x}_k^T\mathbf{P}_j\mathbf{x}_k \\ &\quad = \left[\mathbf{A}_j\mathbf{x}_k + \mathbf{B}(\mathbf{u}_k + \mathbf{K}_j\mathbf{x}_k)\right]^T\mathbf{P}_j\left[\mathbf{A}_j\mathbf{x}_k + \mathbf{B}(\mathbf{u}_k + \mathbf{K}_j\mathbf{x}_k)\right] \\ &\qquad - \mathbf{x}_k^T\mathbf{P}_j\mathbf{x}_k. \end{aligned} \tag{9}$$

Then, using (6) we have:

$$\begin{aligned}&\mathbf{x}_{k+1}^T\mathbf{P}_j\mathbf{x}_{k+1} - \mathbf{x}_k^T\mathbf{P}_j\mathbf{x}_k + \mathbf{x}_k^T\mathbf{Q}_j\mathbf{x}_k \\ &= 2\mathbf{x}_k^T\mathbf{A}^T\mathbf{P}_j\mathbf{B}\mathbf{u}_k + 2\mathbf{x}_k^T\mathbf{A}^T\mathbf{P}_j\mathbf{B}\mathbf{K}_j\mathbf{x}_k - \mathbf{x}_k^T\mathbf{K}_j^T\mathbf{B}^T\mathbf{P}_j\mathbf{B}\mathbf{K}_j\mathbf{x}_k \\ &\qquad + \mathbf{u}_k^T\mathbf{B}^T\mathbf{P}_j\mathbf{B}\mathbf{u}_k \end{aligned} \tag{10}$$

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

$$\begin{aligned}&\left[(\mathbf{x}_{k+1}^T \otimes \mathbf{x}_{k+1}^T) - (\mathbf{x}_k^T \otimes \mathbf{x}_k^T)\right]\text{vec}(\mathbf{P}_j) + (\mathbf{x}_k^T \otimes \mathbf{x}_k^T)\text{vec}(\mathbf{Q}_j) \\ &\quad = \left[2(\mathbf{x}_k^T \otimes \mathbf{u}_k^T) + 2(\mathbf{x}_k^T \otimes \mathbf{x}_k^T)(\mathbf{I}_n \otimes \mathbf{K}_j^T)\right]\text{vec}(\mathbf{B}^T\mathbf{P}_j\mathbf{A}) + \\ &\qquad \left[-(\mathbf{K}_j\mathbf{x}_k)^T \otimes (\mathbf{K}_j\mathbf{x}_k)^T + (\mathbf{u}_k^T \otimes \mathbf{u}_k^T)\right]\text{vec}(\mathbf{B}^T\mathbf{P}_j\mathbf{B}). \end{aligned} \tag{11}$$

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

$$\mathbf{\Psi}_j\boldsymbol{\theta}_j = -\mathbf{I}_{\mathbf{x},\mathbf{x}}\text{vec}(\mathbf{Q}_j), \tag{12}$$

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

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

$$\text{rank}([\mathbf{I}_{\mathbf{x},\mathbf{x}}, \mathbf{I}_{\mathbf{x},\mathbf{u}}, \mathbf{I}_{\mathbf{u},\mathbf{u}}]) = \frac{n(n+1)}{2} + nm + \frac{m(m+1)}{2}. \tag{13}$$

Remark 1: A choice of $s^* \geq \frac{n(n+1)}{2} + nm + \frac{m(m+1)}{2}$ to guarantee the feasibility of (13).

Remark 2: Under Assumption 2.1, $\mathbf{\Psi}_j$ has full column rank for all $j \in \mathbb{Z}_+$ [14].

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 $\boldsymbol{\Delta}_{\mathbf{x},\mathbf{x}}, \mathbf{I}_{\mathbf{x},\mathbf{x}}, \mathbf{I}_{\mathbf{x},\mathbf{u}}, \mathbf{I}_{\mathbf{u},\mathbf{u}}$ until the rank condition in (13) is satisfied. Let $j = 0$.
3: Solve for $\boldsymbol{\theta}_j$ from (12). 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 $\|\mathbf{P}_j - \mathbf{P}_{j-1}\| \leq \varepsilon_0$ for $j \geq 1$, where the constant $\varepsilon_0 > 0$ is a predefined small threshold.

Remark 3: Note that (13) is like persistent of excitation in adaptive control. Like other ADP algorithms, an exploration/probing noise is added to the input to satisfy (13) [14].

III. EXPERIMENTS

A. Car Model

We build a compact car model to analyze and validate the performance of algorithm designed for lane change. In fact, the car model is designed not only for lane-changing but also for lane-following. The car model is attached several sensors such as GPS, IMU, and camera to collect as much data as and possible. An Nvidia Jetson AGX Xavier board, small, lightweight computer, is deployed to our car model while having the ability to process the data created by sensors in real-time. Therefore, suitable sensors for the experiment and computational module are carefully selected to be compatible with the experimental platform.

Our car model uses Traxxas Inc's TRX-4 RC as its base body. The TRX-4 car has high power, delicate steering angle adjustment, and various additional parts such as differential gears are built-in so that it works close to an actual car [18]. And the TRX-4 can control all the attached motors with PWM signals. The total weight of sensors (GPS, IMU, and camera: $0.21kg$), computers (Jetson board: $0.27kg$), batteries ($1.1kg$), etc ($0.53kg$), which we must attach to the car, is over $2.1kg$. The TRX-4 can perform delicate movements even after attaching all of these parts and produce a trajectory similar to a real car.

TABLE I
SENSOR SPECIFICATIONS

Sensor	Brand	Purpose
GPS	Marvelmind Indoor GPS	Location
IMU	Marvelmind Indoor GPS	Orientation
Camera	NexiGo N980P	Vision

The hardware of our car model is connected shown in Fig. 1. We use Marvelmind's Indoor GPS devices, which has GPS and IMU sensors to collect real-time car position and orientation data. Marvelmind's Indoor GPS has an error of $\pm 2cm$ and has a significant data disturbance due to nearby obstruction [19]. Therefore, we use two beacons to minimize data errors in GPS and improve data accuracy. Also, we use a Kalman filter to reduce the noise of the IMU sensor [20]. The error of the IMU sensor is highly dependent on the frequency and location of the beacon. Therefore, filtering is necessary to reduce high-pitching errors. We also use wide-angle camera to detect the lane for the lane-following algorithm. All the sensors specification are shown table I.

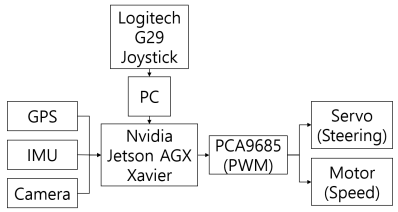

Fig. 1. The diagram above shows the hardware connection of the car model through the flow of data. The Jetson board receives data from multiple sensors in real-time and can process the necessary data.

The car model uses Nvidia's Jetson AGX Xavier for computing and vccollecting data from all sensors [21]. The Jetson board receives all data using UDP packages, which processes the necessary data and discards the unrelated data [22]. All sensors are designed to send data to the Jetson board even if it is not currently needed, making it easy to ensure the integrity of data. PWM signals control the steering and acceleration of the TRX-4. Therefore, to obtain the steering angle and speed we need, the Jetson board must generate a PWM signal and send it to the motor. However, the PWM signal to operate the TRX-4 steering servomotor requires $2.5W$, and the Jetson board's own PWM generator cannot give enough power to operate it. Therefore, we should use an external PWM generator that gives higher power by connecting the external battery.

The PCA9685 board is utilized to solve this power problem. First, connect PCA9685 with the Jetson board using the I2C interface to communicate. And then PCA9685 connects to external batteries with 5V 0.5A in order to provide enough power to control the servomotor. The acceleration motor is already connected to its external battery, so the speed can be controlled by sending only PWM signals to meet PWM frequency conditions specified by the motor. Although we can use Jetson board's own PWM generator to control the acceleration motor, we only use PCA9685 because using PCA9685 and Jetson board's PWM generator simultaneously can cause data collisions in the I2C interface. Also, to simplify the PWM controlling software, we connect the acceleration motor to the PCA9685 board[23]. Table II shows the specification of motors attached to our car model through PCA9685 board.

We also connect the joystick to the car model to manually control it to enhance the convenience of the experiment. Joystick is connected by wireless communication because car modules must be entirely wireless connected to conduct con-

TABLE II

TRX-4 CONTROLLABLE MOTOR SPECIFICATION

Motor Type	Purpose	Power	PWM Frequency
Servomotor	Steering Angle	91W	60
OGRC Motor 550	Acceleration	2.5W	60

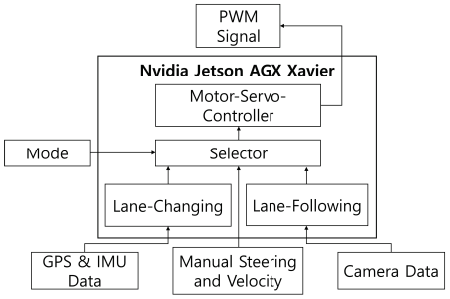

Fig. 2. The diagram above is the software system of the car model and the flow of data. The figure shows that all data is processed on the Jetson board. Therefore, several data processing and computation programs continue to operate simultaneously.

venient experiments without being restricted by cables. The joystick we use is Logitech Inc's G29 Driving Force Racing Wheel. This joystick is a wheel specialized in automobile simulation and has similar functions to actual automobile driving. In addition, several programmable buttons are attached to the wheel, so various functions, such as starting autonomous driving, going backward, and so on, can be executed without a keyboard, increasing convenience in the experiment.

The central processing unit of our car model is the Ndivia's Jetson AGX Xavier Board, a high-end compact computer. For reducing the sensor's latency and raising scalability, the sensors are directly connected to the Jetson board by serial ports. Fig. 2 shows the whole procedure on the Jetson board. The Jetson board has three programs running simultaneously. Motor-Servo-Controller controls all the motors: acceleration motor and servomotor. This program only uses the car's steering angle, velocity, and driving type as parameters to move the car. All parameter data are accepted as UDP package type. In addition, the Motor-Servo-Controller only receives the desired values of parameters. In other words, it does not share any of result with Lane-Changing or

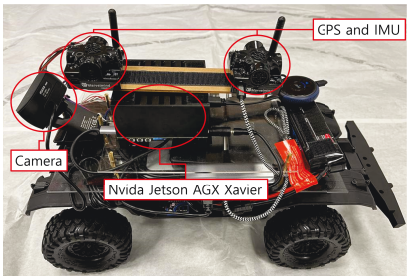

Fig. 3. The car model with all the hardware. Each piece of hardware is powered by a battery and is located in consideration of maximum safety, depending on its weight.

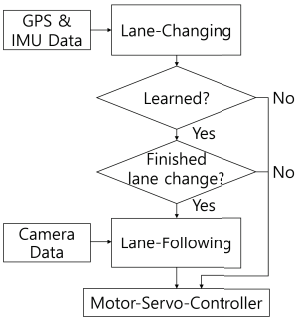

Fig. 4. Basic algorithmic structure of the autonomous driving. Each algorithm should be able to apply the result value to the motor only when certain conditions are met.

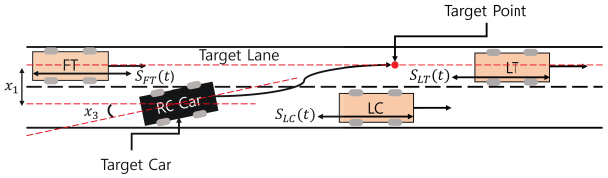

Fig. 5. The figure shows the desirable movement of the experiments. Our ultimate goal is to reach the target lane stably. Therefore, the value we need to measure is the shortest distance from the target car and the target lane, x_1, and the orientation of the target car, x_3.

Lane-Following. Motor-Servo-Controller is written in C++ to easily combine with a driver that fits the PCA9685.

Lane-Changing and Lane-Following are programs that contain algorithms which yield the direction and velocity of the car [24], [25]. Each program receives data from the desired sensor as parameters for the algorithm through serial ports. After processing the data through algorithm, the result (steering angle, speed, and driver type), send through localhost address with assigned port number to Motor-Servo-Controller. At this time, the driving mode is used to distinguish the transmission data of each program; for example, steering angle and speed made by Lane-Changing has a driving mode of 1, and the case of Lane-Following is 2. Then, Motor-Servo-Controller recognizes the driving way by the driving mode. The detail of driving mode is shown in table III.

TABLE III

DRIVING TYPE SPECIFICATION

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

Fig. 3 is the car model that we used in the experiment. The body size is 50cm, 24cm, and 26cm in width, length, and height, respectively. In addition, the maximum steering angle is ± 35 degrees. For the safety of our experiments, we limited the steering angle to within 30 degrees because any higher steering angle may lead the wheel to touch the body frame. The motor has a maximum output of 46W and a maximum speed of 15km/h. The car's speed can

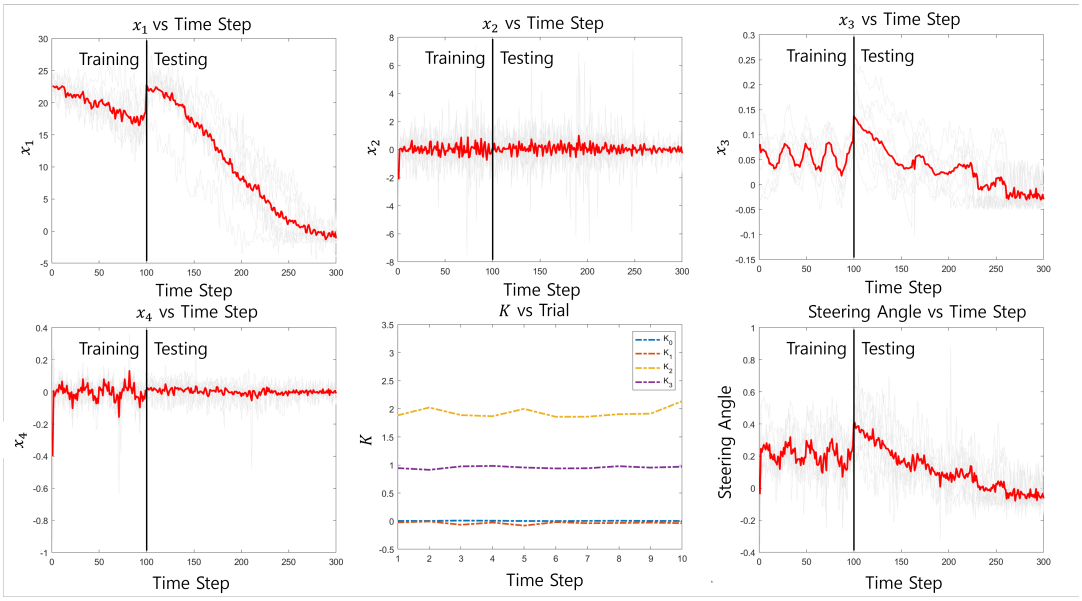

Fig. 6. Training and testing experiments are conducted in the same environment. Units for x_1, x_2, x_3, and x_4 are cm, cm/sec, $radian$, and $radian/sec$, respectively. The gray line is each experiment raw data and the red line represents the average value of the 10 experiments. The training part is until time-step 100, and then testing part. $\mathbf{K}$ vs Trial graph shows the $\mathbf{K}$ value after the training part in each experiment. The Steering Angle vs Time Step graph shows the actual steering angle that was calculated by the proposed algorithm.

maintain until approximately 80% of the battery. Hence, the experiments must have constant battery life above 80%.

The lane-changing algorithm of our car model collects data during the first 100 time-step (1 time-step = $0.083sec$) to find the near-optimal $\mathbf{K}$. In addition, when lane changing is completed, the car immediately executes the lane-following algorithm to induce continuous movement of the car. Therefore, the overall flow of data is the same as Fig. 4.

B. Experiments Environment

We have developed a two-lane road section in our lab for the purpose of experiments. The total length of the road is $4.5m$, and the width of each lane is $0.3m$. As shown in Fig. 5, there are four vehicles presented in the environment, where RC car denotes the remote controlled car, LT denotes the leader in the target lane, LC denotes the leader in the current lane, FT denotes the follower in the target lane, $S_{FT}(t)$, $S_{LT}(t)$, $S_{LC}(t)$ are the safety distances. In Fig. 5, x_1 denotes the distance of the center of gravity of the RC car from the center line of the target lane, x_3 denotes the orientation error of the RC car with respect to the road. The aim is that the RC car moves from the current lane to the target lane avoiding collision with the surrounding vehicles. The lane-change decision-making algorithm used in this work can be found in [26]. In this work, the RC car state vector is assumed to be $\mathbf{x} = [x_1, x_2, x_3, x_4]$, where x_2 is defined as the change in x_1 and x_4 is defined as the change in x_3. The control input $\mathbf{u}_k$ to the RC car is the steering wheel angle denoted as δ_k. The details on the lateral dynamic model of RC car can be found in [27].

We aim at learning a data-driven optimal controller to compute δ_k such that the RC car can successfully perform a lane change maneuver from the current lane to the target lane. Also, the safety from the surrounding vehicles is ensured by our proposed algorithms. We use data-driven Algorithm 2 to compute the optimal δ_k, where the data matrices in Algorithm 2 are formed by collecting real-time RC car state ($\mathbf{x}_k$) and input (δ_k) data. The initial stabilizing gain in Algorithm 2 is given as $\mathbf{K}_0 = [0.0047, -0.0447, 2.0002, 0.0002]$. The weight matrices are chosen as $\mathbf{Q} = \mathbf{I}_4$, and $\mathbf{R} = \mathbf{I}_1$. The exploration noise $\boldsymbol{\eta}_k$ is chosen as sum of sinusoidal waves.

C. Result and Analysis

In this section, we discuss the obtained experimental results for the RC car lane change. During the learning phase, we have implemented the initial stabilizing controller gain $\mathbf{K}_0$ to compute the steering angle $\delta_{0k} = -\mathbf{K}_0\mathbf{x}_k + \boldsymbol{\eta}_k$. This steering angle input is used as the control input to the RC car to collect data for 100 time-steps such that the rank condition in (13) holds. The training data $\{\mathbf{x}_k, \delta_k\}_{k=0}^{100}$ were collected by using the GPS sensors mounted on the RC car.

This section shows and analyzes the values and results obtained through experiment. We perform 10 experiments with the RC car to validate our proposed methodology. The mean values of x_1, x_2, x_3, and x_4 is denoted with red color plots in Fig. 6. For each experiment, we collect 100 data samples that is used for learning the optimal control gain for that experiment run. The training phase is marked complete when the optimal control gain is learned. After training, the RC car moves back to the starting position for the testing phase. For each experiment run, it is clear from Fig. 6 for the testing phase that, when the trained $\mathbf{K}$ is used by the RC car to perform a lane change the states of the RC car converges to zero. This implies that the lane change has been successful for each experiment run. Thus, the RC car could

learn optimal control policy for a lane change maneuver in real-time using sensor data.

As evident from the **K** vs Trial plot in Fig. 6, the trained **K** obtained for each experiment run is approximately the same, where K_0, K_1, K_2, and K_3 are the entries of the control gain vector **K**. The small variations of each **K** is possible because of the sensor noise. This shows the robustness of the proposed data-driven algorithm to noisy sensor data. For each experiment run, it was found that the RC car could perform smoother lane-changing maneuvers using the learned **K**. Our experimental research has shown that, through trials and learning, the car manages to learn a good estimate of the optimal control policy **K** from data as shown in table IV.

TABLE IV

INITIAL AND TRAINED K COMPARISON

	K_0	K_1	K_2	K_3
Initial $\mathbf{K}_0$	0.0047	−0.0447	2.0002	0.0002
Average Trained K	**0.0053**	**−0.0339**	**1.933**	**0.9558**

IV. CONCLUSIONS

In this study, we have presented a novel learning-based methodology for developing an optimal lane-changing control policy for an RC car using real-time sensor data. Our approach involves outfitting the RC car with a suite of sophisticated instruments, including GPS, IMU devices, a camera, and a Jetson board. Moreover, to ensure the accuracy and efficiency of our methodology, we equip the surrounding RC car with sensors (GPS, IMU devices and a camera). By processing real-time data from these sources, our RC car can compute the optimal lane-changing control policy online. We have implemented our algorithm on the Jetson board, which is expertly connected to the sensors. This optimal learning-based control approach utilizes the real-time data received from the sensors to compute the optimal lane-changing policy with exceptional accuracy. By utilizing an RC car, we are able to gather data that emulates the autonomous driving of full-size vehicles, resulting in significant reductions in experimental expenses, shorter experiment duration, and, most importantly, decreased safety risks associated with field experiments. We also demonstrate that our data-driven algorithm works successfully in automotive models using actual sensors. However, we have performed experiments in a laboratory environment. In the future, we plan on performing experimental studies in realistic environments.

ACKNOWLEDGMENT

We would like to thank Tong Liu for his constructive comments on our manuscript.

REFERENCES

[1] Y. Ma, Z. Wang, H. Yang, and L. Yang, "Artificial intelligence applications in the development of autonomous vehicles: A survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 2, pp. 315–329, 2020.

[2] J.-F. Bonnefon, *The car that knew too much: can a Machine be moral?* MIT Press, 2021.

[3] Y. Dou, F. Yan, and D. Feng, "Lane changing prediction at highway lane drops using support vector machine and artificial neural network classifiers," in *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2016, pp. 901–906.

[4] 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, p. 2140, 2021.

[5] 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.

[6] 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.

[7] T. Li, J. Wu, C.-Y. Chan, M. Liu, C. Zhu, W. Lu, and K. Hu, "A cooperative lane change model for connected and automated vehicles," *IEEE Access*, vol. 8, pp. 54 940–54 951, 2020.

[8] J. Wang, Q. Zhang, D. Zhao, and Y. Chen, "Lane change decision-making through deep reinforcement learning with rule-based constraints," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–6.

[9] Y. Liu, X. Wang, L. Li, S. Cheng, and Z. Chen, "A novel lane change decision-making model of autonomous vehicle based on support vector machine," *IEEE Access*, vol. 7, pp. 26 543–26 550, 2019.

[10] D.-F. Xie, Z.-Z. Fang, B. Jia, and Z. He, "A data-driven lane-changing model based on deep learning," *Transportation Research Part C: Emerging Technologies*, vol. 106, pp. 41–60, 2019.

[11] X. Gu, Y. Han, and J. Yu, "A novel lane-changing decision model for autonomous vehicles based on deep autoencoder network and xgboost," *IEEE Access*, vol. 8, pp. 9846–9863, 2020.

[12] L. Tang, H. Wang, W. Zhang, Z. Mei, and L. Li, "Driver lane change intention recognition of intelligent vehicle based on long short-term memory network," *IEEE Access*, vol. 8, pp. 136 898–136 905, 2020.

[13] D. of Transportation USA, "The next generation simulation programs," 2016, http://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm. [Accessed: (2023-05-27)].

[14] 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.

[15] T. Liu, L. Cui, B. Pang, and Z.-P. Jiang, "Data-driven adaptive optimal control of mixed-traffic connected vehicles in a ring road," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 77–82.

[16] ——, "Learning-based control of multiple connected vehicles in the mixed traffic by adaptive dynamic programming," *IFAC-PapersOnLine*, vol. 54, no. 14, pp. 370–375, 2021.

[17] 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.

[18] L. Traxxas, "The new traxxas summit 16.8 v electric extreme terrain monster truck," 2008.

[19] M. Robotics, "Marvelmind indoor navigation system operating manual," 2023.

[20] R. V. Vitali, R. S. McGinnis, and N. C. Perkins, "Robust error-state kalman filter for estimating imu orientation," *IEEE Sensors Journal*, vol. 21, no. 3, pp. 3561–3569, 2020.

[21] Nvidia, "Nvidia jetson agx xavier developer kit manual," 2019.

[22] W. Stevens and G. Wright, *TCP/IP Illustrated: The protocols*, ser. Addison-Wesley professional computing series. Addison-Wesley, 1994. [Online]. Available: https://books.google.co.kr/books?id=-btNds68w84C

[23] B. Earl, "Adafruit pca9685 16-channel servo driver," 2023.

[24] A. S. Rathore, "Lane detection for autonomous vehicles using opencv library," *International Research Journal of Engineering and Technology*, vol. 6, no. 1, pp. 1326–1332, 2019.

[25] 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*, 2023.

[26] 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)*. IEEE, 2022, pp. 1823–1828.

[27] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.