Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 4801985, 18 pages
https://doi.org/10.1155/2019/4801985

Research Article

Automatic Parking Controller with a Twin Artificial Neural Network Architecture

Jaeyoung Moon, Il Bae, and Shiho Kim

School of Integrated Technology, Yonsei Institute of Convergence Technology, Yonsei University, Incheon 21983, Republic of Korea

Correspondence should be addressed to Shiho Kim; shiho@yonsei.ac.kr

Received 9 July 2019; Accepted 3 September 2019; Published 18 September 2019

Academic Editor: Alberto Olivares

We propose an artificial deep neural network- (ANN-) based automatic parking controller that overcomes a stubborn restriction prevalent in traditional approaches. The proposed ANN learns human-like control laws for automatic parking through supervised learning from a training database generated by computer-aided optimizations or real experiments. By learning the relationships between the instantaneous vehicle states and the corresponding maneuver parameters, the proposed twin controller yields lateral and longitudinal maneuvering parameters for executing automatic parking tasks in confined spaces. The proposed automatic parking controller exhibits a twin architecture comprising a main agent and its cloned agent. Before the main agent assumes a maneuvering action, the cloned agent predicts the consequences of the maneuvering action through a Collision Checking and Adjustment (CCA) system. The proposed parking agent operates like a human driver in a manner that is characterized by an unplanned trajectory. In addition, the kinematics of the subject vehicle is not exactly modelled for parking control. The simulation results demonstrate that the proposed twin agent emulates the attributes of a human driver such as adaptive control and determines the consequences of the tentative maneuvering action under varying kinematic models of the subject vehicle. We validate the proposed parking controller by simulating the software-in-the-loop architecture using a PreScan simulator in which the dynamics of the virtual vehicle's behavior resemble a real vehicle.

1. Introduction

Automatic parking systems enhance the safety and comfort of drivers. The automatic parking system of a vehicle must perform numerous complicated tasks to safely park a vehicle in a confined space within a limited time, including environment detection, steering, acceleration, braking, and gear shifting while moving the car. Numerous studies on automatic parking systems have been conducted; however, achieving automatic parking of a car within a tiny space is a problem that must be overcome for the implementation of advanced driver-assistance systems or autonomous vehicles.

Most current parking systems exploit a step-by-step approach in which the reference trajectory is first determined, and then the desired reference trajectory for moving the car to the parking destination is tracked using an online feedback control loop. These methodologies have been broadly classified into two branches: geometric approaches and mathematical optimization approaches. In geometric approaches, a reference trajectory is usually denoted by a set of special curves (e.g., β-spline curves [1], Bézier curves [2], clothoid curves [3], and polynomial curves [4]) that exhibit geometrical properties. Mathematical optimization approaches [5–9] formulate parking problems in nonlinear programs (NLPs) and solve problems to minimize an objective function. A primary feature of this type of method with offline planning and online maneuvering is that the reference parking trajectory is generated by a set of curves that consider the kinematic model of a vehicle. In addition, the predefined path is traversed by controlling for a sequence of maneuvering actions associated with steering and velocity control to track the trajectory of the vehicle as closely as possible within a tolerable error range.

The difference between the actual parking trajectory and the desired trajectory can be controlled by a specific negative feedback loop to maneuver the vehicle and track the reference trajectory. However, such indirect planning and tracking methods may present several challenges in ensuring that the vehicle follows a predetermined trajectory within a permissible error in a confined parking space, owing to uncertainty and inaccuracy in the kinematic and/or dynamic model with a real vehicle under maneuvering conditions. The model parameters of the vehicle may vary with environmental changes such as tire pressure, surface conditions of the road, weight and distribution of the mounted load, and aging of the vehicle, which causes uncertainty in the kinematic and dynamic model, even for the same individual vehicle. Such problems can hinder the accomplishment of autonomous parking under restricted conditions of time and space with multiple back-and-forth movements.

On the contrary, a human driver can drive a vehicle without knowledge of the exact kinematic/dynamic parameters of the vehicle [10–12]. Human drivers implicitly consider environmental conditions and obstacles, including the dynamics and kinematics of the vehicle while driving a car. Skilled drivers intuitively and appropriately maneuver the vehicle by observing vehicle states and the relative positions with respect to the obstacles and goal position. One of the objectives of this study is to develop an automatic parking controller that exhibits tolerance to variations in the kinematic parameters of the vehicle. The parking controller operates under conditions in which the kinematics of the subject vehicle have not been exactly planned or modelled. Moreover, human drivers anticipate the consequences of their actions before they make the actual maneuvering decision that affects the movement of the vehicle [11, 13]. This allows the driver to curtail the potential risk of collision by anticipating, in advance, the consequence of driver maneuvers. The objective of this study is to implement an automatic parking controller that considers the aforementioned characteristics of human drivers, maneuvering of the vehicle without explicit vehicle modelling and foreseeing capabilities.

Pioneering work that attempted to mimic the behavior of a human driver in an automatic parking controller using a neural and fuzzy network was conducted in 1995 [14, 15]. After recording the parking behavior of skilled drivers, these datasets were used to train an artificial neural network (ANN) that mimics the expert driver's parking strategy. This approach has resulted in breakthroughs in state-of-the-art automatic parking controller systems; however, limited training datasets and the lack of premature processing devices in 1995 prevented the full realization of human-like parking controllers for autonomous vehicles. Heinen et al. also proposed autonomous vehicle parking control using artificial neural networks in 2006 [16]. The artificial neural network (ANN) models used in prior works output the vehicle's steering maneuvers by sensing the vehicle's current states and the target parking slot as the neural network's inputs. However, the sparse finite action spaces within those networks (such as straight and turning right/left) cause abrupt changes in input values. Li et al. [17] also proposed an end-to-end automatic parking approach using ANNs. However, these approaches handle automatic parking tasks in the case of simple trajectories without multiple maneuvering in confined spaces. In addition, the networks used in prior studies did not exhibit tolerance to variations in vehicle parameters.

Liu et al. proposed a trajectory planning system using ANN combined with online negative feedback control [18]. The fundamental idea of trajectory planning is to enumerate all possible parking trajectories and corresponding steering actions so that the parking controller learns the relationship between the given initial- and final-state pair and the corresponding sequence of steering actions to accomplish the parking task. This planning method imitates a human driver such that the parking controller recalls the desired steering actions in accordance with the parking trajectory based on the learning experience. However, this method still exploits the step-by-step approach that plans a reference trajectory first and then tracks the desired reference trajectory.

The primary aim of this study is to develop an automatic parking agent by learning the driving attributes of human drivers such as the inherence concept of a vehicle model and preview utilization for autonomous vehicles using ANNs. The proposed parking controller, comprising a main agent and a cloned agent, operates like a human driver in a manner that has not been exactly planned such that the kinematics of the subject vehicle have not been exactly modelled. The cloned ANN with a virtual vehicle model enables the controller to pre-evaluate the consequences of maneuvering actions in a manner that does not result in a collision, resembling a look-ahead searcher. Using a motion planner based on simultaneous dynamic optimization [6–9], we can easily collect a large amount of data to train the deep neural networks. We developed a near-optimal motion and trajectory planner using an interior-point method- (IPM-) based simultaneous dynamic optimization process to generate the training data. We generate vehicle states/input profiles that describe vehicle motions under conditions of mechanical restrictions and geometric constraints. Therefore, we can derive several parking scenarios in the case of single and multiple maneuvers using the parameters of the initial/final position of a vehicle and a parking space. By learning the relationships between the current states of the vehicle and the corresponding vehicle control inputs, the proposed approach yields a direct steering angle and velocity while executing automatic parking maneuvers. We validated that the proposed ANN-based parking agent mimics human-like common sense in terms of the manner in which the car is moved from the starting position to the destination to complete the parking task.

This paper has been organized into six sections. Section 2 provides an explanation for generating a training dataset that learns parking maneuvers. Section 3 presents a description of the proposed automatic parking algorithm, which is based on a deep neural network model. Simulation results and discussions are presented in Sections 4 and 5, and the conclusions are finally drawn in Section 6.

2. Generation of Training Dataset

2.1. Interior-Point Method-Based Simultaneous Dynamic Optimization. A recent IPM-based simultaneous dynamic optimization methodology [19, 20] which can be classified as a numerical method, that includes vehicle kinematics and collision-avoidance constraints was adopted to generate time-optimal parallel parking maneuvers in a straightforward manner. An IPM-based simultaneous dynamic optimization technique was successfully applied to offline near-optimal automatic path and maneuver planning for automatic parking [6–9]. The IPM-based simultaneous dynamic optimization technique briefly includes the following four processes: (1) obtaining a set of differential equations of control and state variables describing the kinematics of a vehicle using a kinematic bicycle model as shown in Figure 1; (2) defining the geometric environment of a parking space with obstacles as shown in Figure 2; (3) discretizing the control and state variables, the original problem is transformed into an IPM-based simultaneous dynamic optimization of nonlinear program (NLP), which is equivalent to a fully implicit Runge–Kutta method [21]; and (4) solving the set of equations with equality and inequality constraints using the Interior Point OPTimizer (IPOPT) solver [22], which is an open-source software package of IPM.

We first used Ackerman's kinematic bicycle model to describe a front-wheel drive vehicle, as shown in Figure 1 [2, 23]. The kinematics of an autonomous vehicle can be expressed as follows:

$$\frac{dt}{d}\begin{pmatrix} x(t) \\ y(t) \\ v(t) \\ \Psi(t) \\ \delta(t) \end{pmatrix} = \begin{pmatrix} v(t)\cdot\cos\Psi(t) \\ v(t)\cdot\sin\Psi(t) \\ a(t) \\ \dfrac{v(t)\cdot\tan\delta(t)}{l_w} \\ \omega(t) \end{pmatrix}, \quad \forall t \in [0, t_f], \qquad (1)$$

where t_f indicates the completion time of the entire process, (x, y) denotes the coordinate of E (the mid-point of the rear-wheel axle), $v(t)$ is the velocity of point E, $a(t)$ denotes the corresponding acceleration, $\Psi(t)$ is the yaw angle of the vehicle body, $\delta(t)$ refers to the steering angle of the front wheel, and $\omega(t)$ is the angular velocity of the front wheel. The parametric notations are illustrated in Figure 1 and listed in Table 1.

In the general case of a parallel parking scenario, a subject vehicle moves from an initial position to the ready-to-reverse position (RRP). The parking task begins at the RRP, where the vehicle starts to move backward toward the destination position in the parking slot, as shown in Figure 2(a). In this study, we assume that the parking maneuver begins from RRP and ends at the destination position by satisfying the completion condition of the target pose (position and orientation), as defined in the ISO 16787 standard [24].

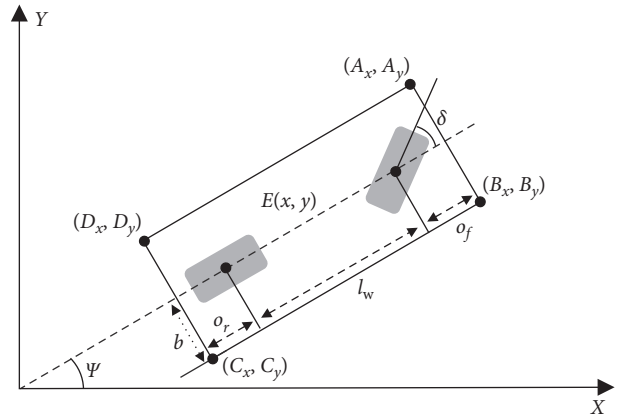


Figure 1: Bicycle kinematic model of front-steering vehicles.

An example of a parking geometry with obstacles and an illustration of collision cases in a parallel parking geometry are shown in Figure 2(b). The parameters of the parking slot include the following: parking slot length (SL), parking slot width (SW), and lane width of parking area (LW). A completion condition of the parking scenario follows the minimum performance requirements for the final pose of the automatic parking process described by the ISO 16787 standard. In the terminal condition of the parking problem, the orientation of the vehicle relative to the road edge is in the range of -3° to $+3^\circ$, where 0° is the target value. The distances from the points of the outermost tire-contact with the ground to the outer edge of the parking slot line are greater than 0 m, which are the minimum performance requirements for the end position of the automatic parking process.

The parking scenario is formulated via mathematical programming to obtain a training dataset. The x- and y-coordinates of the maneuvering zone are also defined as shown in Figure 2(b). Three collision cases exist in a parallel parking geometry marked as ①, ②, and ③ in Figure 2(b). If SL is not sufficiently long to park a vehicle in a single maneuver, the corresponding back-and-forth shuffling results obtained from additional possibilities of collisions with neighboring left and right obstacles and with the curb are labelled as ③. The strategy for formulating the problem and avoiding collision cases in the parking scenario is based on the "partitioning maneuvering zone method" in [9]. We recommend Ref. [9] to interested readers for more detailed information with respect to collision cases ①, ②, and ③.

After formulating the parking problem, the simultaneous dynamic optimization method [6–9, 20] discretizes differential equations that can describe vehicle kinematics including mechanical/physical constraints on state/control variables and solve. The IPOPT optimizer using IPM-simultaneous dynamic optimization methodology [22] is adopted to solve the formulated problem.

2.2. Training Dataset Generation. This subsection presents the criteria for generating a training dataset and for training an ANN model for automatic parking. To collect a training

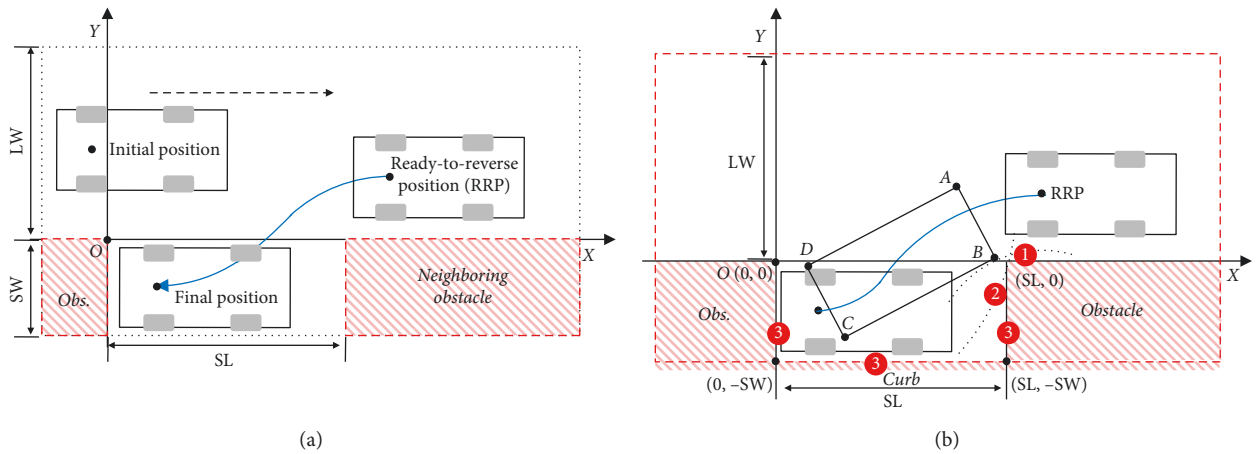


Figure 2: Example definition in a parallel parking geometry. (a) General case of a parallel parking scenario. (b) Illustration of collision cases.

Table 1: Variables and parameters used in bicycle kinematic model of vehicles.

Symbol	Description
$E(x, y)$	Center of the rear wheel in the bicycle kinematic model
$A(A_x, A_y)$	Coordinates of front-left corner of a vehicle
$B(B_x, B_y)$	Coordinates of front-right corner of a vehicle
$C(C_x, C_y)$	Coordinates of rear-right corner of a vehicle
$D(D_x, D_y)$	Coordinates of rear-left corner of a vehicle
l_w	Distance between front and rear wheelbase
$o_{f(r)}$	Distance of front (rear) overhang of the vehicle
$2b$	Width of the vehicle
v	Velocity of the vehicle at the center of rear wheels
a	Corresponding acceleration
Ψ	Yaw angle
δ	Front wheel angle
ω	Angular velocity of wheel steering angle

dataset, the formulated parking problem is varied by three parameters, i.e., the initial x- and y-coordinates of the vehicle's position and the length of the parking slots. These problems can be solved mathematically using the IPM [20]. The optimal solver obtained the vehicle's state/control profiles and determined whether the results of the parking problems are single- or multiple-maneuver tasks. Figure 3 geometrically illustrates the manner in which various parking scenarios can be set to generate a training dataset. The symbol b denotes the half-width of the vehicle. SW is the width of the parking slot. We set three variables as the length of the parking slot (SL) and the x- and y-coordinates of the RRP of a vehicle (x_{RRP}, y_{RRP}). The parking process can be varied depending on these three parameters. If the parking maneuver is aborted before arriving at the final position, we eliminated the respective RRP from the training dataset. We only included successfully completed parking maneuvers in the dataset.

First, we generate cases with SL in the range from 4.4 m to 5.4 m to include both the single- and multiple-maneuver parking scenarios. Next, we defined the RRP region with the x- and y-coordinates in an area defined by the union of two boundaries, $\mathrm{SL} + 0.8 + (y_{RRP} - 1.0) \le x_{RRP} \le \mathrm{SL} + 2.0$ and $b + 0.2 \le y_{RRP} \le b + 1.0$, as shown in Figure 3.

To generate the training dataset, simulations under various parking conditions and environments were conducted with the IPOPT solver (version 3.8.0) [22] and executed on an Intel Core i7-3770 CPU. We altered the parameters, i.e., the parking slot length (SL) and the x- and y-coordinates of RRP (x_{RRP}, y_{RRP}) in the parallel parking process, while the dimensions of the parking slot width (SW) were fixed at 2 m. The boundaries of the variables include: $v_{max/min} = \pm 2$ m/s, $a_{max/min} = \pm 0.5$ m/s^2, $\delta_{max/min} = \pm 33^\circ$, and angular velocity $\omega_{max/min} = \pm 1$ rad/s. The vehicle parameters used for training data generation include the following: overall width ($2 \times b = 1.6$ m), length of wheelbase ($l_w = 2.54$ m), and front/rear overhang ($o_{f/r} = 0.54/0.54$ m).

We simulated parking problems for 891 cases from easy to difficult conditions, where the dataset includes single- and multiple-maneuver cases with back-and-forth movements and gear changes. Table 2 presents a summary of the selected cases of parking geometry and parking maneuvers. The generated dataset was submitted along with this manuscript in the Supplementary Materials (available here). The number of gear changes and generated completion times of the entire process vary depending on the positions of SL and the RRP. Examples of the generated trajectories and the control/state profile results are depicted in Figure 4 for zero gear change, and in Figure 5 for four gear changes, causing multiple back-and-forth movements owing to the confined parking space. The inset box in Figure 5(a) presents an enlarged view of the trajectory where multiple maneuvers occur.

3. Proposed Approach and Architecture of Automatic Parking Controller Based on ANN

3.1. Proposed ANN Architecture and Training the ANN. We built an architecture of a deep ANN for an automatic parking agent. The proposed ANN is a basic component of the automatic parking agent, which maneuvers a vehicle with parking skills learnt by training data, so as to produce the appropriate action outputs while maneuvering the vehicle with respect to the current state and previous actions.

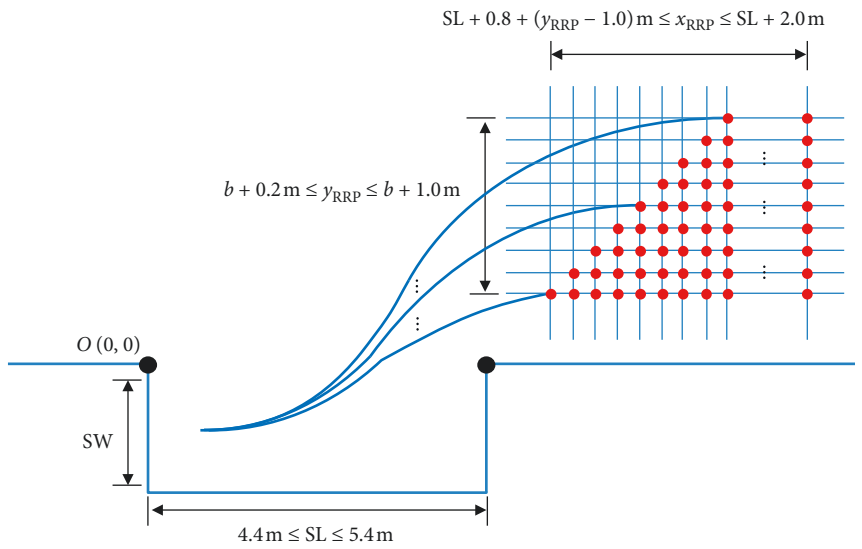


Figure 3: Schematic illustrating zone and RRP of parking for collecting training dataset.

Table 2: Selected examples of generated training dataset.

Case no.	Parking slot length (m)	RRP position (x, y) (m)	Number of gear change	Time to complete parking t_m (s)
1	SL = 5.4	(6.4, 1.0)	0	5.71
2	SL = 5.2	(6.2, 1.0)	1	6.71
3	SL = 5.0	(6.0, 1.0)	1	7.06
4	SL = 4.8	(5.8, 1.0)	1	7.43
5	SL = 4.6	(5.6, 1.0)	2	8.74
6	SL = 4.4	(5.4, 1.0)	4	10.75
7	SL = 5.4	(7.2, 1.8)	0	6.26
8	SL = 5.2	(7.0, 1.8)	1	7.26
9	SL = 5.0	(6.8, 1.8)	1	7.71
10	SL = 4.8	(6.6, 1.8)	2	8.99
11	SL = 4.6	(6.4, 1.8)	2	9.39
12	SL = 4.4	(6.2, 1.8)	4	11.16

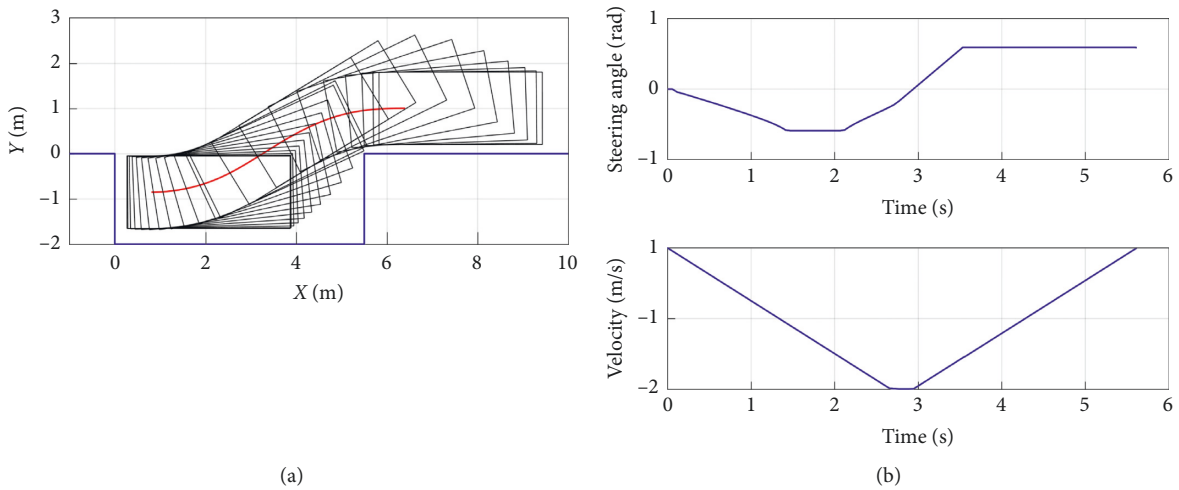
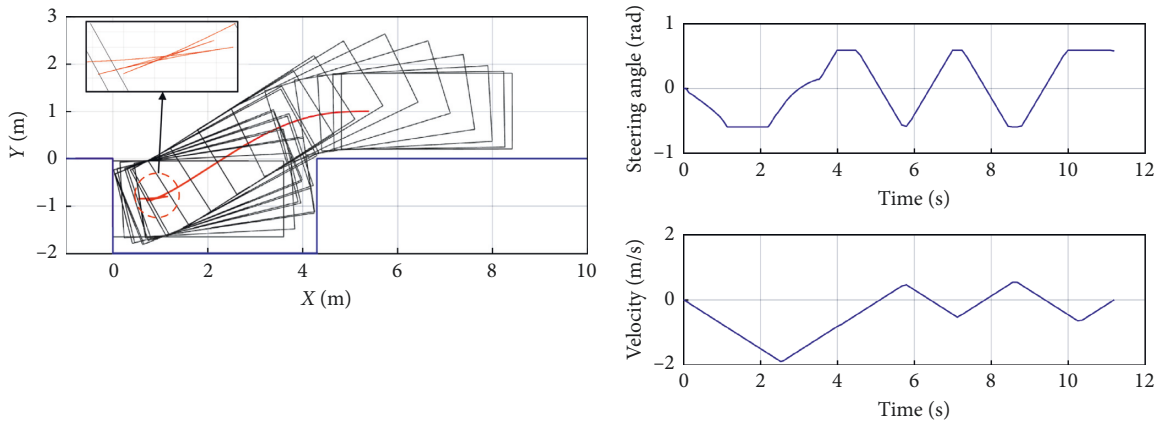


Figure 4: Training data of Case 1 in Table 2. (a) Generated path and parking trajectory. (b) Steering angle profile and velocity profile.

The proposed ANN consists of seven fully connected layers with 128 neurons in each layer and six inputs and two outputs. Hyperbolic tangent activation functions (*TanH*) are used in each fully connected layer. We decided the number of deep layers and the number of neurons per layer using a try and modify method based on some rules-of-thumb method [25, 26] until the appropriate performance was achieved during training based on the generated datasets.



(a) (b)

Figure 5: Training data of Case 6 in Table 2. (a) Generated path and parking trajectory. (b) Steering angle profile and velocity profile.

We denote a dataset by $\mathbb{D}$ and an individual recorded sequence of a successful parking trajectory and maneuvering by $\mathscr{A} \sim \mathbb{D}$. $\mathscr{A}$ is a sequence of vehicle poses and actions: $\mathscr{A} = [(s_1, 0, a_1), (s_2, a_1, a_2), \ldots, (s_n, a_{n-1}, a_n)]$, where s_k is a pose of the vehicle and environment, i.e., $s_k = (x(k), y(k), \Psi(k), \mathrm{SL}(k))$ and $a_k = (v(k), \delta(k))$, which is a maneuvering action of the vehicle at the k^{th} discrete time. We train the ANN by applying a supervised learning method using the whole dataset, $\mathbb{D}$, with a mean squared error (MSE) loss function. The ANN is trained using maneuvering parameters $(v(k-1), \delta(k-1))$ and s_k of the vehicle as inputs and the optimum maneuvering parameters at the current state and a_k as outputs. Single-maneuver scenarios produce approximately 60–70 pairs of states, and multiple-maneuver scenarios produce approximately 90–150 pairs of states. 103,650 pairs of inputs and outputs are generated using 891 scenarios for the training dataset.

By applying an ANN, the control laws for automatic parking are learnt through a dynamic optimization database and, particularly, by following optimized actions that provide the appropriate maneuvers for each specific situation with respect to the vehicle pose and environment.

The output range of the steering angle, $\delta(k)$, must be limited by the maximum values associated with the vehicle's mechanical/physical constraints. The velocity of the vehicle, $v(k)$, should be bounded by considering the parking environment. Therefore, the ANN model can consider the output's maximum/minimum values. In the output layer, we apply an activation function *TanH* to saturate the output value at or below the maximum value.

The structure of the generated dataset and the arrangement of the input and output data for training the ANN are shown in Figure 6. The current states of the vehicle at time k ($x(k)$, $y(k)$, and $\Psi(k)$), parking slot information (SL), and actions of the vehicle at time $k-1$ ($v(k-1)$ and $\delta(k-1)$) are used as inputs of the model for supervised learning.

We trained our ANN model for automatic parking using the aforementioned training dataset. The proposed deep neural network for automatic parking was implemented using the Caffe deep learning framework [27]. The ANN model was trained on 1 million iterations with a random sampled batch (size of 64). The learning rate was 0.001 with a descent ratio of 0.96 per 10,000 iterations.

3.2. Implementation of Parking Controller Using ANN. This subsection describes the implementation of an automatic parking controller using the trained ANN. The trained ANN operates as an automatic parking controller to produce action signals at every time step. The proposed simple controller with the trained ANN and kinematic model of a vehicle constitute a software-in-the loop (SIL) architecture that validates the proposed approach, as illustrated in Figure 7. The SIL architecture includes a closed feedback control loop, wherein the trained ANN generates action signals as outputs, and the vehicle pose and previous actions are fed into the ANN at every time step while executing parking tasks.

An inherent drawback of the simple controller with an ANN model is that the ANN produces maneuvers without predicting the consequences of its action signals, which may result in unwanted collisions with obstacles, even if the ANN were fully trained with an extensive optimized dataset. On the contrary, with respect to the skills of human drivers [11], human drivers anticipate the consequences of their actions before they make the actual maneuvering decision, which affects the movement of the vehicle.

We proposed a twin architecture by combining the main parking agent and a clone for pre-evaluation of the pose of a virtual vehicle in advance by λ steps before applying the real action signal to the main vehicle to avoid collisions. Figure 8 presents an overview of the proposed automatic parking controller with a twin architecture, including the main vehicle-in-the-loop and its clone with a collision-avoidance algorithm. The dotted lines denote the machine boundary between the twin agents. A duplicated SIL architecture comprising the cloned ANN with a virtual vehicle model is used to predict the vehicle pose by looking λ steps ahead. Simulation of the clone agent provides a prediction of

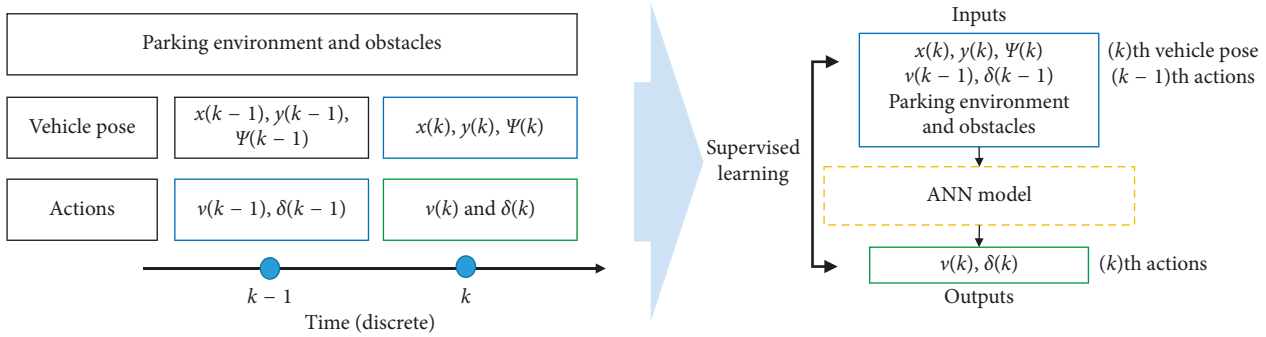


Figure 6: Structure of generated data and arrangement of input and output data for training the ANN.

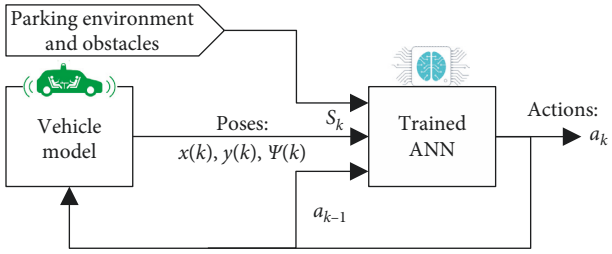


Figure 7: Simple parking controller based on the trained ANN model. A trained ANN model and kinematic model of a vehicle constitutes a software-in-the loop architecture for validating the proposed approach without predicting the consequences of its actions.

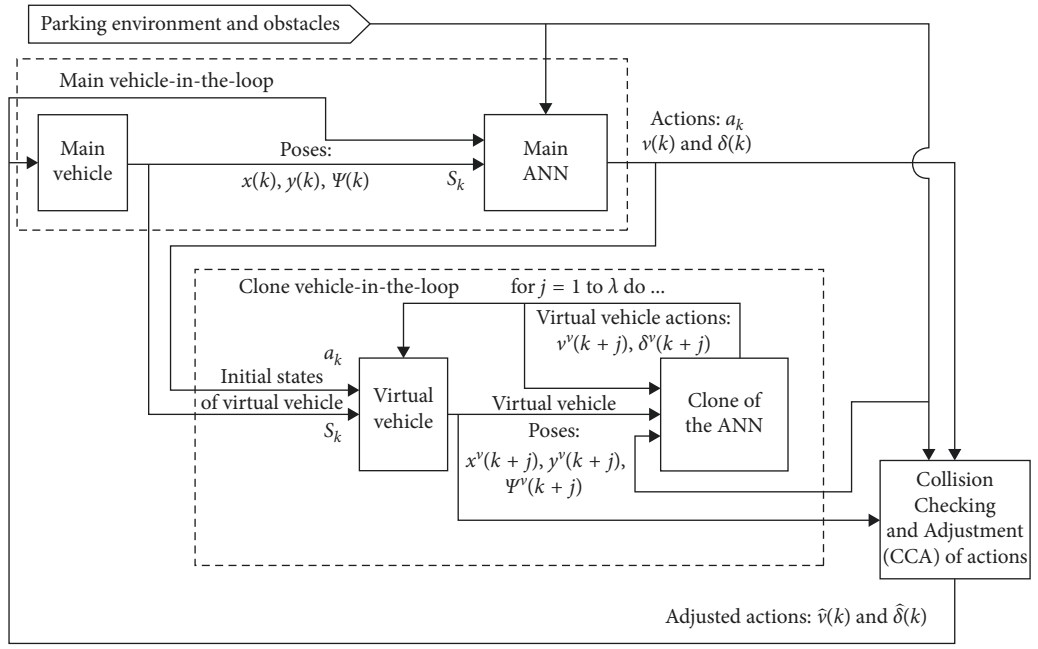


Figure 8: Block diagram of proposed twin architecture of parking controller including main vehicle-in-the-loop and its clone with collision-avoidance algorithm.

potential collisions with obstacles, including modifications to the action signal of the main agent before applying the signal to the main vehicle. The prediction of the clone agent is tracked and reflected in the results of the main agents' actions by employing the twin vehicle-in-the loop mechanism.

The outputs from the main ANN and pose of the main vehicle are fed into the initial pose and initial action of the cloned vehicle-in-the-loop, so as to simulate a potential sequence of the virtual vehicle pose for λ steps ahead, to determine whether any type of collision occurs and to make adjustments to actions in response to a detected collision while simulating a virtual vehicle λ steps ahead. We can empirically choose λ with a safe distance margin to identify collisions by considering the maximum allowed speed and period of discrete time sampling of an autonomous vehicle

during parking. The Collision Checking and Adjustment (CCA) module is a functional block that checks for possible collisions and then adjusts the action signals. Algorithm 1 presents the pseudocode of the predictive collision-avoidance algorithm, which is a core part of the twin architecture of the parking controller.

4. Validation of Proposed Architecture

This section investigates the functionality and the ability of the proposed parking controllers to perform the automatic parking task in various environments. To validate the proposed parking controllers based on the trained ANN model, we simulated the software-in-the-loop architecture for various parking environments, where the vehicle model in the loop exhibits the same kinematic model parameters as the bicycle model used for simultaneous dynamic optimization. The first step in simulation is placing the initial pose of the vehicle in the zone of RRP. We selected arbitrary points for the initial pose and performed simulations while tracing the trajectory and maneuvers of the vehicle. The RRP region was defined when we generated the training dataset. The parking process is complete when the vehicle reaches the designated parking slot with the requirements for the final pose of automatic parking, as described by the ISO 16787 standard.

4.1. Simulation Results of the Proposed Simple Controller. To verify the feasibility of the proposed simple parking controller based on the trained ANN model, the SIL architecture represented by the block diagram shown in Figure 7 was implemented in the software model using MATLAB, and we performed simulations for various parking environments and RRP, which exhibit different positions from training data generation. In the simulation of the proposed simple parking controller, we used the same bicycle steering model and the same kinematic model parameters of the bicycle model used in the dynamic optimization for training data generation. The vehicular parameters used for the simulations are the same as that of the training data generation.

When the given space is wide enough to complete automatic parking without gear changes for back-and-forth movements, from the initial RRP to the destination position while satisfying the ISO 16787 requirements, most simulated results demonstrate that parking can be achieved without collision with fixed obstacle barriers, as shown in Figure 9 when SL = 5.4. However, when the parking space is so small that the vehicle must change gears for back-and-forth movement, the vehicle collided with fixed obstacles, as shown by the results in Figures 10 and 11. Figure 10 illustrates that collision case ② occurred during back-and-forth movements in a confined space when SL = 4.8 m. In this case, the vehicle's front-right side collides with the vertex at the right side as shown in the enlarged subfigure in Figure 10(a). Figure 11 shows that collision case ③ as well as case ② occurred in the case of a confined parking space when SL = 4.4.

Figures 9–11 present ordinary cases of simulation results on the simple controller with an ANN model. We simulated several parking cases with the starting points of parking in the region of RRP. In the case of SL > 5.3, the controller parks the vehicle with a success rate of 99.8%. However, in the case of SL < 5.3, the controller requires multiple maneuverings for back-and-forth movements during parking, and the vehicle body at times intruded border line, resulting in collisions with obstacles.

As indicated in the simulated results, the artificial neural network was trained to output nearly appropriate sequences of back-and-forth maneuverings through supervised learning without a preplanned trajectory. However, the vehicle could not avoid collisions in confined parking spaces.

4.2. Simulation Results of the Proposed Controller with a Twin Architecture. We implemented a software-in-the-loop automatic parking simulation environment for the proposed controller with twin ANN architecture using MATLAB. In the predictive collision-avoidance algorithm based on the twin ANN, the clone agent in the cloned vehicle-in-the-loop executes subsequences of the parking maneuvers with the virtual vehicle while checking for possible collisions with obstacles. If a collision is detected, the CCA then adjusts the velocity and steering angle coming from the main ANN before applying to the main vehicle. To compare differences in collision-avoidance capabilities between single agent and twin agent systems, we conducted automatic parking simulations in the exact same starting points and under environmental conditions of the aforementioned simulations with a simple ANN.

The proposed predictive collision-avoidance algorithm bypasses actions from the main ANN if no collision in the cloned vehicle-in-the-loop is detected during maneuvering for the automatic parking task. As expected, no differences can be observed in the simulated results between the single agent and the twin agent in the case when SL = 5.4 m and RRP at position $(x_{\mathrm{RRP}}, y_{\mathrm{RRP}}) = (6.7, 1.4)$, which indicates the same simulation presented in Figure 9.

Figure 12 presents the simulation results of the proposed twin ANN agent in the case when SL = 4.8 and RRP at position $(x_{\mathrm{RRP}}, y_{\mathrm{RRP}}) = (6.0, 1.2)$, which is the condition for the collision case ② occurring while the single agent was maneuvering the vehicle, as shown in Figure 10. The enlarged subfigure shown in Figure 12(a) demonstrated that the front end of the main vehicle does not intrude the border of the obstacle barrier during the back-and-forth movement for automatic parking maneuvers. The CCA block in the twin agent architecture checks for possible collisions; whenever collision case ② occurs with the virtual vehicle, the CCA adjusts the steering angle so as to avoid the collision. The simulated results demonstrate that CCA adjusted the steering angle approximately from 3.3 to 3.9 seconds to avoid collision case ②, and so the parking task takes approximately 7.2 s to complete the parking task, which is slightly longer than 6.8 s in the single ANN case shown in Figure 10.

Figure 13 presents the results of the trajectories in the case of SL = 4.4 m and RRP at position $(x_{\mathrm{RRP}}, y_{\mathrm{RRP}}) = (5.7, 1.4)$, which are the same conditions of the single ANN shown

Input: // *Input to the Clone agent for every decision step of the Main agent*
$S^v(k) = \{x(k), y(k), \Psi(k)\}$; // *Input Pose of the Virtual vehicle = Current pose of the main vehicle*
$A^v(k) = \{v(k), \delta(k)\}$; // *Input Action of the clone agent = Output of the main agent's action*
Parking *Environment* & obstacles SL; // *Parking slot length and neighboring obstacles*
Output: Adjusted Action to the main vehicle $\widehat{A}(k) = \{\widehat{v}(k), \widehat{\delta}(k)\}$;
Parameters λ, $A1$, $A2$, $A3$; // *# of looking forward steps* (λ), *adjustment parameters for desired actions* ($A1$, $A2$, $A3$)
(1) **Procedure** forecasting λ step forward poses and actions for the virtual vehicle
// *Predicting Pose at j forward steps* $S^v(k+j) = \{x^v(k+j), y^v(k+j), \Psi^v(k+j)\}$
// *Simulating j step forward actions* $A^v(k+j) = \{v^v(k+j), \delta^v(k+j)\}$
(2) **for** $j = 1$ to λ **do**
(3) Calculate j step forward state of the virtual vehicle $S^v(k+j)$;
(4) Calculate j step forward actions of the virtual vehicle $A^v(k+j)$;
// *Checking collision occurrence and adjustment of actions if collision detected*
(5) **if** Collision Case ①:
(6) adjust steering angle $\{\widehat{\delta}(k) = A1 * \delta(k), \widehat{v}(k) = v(k)\}$;
(7) **break for-loop**
(8) **else if** Collision Case ②:
(9) adjust steering angle $\{\widehat{\delta}(k) = A2 * \delta(k), \widehat{v}(k) = v(k)\}$;
(10) **break for-loop**
(11) **else if** Collision Case ③:
(12) adjust desired velocity $\{\widehat{\delta}(k) = \delta(k), \widehat{v}(k) = A3 * v(k)\}$;
(13) **break for-loop**
(14) **else** // *If no collision is predicted, then do not adjust inputs*
(15) $\{\widehat{\delta}(k) = \delta(k), \widehat{v}(k) = v(k)\}$;
(16) **end if**
(17) **end for**
(18) **return** $\widehat{A}(k) = \{\widehat{v}(k), \widehat{\delta}(k)\}$;
(19) **end of Procedure**

Algorithm 1: Predictive collision-avoidance algorithm based on twin ANN parking agent.

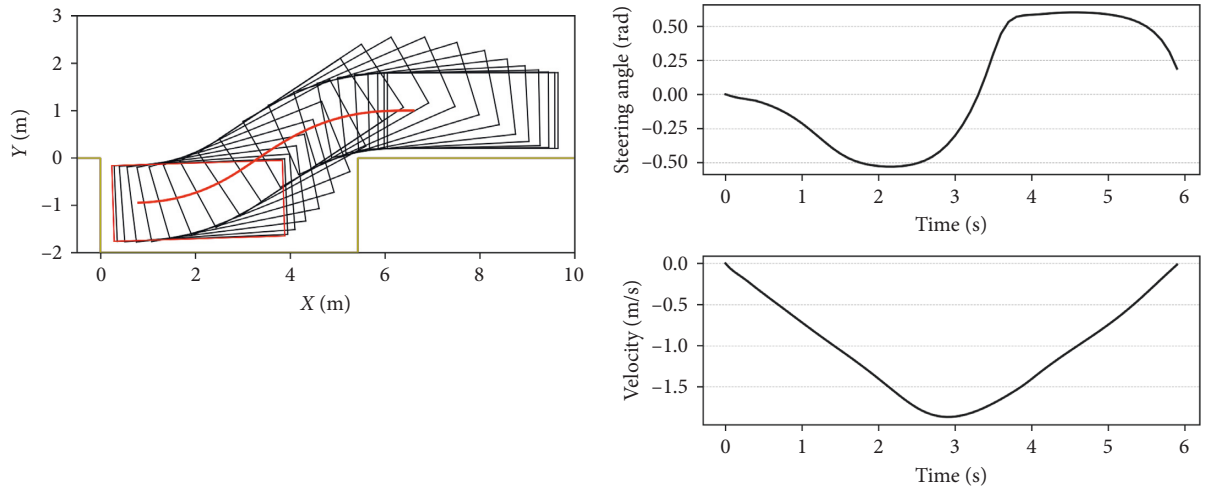


Figure 9: Simulated results of proposed simple ANN agent when SL = 5.4 m and RRP at position $(x_{RRP}, y_{RRP}) = (6.7, 1.4)$. (a) Simulated parking trajectory. (b) Steering angle profile and velocity profile.

in Figure 11. As shown in the enlarged subfigure in Figure 13(a), the proposed twin agent controller can complete the automatic parking task without any collisions. The simulation results shown in Figure 13 demonstrate that CCA resolved case ② and case ③ collisions shown in Figure 11 while parking in a tiny space.

Human drivers implicitly make tentative decisions, predict the consequences of an action in their mind before applying and then decide on the final action that produces the appropriate consequence. The simulation results demonstrate that the automatic parking skill of the twin agent emulates a human driver's attributes to the extent that the main agent

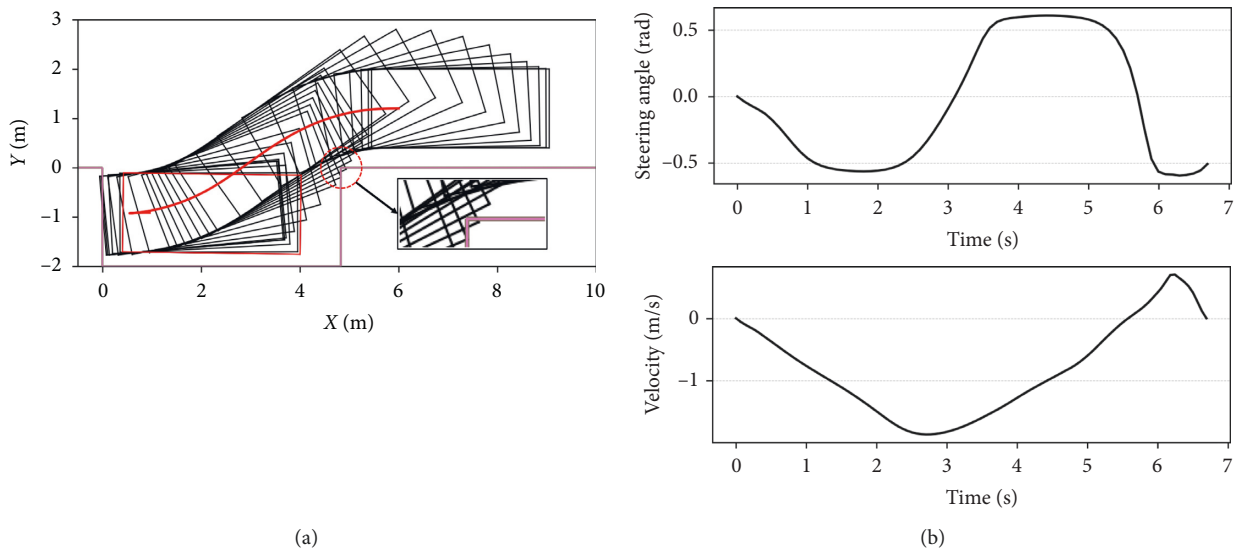


Figure 10: Simulated results of proposed simple ANN agent when SL = 4.8 m and RRP at position $(x_{RRP}, y_{RRP}) = (6.0, 1.2)$. BC side front end of the vehicle collides with the vertex of the parking slot while the vehicle is moving backward (i.e., collision case ②). (a) Simulated parking trajectory. (b) Steering angle profile and velocity profile.

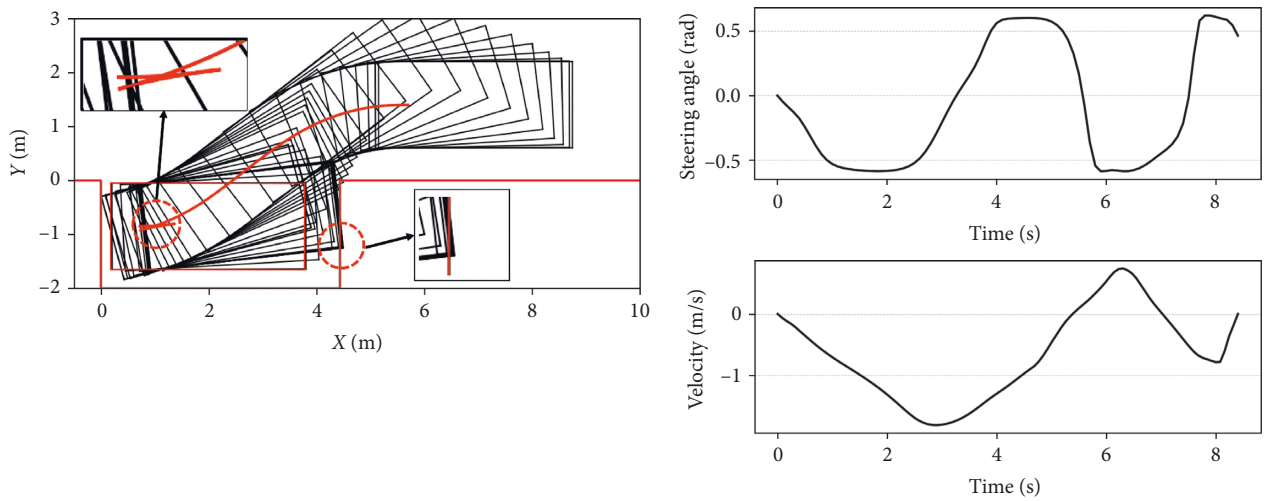


Figure 11: Simulated results of the proposed simple ANN agent without a collision prediction algorithm when SL = 4.4 m and RRP at position $(x_{RRP}, y_{RRP}) = (5.7, 1.4)$. AB side of the vehicle collides with the border of the parking slot while the vehicle moves back and forth (i.e., collision case ③ in addition to collision case ②). (a) Simulated parking trajectory. (b) Steering angle profile and velocity profile.

makes tentative actions, the cloned agent performs a virtual activity to predict the consequence of the tentative actions, and then the CCA adjusts the actions applied to the main vehicle.

4.3. Tolerance of Parking Agent against Variation in Kinematic Vehicle Model Parameters. Vehicle parameters can vary because of many reasons during the life cycle of the vehicle. In this subsection, we verify that the proposed twin agent exhibits tolerance against variations in the kinematic parameters of the vehicle. A skilled human can drive a car under uncertain knowledge of kinematic/dynamic parameters of the vehicle [10–12].

Figure 14presents the simulated turning-radius of vehicles with different geometric parameters with (L/l_w = 3.4/2.38, 3.6/2.53, and 3.8/2.66) under the assumption of the bicycle kinematic model of the vehicle shown in Figure 1, where L is the length of the vehicle and l_w denotes the distance between the centers of the front and rear wheelbase. We applied identical steering control input to vehicles with different geometric parameters, and the corresponding simulated parking trajectories are presented in Figure 15. As expected, different geometric parameters result in different turning radii and parking trajectories when an equivalent steering input is applied as shown in Figures 14 and 15. The geometric parameters

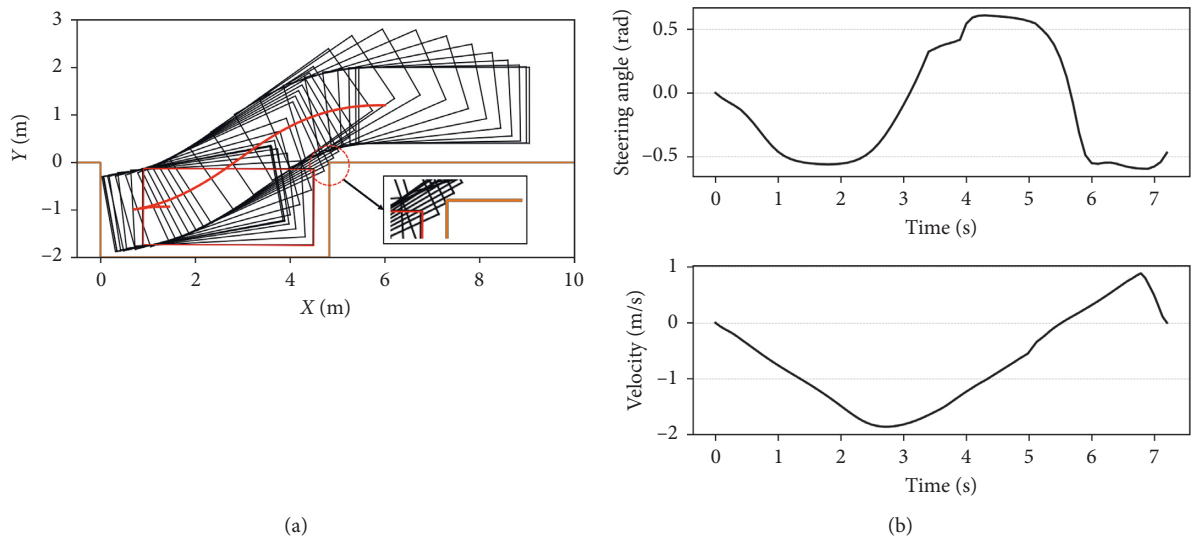


Figure 12: Simulated results of the proposed twin ANN agent when SL = 4.8 and RRP at position $(x_{RRP}, y_{RRP}) = (6.0, 1.2)$. BC side front end of the vehicle does not collide with the vertex of the parking slot. (a) Simulated profile of parking trajectory. (b) Steering angle profile and velocity profile.

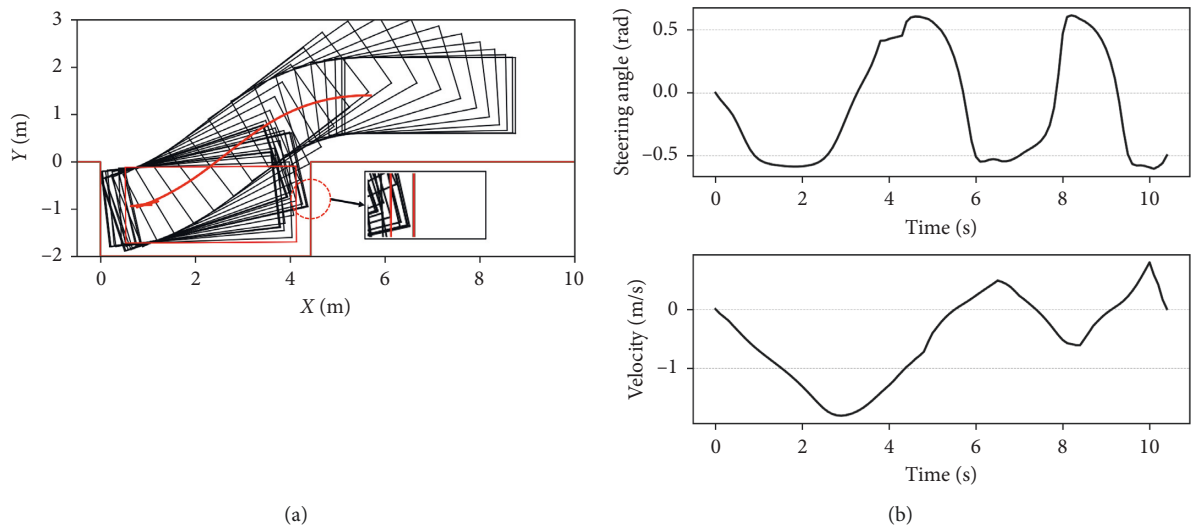


Figure 13: Simulated results of the proposed twin ANN agent when SL = 4.4 and RRP at position $(x_{RRP}, y_{RRP}) = (5.7, 1.4)$. It is clear that case ③ and ② collisions can be avoided. (a) Simulated profile of parking trajectory. (b) Steering angle profile and velocity profile.

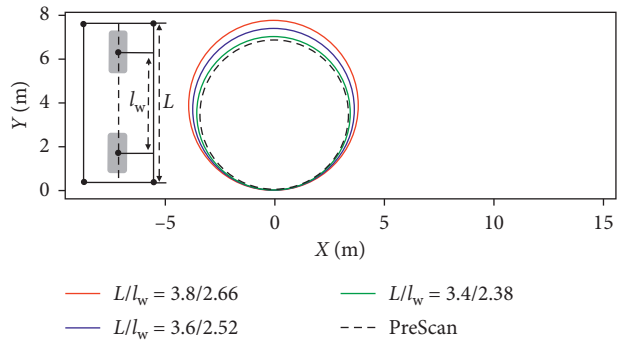


Figure 14: Simulated turning radius of vehicles with different geometric parameters with (L/l_w = 3.4/2.38, 3.6/2.53, and 3.8/2.66) assuming bicycle kinematic models of the vehicle shown in Figure 1.

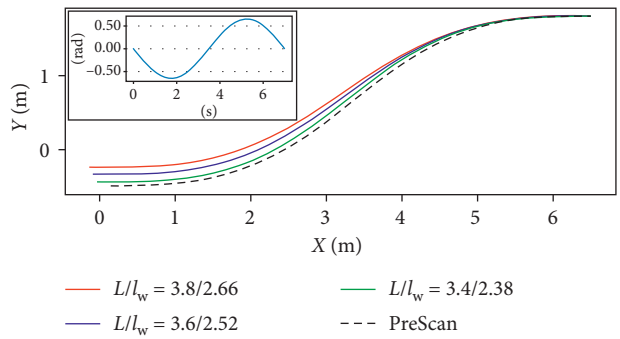


FIGURE 15: Simulated trajectories of vehicles with different geometric parameters as a result of equivalent steering control input shown in the figure. L is the length of the vehicle and l_w is a distance between the centers of the front and rear wheelbase.

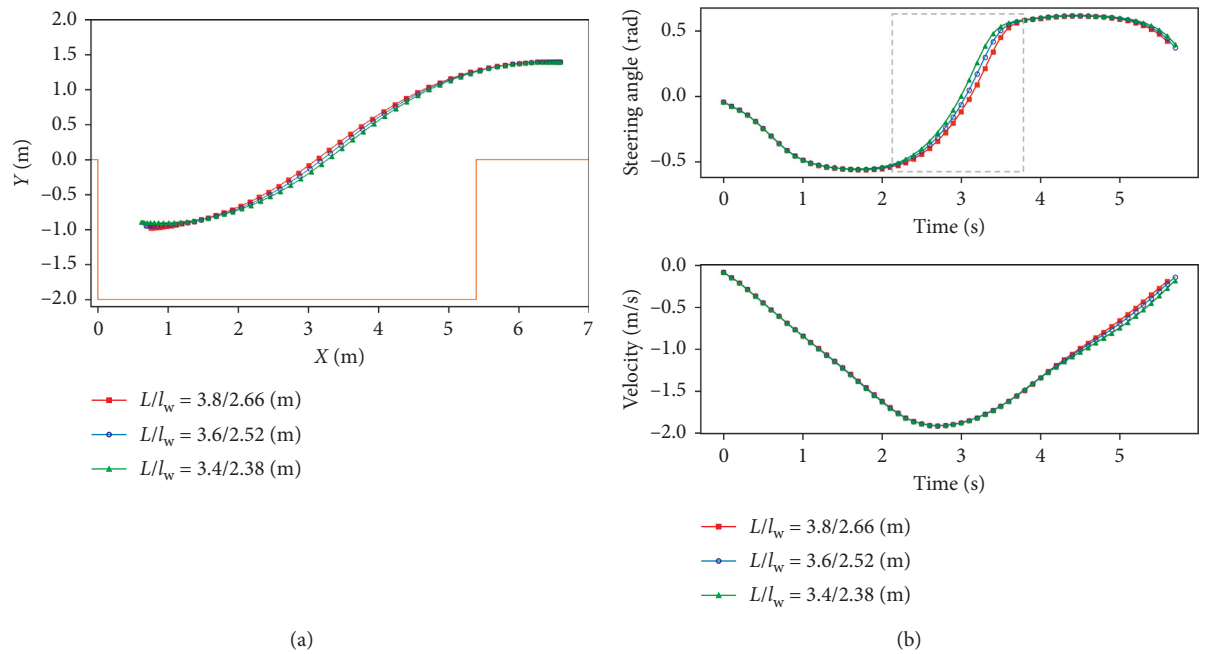
of ($L/l_w = 3.6/2.53$) include the kinematic bicycle model parameters used in dynamic optimization for training dataset generation.

We performed more than 50,000 simulations in different environments and RRPs to verify the tolerance of the proposed parking agent. The ANN model was trained by a dataset generated using the geometric parameters ($L/l_w = 3.6/2.53$). In this subsection, we illustrate two typical cases, one for parking with a single gear-shift, and another for multiple maneuvering of gear-shifts for back-and-forth movements in a narrow parking space. In the simulation depicted in Figures 16 and 17, the kinematic parameters of the main vehicle and the virtual vehicle in the cloned vehicle-in-the-loop are the same. The steering angle and velocity profiles are adjusted to reflect variations in vehicle kinematic parameters in the loop. The ANN was trained with the vehicle kinematic parameter ($L/l_w = 3.6/2.53$); therefore, the vehicle model used to train the ANN model is different from the vehicle model in the main and the cloned loop. The simulation results demonstrate that the trained twin ANN agent can drive the main vehicle even with multiple back-and-forth movements in a confined space, regardless of specific variations in the kinematic model. The proposed automatic parking agent can output adjusted actions without retraining using revised model parameters to adapt the specific variations in kinematic parameters.

4.4. Proposed Controller with a Twin Architecture Using PreScan Simulator. In this subsection, we determine whether the proposed parking controller works in a vehicle simulator while adapting to changes in the vehicle model in terms of kinematics and dynamics. We simulated the proposed twin architecture using the PreScan driving simulator in several parking environments. The PreScan driving simulator builds a virtual validation environment with realistic configurations using a vehicle dynamics model [28]. The main vehicle in the twin architecture implemented in the PreScan simulator includes the same kinematic model parameters ($L/l_w = 3.6/2.53$) used for data generation. However, the primary vehicle model is the 4-wheel vehicle dynamics model which closely resembles the real vehicle than the simple kinematic bicycle models used for generating training data. The dynamics model consists of the vehicle components of an engine, transmission, chassis, and gear-shift logic. The controllable steering wheel angle range of the vehicle is from −500° to +500°. The sign of a steering wheel angle indicates the direction of the steering wheel, which can be negative for clockwise and positive for counterclockwise movements, and the steering ratio of the vehicle is set to 15 : 1. In addition, gear-shift logic is considered as in the real vehicle when the direction of the vehicle must be changed for multiple maneuvering. The gear-shift logic changes from drive mode to reverse mode and vice versa with a 0.8 second delay. The logic holds the velocity at 0 while the gear is changed. Except for the kinematic parameters, other parameters of the dynamics model are referred to the default values of Yaris vehicle model in PreScan simulator. Additional details on the vehicle model that PreScan provides can be found in [28].

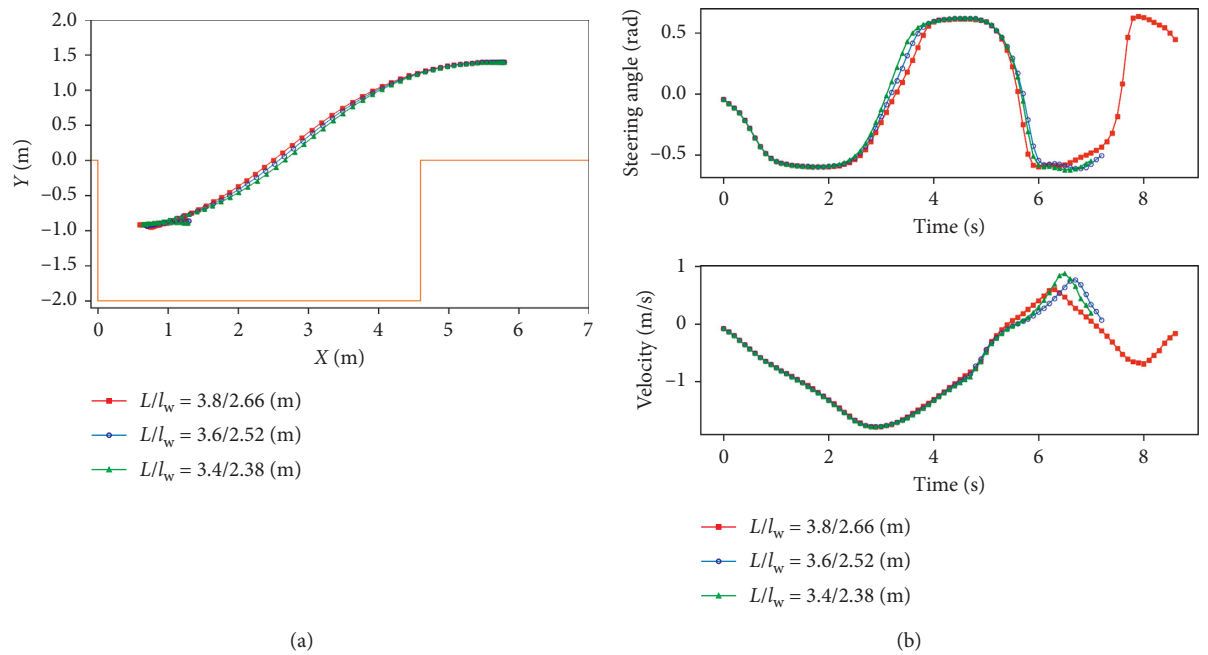
A block diagram of an automatic parking system via PreScan is presented in Figure 18. A software-in-the-loop (SIL) configuration is implemented with the main vehicle in the PreScan simulator and the twin architecture controller on a Caffe deep learning framework. The two separated parts communicate vehicle reference inputs and vehicle current states via a UDP network. PreScan sends the vehicles states ($x(k)$, $y(k)$, and $\Psi(k)$). The parking agent with a twin controller infers the adjusted vehicle actions ($\widehat{v}(k)$ and $\widehat{\delta}(k)$) from the vehicle states. By considering the velocity from the agent, the longitudinal low-level controller in PreScan yields the vehicle's throttle/brake inputs.

We performed 100 simulations under different environments and RRPs to verify the tolerance of the proposed parking agent. We illustrate two selected typical cases, one for parking with two gear-shifts, and another for multiple maneuvering of gear-shifts for back-and-forth movements in a narrow parking space. Figure 19 illustrates the simulated results of the proposed twin ANN agent via the PreScan simulator in the case of SL = 4.8 and RRP at position (x_{RRP}, y_{RRP}) = (6.0, 1.2), which are the same initial conditions of the results shown in Figures 10 and 12 while the proposed controller operates with the kinematics vehicle model. Figure 20 presents the result of the trajectories in the case of SL = 4.4 m and RRP at position of (x_{RRP}, y_{RRP}) = (5.7, 1.4), which are the same initial conditions of the results shown in Figures 11 and 13. In the simulation presented in Figures 19 and 20, the kinematic parameters of



(a) (b)

FIGURE 16: Simulated results of the proposed twin architecture with different kinematic model parameters of a vehicle when SL = 5.4 and RRP at position $(x_{RRP}, y_{RRP}) = (6.6, 1.4)$. (a) Simulated profile of parking trajectories. (b) Steering angle profile and velocity profile.



(a) (b)

FIGURE 17: Simulated results of the proposed twin architecture with different kinematic model parameters of a vehicle when SL = 4.6 and RRP at position $(x_{RRP}, y_{RRP}) = (5.8, 1.4)$. (a) Simulated profile of parking trajectories. (b) Steering angle profile and velocity profile.

both the main vehicle and the virtual vehicle in the cloned loop are the same.

The velocity profiles depicted in Figures 19(b) and 20(b) present the desired velocity indicated by a solid line and the current velocity of the vehicle indicated by a dashed line. As we can observe, there is a delay in car motion. Changes in gear-shift presented in Figure 20 increase from three times to five times compared to the results in Figure 13. Nevertheless, the simulation results demonstrate that the trained twin ANN agent can park the vehicle even with multiple back-and-forth movements in a very tiny space, regardless of variations of the vehicle model and longitudinal control. The steering angle and velocity profiles are adjusted to reflect variations in vehicle parameters. The proposed automatic

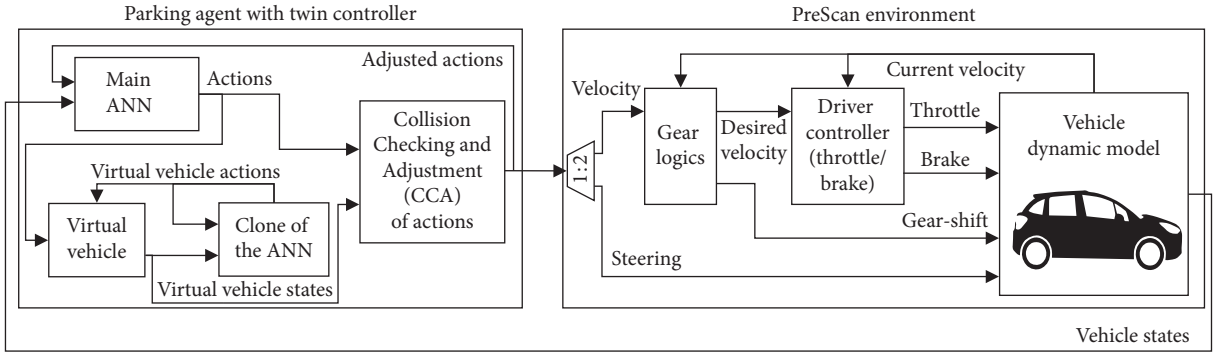


Figure 18: Block diagram of proposed twin architecture of parking controller via the PreScan simulator.

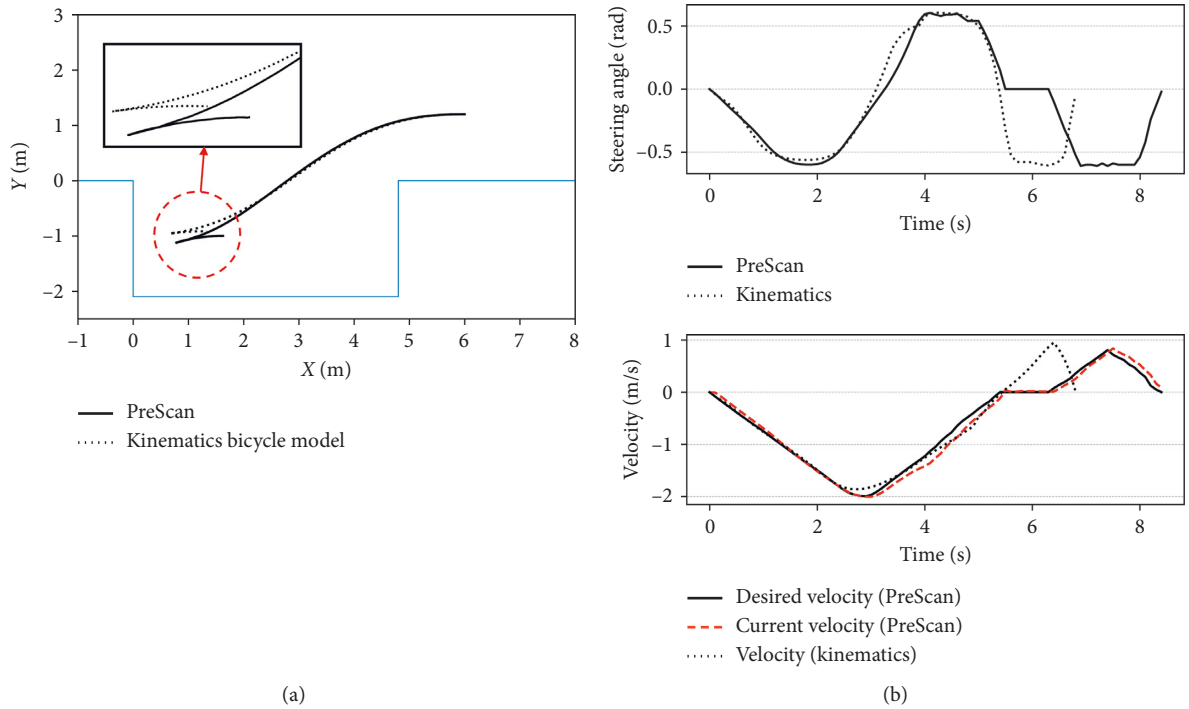


Figure 19: Simulated results via the PreScan simulator when SL = 4.8 and RRP at position $(x_{RRP}, y_{RRP}) = (6.0, 1.2)$. (a) Simulated profile of parking trajectories. (b) Steering angle profile and velocity profile.

parking agent can output adjusted actions without retraining using revised model parameters in adapting to variations in the vehicle models.

5. Discussions

Through simulations, we confirmed that the proposed twin agent with a trained ANN exhibits automatic parking with human-like parking skills. We further demonstrated the robustness of the proposed twin agent with respect to variations in kinematic parameters and environmental changes in the parking slot length and RRP. The time taken to complete the automatic parking process indicates a measure of the cost function for determining the ease with which the parking task can be achieved. As we mentioned in the previous section, the ANN was trained with a dataset generated by the vehicle parameters $(L/l_w = 3.6/2.52)$. We simulated the automatic parking task for different vehicle kinematic parameters and environments, as shown in Table 3. For Case 2 and Case 3, we set various kinematic parameters for the main and virtual vehicles of the twin agent using the trained vehicle parameters. We set different kinematic parameters for the main and virtual vehicles in Case 4. For cases 1–4, we set the same initial heading angle of the vehicle as 0° while altering the kinematic parameters. In cases 5 and 6, we set the same kinematic parameters; but we set different initial heading angles that are not trained for the ANN model.

We selected three different sizes of parking slot lengths: a tiny space (4.4 m) for multiple gear-shifts, a middle space (4.9 m) for mostly one or two gear-shifts, and a wide space (5.4 m) for easy automatic parking without multiple gear-shifts. For simulations in each case, we select 10,000 starting points that are not included in the RRP of the training dataset.

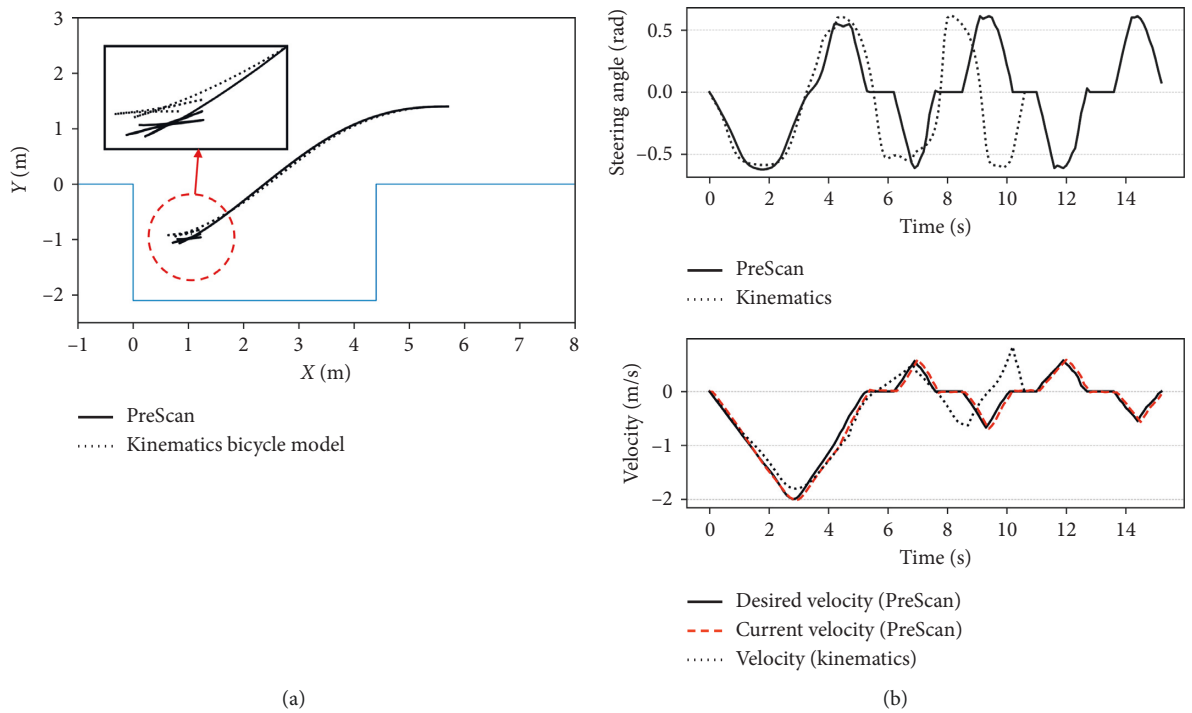


Figure 20: Simulated results via the PreScan simulator when SL = 4.4 and RRP at position $(x_{RRP}, y_{RRP}) = (5.7, 1.4)$. (a) Simulated profile of parking trajectories. (b) Steering angle profile and velocity profile.

Table 3: Success rate of automatic parking depending on test conditions.

Test condition	Kinematic parameters of main/virtual vehicle (L/l_w)	Initial heading angle (°)	No. of test	No. of successes	No. of failures	Success rate (%)
Case 1	(3.6/2.52)/(3.6/2.52)	±0	10,000	9,941	59	99.4
Case 2	(3.4/2.38)/(3.4/2.38)	±0	10,000	9,901	99	99.0
Case 3	(3.8/2.66)/(3.8/2.66)	±0	10,000	9,546	454	95.5
Case 4	(3.4/2.38)/(3.6/2.52)	±0	10,000	9,978	22	99.8
Case 5	(3.6/2.52)/(3.6/2.52)	+3	10,000	9,799	201	98.0
Case 6	(3.6/2.52)/(3.6/2.52)	+5	10,000	9,436	574	94.4

Table 3 presents a summary of the simulated success rate and number of failed trials of automatic parking, depending on the parking environments and vehicle parameters of the main and virtual vehicle. We defined the fail condition if the main vehicle fails to reach the end position that satisfies the requirement defined in ISO 16787 within 21 s.

Figure 21 illustrates the simulated contour plots for the time taken to complete automatic parking with respect to the starting positions and slot length of parking lots. The contour plots include a cost function of parking as a function of the environment and RRP; when the agent selects an RRP, the contour plot provides a good guideline for choosing the starting point of automatic parking.

In the simulations, the time taken to complete automatic parking ranges from 5 seconds to 21 seconds. We chose a reference time of 21 seconds to determine whether a parking task is successful. If the parking task was not completed within 21 seconds, we consider it a fail case. When the vehicle parameters of the training data are equivalent to the main vehicle and virtual vehicle, the simulated success rate is 99.4%, as shown in Case 1. The fails occurred among 10,000 trials only at the near upper boundary and the lower right-end boundary of the tiny spaces when SL = 4.4 m, as shown in Figure 21(a). For a slot length larger than 4.9 m, most trials were successfully completed.

In Case 2, the main vehicle length is shorter than that of the trained vehicle. Most of the trials succeeded, except for a certain boundary of the region when the slot length was 4.4 m even though the conditions of vehicle parameters were not trained.

In Case 3, the proposed parking agent failed to complete the maneuver in almost 25% of the trials when the parking slot length was 4.4 m. The mathematical limitation of the slot length should be larger than the diagonal distance of the vehicle. When the slot length was 4.4 m, the slot length of the lot was only 6.7% longer than the diagonal length of the main vehicle (which was 4.1 m hence $L = 3.8$ and width $(2 \times b) = 1.6$ m). Even for a skilled human driver, it is very difficult to park when the slot length is only 30 cm longer than the vehicle's diagonal length. Nevertheless, with limited tiny

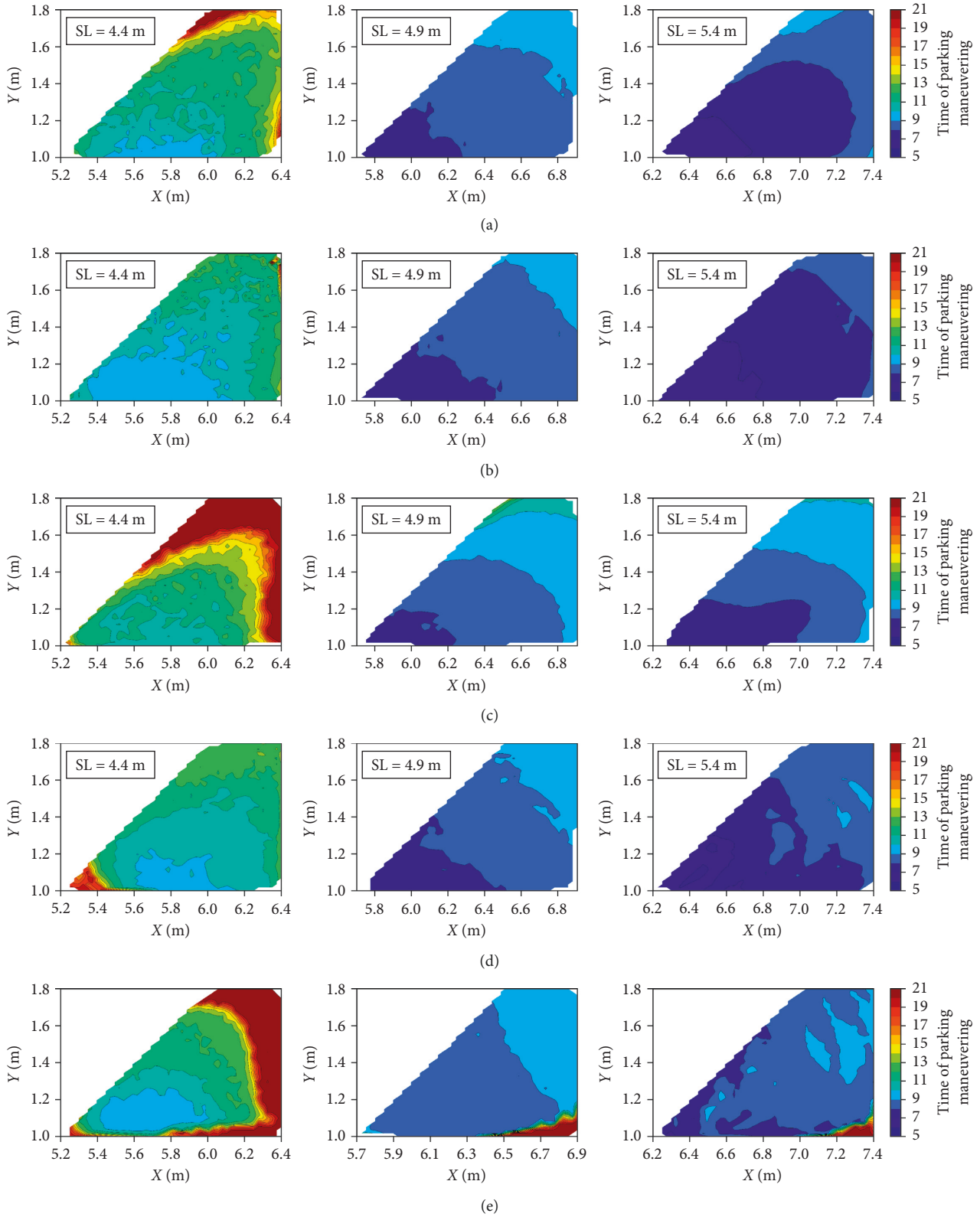


Figure 21: Contour plots for time taken to complete automatic parking with respect to RRP and slot length of parking lots. (a) Test condition: Case 1 (main/virtual vehicle) = (3.6/2.52)/(3.6/2.52); $\Psi_{init} = 0$. (b) Test condition: Case 2 (main/virtual vehicle) = (3.4/2.52)/(3.4/2.52); $\Psi_{init} = 0$. (c) Test condition: Case 3 (main/virtual vehicle) = (3.8/2.52)/(3.8/2.52); $\Psi_{init} = 0$. (d) Test condition: Case 4 (main/virtual vehicle) = (3.4/2.52)/(3.6/2.52); $\Psi_{init} = 0$. (e) Test condition: Case 6 (main/virtual vehicle) = (3.6/2.52)/(3.6/2.52); $\Psi_{init} = 5$.

parking slot length, the proposed parking agent succeeded in 75% of trials under this harsh condition.

In Case 4, we set different kinematic parameters for the main and virtual vehicle. The model parameters of the vehicle may vary with environmental changes such as tire pressure, weight and distribution of mounted load, and aging of the vehicle, which causes uncertainty in the kinematic and dynamic models, even for the same individual vehicle. Therefore, we exaggerate by changing the parameters of the main vehicle to verify the tolerance. Most of the trials succeeded, except for a certain boundary of the region when the slot length was 4.4 m even though the conditions of vehicle parameters were not trained. The success rate is comparable to that in Case 1 and even higher success rate when only the main vehicle is smaller while the virtual vehicle is equivalent to the trained vehicle model.

In Case 5 and Case 6, the fails occurred in almost 20% of the trials when the parking slot length is 4.4 m. Most of failure trials occurred at a certain boundary of the region. If we choose a starting point around the core area and not in the periphery of the RRP region shown in the contour plots, a success rate of 100% can be achieved, for all simulated cases.

Simulation results of variations in kinematic parameters provide substantial evidence that the proposed twin agent learns parking skills in a manner such that it can successfully execute the parking maneuver of a car. The proposed technique can handle inexact and imprecise problem domains and has been demonstrated to be useful in solving the parking problem under cases of uncertain knowledge of a vehicle's kinematics or dynamics. By applying ANN, the agent learns the control laws of the vehicle through a dataset-based pseudo-experience, and in particular, by cooperating with the virtual agent, the trained main agent provides the correct actions for each specific situation. The proposed ANN-based controller may substantially simplify the development process; in fact, it uses a training dataset generated by simultaneous dynamic optimization, which is a very well-known and low-cost method of generating training datasets.

An artificial intelligence system may present a serious problem in the inference stage if it encounters unexperienced situations during operation. Simulated experiments of both single and twin agents have shown that an automatic parking agent comprising a twin architecture performs human-like parking maneuvers by predicting the consequence of tentative actions and adjusting the action applied to the vehicle if a collision predicted. Predicting a consequence of tentative actions enables the automatic parking agent to cope with the potential risk of collision under unexperienced situations without increased computational burden or design complexity.

Because the proposed work was verified in simulation environment, some issues have remained for future work in order to implement proposed automatic parking controller to real-world scenarios with real vehicles. In the simulation environment, simulator provides precise or even exact values of the vehicle's current pose. However, it is hard to get a very precise pose of the vehicle in real-world application. In the future work, we must consider in-vehicle sensors for environmental perception and detecting potential collisions with errors, imprecisions, and noise. Moreover, we need to consider a reactive architecture or a hybrid control architecture [16], based on real-world sensors to avoid collisions.

6. Conclusion

We proposed an automatic parking agent to implement human-like parking skills using ANNs that comprise a main agent and its cloned agent. By learning the relationships between the current vehicle states and the corresponding actions from the generated dataset through simultaneous dynamic optimization, the proposed approach yields a direct steering angle and velocity and executes automatic parking in a tiny space with multiple maneuverings. The simulation results demonstrate that the proposed twin agent for automatic parking emulates the attributes of a human driver including the inherence concept of a vehicle model and preview utilization. The main agent decides the potential actions, and the cloned agent simulates and pre-evaluates the consequences of the potential actions that produce results. Following this, the Collision Checking and Adjustment (CCA) system adjusts the actions applied to the main vehicle. We validated that the trained twin ANN agent can drive the main vehicle, even with multiple back-and-forth movements in confined spaces, regardless of variations in the kinematic model, through simulations. The proposed automatic parking agent can output adjusted actions without retraining in adapting to variations in kinematic parameters. The agent can complete parking without retraining under several conditions of ±5% increments in the vehicle's overall length and wheel-base length and 5° increments of the initial heading angle of the vehicle. Our proposed parking agent operates like a human driver in a manner that is not exactly planned and not exactly modelled for the kinematics of a subject vehicle; that is, vehicle model fluctuations are considered by applying the trained model to other vehicles with different kinematic parameters.

We only verified the proposed work in simulation environments, there are many remaining issues in a future work, to test and implement the proposed approach in real-world scenarios with real vehicles.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the "ICT Consilience Creative Program" (IITP-2019-2017-0-01015) supervised by the IITP (Institute for Information and communication Technology Planning and Evaluation).

Supplementary Materials

The supplementary materials include generated datasets of this work. Video of automatic parking in a tiny space with back-and-forth shuffling using PreScan simulator is available at: https://youtu.be/UI6csjG5qdQ. (*Supplementary Materials*)

References

[1] F. Gómez-Bravo, F. Cuesta, A. Ollero, and A. Viguria, "Continuous curvature path generation based on -spline curves for parking manoeuvres," *Robotics and Autonomous Systems*, vol. 56, no. 4, pp. 360–372, 2008.

[2] J. Moon, I. Bae, J. G. Cha, and S. Kim, "A trajectory planning method based on forward path generation and backward tracking algorithm for Automatic Parking Systems," in *Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems (ITSC) 2014*, pp. 719–724, Qingdao, China, October 2014.

[3] H. Vorobieva, S. Glaser, N. Minoiu-Enache, and S. Mammar, "Automatic parallel parking in tiny spots: path planning and control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 396–410, 2015.

[4] X. Du and K. K. Tan, "Autonomous reverse parking system based on robust path generation and improved sliding mode control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1225–1237, 2015.

[5] P. Zips, M. Böck, and A. Kugi, "Optimisation based path planning for car parking in narrow environments," *Robotics and Autonomous Systems*, vol. 79, pp. 1–11, 2015.

[6] B. Li and Z. Shao, "Simultaneous dynamic optimization: a trajectory planning method for nonholonomic car-like robots," *Advances in Engineering Software*, vol. 87, pp. 30–42, 2015.

[7] B. Li and Z. Shao, "A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles," *Knowledge-Based Systems*, vol. 86, pp. 11–20, 2015.

[8] B. Li, K. Wang, and Z. Shao, "Time-optimal maneuver planning in automatic parallel parking using a simultaneous dynamic optimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3263–3274, 2016.

[9] J. Moon, I. Bae, and S. Kim, "Real-Time near-optimal path and maneuver planning in automatic parking using a simultaneous dynamic optimization approach," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 193–196, Los Angeles, CA, USA, June 2017.

[10] D. McRuer, "Human dynamics in man-machine systems," *Automatica*, vol. 16, no. 3, pp. 237–253, 1980.

[11] C. C. Macadam, "Understanding and modeling the human driver," *Vehicle System Dynamics*, vol. 40, no. 1–3, pp. 101–134, 2003.

[12] A. V. Phatak and G. A. Bekey, "Model of the adaptive behavior of the human operator in response to a sudden change in the control situation," *IEEE Transactions on Man Machine Systems*, vol. 10, no. 3, pp. 72–80, 1969.

[13] A. Y. Ungoren and H. Peng, "An adaptive lateral preview driver model," *Vehicle System Dynamics*, vol. 43, no. 4, pp. 245–259, 2005.

[14] W. Daxwanger and G. K. Schmidt, "Skill-based visual parking control using neural and fuzzy networks," in *Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, vol. 2, no. 1, pp. 1659–1664, Vancouver, BC, Canada, October 1995.

[15] W. A. Daxwanger and G. Schmidt, "Neural and fuzzy approaches to vision-based parking control," *Control Engineering Practice*, vol. 4, no. 11, pp. 1607–1614, 1996.

[16] M. Heinen, F. S. Osorio, F. J. Heinen, and C. Kelber, "SEVA3D: Using artificial neural networks to autonomous vehicle parking control," in *Proceedings of the 2006 IEEE International Joint Conference on Neural Network*, pp. 4704–4711, Vancouver, BC, Canada, July 2006.

[17] R. Li, W. Wang, Y. Chen, S. Srinivasan, and V. N. Krovi, "An end-to-end fully automatic bay parking approach for autonomous vehicles," in *Proceedings of the ASME 2018 Dynamic Systems and Control Conference. Volume 2: Control and Optimization of Connected and Automated Ground Vehicles; Dynamic Systems and Control Education; Dynamics and Control of Renewable Energy Systems; Energy Harvesting, p.* V002T15A004, Atlanta, Georgia, USA, September–October 2018.

[18] W. Liu, Z. Li, L. Li, and F. Y. Wang, "Parking like a human: a direct trajectory planning solution," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 12, pp. 3388–3397, 2017.

[19] A. Wächter and L. T. Biegler, "Line search filter methods for nonlinear programming: motivation and global convergence," *SIAM Journal on Optimization*, vol. 16, no. 1, pp. 1–31, 2005.

[20] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[21] L. T. Biegler, "Nonlinear programming: concepts, algorithms and applications to chemical processes," *Siam*, vol. 40, no. 6, 2010.

[22] R. Lougee-Heimer, "The Common Optimization INterface for Operations Research: promoting open-source software in the operations research community," *IBM Journal of Research and Development*, vol. 47, no. 1, pp. 57–66, 2003.

[23] K. Kondak and G. Hommel, "Computation of time optimal movements for autonomous parking of non holonomic mobile platforms," in *Proceedings of the 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 3, pp. 2698–2703, Seoul, South Korea, May 2001.

[24] ISO 16787:2017 Intelligent transport systems—assisted Parking System (APS)—performance requirements and test procedures, https://www.iso.org/standard/73768.html, 2019.

[25] D. Hunter, H. Yu, M. S. Pukish III, J. Kolbusz, and B. M. Wilamowski, "Selection of proper neural network sizes and architectures—a comparative study," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 228–240, 2012.

[26] K. G. Sheela and S. N. Deepa, "Review on methods to fix number of hidden neurons in neural networks," *Mathematical Problems in Engineering*, vol. 2013, Article ID 425740, 11 pages, 2013.

[27] Y. Jia, "Caffe: convolutional architecture for fast feature embedding," 2014, https://arxiv.org/abs/1408.5093.

[28] Tass International, *PreScan: Simulation of ADAS and Active Safety*, Tass International, Helmond, Netherlands, 2018.