applied sciences

MDPI

*Article*

# Decision-Making in Fallback Scenarios for Autonomous Vehicles: Deep Reinforcement Learning Approach

**Cheonghwa Lee and Dawn An ***

Advanced Mechatronics R&D Group, Daegyeong Division, Korea Institute of Industrial Technology, Daegu 42994, Republic of Korea; haya.c.lee@gmail.com
* Correspondence: dawnan@kitech.re.kr

**Abstract:** This paper proposes a decision-making algorithm based on deep reinforcement learning to support fallback techniques in autonomous vehicles. The fallback technique attempts to mitigate or escape risky driving conditions by responding to appropriate avoidance maneuvers essential for achieving a Level 4+ autonomous driving system. However, developing a fallback technique is difficult because of the innumerable fallback situations to address and eligible optimal decision-making among multiple maneuvers. We employed a decision-making algorithm utilizing a scenario-based learning approach to address these issues. First, we crafted a specific fallback scenario encompassing the challenges to be addressed and matched the anticipated optimal maneuvers as determined by heuristic methods. In this scenario, the ego vehicle learns through trial and error to determine the most effective maneuver. We conducted 100 independent training sessions to evaluate the proposed algorithm and compared the results with those of heuristic-derived maneuvers. The results were promising; 38% of the training sessions resulted in the vehicle learning lane-change maneuvers, whereas 9% mastered slow following. Thus, the proposed algorithm successfully learned human-equivalent fallback capabilities from scratch within the provided scenario.

**Keywords:** autonomous vehicle; levels of driving automation; scenario-based testing; decision-making algorithm; deep reinforcement learning



**Citation:** Lee, C.; An, D. Decision-Making in Fallback Scenarios for Autonomous Vehicles: Deep Reinforcement Learning Approach. *Appl. Sci.* **2023**, *13*, 12258. https://doi.org/10.3390/app132212258

Academic Editors: Huizilopoztli Luna García, Hamurabi Gamboa-Rosales, Jose M. Celaya-Padilla and Antonio Martínez Torteya

Received: 20 October 2023
Revised: 9 November 2023
Accepted: 10 November 2023
Published: 13 November 2023



## 1. Introduction

Autonomous vehicles (AVs) are designed to navigate unexpected and potentially dangerous driving conditions by engaging in fallback maneuvers or achieving minimum risk conditions when necessary [1,2]. For example, an AV may need to be pulled to the roadside if a critical sensor fails or swiftly changes lanes to avoid suddenly stopping the vehicle ahead [3]. Current AVs from industry leaders such as Tesla, Mobileye [4], and Waymo operate at Level 2 or 3 autonomy, as defined by the Society of Automotive Engineers (SAE) [1]. AVs can handle certain risky situations at these levels, but they are engineered to transfer control back to a human driver when the system reaches its limits. Therefore, drivers must be ready to take over from either in the vehicle or from a remote location. The ultimate aim of AV technology is to achieve Level 4+ (4 and 5) autonomy, where vehicles can handle all driving tasks independently without human intervention, even in complex driving scenarios.

To understand the evolution of AVs, it is important to be familiar with SAE-defined levels of autonomy to understand the evolution of AVs. Starting from Level 0, where there is no automation, the scale progresses to Level 1, with basic driver assistance, such as cruise control. At Level 2, we observe partial automation that can control steering and acceleration under certain conditions. Level 3 allows the vehicle to take over many driving functions yet still requires a driver to intervene when prompted. Level 4 introduces the capability of the vehicle to operate independently in most environments, and Level 5 represents full automation: a vehicle that is fully autonomous in every driving scenario, eliminating

the need for any human intervention. The leap to autonomy in Levels 4 and 5 entails a complex blend of technologies enabling vehicles to navigate and respond to all driving situations autonomously.

Progression towards higher levels of AV technology is ready to yield transformative societal rewards such as increased safety by reducing driver error, enhanced mobility for those who cannot drive, and increased free time for drivers on the move. As we advance towards Level 4+ automation, the implications for society are immense, offering not only safer and more accessible transportation options but also the promise of streamlined traffic management, the reshaping of transportation across personal and public domains, and environmental benefits via lowered emissions. Achieving this high level of autonomous operation necessitates a harmonious blend of cutting-edge technologies, including advanced machine learning, sensor fusion, and computer vision, coupled with the development of inter-vehicle and vehicle–infrastructure communication capabilities. The evolution of these technologies is expected to lead to a future in which intelligent, interconnected vehicles will form an efficient smart transportation network.

Generic autonomous driving system (ADS) research has two primary focus areas: development and validation. The ADS research development phase focuses on the design of algorithms, hardware, and software for autonomous navigation. A holistic ADS is operationalized through a sequential triad of perception, decision-making, and control components. Employing this mechanism, an AV interprets its immediate driving environment by analyzing the data gathered from an ensemble of sensors. In the decision-making phase, viable actions are determined based on this interpretation. During the control phase, the AV performs optimally, ensuring adherence to its intended trajectory.

The decision-making component is the most challenging to optimize within ADS architecture. This challenge arises primarily because of two factors. First, they must manage various unpredictable and potentially dangerous situations. Although these two situations pose the same risk, they can differ depending on when and where they occur. Second, the decision-making process is complex. The best course of action for a vehicle can vary significantly depending on the specific fallback scenario, and several viable actions are often available in response to a single situation.

To address these issues, researchers typically employ two main methodologies: (1) heuristic-based techniques [5,6] and (2) data-driven learning strategies [7–9].

A *heuristic-based* approach in ADS development primarily relies on expert-defined rules to guide vehicular decision-making. These rules simplify the complex decisions encountered by vehicles, drawing on expert knowledge and the fundamental principles of physics. One primary advantage of heuristic methods is their predictability and transparency, which render vehicle behavior foreseeable and interpretable. Moreover, these methods offer faster deployment than certain data-driven methods. Historically, heuristics were found in early ADS Levels 1–2, such as advanced driver assistance systems (ADAS), exemplified in lane-keeping and emergency braking systems. However, they encounter challenges in addressing many complex and dynamic driving scenarios that often require frequent updates. The primary limitation is the dependence on human intervention, which comprises the objective of achieving ADS levels of 4+. Orzechowski et al. (2020) developed a scalable decision-making framework for automated driving utilizing a heuristic approach, emphasizing a hierarchical behavior-based architecture [10]. This system employs modular behavioral units combined in a bottom-up manner to devise more intricate driving behaviors. Although state-of-the-art automated-vehicle platforms often utilize finite-state machines, they lack transparency, scalability, and ease of maintenance. This model merges knowledge-based systems with behavior-based systems to harness the strengths of both. However, its reliance on human intervention affects its adaptability.

*Data-driven learning* approaches, particularly those rooted in deep learning, have led to advancements in ADS. These methods train an ADS to recognize patterns and make informed decisions by harnessing extensive datasets from diverse driving conditions. Reinforcement learning, an offshoot of this approach, enables the system to learn optimal

strategies iteratively by interacting with the environment. Such data-driven techniques demonstrate heightened adaptability, can handle intricate driving scenarios that may not be hard-coded, and often outperform heuristic methods in unpredictable situations; data-driven and reinforcement learning methods represent significant progress in vehicle autonomy. They introduced a dynamic learning process grounded in real-world experiences compared to more fixed heuristic approaches. Their effectiveness, particularly for tasks such as object detection, highlights their potential applications.

Consequently, these methods are increasingly considered the way forward and are likely to dominate the subsequent stages of autonomous driving. Owais et al. (2020) proposed a novel algorithm for generating Pareto-optimal paths in stochastic transportation networks by considering the variability in travel times correlated with traffic flow [11]. The algorithm utilizes a multi-objective analysis to provide a set of optimal paths for each origin–destination (O-D) pair based on successive simulations that reflect changing traffic conditions within different time slots. However, this study did not explicitly discuss the computational complexity or scalability of the algorithm, which may be important when applying this method to vast and complex networks. It also does not address the potential limitations in the accuracy of traffic-flow predictions, which may affect the reliability of the generated routes. Alshehri et al. (2023) presented a novel application of deep residual neural networks to estimate O-D trip matrices utilizing traffic-flow data from strategically placed sensors, inversely addressing the conventional traffic assignment problem [12]. This research seeks to determine the correct O-D matrix from traffic counts and the optimal number and placement of sensors for efficient data gathering. Despite this innovative approach, the study may face challenges, such as the need for high-quality historical traffic data for model training, computational complexity of the deep learning model, and practical constraints of sensor placement in real-world scenarios. Pini et al. (2023) developed SafePathNet, a novel machine-learning system designed for trajectory prediction and planning in AVs [13]. SafePathNet utilizes a unified neural network to anticipate future pathways for AVs and the surrounding road agents to avoid conventional rule-based methodologies. It integrates the attention mechanism of a transformer module with an expert mixing strategy to predict different trajectories and prioritize safety and reliability. SafePathNet chooses a trajectory in real-time decision-making based on safety metrics and the predicted likelihood. Simulators and real-world tests have underscored their safety and effectiveness, distinguishing them from contemporary data-driven methods. However, this study did not focus on achieving an ADS level of 4+.

By leveraging deep reinforcement learning, the proposed system significantly advances the adaptive capabilities of AVs, aiming at ADS Level 4+ autonomy. Deep reinforcement learning overcomes the limitations of heuristic and static rule-based systems, such as fuzzy logic and expert systems, enabling AVs to interpret extensive sensor data and iteratively refine their driving strategies from real-world interactions. Although fuzzy logic copes with ambiguity, and expert systems execute preprogrammed rules, they cannot self-optimize in response to novel stimuli. However, our deep reinforcement learning approach excels in assimilating experiences from unpredictable and intricate driving conditions, fostering a learning mechanism that continually enhances decision-making and adaptability. This empowers AVs with the decision-making agility for high-risk environments, ensuring robust performance across a diverse spectrum of real-world driving situations to achieve ADS Level 4+.

Second, a validation phase for AVs and their ADS is imperative to ensure their safety, reliability, and functionality. Although a wide array of tests is utilized in the development and validation process, the two critical testing methods for AVs and ADS are (1) miles driven [14–18] and (2) scenario-based testing [19–24]. The other validation approaches are presented in Appendix A.

*Miles-driven validation* quantifies the performance of an ADS by measuring the distance traveled by an AV without significant safety incidents. This method offers a direct and easily comprehensible metric and provides an ADS with a genuine and varied real-world

driving experience. However, these methods exhibit several limitations. Relying solely on mileage does not necessarily represent the diversity and complexity of the driving scenarios a vehicle encounters. Moreover, accruing substantial mileage for testing can be both resource intensive and time consuming, particularly when the objective is to validate a system's response to rare events. Shalev–Shwartz et al. (2017) asserted that evaluating the safety of AVs utilizing a mile-driven approach is problematic [4]. This method measures the number of miles that an AV drives without an accident and compares its safety with that of human drivers. To statistically prove that an AV is as safe as a human, it would need to drive 30 million miles, given a specific fatality probability. If the goal is for AVs to become significantly safer, this will increase to 30 billion miles. In addition, any software update can render previous tests obsolete, and there is concern regarding whether these miles genuinely represent real-world conditions.

*Scenario-based validation* emphasizes the evaluation of an ADS within specific predefined scenarios. These scenarios depict challenging or critical driving situations that can be drawn from real-world datasets or constructed based on conceivable yet complex events. This method enabled a more structured evaluation, ensuring the ADS was tested across various scenarios. The focused nature of this approach facilitates efficient testing, particularly in situations where assessing safety and performance is paramount. It also offers a mechanism for repeatedly reproducing challenging driving situations, which may be infrequent in standard driving but are vital for comprehensive validation. However, this methodology has several challenges. Constructing a comprehensive scenario library is paramount, and this process can be demanding.

Furthermore, despite extensive scenario libraries, the inherent unpredictability of real-world driving is not always captured. Galko et al. (2014) introduced a vehicle-hardware-in-the-loop system designed to prototype and validate an ADAS for vehicles [25]. The SERBER system provides a scenario-based testing platform for ADAS in vehicles. Instead of outdoor testing or conventional test benches, SERBER utilizes a controlled environment to simulate real-world scenarios. This approach addresses the challenges of the consistency and safety of conventional methods. By replicating real-world indoor scenarios without large spaces or high-speed mobile bases, SERBER offers a safe and effective ADAS validation process.

In our quest to advance ADS, we considered a data-driven learning method over a heuristic-based method. This approach enables our systems to adapt and improve their performance utilizing real-world data, enhancing their power and efficiency. When assessing the ADS performance, we did not rely solely on mileage. Instead, we prioritized simulation and evaluation under specific driving scenarios. This approach is instrumental for developing a safer and more reliable ADS by providing a clear understanding of the performance of the proposed system under challenging situations.

Here, we propose a decision-making algorithm based on deep reinforcement learning for a fallback scenario. Decision-making in fallback scenarios is an optimization technique in which autonomous vehicles select an appropriate optimal maneuver under certain risky driving conditions. The designed fallback scenario occurred when the vehicle in front of the ego vehicle decelerated rapidly, and the vehicle on the right drove at high speed with three vehicles driving on a highway. We derived three expected optimal maneuvers in advance: slow following, lane change, and lane change after yielding. We then formulated a given fallback scenario based on deep reinforcement learning. In addition, we adopted a proportional-integral-differential (PID) control algorithm for low-level control and provided closed-form equations. Therefore, the ego vehicle was trained in a fallback scenario employing the proposed decision-making algorithm.

The contributions of this study include significant advancements in the algorithm development and testing methodologies for ADS.

1. **Learning Optimal Policy without Human Intervention**: We introduced an algorithm to mimic human-driver decision-making. Uniquely developed from scratch, it exhibits exceptional adaptability across different fallback scenarios. It was rooted in a "learning from scratch" approach and possessed inherent flexibility and versatility. This algorithm captures the characteristics of human decision-making. Thus, they can efficiently return to a myriad of diverse driving situations.
2. **Scenario-Based Simulation Testing for Efficient Validation**: The test system setup process was streamlined, minimizing the required time to a few hours. A notable aspect of our methodology is its open-source framework, which ensures broad accessibility at no cost. While conventional deterministic and heuristic methods often struggle in varied environments, reinforcement learning enhances the adaptability of our model, enabling it to address fallbacks even in uncharted situations.
3. **Pioneering Level 4+ ADS Deployment**: Previous efforts utilizing scenario-based reinforcement learning primarily targeted Level 1–3 autonomy, wherein human intervention still played a role. Our research attempts to enhance ADS for Level 4+ operations, wherein the system operates autonomously. Unlike earlier studies with potentially vague objectives, our approach was methodically tailored with a clear objective to achieve Level 4+ autonomy. This emphasizes our contributions to evolutionary and groundbreaking ADS.

Thus, this study provides innovative algorithmic developments, advanced testing techniques, and a pioneering vision for next-generation ADS autonomy.

The remainder of this paper is organized as follows: Section 2 examines the background knowledge on deep reinforcement learning, which forms the basis for the proposed decision-making algorithm. In Section 3, a fallback scenario is designed, and the optimal fallback is derived with a heuristic method. Section 4 formulates the designed fallback scenario with respect to deep reinforcement learning. Section 5 details the low-level control algorithm, system, network, and hyperparameters of the decision-making algorithm. Section 6 evaluates and discusses whether the results of the optimal avoidance maneuver and reinforcement learning derived with the heuristic method are the same as those in the proposed fallback scenario. Finally, Section 7 summarizes the proposed research and describes future research directions.

## 2. Background

The proposed decision-making algorithm is primarily based on reinforcement learning. Recently, naïve reinforcement learning has utilized artificial neural networks of supervised learning as a plugin to overcome the typical drawbacks (discrete) and achieve synergy through collaboration. Various advanced deep reinforcement learning algorithms have been developed, for example, deep deterministic policy gradient (DDPG), trust region policy optimization (TRPO), proximal policy optimization (PPO), etc. These advanced algorithms demonstrated outstanding results in various applications. However, we implemented the proposed algorithm as a deep Q-network (DQN), which is outdated but considered adequate for our research for the following reasons. The DQN is a neural network that renders the state space continuous and the action space discrete. Furthermore, it has a lower computational strength than the above algorithms.

Reinforcement learning is a machine-learning algorithm that learns optimal decision-making patterns based on sequential interaction data between agents and the environment. The aim is to determine the optimal policy $\pi^*$. In the current state $s$ with the optimal policy, the agent selects the optimal action $a$ that is expected to yield the largest total reward $Q(s,a)$ in the future.

$$\pi^*(s) = \underset{a}{\text{argmax}} Q(s,a),$$

where $Q(s,a)$ denotes the state–action value function. A state–action value function that adopts an artificial neural network is called deep reinforcement learning. Because the artificial neural network contains the weights, we denote it again as $Q(s,a;\theta)$ and refer

to it as the Q-network. As the interaction continued, more accurate and diverse driving data containing collision and/or collision-free data were utilized to train the Q-network. Therefore, the Q-network network exhibits higher values.

Sequential interaction data for learning the Q-network were collected through the interaction between the agent and the environment. The agent receives State $s$ from the environment and outputs action $a$. The environment receives the behavior of the agent and outputs the next state $s'$ and reward $r$. Through one interaction, data $d$ are collected as $(s, a, s', r)$.

The training data for the Q-network were such that the input data were the state $s$, and the output data were the temporal difference (TD) target $r + \gamma \max Q(s', a'; \theta)$, calculated from the interaction data where $a'$ is an arbitrary next action.

The Q-network is an approximation function that minimizes the optimization problem by estimating training-output data. The loss function of the Q-network is expressed as

$$L(\theta) = \frac{1}{2}\left[\{r + \gamma \max Q(s', a'; \omega)\} - Q(s, a; \theta)\right]^2.$$

The gradient is as follows:

$$\nabla_\theta L(\theta) = \left[\{r + \gamma \max Q(s', a'; \theta)\} - Q(s, a; \theta)\right] \nabla_\theta Q(s, a; \theta).$$

It is updated with a batch and determines the minimum such that stochastic gradient descent is utilized to minimize the objective function and gradient value.

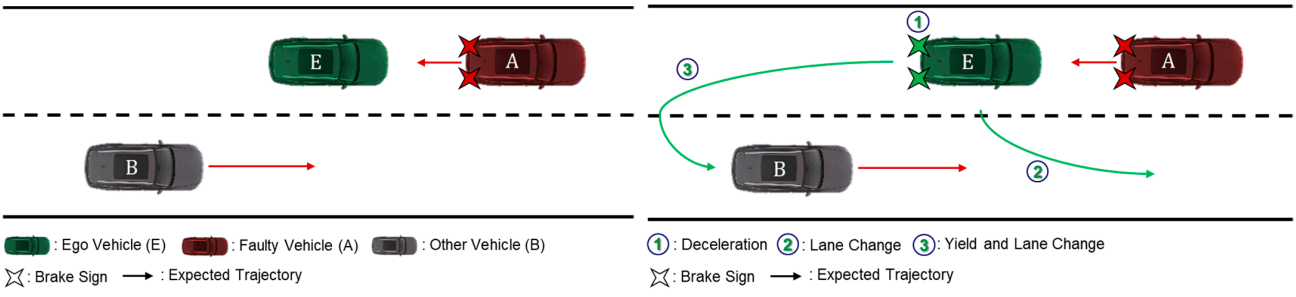
$$\theta' = \theta + \alpha \nabla_\theta L(\theta),$$

Reinforcement learning algorithms repeat learning until an optimal policy is found.

The remainder of the detailed implementation-associated background, such as hyper-parameters, is explained in Section 5.

## 3. Fallback Scenario Design

We designed a fallback scenario that included the risks to be addressed, as illustrated in Figure 1. The fallback scenario describes the risky driving conditions with assumptions and optimal maneuvers based on heuristics.




(**a**) (**b**)

**Figure 1.** Fallback scenario design: (**a**) risk driving conditions, and (**b**) optimal fallback maneuvers through heuristic method.

### *3.1. Driving Conditions with Risks and Assumptions*

As illustrated in Figure 1a, three vehicles were driven on a two-lane straight highway. Vehicle A was a defective vehicle (resulting in risky driving conditions for the ego vehicle) that experienced rapid deceleration due to a mechanical error and drove at a constant low speed. Vehicle B was a normal vehicle that continued to drive at high speed in the right lane. The ego vehicle began the fallback execution immediately after recognizing a potential front-end collision owing to the front-defective Vehicle A and a potential side collision owing to the high-speed drive of Vehicle B in the right lane.

In this scenario, it was assumed that the recognition and control of the autonomous vehicles were perfect. The proposed fallback approach assumes that the driver of the surrounding vehicle performs predetermined driving tasks. We designed a fallback scenario with three vehicles because the maneuver chosen by the ego vehicle while avoiding a defective vehicle may affect the normal vehicle. The utilization of two lanes provides numerous opportunities to perform diverse maneuvers. A highway was selected because of its ease of handling. The scope of learning was not to train in general driving but to learn appropriate decision-making under one fallback situation. Therefore, driving before and after the occurrence of fallback was not considered.

### 3.2. Optimal Maneuvers Based on Heuristics

The decision-making process in the fallback scenario was for the ego vehicle to learn to reach a target point without colliding with the surrounding environment or Vehicles A and B. The optimal maneuvers were expected to be appropriate under the given fallback state. The easiest method for deriving optimal maneuvers is based on a heuristic method that aligns with the driver's common sense. At this time, the ego vehicle selects one of the three fallback maneuvers. The three heuristic maneuvers are illustrated in Figure 1b.

- *Slow following* involves rapid deceleration according to the deceleration speed of the front defective vehicle. This fallback maneuver only needs to consider the front defective vehicle and not its interaction with the vehicle on the right side. However, this does not make sense as a normal or efficient driving policy. Thus, this maneuver is easy but inefficient.
- *Lane change* involves changing to the right lane. At this time, the ego vehicle must consider the distance to the front defective vehicle, distance to the right vehicle, and its speed.
- *Lane change after yield* occurs after the side vehicle passes through the ego and front vehicles. This maneuver is also reasonable but requires two abstract maneuvers; thus, it is expected to be difficult to learn.

These heuristic-based maneuvers are adopted as metrics to evaluate the decision-making algorithms.

## 4. Fallback Scenario Formulation

To formulate a given fallback scenario, we mapped it to reinforcement learning concepts, such as agent, environment, and learning goals.

*Agent* denotes the ego vehicle, and the *environment* denotes the driving conditions, including the ego vehicle and two other vehicles, Vehicles A and B. The ego vehicle set the *objective* of learning to reach the target destination without colliding with Vehicles A and B and did not cross either side.

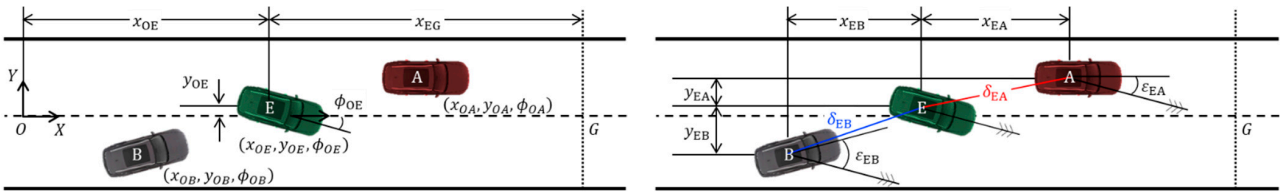
### 4.1. State and Perception

*State s* denotes an ego vehicle that collects physical information to perceive surrounding vehicles and road environments. It comprises nine parameters, as expressed in Equation (1).

$$s = (x_{EG}, y_{OE}, \phi_{OE}, x_{EA}, y_{EA}, \phi_{EA}, x_{EB}, y_{EB}, \phi_{EB}). \quad (1)$$

The details of the coordination and spatial variables are presented in Figure 2. These nine parameters represent three pieces of information: goal-related, localization, and relationship states. A goal-related state is the information utilized to achieve a learning goal. The ego vehicle attempts to reach the goal line such that the goal-related state is the longitudinal relative distance $x_{EG}$ between the current $x$ position of the ego vehicle $x_{OE}$ and the goal line $x_{OG}$. The localization state represents the information through which an ego vehicle perceives its location in the environment. The chosen localization state parameters were the lateral absolute position $y_{OE}$ and rotational absolute yaw angle $\phi_{OE}$, as illustrated in Figure 2a. The longitudinal absolute position $x_{OE}$ was ignored because it was already

included in the goal-related state $x_{EG}$ which is the distance between the ego vehicle and the goal position. The relationship state is the information between the ego vehicle and the other vehicles. This state is scalable depending on the number of surrounding vehicles, and each relationship state has three degrees of freedom for each vehicle: one longitudinal and lateral distance each and one rotation angle $(x, y, \phi)$, as illustrated in Figure 2b. The ego vehicle had six relationship state parameters: $x_{EA}$, $y_{EA}$, and $\phi_{EA}$ for Vehicle A and $x_{EB}$, $y_{EB}$, and $\phi_{EB}$ for Vehicle B.




(**a**) (**b**)

**Figure 2.** Coordination and spatial variables: (**a**) origin and goal, and (**b**) relative with other vehicles.

State $s$, which includes spatial variables, serves as a training input for the deep reinforcement learning algorithm, allowing the autonomous system to make informed decisions based on comprehensive environmental awareness. By processing this input, deep reinforcement learning can train AVs to execute complex tasks, such as navigation and obstacle avoidance, and learn optimal strategies through repeated interactions with various simulated scenarios.

More parameters must be considered as states, such as velocity and acceleration, that is $\dot{x}_{OE}$ and $\ddot{x}_{OE}$, respectively. This information was correlated with distance information. In other words, a part of each piece of information is already contained in its parameters. Hence, they need not be included in the state; they are duplicate or trivial.

### *4.2. Action and Abstract Control*

*Action a* comprises nine abstract actions and fallback maneuvers, as expressed in Equation (2). Each action of the ego vehicle contained commands for the longitudinal and lateral controls. The details of each action are presented in Table 1.

$$a = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9\}. \tag{2}$$

**Table 1.** Ego vehicle behavior list.

| Action $a$ | Description | Velocity $\dot{x}$ (m/s) | Driving Lane $y$ (m) |
|---|---|---|---|
| $a_1$ | Turn left with fastest velocity | 0.20 | 0.15 |
| $a_2$ | Turn left with fast velocity | 0.15 | 0.15 |
| $a_3$ | Turn left with slow velocity | 0.10 | 0.15 |
| $a_4$ | Turn left with slowest velocity | 0.05 | 0.15 |
| $a_5$ | Turn right with fastest velocity | 0.20 | −0.15 |
| $a_6$ | Turn right with fast velocity | 0.15 | −0.15 |
| $a_7$ | Turn right with slow velocity | 0.10 | −0.15 |
| $a_8$ | Turn right with slowest velocity | 0.05 | −0.15 |
| $a_9$ | Emergency stop | 0.00 | null |

Each action had specific control commands for longitudinal and rotational control. Longitudinal control was performed with velocity command $\dot{x}_{OE}$. This command is explicitly expressed. However, the lateral control was performed utilizing the driving lane $y_{OE}$. This command was expressed implicitly. PID control with sensor fusion is necessary to render the lateral control explicit. Further details are provided in Appendix B.

### 4.3. Reward for Achieving the Goal

An ego vehicle must be designed to reach the goal line without collision. A reward function was defined to solve the fallback scenario. The reward function adequately provided success, failure, and transfer rewards to drive safety. The reward for success was reaching the destination without colliding with nearby vehicles.

A reward is a key component that affects learning; a bad reward design results in a bad policy or even divergence of learning. Rewards can be classified as episodic, transitional, or conditional. An episodic reward was provided when learning episodes were complete. When an episode was completed while achieving the goal, the agent was awarded a positive reward of +100; otherwise, the agent did not receive that reward. A transitional reward was awarded at each conditional time step during an episode. This depends on the design and can be constant or variable. Furthermore, both rewards can be awarded simultaneously.

*Reward r* was formulated to encapsulate a tripartite sub-reward ($r_1$, $r_2$, $r_3$), as expressed in Equation (3).

$$r = \sum_{j=1}^{3} r_j, \tag{3}$$

The first sub-reward $r_1$ is a sparse reward awarded only when a goal is achieved, as expressed in Equation (4). This reward is a strong motivation for the ego vehicle to achieve a goal because it provides many rewards simultaneously compared to subsequent rewards. Specifically, when the ego vehicle successfully navigated to the target location without colliding, a large reward of +100 was awarded. Conversely, nothing (zero reward) was awarded if the ego vehicle failed and the current episode ended.

$$r_1 = \begin{cases} +100, & \text{if } x_{OE} \geq x_{OG} \\ 0, & \text{otherwise} \end{cases}. \tag{4}$$

Here, $x_{OE}$ and $x_{OG}$ denote the current and target locations of the ego vehicle, respectively.

The second sub-reward $r_2$ is a positive and dense reward awarded at every time step after the interaction, as described in Equation (5). This sub-reward serves as a directional indicator, revealing whether the current behavior positively or negatively influences goal attainment. It offers vital navigational insights by progressively rewarding the agents as they approach their desired outcomes. For example, if the ego vehicle starts at 1.0 m and the position of $n_k$-step is 3.5 m, the reward is $100 \times (3.5 - 1.0) = 250$. In the next step $n_{k+1}$, if the position of the ego vehicle is 2.5 m, the reward is $100 \times (3.5 - 1.0) = 150$. When comparing the rewards of steps $n_k$ and $n_{k+1}$, it can be concluded that this is a bad case because the position of Step $n_{k+1}$ is lower than that of Step $n_k$.

$$r_2 = 100 \times (x_{OE} - x_{OE,i}), \tag{5}$$

where $x_{OE,i}$ is an initial position of the ego vehicle and is a constant value.

The third sub-reward, $r_3$ was a negative, dense reward awarded at every time step after the interaction, as described in Equation (6). This sub-reward imposes a time penalty to increase the efficiency of the learning process, leading to faster optimal learning. An ego vehicle can reach a target point at low speed, as a practical example. However, this policy was inefficient in terms of time consumption. Therefore, the ego vehicle must achieve its goal at the highest speed to reduce the penalty and increase the total reward.

$$r_3 = -1 \times n_{step}, \tag{6}$$

where $n_{step}$ is the number of steps in an episode.

The expected maximum total reward, according to Equations (3)–(6), was computed as 480. When the ego vehicle reaches the goal point, the maximum value of the first sub-reward is +100. The maximum value of the second sub-reward was 400, where the driving distance $x_{EG}$ was 400 from the initial position of ego vehicle $x_{OE,i}$, and 1.00 m to the goal position of $x_{OG}$ 5.00 m. To obtain the maximum total reward, the third sub-reward must be minimized. Then, the minimum value of the third sub-reward was $-20$ when the ego vehicle drove a distance of 4.00 m with a maximum velocity of 0.2 m/s. The minimum time required was 20 s when the min steps $n_{step}$ was 20. However, the ego vehicle must change lanes. Therefore, the actual maximum total reward was slightly less than 480.

## 5. Decision-Making Algorithm Design

In this section, we describe the structure of the decision-making algorithm and the learning hyperparameters. The structure of a Q-network based on deep reinforcement learning is explained. Moreover, diverse learning hyperparameters were specified.

### *5.1. Structure of Q-Network*

The network utilized for training was a forward neural network model with four layers: an input layer, two hidden layers, and one output layer. The number of nodes in the input layer was equal to the number of parameters in the state. Each of the two hidden layer nodes were empirically set to 64, and a rectified linear unit (ReLU) was utilized as the activation function. The number of nodes in the output layer was equal to the number of actions, and a purely linear function was utilized as the activation function. The network is illustrated in Figure 3. It is formatted with Equations (7)–(9) as follows.

$$h_1 = f_{h_1}\left(W_{h_1} \cdot s + b_{h_1}\right), \tag{7}$$

$$h_2 = f_{h_2}\left(W_{h_2} \cdot h_1 + b_{h_2}\right), \tag{8}$$

$$Q(s,a) = f_Q\left(W_Q \cdot h_2 + b_Q\right), \tag{9}$$

where $s$ is the state and input layers; $W$ is the weight matrix; $b$ is the bias matrix; $f$ is the activation function; $h$ is the hidden layer; and $Q$ is the state–action value function and output layer.

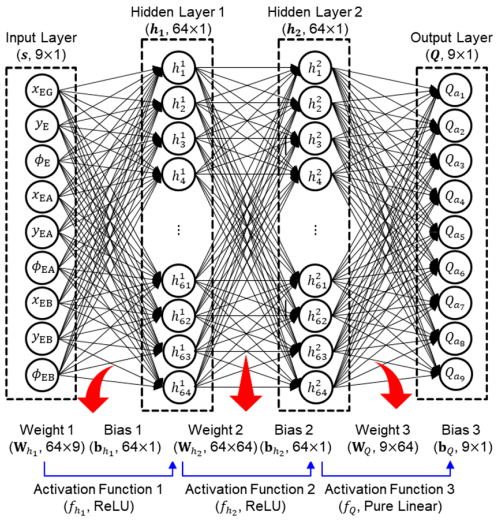



**Figure 3.** Schematic structure of neural network.

### 5.2. Hyper-Parameters for Learning

In the deep reinforcement learning environment, various hyperparameters were configured to establish the learning conditions. A total of 500 training and learning episodes were conducted. Learning generally converges in 150–200 episodes but provides sufficient episodes. The maximum step size for each episode was 500. The learning rate was 0.1, dropout rate was 0.2, and batch size was 64. The initial exploration and exploitation ratio $\varepsilon$ was 1.0, and as the episode progressed, the ratio $\varepsilon$ decreased by a factor of 0.99. With a high epsilon value, ego vehicle exploration was random. Therefore, the ego vehicle collects diverse data patterns through trial-and-error interactions. The hyperparameters are listed in Table 2.

**Table 2.** Hyper-Parameter of Learning.

| Category | Value |
|---|---|
| Episode size | 500 |
| Max. step size | 500 |
| Learning rate | 0.1 |
| Dropout | 0.2 |
| Batch size | 64 |
| Exploration and exploitation rate $\varepsilon$ | 0.99 |

## 6. Training and Evaluation

To validate the fallback capability of the proposed decision-making algorithm, we conducted a simulated training experiment on an ego vehicle in a given fallback scenario. This section describes the experimental setup, coordinates, and initial conditions. Subsequently, we present and discuss the metrics and training results. Detailed calculations for the initial process of training are described in Appendix C.

### 6.1. Experimental Setup

The training simulation was conducted on a computer with the following specifications: Intel Core i7-10700 CPU, NVIDIA GeForce RTX 2060 GPU, and Samsung DDR4 with 16 GB of RAM (Samsung Electronics, Republic of Korea). We provide the hardware specifications because we present the average training time in the training results, which indicates a highly hardware-sensitive performance.

The learning environment was built on the Gazebo simulator in the ROS framework: Gazebo 9.0.0 and ROS Melodic Morenia on the Ubuntu 18.04 Bionic Beaver OS. It provides a robust physics engine, high-quality graphics, convenient programming, and graphical interface. ROS provides many robotics libraries and interoperates with multiple plugins.

We employed the Turtlebot3 (©ROBOTIS) as AVs, which can be easily reprogrammed [26]. Turtlebot3 is an ROS-based mobile robot that provides low-level control algorithms such as simultaneous localization and mapping, navigation, and collision avoidance.

The specifications of the Turtlebot3 are maximum linear velocity $\dot{x}$ of 0.22 m/s, maximum angular velocity $\dot{\phi}$ of 284 rad/s (162.72 deg/s), length $l$ of 138 mm, width $w$ of 178 mm, and height $h$ of 192 mm.

The source code was written in Python utilizing the Keras deep learning libraries Python 2.7.17 and Keras 2.1.5. The code utilized for the training was obtained from GitHub [27]. The code was based on the ROBOTIS e-manual [26].

### 6.2. Coordinates and Initial Conditions

The training began with the initial conditions presented in Figure 4 and detailed in Table 3. The origin of the coordinates is located at the left center, and the horizontal and vertical lines represent the $x$- and $y$-axes, respectively. The goal line was 5.00 m from the origin along the $x$-axis. The ego vehicle attempts to learn the speed and angular velocity required to reach the goal line without colliding with Vehicle A or B. The initial position $(x, y)$ and orientation $\phi$ of the ego vehicle were 2.00 m, 0.15 m, and 0.00 rad. Vehicles A

and B (which did not learn) were driven according to a constant preset control command. Vehicle A drove at a low speed (0.05 m/s) owing to a defect and did not steer (angular velocity of 0.00 rad/s). Vehicle A's initial positions and orientations were 2.00 m, 0.15 m, and 0.00 rad, respectively. As Vehicle B was not affected by the defective vehicle, it traveled at a high speed of 0.15 m/s and did not steer (angular velocity is 0.00 rad/s). Thus, Vehicle B's initial position and orientation were 0.00 m, −0.15 m, and 0.00 rad.

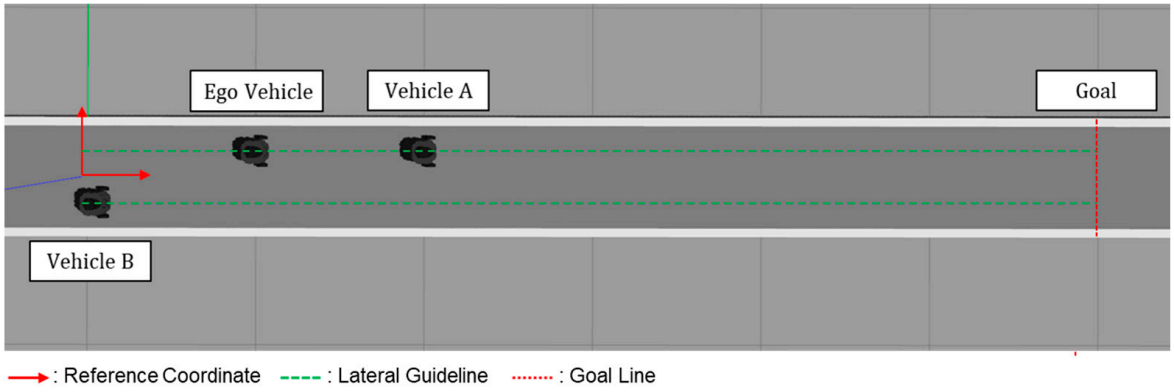



**Figure 4.** Learning environment of fallback scenario.

**Table 3.** Initial conditions of ego vehicle, Vehicles A and B.

| Parameter | Ego Vehicle | Vehicle A | Vehicle B |
|---|---|---|---|
| $x$ (m) | 1.00 | 2.00 | 0.00 |
| $y$ (m) | 0.15 | 0.15 | −0.15 |
| $\phi$ (rad) | 0.00 | 0.00 | 0.00 |
| $\dot{x}$ (m/s) | - | 0.05 | 0.15 |
| $\dot{\phi}$ (rad/s) | - | 0.00 | 0.00 |

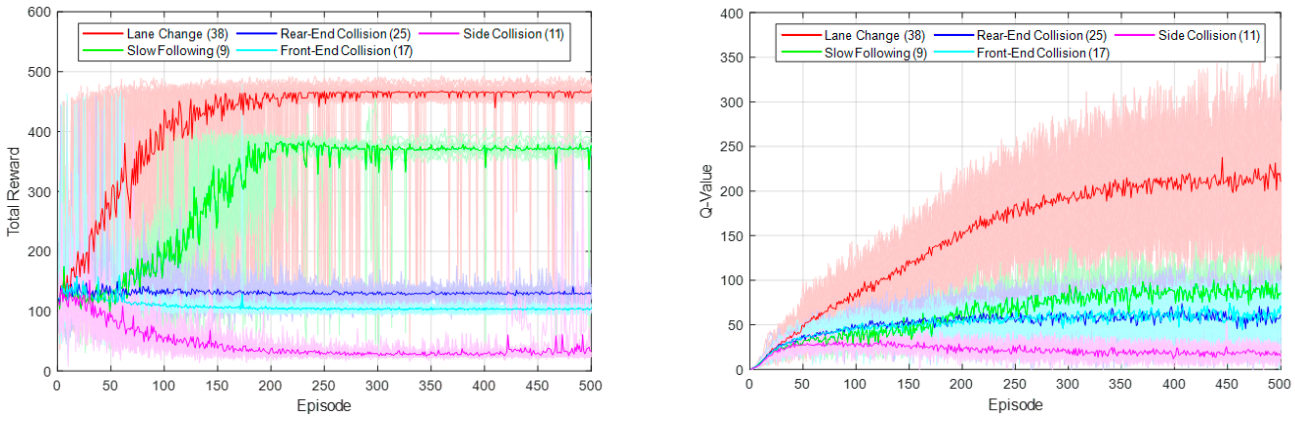
### *6.3. Metric and Training Results*

The metrics considered for the fallback capacity were the three optimal heuristic maneuvers derived in Section 3.2: slow following, lane change, and lane change after yielding. We evaluated the decision-making algorithm by comparing the training results from heuristic maneuvers. If the training results converge to a particular heuristic maneuver, the ego vehicle successfully learns the optimal policy to respond appropriately to the corresponding fallback scenario.

The proposed decision-making algorithm was trained 100 times with the same hyper-parameters. The training was performed in a single synchronous environment, and the time required to complete the training was measured and presented. This was conducted to generalize the training results based on the statistical evaluation. The 100 results were converged into five groups: (1) lane changes, (2) slow following, (3) rear-end collisions, (4) front-end collisions, and (5) side collisions. Therefore, lane changes and slow following were successful results of the training fallback technique, and the three collisions were considered failures. Figure 5 presents the quantitative training results for each episode's total reward and maximum Q-value. The Q-value is related to the loss function and is presented as an indicator for evaluating the learning process and policy results.

1. *Lane change* involved the ego vehicle moving from the current lane to the right lane, which occurred 38 of 100 times (38%). In Figure 5a,b, the average of 38 training results and each training result are depicted in red and shaded red, respectively. The total reward converged within the range of 470–480 after 200 episodes. The maximum Q-value increased monotonically. Moreover, each training time required an average of 2.5 h for 500 episodes.

2. *Slow following* involved the ego vehicle following the front slow vehicle, which occurred 9 out of 100 times (9%). In Figure 5a,b, the averages of the nine training results and each training result are depicted in green and shaded green, respectively. The total reward converges to 370–380 after 200 episodes. The maximum Q-value increases slightly. Moreover, each training time required an average of 8.5 h for 500 episodes.
3. *Rear-end collision* involved the front of Vehicle B colliding with the rear of the ego vehicle, which occurred 25 of 100 times (25%). In Figure 5a,b, the average of the 25 training results and each training result are depicted in blue and shaded blue, respectively. The total reward converged in the range of 120–130 after 100 episodes. The maximum Q-value increased slightly.
4. *Front-end collision* involved the front of the ego vehicle colliding with the rear of Vehicle A, which occurred 17 of 100 times (17%). In Figure 5a,b, the average of the 17 training results and each training result are depicted in cyan and shaded cyan, respectively. The total reward converged in the range of 100–110 after 100 episodes. The maximum Q-value increased slightly and was indistinguishable from the rear-end collision results.
5. *Side collision* involved the side of the ego vehicle colliding with the front of Vehicle B, which occurred 11 of 100 times (11%). In Figure 5a,b, the average of the 11 training results and each training result are depicted in magenta and shaded magenta, respectively. The total reward converged within the range of 30–40 after 200 episodes. The maximum Q-value decreased slightly.




(**a**) (**b**)

**Figure 5.** Training decision-making algorithm in fallback scenario: (**a**) total reward in each episode, and (**b**) max. Q-value in each episode.

Thus, the *lane change* maneuver was the optimal fallback maneuver that achieved its purpose and provided the most efficient (speed) results. This analysis demonstrated that the *lane change* maneuver received the highest reward and Q-value in the experimental results. The *slow following* maneuver is a suboptimal fallback maneuver. This achieved its purpose; however, it was relatively slow and inefficient compared to the *lane change* maneuver. The *rear*, *front*, and *side-end collision* maneuvers are poor fallback maneuvers. This is because these maneuvers failed to achieve their goals. In addition, the *lane change* and *slow following* maneuvers were consistent with the heuristic optimal fallback maneuvers. This implies that the ego vehicle learned the appropriate fallback maneuvers utilizing the proposed algorithm. The ego vehicle accepts one of the trained models among the 38 *lane change* models or the 9 *slow following* models and considers it in the fallback scenario. In addition, the *lane change* maneuver is recommended for the *slow following* maneuver because of driving efficiency.

### 6.4. Discussion

Next, we discuss why the training experiment yielded five convergence results even though the proposed algorithm was independently trained 100 times with the same hyperparameters.

Convergence was primarily determined by the existence and frequency of successful experiences within the 64 episodes. This was the batch size, and the training started at this time. The experience data collected during 64 episodes were utilized more frequently and repeatedly to train the neural network model.

Therefore, if an initial neural network is trained on successful data containing optimal decision-making patterns, it converges to lane change or slow following. Eventually, active behavioral maneuvers (lane changes) and passive behavioral maneuvers (slow following) occur. Otherwise, if it is trained with poor data, it converges to a collision. Eventually, five convergences appeared. If learning fails, no pattern can be learned from the data. In other words, it does not capture the sense of driving.

To enhance the success of the experience, the data were related to the exploration and exploitation rates $\varepsilon$. The current exploration and exploitation rates decreased as the episode progressed; although, the ego vehicle did not have a successful experience. In other words, the exploration and exploitation rates decayed early. Methods for delaying the decay of exploration and exploitation rates are considered acceptable, for example, decaying the rate whenever an ego vehicle experiences success.

Furthermore, we discuss why lane changes after yielding did not occur owing to training. According to the training results, the proposed algorithm was expected to learn only one abstract maneuver. In addition, lane changes after yielding combine two abstract maneuvers. Therefore, lane changes did not occur after yielding. A neural network must be designed with more complex or additional networks for more complex abstract maneuvers.

The deployment of deep reinforcement learning is distinguished by its capacity to learn from scratch, which is instrumental in its adaptiveness. This feature of deep reinforcement learning is particularly advantageous because it facilitates the ability of an AV to discover and refine optimal driving policies in a wide range of fallback scenarios. Notably, this adaptiveness is a theoretical benefit and a practical necessity, given the unpredictable nature of real-world driving environments. Starting from basic principles and learning via trial and error, a deep reinforcement learning framework was designed to cope with the nuances and complexities of such scenarios. This approach significantly boosts the system's scalability, allowing it to maintain its robustness and efficacy. As the diversity and complexity of potential driving situations expand, the ability of our deep reinforcement learning system to adapt and scale ensures that autonomous vehicles can navigate through these challenges, while increasing autonomy and safety. Therefore, this ongoing learning process is crucial for advancing towards fully autonomous driving capabilities, where the system must handle an ever-growing array of situations with minimal human oversight.

## 7. Conclusions

This study developed a decision-making algorithm based on deep reinforcement learning to address a particular fallback scenario in AVs. We designed a fallback scenario and derived heuristic maneuvers. The proposed algorithm selects an appropriate response action for dangerous situations while driving AVs. Owing to deep reinforcement learning, the heuristic method yielded the same results as expected. The fallback scenario was solved with 38% lane changes and 9% slow following out of 100 training sessions. Thus, it was confirmed that the decision-making algorithm for AV can be learned based on deep reinforcement learning.

Future research can be extended in terms of three aspects: (1) diverse and complex fallback scenarios, (2) advances in deep reinforcement learning, and (3) collective fallback techniques utilizing multiagent learning. First, we will design more diverse simple-to-complex fallback scenarios, for example, based on various operational design areas utilizing the PEGASUS 6-layer model [28,29] and apply deep reinforcement learning to broaden

the scope of the application of the proposed algorithm to demonstrate its adaptability. While the current research has been conducted with simulated data, future work will focus on employing real-world open datasets to refine ADSs for genuine on-road applications, further enhancing their real-life performance and adaptability. Second, advanced deep reinforcement learning algorithms such as A3C, DDPG, TRPO, PPO, SAC, and TD3 can be applied to increase learning performance and accuracy. Finally, instead of solving a given fallback scenario with only one agent, multiagent reinforcement learning will be applied such that multiple agents can learn. This ensures that the vehicle cluster can escape dangerous situations utilizing collective intelligence.

**Author Contributions:** Conceptualization, C.L. and D.A.; methodology, D.A.; software, C.L.; validation, C.L. and D.A.; formal analysis, D.A.; investigation, C.L.; resources, D.A.; data curation, C.L.; writing—original draft preparation, C.L.; writing—review and editing, D.A.; visualization, C.L.; supervision, D.A.; project administration, D.A.; and funding acquisition, D.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This study has been conducted with the support of the Korea Institute of Industrial Technology as "Development of Core Technologies for a Working Partner Robot in the Manufacturing Field (kitech EO-23-0009)" and the Technology Innovation Program (20016970, Development of Cooperative Robot SI (System Integration) Service based on Safety Intelligence) funded By the Ministry of Trade, Industry & Energy (MOTIE, Korea).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data presented in this study are available in Appendix C.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Appendix A

This section introduces other validation methods based on the mile-driven and scenario-based approaches mentioned here.

We highlighted the validation phase in the introduction, emphasizing mile-driven and scenario-based methods. Nonetheless, various methods exist for assessing the safety of AVs and their ADSs and have been elaborated in this context: (1) public road testing, (2) closed-course testing, and (3) simulation testing.

*Public road testing* was conducted to validate the performance of the ADS under real-world conditions and to expose it to unpredictable events. The AV was operated on public roads with an onboard safety driver for control, if necessary. This was the ultimate test for the readiness of AV for real-world deployment. Advantages: Real-world variability and unpredictability cannot be fully mimicked in simulations or closed courses. Public road testing provides the most accurate understanding of how an ADS performs under typical and unexpected conditions.

*Closed-course testing* is intended to test the AV sensors, software, and decision-making algorithms in controlled environments. The vehicle was tested on private tracks or facilities that mimic real-world driving conditions. These scenarios could be carefully staged with props, other vehicles, and robotic pedestrians. This approach allows real-world physical testing, while minimizing the unpredictability associated with public roads. This facilitates reproducibility and careful staging of specific scenarios.

*Simulation testing* is a safe, cost-effective, and scalable method for testing the decision-making algorithms of an ADS across millions of virtual miles in a wide array of scenarios. This involves creating virtual environments in which AV software is subjected to numerous scenarios, including those too dangerous to be tested in the real world. Simulation testing can cover rare-edge cases, reproduce complex scenarios, and run continuously, accelerating the validation process and ensuring broader coverage than real-world testing alone.

## Appendix B

This section describes the sensor-fusion and P-control algorithm validation approaches and the control algorithm.

Gazebo has a physics engine; therefore, it does not move according to the command value owing to inertia and friction and has certain errors. Therefore, an appropriate controller was required. An abstract action replaced the discontinuous action space, and PID control was performed at the base.

The objective of PID control with sensor fusion is to maintain vehicles that follow a straight-line $y_{OG}$. The lateral values and yaw angle controls are maintained constant. However, the ego vehicle can control its lateral direction and yaw angle utilizing only one steering angle and angular velocity. Therefore, sensor fusion is required for multivariate P control, and the formula is the same as in Equations (A1) and (A2). The results are presented in Figure A1.

$$u_\theta = K_{P,y} \cdot \overline{y}_{ri} + K_{P,\phi} \cdot \phi_{ri}, \tag{A1}$$

$$\overline{y}_{ri} = \tan^{-1}\left(\frac{y_{ri}}{c_x}\right), \tag{A2}$$

where $K_{P,y}$ is the proportional control gain with a constant value of 1.5. The two error values are different in terms of $\overline{y}_{ri}$ mm and $\phi_{ri}$ rad. The $\overline{y}_{ri}$ is a dimension transition that equalizes the dimensions of the two error values and is a constant value $y_{ri}$ with a hypotenuse value $c_x$ of 0.3. Further, $K_{P,\phi}$ is the proportional control gain that has a constant value of 1.0, as obtained through an experiment.

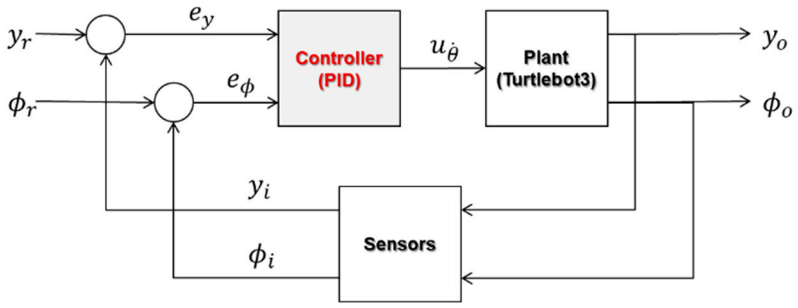



**Figure A1.** Schematic overview of PID control algorithm with sensor fusion.

## Appendix C

In this section, the initial iteration of the interaction is calculated with Equations (1)–(9) and the equations in the background section. The initial information provided by the AVs before the calculations is presented in Table A1.

$$s = (-3.9930, 0.1500, 0.0000, -0.9930, 0.0000, 0.0000, 1.0070, 0.3000, 0.0000).$$

**Table A1.** Training input and output data through agent and environment interaction.

| Step | Input Data | | | | | | | | | Output Data |
|---|---|---|---|---|---|---|---|---|---|---|
| | $x_{EG}$ | $y_{OE}$ | $\phi_{OE}$ | $x_{EA}$ | $y_{EA}$ | $\phi_{EA}$ | $x_{EB}$ | $y_{EB}$ | $\phi_{EB}$ | $Q$ |
| 1 | −3.9930 | 0.1500 | 0.0000 | −0.9930 | 0.0000 | 0.0000 | 1.0070 | 0.3000 | 0.0000 | 0.0000 |
| 2 | −3.9621 | 0.1500 | −0.0076 | −0.9706 | 0.0000 | −0.0076 | 1.0222 | 0.3000 | −0.0077 | 0.0000 |
| 3 | −3.9367 | 0.1496 | −0.0138 | −0.9553 | −0.0004 | −0.0138 | 1.0226 | 0.2996 | −0.0139 | 0.9352 |
| 4 | −3.9163 | 0.1495 | −0.0024 | −0.9433 | −0.0005 | −0.0022 | 1.0133 | 0.2995 | −0.0024 | 2.0041 |
| 5 | −3.9033 | 0.1495 | −0.0214 | −0.9388 | −0.0005 | −0.0212 | 0.9963 | 0.2994 | −0.0215 | 2.7025 |
| 6 | −3.8924 | 0.1482 | −0.1458 | −0.9429 | −0.0018 | −0.1454 | 0.9720 | 0.2982 | −0.1458 | 3.1954 |
| 7 | −3.8703 | 0.1442 | −0.2348 | −0.9286 | −0.0058 | −0.2344 | 0.9643 | 0.2942 | −0.2348 | 2.9132 |

**Table A1.** *Cont.*

| Step | Input Data | | | | | | | | | Output Data |
|---|---|---|---|---|---|---|---|---|---|---|
| | $x_{EG}$ | $y_{OE}$ | $\phi_{OE}$ | $x_{EA}$ | $y_{EA}$ | $\phi_{EA}$ | $x_{EB}$ | $y_{EB}$ | $\phi_{EB}$ | $Q$ |
| 8 | −3.8506 | 0.1387 | −0.2736 | −0.9187 | −0.0113 | −0.2732 | 0.9541 | 0.2887 | −0.2737 | 5.7062 |
| 9 | −3.8246 | 0.1321 | −0.2625 | −0.9025 | −0.0180 | −0.2624 | 0.9501 | 0.2820 | −0.2626 | 5.5420 |
| 10 | −3.8012 | 0.1245 | −0.3744 | 0.9433 | 0.2745 | −0.3743 | −0.8891 | −0.0255 | −0.3744 | 5.7779 |
| $n$ | - | - | - | - | - | - | - | - | - | - |

The initial actions of the ego vehicle were randomly selected. The $a_E$ is $a_7$ which turns right at a slow speed. According to Action $a_7$, the ego vehicle moves toward the right driving lane −0.15 m at a velocity of 0.15 m/s. Vehicles A and B exhibit constant actions $a_4$ and $a_6$, respectively. At all times, Vehicle A drove in the left driving lane 0.15 m at a velocity of 0.10 m/s, and Vehicle B drove in the right driving lane −0.15 m at a velocity of 0.20 m/s.

$$a_E = a_7, a_A = a_4, a_B = a_6.$$

For the $r_1$ reward function, $r_1$ = 0 because the ego vehicle has not yet reached its goal point.

$$\begin{aligned} r_1 &= \begin{cases} +100, & \text{if } x_{OE} \geq x_{OG} \\ 0, & \text{otherwise} \end{cases} \\ &= 0. \end{aligned}$$

For the $r_2$ reward function, $r_2$ is 0.70 as per the following calculation.

$$\begin{aligned} r_2 &= 100 \times (x_{OE} - x_{OE,i}) \\ &= 100 \times (x_{EG} + x_{OG} - x_{OE,i}) \\ &= 100 \times (-3.9930 + 5.0000 - 1.0000) \\ &= 100 \times 0.0070 \\ &= 0.70, \end{aligned}$$

where the current position of the ego vehicle $x_{OE}$ is $x_{EG} + x_{OG}$ because $x_{EG}$ is $x_{OE} - x_{OG}$, the $x_{EG}$ is −3.9930; the initial position of the ego vehicle $x_{OE,i}$ is 1.0000 m; and the $x_{OG}$ is 5.0000 m.

For the $r_3$ reward function, $r_3$ is −1 because only one step progressed, that $n_{step}$ is 1.

$$\begin{aligned} r_3 &= -1 \times n_{step} \\ &= -1 \times 1 \\ &= -1 \end{aligned}$$

For the integration of the total reward function, $r$ is −0.30 as per the following calculation.

$$\begin{aligned} r &= \sum_{j=1}^{3} r_j \\ &= r_1 + r_2 + r_3 \\ &= 0 + 0.70 - 1 \\ &= -0.30. \end{aligned}$$

Depending on the progress made thus far, move on to the next state $s'$.

$$s' = (-3.9621, 0.1500, -0.0076, -0.9706, 0.0000, -0.0076, 1.0222, 0.3000, -0.0077).$$

The one-step interaction data $(s, a, r, s')$ are recorded in the reply buffer. The current state $s$ is updated with $s'$.

$$s = s'.$$

The next action was selected utilizing the following equations. The value of the Q function was calculated and updated as follows:

$$h_1 = f_{h_1}(W_{h_1} \cdot s + b_{h_1})$$
$$h_2 = f_{h_2}(W_{h_2} \cdot h_1 + b_{h_2})$$
$$Q(s,a) = f_Q(W_Q \cdot h_2 + b_Q).$$

We then apply epsilon greed, where exploration and exploitation are determined using the epsilon greed rate $\varepsilon$. The initial exploration and exploitation ratio is 0.99. This implies a 99% exploration probability and a 1% exploitation probability. The exploration and exploitation rates decrease as the stages progress. During the exploration, an action was selected randomly. If exploitation is selected, policy $\pi$ outputs an action $a$ for the maximum value of Q.

$$\pi(s) = \underset{a}{\text{argmax}} Q(s,a).$$

When as many interaction data as the number of batch sizes are collected in the replay buffer, the Q function is later learned by updating the weight bias according to the following formula:

$$L(\theta) = \tfrac{1}{2}[\{r + \gamma \text{max} Q(s', a'; \boldsymbol{\omega})\} - Q(s,a;\boldsymbol{\theta})]^2$$
$$\nabla_\theta L(\theta) = [\{r + \gamma \text{max} Q(s', a'; \boldsymbol{\theta})\} - Q(s,a;\theta)] \nabla_\theta Q(s,a;\boldsymbol{\theta})$$
$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

The input and output data obtained by repeating Steps 1–10 are summarized in Table A1. This interaction is repeated until the end of an episode.

## References


1. *SAE-J3016*; Taxonomy and Definitions for terms Related to Driving Automation Systems for On Road Motor Vehicles. Society of Automotive Engineers (SAE): Warrendale, PA, USA, 2021. Available online: https://www.sae.org/standards/content/j3016_202104/ (accessed on 9 November 2023).
2. *ISO/PAS 21448*; Road Vehicles: Safety of Intended Functionality (SOTIF). International Organization for Standardization (ISO): Geneva, Switzerland, 2022. Available online: https://www.iso.org/standard/77490.html (accessed on 9 November 2023).
3. Yu, J.; Luo, F. Fallback strategy for level 4+ automated driving system. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; pp. 156–162.
4. Shalev-Shwartz, S.; Shammah, S.; Shashua, A. On a formal model of safe and scalable self-driving car. *arXiv* **2017**, arXiv:1708.06374.
5. Riedmaier, S.; Ponn, T.; Ludwig, D.; Schick, B.; Diermeyer, F. Survey on scenario-based safety assessment of automated vehicles. *IEEE Access* **2020**, *8*, 87456–87477. [CrossRef]
6. Huang, W.; Wang, K.; Lv, Y.; Zhu, F. Autonomous vehicles testing methods review. In Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1–4 November 2016; pp. 163–168.
7. Xu, R.; Xia, X.; Li, J.; Li, H.; Zhang, S.; Tu, Z.; Ma, J. V2v4real: A real-world large-scale dataset for vehicle-to-vehicle cooperative perception. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, Canada, 18–22 June 2023; pp. 13712–13722.
8. Guo, X.; Zhang, Y. Maturity in automated driving on public roads: A review of the six-year autonomous vehicle tester program. *Transp. Res. Rec.* **2022**, *2676*, 352–362. [CrossRef]
9. Vitelli, M.; Chang, Y.; Ye, Y.; Ferreira, A.; Wołczyk, M.; Osiński, B.; Ondruska, P. Safetynet: Safe planning for real-world self-driving vehicles using machine-learned policies. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 897–904.
10. Orzechowski, P.F.; Burger, C.; Lauer, M. Decision-making for automated vehicles using a hierarchical behavior-based arbitration scheme. In Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, NV, USA, 19 October–13 November 2020; pp. 767–774.
11. Owais, M.; Alshehri, A. Pareto optimal path generation algorithm in stochastic transportation networks. *IEEE Access* **2020**, *8*, 58970–58981. [CrossRef]
12. Alshehri, A.; Owais, M.; Gyani, J.; Aljarbou, M.H.; Alsulamy, S. Residual Neural Networks for Origin–Destination Trip Matrix Estimation from Traffic Sensor Information. *Sustainability* **2023**, *15*, 9881. [CrossRef]
13. Pini, S.; Perone, C.S.; Ahuja, A.; Ferreira, A.S.R.; Niendorf, M.; Zagoruyko, S. Safe real-world autonomous driving by learning to predict and plan with a mixture of experts. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 29 May–2 June 2023; pp. 10069–10075.

14. Allamaa, J.P.; Listov, P.; Van der Auweraer, H.; Jones, C.; Son, T.D. Real-time nonlinear mpc strategy with full vehicle validation for autonomous driving. In Proceedings of the 2022 American Control Conference (ACC), Atlanta, GA, USA, 8–10 June 2022; pp. 1982–1987.
15. Wei, C.; Romano, R.; Merat, N.; Wang, Y.; Hu, C.; Taghavifar, H.; Boer, E.R. Risk-based autonomous vehicle motion control with considering human driver's behaviour. *Transp. Res. Part C Emerg. Technol.* **2019**, *107*, 1–14. [CrossRef]
16. Walch, M.; Woide, M.; Mühl, K.; Baumann, M.; Weber, M. Cooperative overtaking: Overcoming automated vehicles' obstructed sensor range via driver help. In Proceedings of the 11th international conference on automotive user interfaces and interactive vehicular applications, Utrecht, The Netherlands, 21–25 September 2019; pp. 144–155.
17. Rokonuzzaman, M.; Mohajer, N.; Nahavandi, S.; Mohamed, S. Model predictive control with learned vehicle dynamics for autonomous vehicle path tracking. *IEEE Access* **2021**, *9*, 128233–128249. [CrossRef]
18. Gelbal, S.Y.; Tamilarasan, S.; Cantas, M.R.; Güvenc, L.; Aksun-Güvenc, B. A connected and autonomous vehicle hardware-in-the-loop simulator for developing automated driving algorithms. In Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, Canada, 5–8 October 2017; pp. 3397–3402.
19. Lee, K.; Kum, D. Collision avoidance/mitigation system: Motion planning of autonomous vehicle via predictive occupancy map. *IEEE Access* **2019**, *7*, 52846–52857. [CrossRef]
20. Liu, K.; Gong, J.; Kurt, A.; Chen, H.; Ozguner, U. A model predictive-based approach for longitudinal control in autonomous driving with lateral interruptions. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017; pp. 359–364.
21. Corso, A.; Lee, R.; Kochenderfer, M.J. Scalable autonomous vehicle safety validation through dynamic programming and scene decomposition. In Proceedings of the 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), Rhodes, Greece, 20–23 September 2020; pp. 1–6.
22. Corso, A.; Du, P.; Driggs-Campbell, K.; Kochenderfer, M.J. Adaptive stress testing with reward augmentation for autonomous vehicle validatio. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; pp. 163–168.
23. Frese, C.; Beyerer, J. A comparison of motion planning algorithms for cooperative collision avoidance of multiple cognitive automobiles. In Proceedings of the 2011 IEEE intelligent vehicles symposium (IV), Baden, Germany, 5–9 June 2011; pp. 1156–1162.
24. Xue, W.; Yang, B.; Kaizuka, T.; Nakano, K. A fallback approach for an automated vehicle encountering sensor failure in monitoring environment. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 1807–1812.
25. Galko, C.; Rossi, R.; Savatier, X. Vehicle-Hardware-In-The-Loop system for ADAS prototyping and validation. In Proceedings of the 2014 International conference on embedded computer systems: Architectures, Modeling, and Simulation (SAMOS XIV), Samos Island, Greece, 14–17 July 2014; pp. 329–334.
26. ROBOTIS. TurtleBot3 e-Manual. Available online: https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/ (accessed on 20 October 2023).
27. Lee, C. Learning Codes. Available online: https://github.com/cheonghwa-lee/tb3_ml (accessed on 20 October 2023).
28. Scholtes, M.; Westhofen, L.; Turner, L.R.; Lotto, K.; Schuldes, M.; Weber, H.; Eckstein, L. 6-layer model for a structured description and categorization of urban traffic and environment. *IEEE Access* **2021**, *9*, 59131–59147. [CrossRef]
29. Audi, G.; Volkswagen, A.G. The PEGASUS Method. Available online: https://www.pegasusprojekt.de/en/pegasus-method (accessed on 9 November 2023).