Utilizing Both DDPG and DQN Algorithms for Path-Following Control in Autonomous Vehicles

Ali Rizehvandi, Shahram Azadi

Enhancing autonomous vehicles (AVs) can significantly improve the reliability and safety of transportation systems. To achieve full autonomy (SAE Level 5), AVs must handle complex and unpredictable traffic conditions. A critical aspect of automated driving is path-following, which ensures the vehicle accurately tracks a predefined trajectory. Traditional path-following techniques often rely on parameter tuning or rule-based systems, which may struggle in highly dynamic environments. Reinforcement learning (RL) offers a promising alternative by enabling control strategies to be learned through experience. This study investigates the performance of both Deep Deterministic Policy Gradient (DDPG) and Deep Q Network (DQN) algorithms in managing AV acceleration and steering for path-following. The results show that DDPG and DQN algorithms converge quickly, enabling stable and efficient trajectory tracking while ensuring smooth control inputs. These findings underscore the potential of hybrid DDPG and DQN algorithms in advancing autonomous driving technology.

KEY WORDS
~ Autonomous vehicles
~ DRL method
~ Path-following
~ DDPG algorithm
~ DQN algorithm

Khajeh Nasir Toosi University of Technology, Teheran, Iran
e-mail: ali.rizehvandi75@gmail.com
doi: 10.7225/toms.v15.n01.008

Received: 13 Apr 2025 / Revised: 27 Oct 2025 / Accepted: 3 Apr 2026 / Published: 20 Apr 2026

This work is licensed under CC BY

1. INTRODUCTION

Numerous organizations in industry and academia are advancing autonomous vehicle (AV) technology, which holds great promise for significantly reducing traffic fatalities and saving time during daily commutes. According to the World Health Organization (WHO) (Global Status Report on Road Safety, 2018), approximately 1.35 million people die each year in road accidents, with an additional 20 to 50 million suffering non-fatal injuries. The U.S. National Highway Traffic Safety Administration (NHTSA) (Singh, 2018) reports that 94% of crashes are caused by human error, while only 2% result from vehicle malfunctions. Consequently, broader adoption of AVs is expected to reduce accident rates and, in turn, lower traffic-related deaths. If implemented on a large scale, AVs could provide significant societal benefits, including reduced congestion, fewer collisions, lower energy consumption, and increased efficiency by repurposing time otherwise spent driving. Studies estimate that by 2050, these benefits could result in approximately $800 billion in annual societal gains (Montgomery et al., 2022). However, despite improvements in AV performance in controlled settings (Thrun et al., 2005), the unpredictable and dynamic nature of real-world driving conditions presents major challenges to widespread deployment (Koopman & Wagner, 2017). To evaluate AV technology in actual traffic environments, the U.S. Department of Motor Vehicles (DMV) launched an AV testing program in 2014, issuing permits to manufacturers (Autonomous Vehicles Testing with a Driver, 2022). In California, the DMV requires AV developers to conduct road tests and report any incidents involving property damage, injuries, or fatalities (Autonomous Vehicles Testing with a Driver, 2021).

Path-following has emerged as a viable solution for trajectory tracking in various vehicle applications, due to its adaptability in handling timing constraints (Paden et al., 2016). In trajectory tracking problems, the reference trajectory defines the vehicle's desired position in state space, while path-following aims to keep the vehicle close to a predefined path without relying on real-time timing data (Faulwasser et al., 2009; Aguiar et al., 2008; Rubí et al., 2021). The pure pursuit method, valued for its simplicity and effectiveness in real-time control, was among the earliest path-following strategies (Paden et al., 2016; Amer et al., 2017). This approach assumes the vehicle moves at a fixed, predefined speed along a straight reference path (zero curvature). The controller steers the vehicle by tracking a circle determined by the current position of a key point (such as the rear-wheel position) and a look-ahead distance L to a target point on the path ahead. However, if the vehicle's deviation from the reference path exceeds L, the controller stops evaluating.

Classical control methods offer two prominent approaches for vehicle path-following. The first method (Samson, 1992) models the reference path as a continuous position-dependent function, using the rear-wheel position with a predetermined delay to simultaneously minimize cross-track error and maintain heading stability. While this approach guarantees fast local convergence, it imposes strict requirements on path geometry – specifically, the reference path must be twice continuously differentiable due to its dependence on path curvature. An alternative strategy (Thrun et al., 2006) focuses on minimizing front-wheel cross-track error through nonlinear feedback control. This method demonstrates local exponential convergence to the reference path but requires modifications for reverse driving. Notably, this steering control approach proved its practical effectiveness by powering the winning vehicle in the 2005 DARPA Challenge.

The approaches outlined in (Thrun et al., 2006) serve as benchmark methods, delivering reliable performance with few parameters, low uncertainty requirements, and reasonable model precision. Reinforcement Learning (RL) techniques have seen substantial advancements in gaming, robotics, and self-driving vehicles in recent years, attracting considerable interest (Zhao et al., 2020). RL is a machine learning paradigm in which an agent learns optimal actions through interactions with the environment. These methods enable goal-directed learning through reward functions, similar to humans, while maintaining flexibility in new situations. A key challenge during real-world training is that RL's exploratory nature in the action space may result in potentially dangerous behaviors. The incorporation of deep neural networks (DNNs) to approximate the action-value function (Q-function) (Mnih et al., 2015) has led to Deep Reinforcement Learning (DRL), which can handle complex, high-dimensional state spaces. Major DRL algorithms include deep Q-network (DQN) (Mnih et al., 2015), proximal policy optimization (PPO), and deep deterministic policy gradient (DDPG) (Schulman et al., 2017). DDPG (Lillicrap et al., 2015) merges aspects of both DQN and deterministic policy gradient (DPG) approaches, finding extensive application in robot control and autonomous vehicle guidance. Recent years have seen growing application of DRL to path-following challenges. Rubí et al. (Rubí et al., 2021) developed three DDPG-based techniques that successfully enabled adaptive speed control and path tracking for quadrotors. Cheng et al. (Cheng et al., 2022) simulated effective obstacle avoidance and path-following for wheeled robots, though their method produced significant velocity jerk from overly forceful control actions. Zheng et al. (Zheng et al., 2023) introduced a 3D path-following system for parafoils that combined DDPG with linear active disturbance rejection to handle flight paths and wind resistance. Additionally, Ma et al. (Ma et al., 2023) applied the soft actor-critic (SAC) algorithm to underwater vehicle navigation, demonstrating capable path-tracking performance.

This research focuses on applying Deep Reinforcement Learning (DRL) to autonomous ground vehicles at medium speeds, aiming to reduce lateral drift and maintain stable steering control. The methodology integrates both DDPG and DQN algorithms, utilizing DDPG for speed regulation along the path and DQN for directional control. An optimized reward function facilitates smooth driving dynamics and consistent steering performance during acceleration. Compared to traditional control systems, this combined approach offers simpler training requirements and demonstrates enhanced tracking accuracy along the target path. The trained control system quickly corrects path deviations while limiting excessive overshooting. These characteristics indicate strong potential for real-world implementation in autonomous vehicle systems.

2. DEEP REINFORCEMENT LEARNING (DRL) METHOD

Artificial Intelligence (AI) consists of multiple subfields, with Machine Learning (ML) as a key component that enhances algorithmic performance through data analysis (Mohri et al., 2018). ML methodologies are categorized into three primary approaches: reinforcement learning, supervised learning, and unsupervised learning.

In reinforcement learning (RL), autonomous agents learn optimal behaviors by maximizing cumulative rewards within an environment. The system provides positive reinforcement for beneficial actions and penalties for undesirable ones. Supervised learning relies on pre-labeled training data from domain experts, but this approach faces limitations in interactive scenarios where accurate labeling is challenging. Unsupervised learning focuses on discovering latent patterns in unannotated data, but unlike RL, it does not optimize for specific reward outcomes (R. S. Sutton and A. G. Barto, 2018).

For RL problems with extensive state and action spaces, Artificial Neural Networks (ANNs) can serve as effective approximators. The integration of ANNs with RL forms Deep Reinforcement Learning (DRL). RL frameworks typically use Markov Decision Processes (MDPs), which are mathematical models that formalize decision-making scenarios involving both stochastic elements and agent choices. MDPs are widely applied in reinforcement learning and operations research. The core elements of a Markov Decision Process include:

States (S): MDPs consist of a collection of states that represent all potential situations or arrangements within the environment. The nature of these states can be either discrete or continuous, depending on the specific problem domain.

Actions (A): In an MDP, each state has a specific set of possible actions that can be taken for decision-making. These actions represent the available choices at each state.

Transition Probabilities (P): The probability of moving from one state to another is determined by the transition probabilities after a specific action is taken. These probabilities define the distribution of possible next states based on the current state and action.

Rewards (R): Each state-action pair has an associated reward, representing the immediate benefit or cost of taking that action in that state. Rewards can be positive (rewards) or negative (penalties), and may be deterministic or stochastic.

Policy (π): A policy is a mapping from states to actions, specifying the strategy for selecting actions at each state. The objective of reinforcement learning algorithms is often to find an optimal policy that maximizes the expected cumulative reward over time.

Value Function (Q): The value function represents the expected total reward that can be obtained by following a particular policy or by taking a specific action in a given state. It is used to evaluate the effectiveness of different policies or actions.

The expected discounted reward Rt after the t-th time step can be expressed as:

$$R_t = \sum_t^\infty \gamma^t . r_t \quad (1)$$

Where γ is in the range [0, 1] it is a discount factor. t represents the current time step in the episode

Also, a policy $\pi(a \mid s)$ is mapping from states to action probabilities. A $v_\pi(s)$ be able to be recognized as an expected return regarding a policy π from a state s and is a value function and it is as follows:

$$V^{\pi}(s_t) = E_{\pi}\,[R_t|S_t\,,\ \pi] \quad (2)$$

A $Q_\pi(s, a)$ is action-value function as follow:

$$Q^{\pi}(s_t, a_t) = E_{\pi}[R_t|S_t,\ a_t,\ \pi] \quad (3)$$

Which also the iterative Bellman equation is satisfied:

$$Q^{\pi}(s_t, a_t) = E_{\pi}[r_t + \gamma\ \max Q^{\pi}(s_{t+1},\ a_{t+1})] \quad (4)$$

Not all reinforcement learning problems can be formulated as Markov Decision Processes (MDPs). When environmental states are only partially observable or not directly measurable, these scenarios are better modeled as Partially Observable Markov Decision Processes (POMDPs). A common solution involves augmenting current observations with historical data or prior knowledge, effectively converting the POMDP into a solvable MDP framework (Cao et al., 2023). The fundamental objective of any RL algorithm remains to develop an optimal policy that maximizes cumulative expected rewards.

2.1. Deep Deterministic Policy Gradient (DDPG) Approach

The Deep Deterministic Policy Gradient (DDPG) algorithm is a hybrid reinforcement learning approach that combines elements of value-based and policy-based methods. This algorithm is particularly effective for solving reinforcement learning problems with continuous action domains. DDPG uses a dual-network actor-critic framework consisting of:

• Actor Network: Learns and implements the policy function, directly outputting actions from states to optimize the expected cumulative reward.

• Critic Network: Estimates the value function to assess the quality of the actor's chosen actions by predicting their long-term reward potential.

Key characteristics of DDPG include:

Off-policy learning capability, utilizing past experiences stored in a replay buffer for training [30]; model-free operation, requiring no prior knowledge of environment dynamics; specialization in continuous action spaces, distinguishing it from discrete-action algorithms; and broad applicability to control-oriented domains such as robotics and autonomous systems.

A mean squared Bellman error (MSBE) can be interpreted as:

$$L(\emptyset, \theta) = E_D[(Q_\emptyset(s, a) - (r + \gamma\ (1 - d)\max Q_\emptyset(\acute{s}, \acute{a})))^2] \quad (5)$$

The DDPG algorithm merges components from policy gradient methods and Q-learning. Policy gradient methods are used to train the actor-network to directly maximize the expected return, while Q-learning is used to train the critic network to estimate the value of state-action pairs. DDPG also employs target networks, which are duplicates of the actor and critic networks that are updated less frequently than the primary networks, to stabilize training. Additionally, DDPG utilizes an experience replay buffer to store and sample experiences for training, enhancing data diversity and improving sample efficiency (Rizehvandi et al., 2024).

2.2. Deep Q-Network (DQN) Approach

The DQN algorithm is a deep reinforcement learning method that combines Q-learning with deep neural networks to estimate the optimal action-value function in a continuous state space.

DQN learns to associate states with actions directly from raw sensory inputs, typically images, to handle high-dimensional input spaces. To stabilize training and prevent overfitting, it uses experience replay and a target network by storing and randomly sampling past experiences from a replay buffer and periodically updating a target network with the weights of the trained Q-network. By iteratively updating through gradient descent, DQN minimizes the temporal difference

error between the predicted Q-values and the target Q-values, ultimately learning to maximize cumulative rewards in the environment.

2.3. Modeling and Implementation of the Multi-Agent Reinforcement Learning Environment

The simulation environment was developed in Simulink® and models an ego vehicle and a lead vehicle operating on a single-lane roadway. The ego vehicle dynamics are represented by a simplified bicycle model, while the lead vehicle follows a longitudinal kinematic model. The control system aims for the ego vehicle to maintain a target velocity and a safe following distance from the lead vehicle through longitudinal acceleration and braking, while also keeping its position along the lane centerline using steering control. Two reinforcement learning (RL) agents are used: a DDPG agent for longitudinal control and a DQN agent for lateral control. The simulation episode terminates if any of the following conditions occur: the magnitude of lateral deviation exceeds 1 m, the ego vehicle velocity drops below 0.5 m/s, or the distance between the ego and lead vehicle becomes negative.

2.3.1. Longitudinal Controller (DDPG Agent)

The longitudinal observation vector includes three elements:

Velocity error $e_V = V_{ref} - V$, Integral of velocity error $\int e_V \, dt$, and ego vehicle velocity V.

The reference velocity V_{ref} is dynamically defined based on the relative distance to the lead vehicle. If the relative distance is below the safe distance, V_{ref} is the minimum of the lead vehicle velocity and the driver-set velocity; otherwise, V_{ref} equals the driver-set velocity.

Also, the action space is the longitudinal acceleration of the DDPG agent, bounded within the range $[-3, 2]$ m/s².

The reward at each time step tis computed as $\text{equation } (6)$

$$R_t = -(10\, e^2_v + 100\, a^2_{t-1}) \times 0.001 - 10\, F_t + M_t \quad \text{..... (6)}$$

Where a_{t-1} is the previous acceleration, $F_t = 1$ if the simulation terminates and 0 otherwise, and $M_t = 1$ if $e^2_v < 1$ and 0 otherwise. This reward penalizes large velocity errors, excessive accelerations, and premature terminations, while encouraging accurate velocity tracking.

2.3.2. Lateral Controller (DQN Agent)

The lateral observation vector contains six elements:

Lateral deviation e_1, relative yaw angle e_2, derivatives e_1 and e_2, $\int e_1 \, dt$, and $\int e_2 \, dt$. Also, the action space is a discrete set of steering angle commands ranging from -0.2618 to 0.2618 rad.

The reward at each time step t is given by equation (7) as follows:

$$R_t = -(100\, e^2_1 + 500\, u^2_{t-1}) \times 0.001 - 10\, F_t + 2H_t \quad \text{..... (7)}$$

Where u_{t-1} is the previous steering input, $F_t = 1$ if the simulation terminates and 0 otherwise, and $H_t = 1$ if $e^2_1 < 0.01$ and 0 otherwise. This formulation penalizes large lateral errors and abrupt steering actions while rewarding precise lane centering. The environment object is created in the Matlab – Simulink software and a custom reset function initializes the ego and lead vehicle states at the beginning of each training episode to ensure variability and enhance policy generalization.

Also, the hyper parameters for DDPG and DQN agents are as Table 1.

Agent	Hyper parameter	Value
DDPG (Longitudinal)	Actor Learning Rate	1e-4
	Critic Learning Rate	1e-3
	Discount Factor (γ)	0.99
	Soft Update Rate	0.001
	Batch Size	64
	Replay Buffer Size	1e6
DQN (Lateral)	Learning Rate	1e-3
	Discount Factor (γ)	0.99
	Mini-Batch Size	64
	Replay Buffer Size	1e5

Table 1. Hyper parameter

3. SIMULATION AND RESULTS

In our research, we conducted simulations using MATLAB R2022a, which offers a comprehensive framework for reinforcement learning through its Reinforcement Learning Toolbox. MATLAB also provides a Deep Learning Toolbox to support the training and development of neural networks. We integrated a customized deep neural network into the Reinforcement Learning Toolbox and initialized a Deep Deterministic Policy Gradient (DDPG) agent for training. The DDPG agent uses actor-critic networks, and we applied optimization algorithms such as the Adam optimizer to update network weights during training. Additionally, we developed a tailored path-following environment in Simulink, a MATLAB feature, for agent training. This environment allowed us to assess the agent's performance under different conditions and simulate different scenarios. Figure 1 shows the training progress of the DDPG agent over 1,000 episodes for path-following tasks within the MATLAB environment.

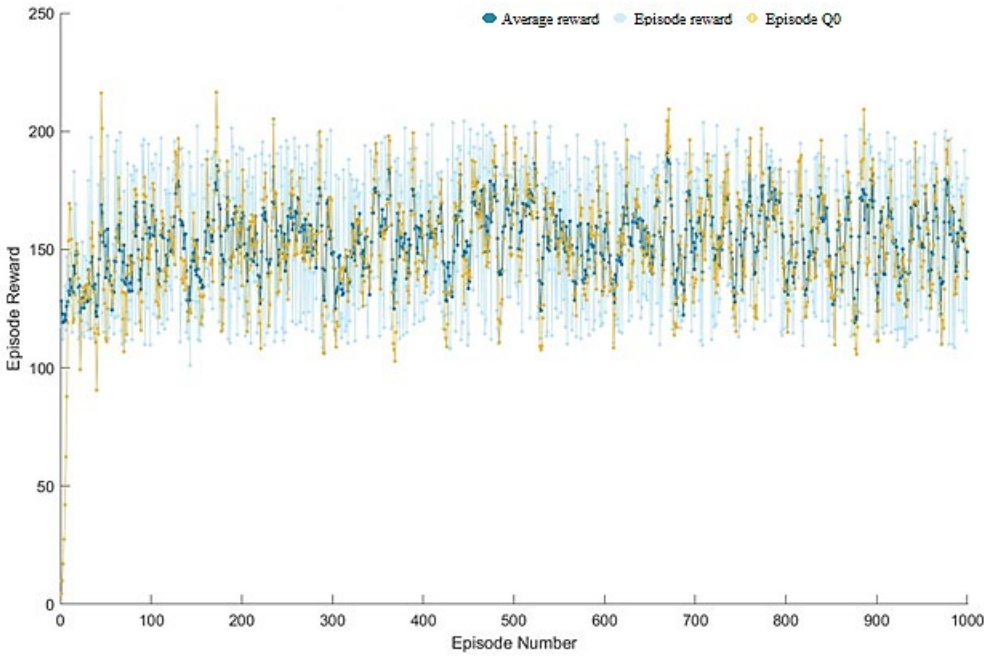

Figure 1. DDPG agent process

Figure 1 illustrates the DDPG agent training process in the Simulink environment at 1000 episodes.

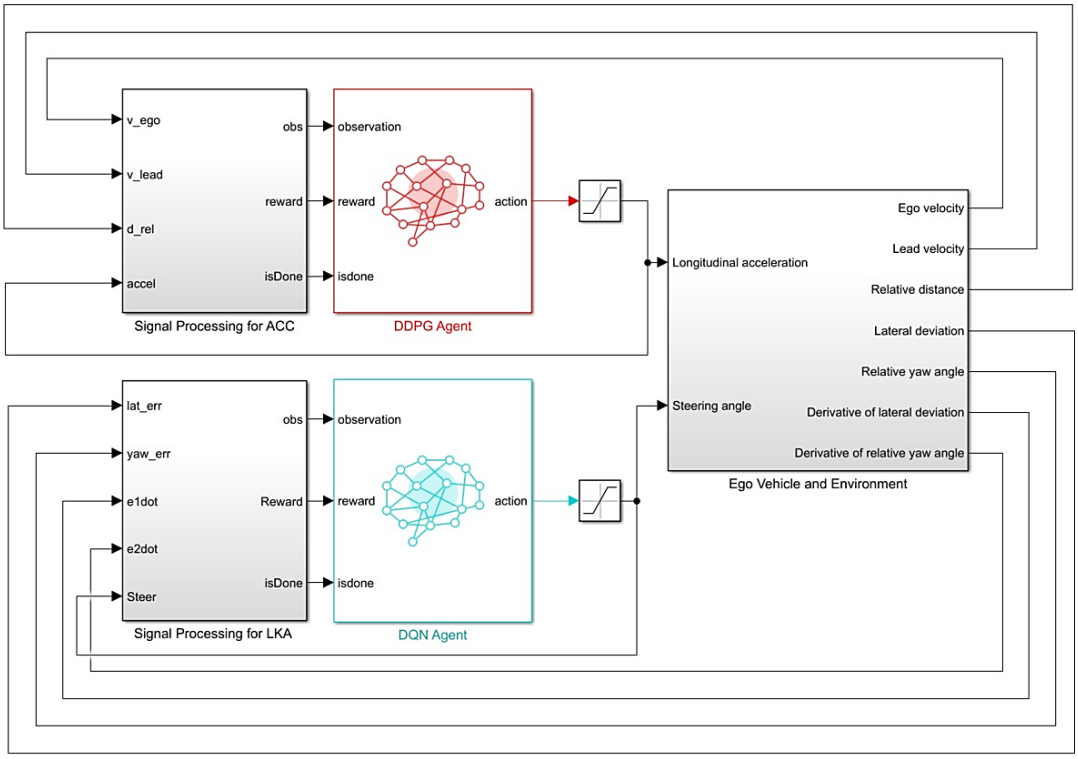

Figure 2. Block diagram

Figure 2 shows the overall architecture of the dual-agent DRL-based control framework for autonomous vehicle path following. The upper module depicts the DDPG agent responsible for longitudinal control, which processes observations such as ego and lead vehicle velocities, relative distance, and acceleration to produce longitudinal acceleration actions. The lower module shows the DQN agent managing lateral control, using lateral error, yaw error, and their derivatives as inputs to generate steering angle actions. Both agents interact with the vehicle and environment model, receiving observations and rewards while executing actions that affect the vehicle's velocity, relative distance, lateral deviation, and yaw dynamics.

In this section, we discuss the results obtained from implementing the Deep Deterministic Policy Gradient (DDPG) and Deep Q Network (DQN) methodologies for the path-following task. This research demonstrates a strategy for coordinating multiple agents to collaboratively perform path-following control (PFC) for a vehicle. The primary objective of PFC is to guide the ego vehicle to maintain a predetermined speed while ensuring a safe distance from a leading vehicle by managing longitudinal acceleration and braking. Additionally, it aims to keep the vehicle aligned with the centerline of its lane by adjusting the steering angle. In this study, two reinforcement learning agents have been developed: the DDPG agent, which generates continuous acceleration values for longitudinal control, and the DQN agent, which provides discrete steering angle values for lateral control. The trained agents work together effectively to implement PFC and achieve favorable outcomes. The scenario examined involves a simplified bicycle model for the ego vehicle and a basic longitudinal model for the lead vehicle. The primary focus during training is to ensure that the ego vehicle maintains a consistent speed while safely following the lead vehicle, achieved through the regulation of longitudinal acceleration and braking. Furthermore, the ego vehicle must remain centered in its lane by adjusting the front steering angle.

The DDPG agent used for the longitudinal control loop evaluates long-term rewards based on its observations and actions. This is accomplished through a critic value function that estimates the value of specific actions in a given state, along with an actor policy that determines the appropriate actions to take in those states. Additional details regarding the development of deep neural network representations for both the value function and policy can be found in existing literature on reinforcement learning algorithms such as DDPG. The DQN agent is also used for lateral control.

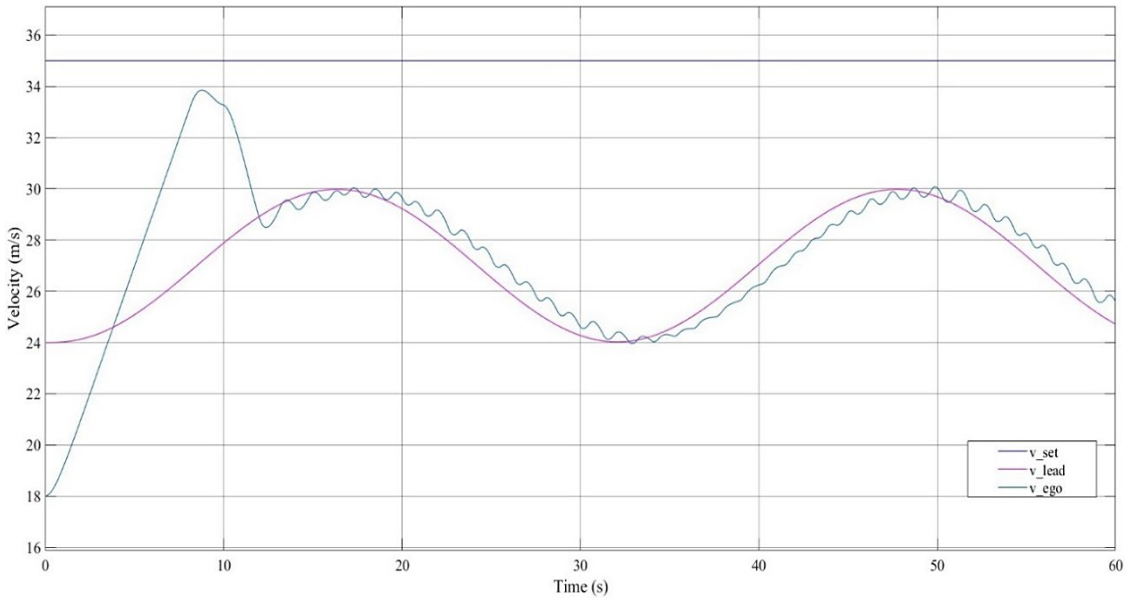

Figure 3. Agent velocity

Figure 3 presents a velocity–time profile illustrating the dynamic interaction among three entities v_{set}, v_{lead}, and v_{ego} over a 60-second simulation period. The vertical axis denotes velocity (0–36 m/s), while the horizontal axis represents time. Conceptually, this figure highlights the comparative motion behavior of the ego vehicle relative to the lead vehicle and the predefined target (set) velocity. The v_{set} trajectory begins at the highest speed, exceeding 30 m/s, representing the desired or driver-set reference velocity. The v_{lead} curve starts at a moderately high velocity of approximately 26 m/s, corresponding to the lead vehicle's motion, while the v_{ego} trace begins near 18 m/s, indicating the ego vehicle's initial speed deficit. The evolving relationship among these curves conceptually demonstrates the control strategy's ability to regulate the ego vehicle's acceleration to track the reference velocity while maintaining a safe distance from the lead vehicle. The convergence or divergence of these traces over time reflects the performance of the reinforcement learning–based controller in achieving stable and adaptive longitudinal behavior.

Figure 4 illustrates the temporal evolution of inter-vehicle spacing by comparing the relative distance and safe distance between the ego and lead vehicles. The vertical axis represents distance (m), while the horizontal axis represents time (s). Conceptually, this figure demonstrates how the ego vehicle adapts its longitudinal motion to maintain a safe following distance under varying traffic conditions. The relative distance curve depicts the instantaneous gap between the two vehicles and fluctuates in response to changes in their velocities. In contrast, the safe distance trajectory represents the dynamically computed minimum separation required for safe operation, typically defined as a linear function of the ego vehicle's velocity and time headway. The regions where the relative distance approaches or falls below the safe distance line indicate instances of potential risk, prompting corrective action by the control policy. The convergence and divergence patterns of these curves provide valuable insight into the reinforcement learning controller's capacity to preserve safety margins and ensure stable car-following behavior throughout the simulation period.

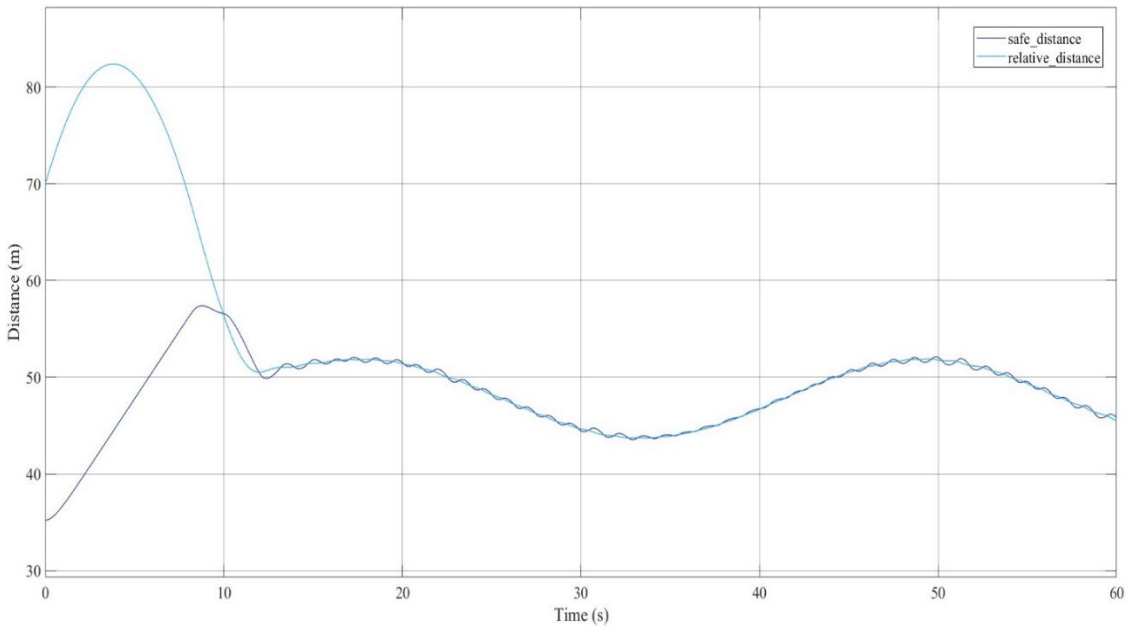

Figure 4. Agent distance

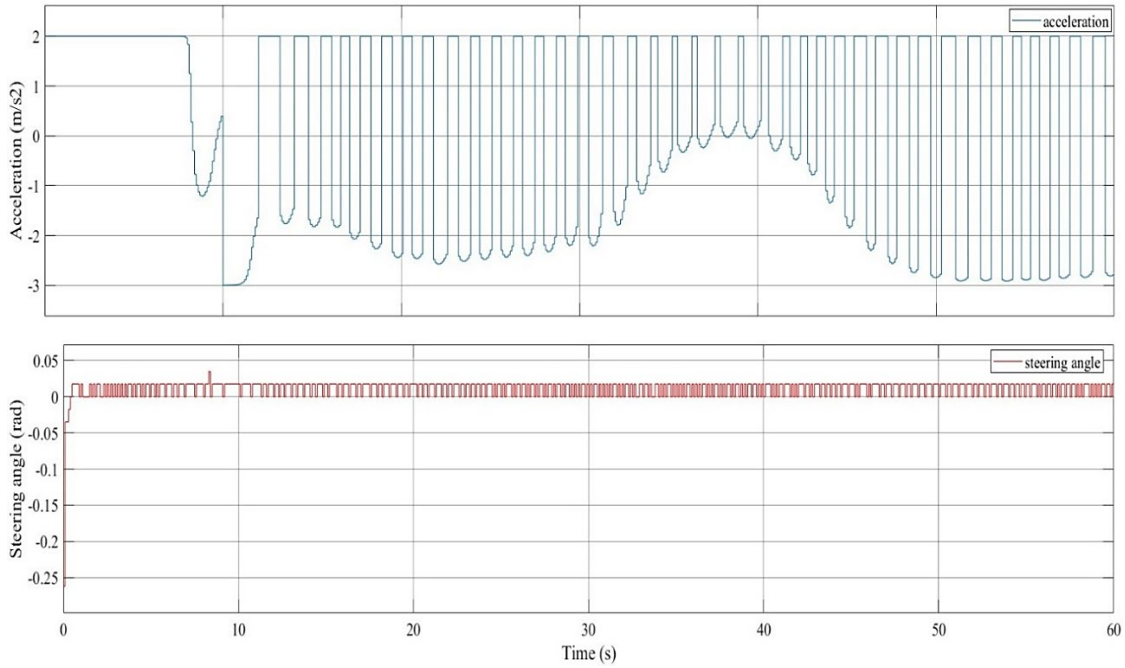

Figure 5. Agent acceleration (using DDPG algorithm), and agent steering angle (using DQN algorithm)

Figure 5 shows the temporal behavior of the vehicle's steering control inputs and the corresponding longitudinal acceleration over a 60-second simulation period. The graph illustrates the coordinated interaction between lateral and longitudinal control actions managed by the reinforcement learning agents. The steering angle (in radians) initially exhibits a small positive deviation of about 0.05 rad, followed by a gradual and sustained shift toward –0.25 rad, indicating a continuous steering maneuver – possibly representing a lane-centering correction, a gentle turn, or curve negotiation. At the same time, the longitudinal acceleration profile (in m/s²) reflects how the controller modulates throttle and braking to maintain stability and velocity during this steering action. Although specific acceleration values are not provided, their temporal correlation with the steering signal highlights the system's capacity for coordinated control – balancing speed regulation with directional stability. Overall, this figure conceptually demonstrates the effectiveness of the multi-agent reinforcement learning framework in achieving smooth, synchronized lateral and longitudinal vehicle responses under dynamic driving conditions.

The simulation models the interaction between a lead vehicle and an ego vehicle under dynamic driving conditions. The lead car intermittently varies its speed between 24 m/s and 30 m/s, simulating realistic traffic scenarios and requiring the ego vehicle to adapt its motion accordingly. The ego car maintains a safe following distance, likely through adaptive cruise control, adjusting its speed to respond to the lead car's accelerations and decelerations, thereby preventing collisions. During the first 30 seconds, the ego vehicle actively modulates its acceleration to achieve or maintain a target velocity, after which its motion stabilizes, indicating convergence to a cruising speed or synchronization with the lead car. Lateral deviation is also monitored, with the ego vehicle correcting any off-center movement within one second and maintaining deviations below 0.1 m, reflecting precise lane-keeping capability. The simulation demonstrates the ego car's ability to adaptively manage longitudinal speed and maintain lateral stability, highlighting the effectiveness of the control system in dynamic driving conditions.

4. CONCLUSION

This study presented a DRL-based control framework for path following in an autonomous ground vehicle. Two distinct agents were used to simultaneously manage steering angle and acceleration control, effectively reducing the action dimension while maintaining stable vehicle behavior. The DDPG agent ensured safe inter-vehicle distance and adaptive velocity tracking, while the DQN agent provided smooth and consistent steering control during navigation. The findings demonstrate the potential of DRL algorithms to enhance vehicle autonomy by achieving reliable path following under dynamic conditions. Future work will extend the current framework to include collision and obstacle avoidance, optimizing performance for multiple objectives such as safety, comfort, and energy efficiency. Further experimental validation of multi-objective control configurations will help compare the performance of DRL-based controllers with classical approaches. Additionally, upcoming studies will investigate the robustness of these controllers in more complex environments, including dense traffic and unpredictable external disturbances, to assess their suitability for real-time and real-world deployment.

CONFLICT OF INTEREST

Authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

REFERENCES

Aguiar, A.P., Hespanha, J.P. and Kokotović, P.V., 2008. Performance limitations in reference tracking and path following for nonlinear systems. Automatica, 44, pp.598–610.

Amer, N.H. et al., 2017. Modelling and control strategies in path tracking control for autonomous ground vehicles: A review of state of the art and challenges. Journal of Intelligent & Robotic Systems, 86, pp.225–254.

Amidi, O. and Thorpe, C.E., 1991. Integrated mobile robot control. In: Proceedings of Mobile Robots V, Boston, MA, pp.504–523.

California Department of Motor Vehicles, 2021. Autonomous vehicles testing with a driver. Sacramento, CA: California DMV.

California Department of Motor Vehicles, 2022. Autonomous vehicles testing with a driver. Sacramento, CA: California DMV.

Cao, Y. et al., 2023. Path following for autonomous ground vehicle using DDPG algorithm: A reinforcement learning approach. Applied Sciences, 13(11), 6847. Available at: https://doi.org/10.3390/app13116847

Cheng, X. et al., 2022. Path-following and obstacle avoidance control of nonholonomic wheeled mobile robot based on deep reinforcement learning. Applied Sciences, 12, 6874. Available at: https://doi.org/10.3390/app12096874

Coulter, R.C., 1992. Implementation of the pure pursuit path tracking algorithm. Pittsburgh, PA: Carnegie Mellon University.

Faulwasser, T. et al., 2009. Model predictive path-following for constrained nonlinear systems. In: Proceedings of the 48th IEEE Conference on Decision and Control (CDC), Shanghai, China, 15–18 December, pp.8642–8647.

Koopman, P. and Wagner, M., 2017. Autonomous vehicle safety: An interdisciplinary challenge. IEEE Intelligent Transportation Systems Magazine, 9, pp.90–96.

Ma, R. et al., 2023. Sample-observed soft actor–critic learning for path following of a biomimetic underwater vehicle. IEEE Transactions on Automation Science and Engineering, pp.1–10.

Mnih, V. et al., 2015. Human-level control through deep reinforcement learning. Nature, 518, pp.529–533.

Mohri, M., Rostamizadeh, A. and Talwalkar, A., 2018. Foundations of machine learning. Cambridge, MA: MIT Press.

Montgomery, W.D. et al., 2022. America's workforce and the self-driving future: Realizing productivity gains and spurring economic growth. Available at: https://trid.trb.org/view/1516782 (Accessed: 8 November 2022).

Paden, B. et al., 2016. A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Transactions on Intelligent Vehicles, 1, pp.33–55.

Rubí, B., Morcego, B. and Pérez, R., 2021. Deep reinforcement learning for quadrotor path following with adaptive velocity. Autonomous Robots, 45, pp.119–134.

Samson, C., 1992. Path following and time-varying feedback stabilization of a wheeled mobile robot. In: Proceedings of the Second International Conference on Automation, Robotics and Computer Vision, pp.1.

Schulman, J. et al., 2017. Proximal policy optimization algorithms. arXiv preprint. Available at: https://arxiv.org/abs/1707.06347

Singh, S., 2018. Critical reasons for crashes investigated in the National Motor Vehicle Crash Causation Survey. Washington, DC: National Highway Traffic Safety Administration.

Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. Cambridge, MA: MIT Press.

Thrun, S. et al., 2006. Stanley: The robot that won the DARPA Grand Challenge. Journal of Field Robotics, 23, pp.661–692.

Thrun, S., Burgard, W. and Fox, D., 2005. Probabilistic robotics. Cambridge, MA: MIT Press.

World Health Organization, 2018. Global status report on road safety. Geneva: World Health Organization.

Yurtsever, E. et al., 2020. A survey of autonomous driving: Common practices and emerging technologies. IEEE Access, 8, pp.58443–58469.

Zhao, W. et al., 2020. Sim-to-real transfer in deep reinforcement learning for robotics: A survey. In: Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, Australia, 1–4 December, pp.737–744.

Zheng, Y. et al., 2023. DDPG-based active disturbance rejection 3D path-following control for powered parafoil under wind disturbances. Nonlinear Dynamics, 111, pp.11205–11221.