INTERNATIONAL JOURNAL OF AUTOMOTIVE AND MECHANICAL ENGINEERING
ISSN: 2229-8649 e-ISSN: 2180-1606
VOL. 21, ISSUE 3, 2024, 11682 – 11694
DOI: https://doi.org/10.15282/ijame.21.3.2024.18.0901

RESEARCH ARTICLE

Design of a Path-Following Controller for Autonomous Vehicles Using an Optimized Deep Deterministic Policy Gradient Method

Ali Rizehvandi*, Shahram Azadi

Faculty of Mechanical Engineering, K.N.Toosi University of Technology, Tehran, Iran

ABSTRACT - The need for a safe and reliable transportation system has made the advancement of autonomous vehicles (Avs) increasingly significant. To achieve Level 5 autonomy, as defined by the Society of Automotive Engineers, AVs must be capable of navigating complex and unconventional traffic environments. Path-following is a crucial task in autonomous driving, requiring precise and safe navigation along a defined path. Traditional path-tracking methods often rely on parameter tuning or rule-based approaches, which may not be suitable for dynamic and complex environments. Reinforcement learning has emerged as a powerful technique for developing effective control strategies through agent-environment interactions. This study investigates the efficiency of an optimized Deep Deterministic Policy Gradient (DDPG) method for controlling acceleration and steering in the path-following of autonomous vehicles. The algorithm demonstrates rapid convergence, enabling stable and efficient path tracking. Additionally, the trained agent achieves smooth control without extreme actions. The performance of the optimized DDPG is compared with the standard DDPG algorithm, with results confirming the improved efficiency of the optimized approach. This advancement could significantly contribute to the development of autonomous driving technology.

ARTICLE HISTORY

Received	:	06th May 2024
Revised	:	22nd Aug. 2024
Accepted	:	02nd Sept. 2024
Published	:	20th Sept. 2024

KEYWORDS

Autonomous vehicles
DRL method
DDPG algorithm
Path-following

1. INTRODUCTION

Several organizations, both in industry and research, are aggressively improving autonomous vehicles (AVs), which can potentially save thousands of lives annually and save time daily. About 1.35 million road traffic accident deaths occur annually, and between 20 and 50 million individuals sustain non-fatal injuries, based on the World Health Organization (WHO) report [1]. In the USA, the National Highway Traffic Safety Administration (NHTSA) report states that 94% of driving accidents happened because of human driving mistakes, while 2% were because of car failure [2, 3]. Therefore, hopefully, enhancing AV adoption will decrease car accidents and thus reduce road traffic fatality statistics. If their extensive placement is fruitful, the annual social forecast profits of AVs, which contain decreasing traffic congestion and the incidence of road accidents, less energy-wasting, and increasing efficiency as a consequence of redistributing driving time, will achieve approximately 800,000 million dollars by 2050 [4]. Though in static environments, the vehicle's moving technology is improved [5], the complex and dynamic conditions of real driving environments have made it sorely challenging for extensive AV acceptance [6]. To test AV technology in the real-world driving environment, the Department of Motor Vehicles (DMV) in the United States issued licenses to producers under the program of AV tester in 2014 [7]. DMV in California needs producers to experiment with AVs and report any crash resulting in property damage, injury, or fatalities [8].

Path-following has been extensively considered as another option for tracking routes in numerous types of automobiles because of its slack, which is subject to time constraints [9, 10]. In problems of tracking of trajectory, the designated trajectory expresses when the automobile is assumed to be within the state space involves maintaining the car as close as possible and navigating along a predetermined path without any time reallocation [10, 12, 13]. One of the first suggested approaches for path-following is pure pursuit [14, 15]. This approach's straightforwardness has caused it to be widely used in many applications where real-time control is vital. Supposing that at a constant predefined speed, the vehicle is in motion, and the reference geometry path is devoid of any curvature. The pure pursuit method is according to appropriate a circle through the structure of the vehicle's current, which is the rear-wheel position in the case of a car, by what is commonly referred to as a lookahead distance (L) to a point along the path in front of the vehicle [9, 16]. However, if the current structure is further away from the reference geometry path than L, then the controller does not consider the case for evaluation. Among the classical control methods, a method in which, as a continuous function, the path of reference geometry is parameterized, and the position of the rear wheel is utilized as a variable with a set delay to minimize the error of cross path between the reference path and the rear wheel, while similarly guaranteeing the car heading stability, has been suggested in [17].

While the control law guarantees rapid local convergence, the curvature of the reference geometry path remains crucial, necessitating the function to be twice continuously differentiable. The alternative strategy of control, explored in [18], focuses on minimizing the error in the front wheel's cross-track position is a reference geometric path. In this method,

***CORRESPONDING AUTHOR | A. Rizehvandi** | ✉ Ali.Rizehvandi75@gmail.com
© 2024 The Author(s). Published by Universiti Malaysia Pahang Al-Sultan Abdullah Publishing. This is an open access article under the CC BY-NC 4.0 International license

to guarantee local exponential convergence towards the reference geometric path, the cross-track error is utilized with a nonlinear feedback mechanism, but to be utilized for reverse driving, it requires some variations. The automobile robot utilizing this control method for steering won the DARPA Challenge in 2005. The above methods [14, 17, 18] are appropriate as standards for comparison and reference because they accomplish acceptable performance with a minimal set of parameter uncertainty requirements and a moderate level of accuracy of the model.

In the last decade, the Reinforcement Learning (RL) method has accomplished meaningful achievements in various fields, like gaming, robotics, and autonomous vehicles, which has brought it widespread recognition and excellent attention [19, 20]. A machine learning method called RL enables an agent to learn to make optimal decisions through its interactions with the environment. While RL methods enable an agent to learn to achieve a goal in a manner similar to humans based on predefined reward functions, they may struggle to adapt to new environments. Safety becomes a critical issue during the training process when RL methods are utilized in real-world situations, as they involve exploration of the action space during training, potentially resulting in unsafe RL actions. The use of deep neural networks (DNNs) to estimate the value function (Q-function) has made it possible to address high-dimensional state space problems in reinforcement learning, a field known as Deep Reinforcement Learning (DRL). More recent examples of DRL algorithms contain deep Q-network (DQN) [21], proximal policy optimization (PPO), and deep deterministic policy gradient (DDPG) [22].

The DDPG method [23], which incorporates the two deep Q-network (DQN) and deterministic policy gradient (DPG) approaches, has thus established extensive uses in fields like control of robotics and control of autonomous vehicles. In recent years, abundant researchers have tried to apply DRL techniques to resolve the path-following problem. Rubi et al. [13] investigated three consecutively developed techniques according to the DDPG algorithm, which realized the adaptive velocity control and path tracking of a quadrotor. Cheng et al. [24] achieved collision avoidance and path following for a non-holonomic wheeled mobile robot in simulation, but leading to a high jerk in robot velocity, the trained agent applied extreme control effort. Also, Zheng et al. [25], for powered parafoils, present a 3D path-following control technique to efficiently control the flight trajectory of the parafoils and disturbances against the wind, using the composition of linear active disturbance rejection control and DDPG algorithm. Ma et al. [26], based on soft actor criticism (SAC), investigate a path-following control system for a submarine, representing effective path tracking. This study presents the DDPG-optimized algorithm application to the path-following system for submarines that travel at a moderate speed, with the main purpose of minimizing lateral deviation and creating a smooth steering angle.

In the current work, we introduce the novel integration of the DDPG algorithm with an optimized approach tailored specifically for autonomous vehicle control. By incorporating advanced optimization techniques into the DDPG framework, we have enhanced the algorithm's ability to effectively learn and execute complex control tasks, such as path following and trajectory tracking. This optimized DDPG framework demonstrates superior performance in terms of both computational efficiency and control precision compared to standard implementations. Through meticulous experimentation and analysis, we have showcased the effectiveness of our approach in reducing lateral deviation, improving steering control, and ensuring smoother acceleration profiles, ultimately enhancing the safety and comfort of autonomous vehicle navigation. By pushing the boundaries of reinforcement learning in autonomous driving applications, our work represents a significant advancement in the field, with implications for the development of more reliable and robust autonomous systems in real-world scenarios.

Our contribution lies in the innovative hybridization of optimization strategies within the DDPG algorithm framework. By employing the Adam optimizer for the critic network and the RMSprop optimizer for the actor network during training, we present a novel approach to balancing exploration and exploitation in DRL methods. This dual-optimizer scheme leverages the strengths of each optimizer, harnessing Adam's adaptive learning rates and momentum for efficient critic network updates while simultaneously utilizing RMSprop's stability and convergence properties for actor-network parameter updates. Through extensive experimentation and analysis, we have demonstrated the effectiveness of this hybrid optimization strategy in enhancing the learning dynamics, accelerating convergence, and improving the overall performance of the DDPG algorithm for autonomous vehicle control tasks. This novel optimization scheme represents a significant advancement in reinforcement learning methodologies, with potential applications across various domains requiring adaptive and efficient learning algorithms.

Also, to accomplish smooth steering and comfortable driving when accelerating during vehicle operation, we utilize a suitable reward function. Compared to standard methods, this approach demonstrates superior performance on the learned trajectory. The control strategy of the trained agent has shown quick responsiveness to lateral deviations from the desired path, with a satisfactory amount of overshoot. Due to these advantages, this technique holds significant promise for practical applications. Finally, the results of the DDPG-optimized algorithm are compared to those of the DDPG algorithm, and the advantages of that are evaluated.

2. VEHICLE DYNAMIC MODEL

In the present study, the bicycle model (2 DOF model) is used to simulate vehicle dynamics. Figure 1 demonstrates the dynamic model of the car.

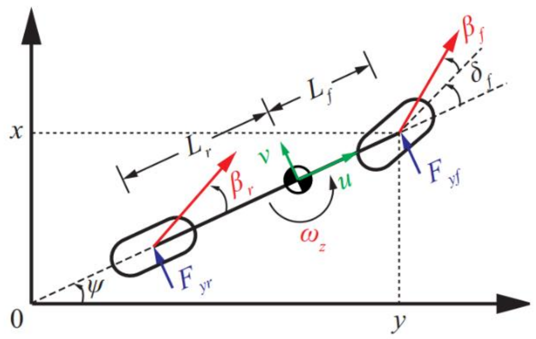


Figure 1. Dynamic model of the car [27]

The state space equations of the vehicle dynamic model are as follows:

$$\begin{bmatrix}\frac{dy}{dt}\\ \frac{d^2y}{dt^2}\\ \frac{d\Psi}{dt}\\ \frac{d^2\Psi}{dt^2}\end{bmatrix} = -\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & \frac{2C_{\alpha f}+2C_{\alpha r}}{mV_x} & 0 & V_x+\frac{2L_fC_{\alpha f}-2L_rC_{\alpha r}}{mV_x}\\ 0 & 0 & 0 & 1\\ 0 & \frac{2L_fC_{\alpha f}-2L_rC_{\alpha r}}{I_zV_x} & 0 & \frac{2l_f^2C_{\alpha f}+2l_r^2C_{\alpha r}}{I_zV_x}\end{bmatrix}\begin{bmatrix}y\\ \frac{dy}{dt}\\ \Psi\\ \frac{d\Psi}{dt}\end{bmatrix} + \begin{bmatrix}0\\ \frac{2C_{\alpha f}}{m}\\ 0\\ \frac{2L_fC_{\alpha f}}{I_z}\end{bmatrix}\delta_f \tag{1}$$

Now, the lateral deviation (e$_1$) and the relative deviation angle (e$_2$) are represented. To redefine the dynamic model, longitudinal velocity (V$_x$) and radius (R) are considered constant. Radius (R) can be defined as the curvature reverse of the path, as well as the deviation rate of the vehicle as a function of the curvature as follows:

$$\frac{d\Psi}{dt}_{des} = kV_x \tag{2}$$

Therefore, the desired acceleration of the car is described as:

$$\frac{d\Psi}{dt}_{des}.V_x = kV_x^2 \tag{3}$$

Hence, the following equations (4 and 5) are achieved:

$$\begin{aligned} m\frac{d^2e_1}{dt^2} = 2C_{\alpha f}\delta_f + \frac{de_1}{dt}.\left[-\frac{2}{V_x}C_{\alpha f}-\frac{2}{V_x}C_{\alpha r}\right] + e_2\left[2C_{\alpha f}+2C_{\alpha r}\right] + \frac{de_2}{dt}.\left[-\frac{2}{V_x}C_{\alpha f}l_f+\frac{2}{V_x}C_{\alpha r}l_r\right]\\ +\frac{d\Psi}{dt}_{des}.\left[-\frac{2}{V_x}C_{\alpha f}l_f+\frac{2}{V_x}C_{\alpha r}l_r\right] \end{aligned} \tag{4}$$

$$\begin{aligned} I_z\frac{d^2e_2}{dt^2} = 2C_{\alpha f}l_f\delta_f + \frac{de_1}{dt}.\left[-\frac{2}{V_x}l_fC_{\alpha f}+\frac{2}{V_x}l_rC_{\alpha r}\right] + e_2\left[2C_{\alpha f}l_f+2C_{\alpha r}l_r\right] + \frac{de_2}{dt}.\left[-\frac{2}{V_x}C_{\alpha f}l_f^2-\frac{2}{V_x}C_{\alpha r}l_r^2\right]\\ -I_z\frac{d^2\Psi_{des}}{dt^2}+\frac{d\Psi}{dt}_{des}.\left[-\frac{2}{V_x}C_{\alpha f}l_f^2+\frac{2}{V_x}C_{\alpha r}l_r^2\right] \end{aligned} \tag{5}$$

Eventually, the equations of the state-space are redefined by the error variables as follows:

$$\begin{aligned} \begin{bmatrix}\frac{de_1}{dt}\\ \frac{d^2e_1}{dt^2}\\ \frac{de_2}{dt}\\ \frac{d^2e_2}{dt^2}\end{bmatrix} = -\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & \frac{2C_{\alpha f}+2C_{\alpha r}}{mV_x} & \frac{-2C_{\alpha f}-2C_{\alpha r}}{m} & \frac{2L_fC_{\alpha f}-2L_rC_{\alpha r}}{mV_x}\\ 0 & 0 & 0 & 1\\ 0 & \frac{2L_fC_{\alpha f}-2L_rC_{\alpha r}}{I_zV_x} & \frac{-2L_fC_{\alpha f}+2L_rC_{\alpha r}}{I_z} & \frac{2l_f^2C_{\alpha f}+2l_r^2C_{\alpha r}}{I_zV_x}\end{bmatrix}\begin{bmatrix}e_1\\ \frac{de_1}{dt}\\ e_2\\ \frac{de_2}{dt}\end{bmatrix} + \begin{bmatrix}0\\ \frac{2C_{\alpha f}}{m}\\ 0\\ \frac{2L_fC_{\alpha f}}{I_z}\end{bmatrix}\delta_f\\ +\begin{bmatrix}0\\ \frac{-2C_{\alpha f}L_f+2C_{\alpha r}L_r}{mV_x}-V_x\\ 0\\ \frac{-2C_{\alpha f}l_f^2-2C_{\alpha r}l_r^2}{I_zV_x}\end{bmatrix}.\frac{d\Psi}{dt}_{des} \end{aligned} \tag{6}$$

Figure 2 also depicts the lateral deviation (e_1) and relative yaw angle (e_2) for the vehicle.

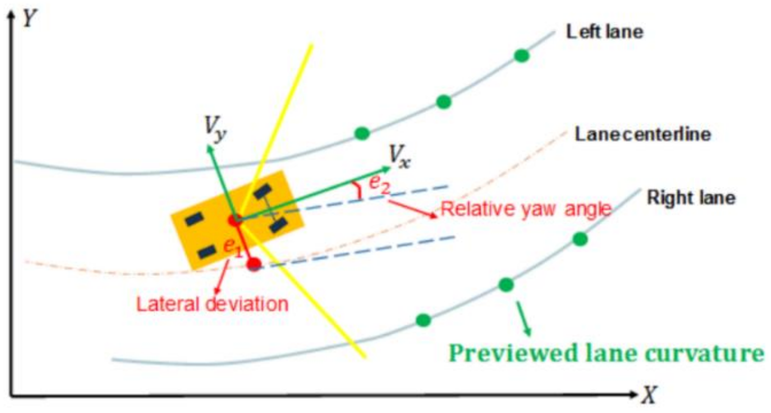


Figure 2. Error variables of the vehicle model

Table 1 presents the parameters and symbols used in the vehicle dynamic model.

Table 1. Parameters of the vehicle

Description	Symbol	Value
The total mass of the vehicle	m	1650 kg
Yaw moment of inertia	I_z	2875 kg.m^2
FrontAxle-C.G.distance	l_f	1.40 m
Rear Axle-C.G. distance	l_r	1.60 m
Front tire cornering stiffness	C_f	19000 N/rad
Rear tire cornering stiffness	C_r	33000 N/rad

3. DEEP REINFORCEMENT LEARNING (DRL) METHOD

Artificial intelligence (AI) encompasses machine learning (ML), a field focused on enhancing computational algorithm performance through data [28]. Machine Learning (ML) consists of three primary types: RL, supervised learning, and unsupervised learning. RL involves an autonomous agent learning to accomplish tasks in an environment by maximizing a predetermined reward function. Agents receive rewards for favorable actions and penalties for unfavorable ones during interactions with their environment. Learning from labeled examples by experts is the essence of supervised learning. Due to the complexity of finding an exact label that represents an interaction, this approach is not suitable for solving interactive problems [29].

Unsupervised learning involves learning to identify a concealed arrangement within data that has not been labeled. While uncovering patterns in gathered data can be beneficial, this method cannot optimize a reward, which is a key goal of reinforcement learning (RL) [29]. RL problems with extensive state and action spaces can utilize an Artificial Neural Network (ANN) to approximate functions. The application of an ANN as a function approximator in RL is known as the DRL method. Issues in reinforcement learning are commonly represented as Markov Decision Processes (MDPs). MDPs are used to model decision-making problems in which outcomes are affected by random elements and the actions taken by an agent. This framework is widely utilized in areas like reinforcement learning and operations research.

The fundamental elements of a Markov Decision Process include the following:

- *States (S):* MDPs are made up of a collection of states that encompass all potential scenarios or configurations of the environment. The states can vary between discrete or continuous, based on the specific problem domain.
- *Actions (A):* In the MDP, every state has a range of possible actions from which the decision-maker can choose. These actions symbolize the options or decisions that are accessible to the decision-maker at every state.
- *Transition Probabilities (P):* The probability of transitioning from one state to another after taking a specific action is determined by the transition probabilities. Put simply, they define the likelihood of moving to the next state based on the current state and action.
- *Rewards (R):* Each state-action pair is linked to a reward that signifies the immediate gain or loss associated with taking that action in that state. The rewards can be either positive (rewards) or negative (penalties), and they could be either deterministic or stochastic.
- *Policy (π):* A policy determines the decision maker's actions in different situations. In reinforcement learning, the goal is to find the best policy that maximizes the total expected reward.

- *Value Function (Q):* The value function represents the expected overall reward that can be obtained by following a certain policy or taking a specific action in a particular state. It assists in assessing the effectiveness of various policies or actions.

MDPs adhere to the Markov property, which means that the subsequent state depends only on the current state and action and is unaffected by the sequence of preceding states and actions. This characteristic simplifies the modeling and examination of decision-making problems and enables the use of different solution methods like dynamic programming, Monte Carlo methods, and temporal difference learning. In general, Markov Decision Processes offer a formal structure for studying decision-making in uncertain situations and are widely utilized in fields such as robotics, autonomous systems, game theory, economics, and more. The value of the discounted reward at time step t is denoted as Rt.

$$\mathrm{R_t} = \sum_{\mathrm{t}}^{\infty} \gamma^{\mathrm{t}}.\,\mathrm{r_t} \tag{7}$$

In the range [0, 1], γ represents a discount factor. Depending on the problem, T can be either an infinite (∞) or a finite value. π (a|s) is a policy that maps states to action probabilities. v_π (s) represents the expected return of a policy π from a state s and is a value function.

$$\mathrm{V^{\pi}(s_t)} = \mathrm{E_{\pi}[R_t | S_t, \pi]} \tag{8}$$

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

$$\mathrm{Q^{\pi}(s_t, a_t)} = \mathrm{E_{\pi}[R_t | S_t, a_t, \pi]} \tag{9}$$

which also the iterative Bellman equation is satisfied:

$$\mathrm{Q^{\pi}(s_t, a_t)} = \mathrm{E_{\pi}[r_t + \gamma \; maxQ^{\pi}(s_{t+1}, a_{t+1})]} \tag{10}$$

However, some RL problems cannot be expressed as MDPs. In certain cases, if the state S is only partially observable from the environment or cannot be observed directly from the defined environment, then our problems can be modeled as Partially Observable Markov Decision Processes (POMDPs). One approach to addressing this issue is to incorporate past knowledge into observations by combining previous observations or prior information with current observations, thereby solving the problem as an MDP [29]. The main goal of the RL algorithm is to learn a policy that maximizes the expected reward. The DDPG algorithm combines elements of value-based and policy-based methods in reinforcement learning. It is especially effective for tackling issues related to continuous action spaces in RL.

The actor-critic framework employed by DDPG consists of two primary networks:

- Actor-Network: The policy function is learned by this network, mapping states to actions with the goal of maximizing the expected return through the direct selection of actions based on the current state.
- Critic Network: The Q-value function is learned by this network, estimating the anticipated return (total reward) from adhering to a particular policy. This aids in assessing the decisions made by the actor-network.

Moreover, DDPG operates as an off-policy algorithm, which implies that it gains knowledge from data obtained from an experience replay buffer without directly adhering to a specific policy. Additionally, it is model-free, indicating that it does not necessitate understanding the underlying dynamics of the environment. In contrast to certain other algorithms that are more suitable for discrete action spaces, DDPG is specifically crafted to manage continuous action spaces, allowing it to be used in a wide range of scenarios, such as robotics and control tasks. A mean squared Bellman error (MSBE) be able to be regarded as:

$$\mathrm{L(\emptyset, \theta)} = \mathrm{E_D}\left[(\mathrm{Q_{\emptyset}(s, a)} - \left(\mathrm{r} + \gamma\,(1 - \mathrm{d})\mathrm{maxQ_{\emptyset}(\acute{s}, \acute{a})}\right))^2\right] \tag{11}$$

The DDPG algorithm merges aspects of policy gradient techniques and Q-learning. Policy gradient methods are utilized to train the actor-network directly in order to maximize the anticipated return, while Q-learning is used to train the critic network to assess the value of state-action pairs. Furthermore, DDPG incorporates target networks in order to stabilize training, consisting of duplicates of the actor and critic networks that are updated less often than the primary networks. In addition, DDPG utilizes an experience replay buffer to store and select experiences during training with the aim of diversifying the data and enhancing sample efficiency.

4. DDPG METHOD IMPLEMENTATION FOR PATH-FOLLOWING CONTROL

4.1 State Space

A state in RL problems is a demonstration of the agent's current environment. This state can be observed by the agent, and it contains all pertinent details regarding the environment that the agent requires to know to make a decision. Figure 3 demonstrates the basic components of the RL model for autonomous vehicles. The DDPG algorithm uses two neural networks, viz. critic and actor networks, which utilize the state as input data. Consequently, all states should be observable.

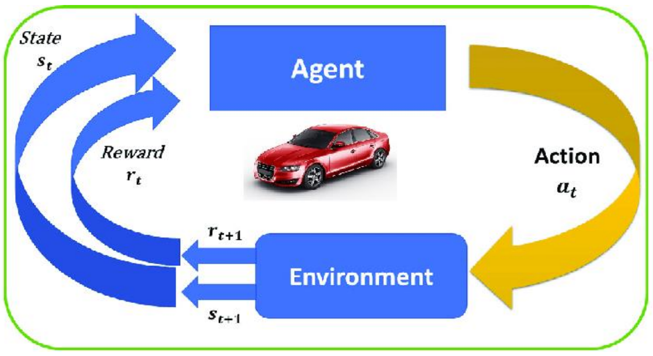


Figure 3. RL model for autonomous vehicle

In the current study, the observations from the environment include longitudinal measurements and lateral measurements. The longitudinal parameters consist of the velocity error, its integral, and the longitudinal speed of the self-driving car. Also, the lateral measurements include the lateral deviation, relative yaw angle, their integrals, and their derivatives as follows:

Longitudinal measurements:

$$S_{Lo} = [V,\ \boldsymbol{e_v},\ \int \boldsymbol{e_v}] \tag{12}$$

Lateral measurements:

$$S_{La} = [\boldsymbol{e_1},\ \boldsymbol{e_2}, \dot{\boldsymbol{e_1}}, \dot{\boldsymbol{e_2}},\ \int \boldsymbol{e_1}\ ,\ \int \boldsymbol{e_2}] \tag{13}$$

4.2 Action Space

In this research, the action space (based on Figure 3) includes steering angle and acceleration actions. The acceleration action signal ranges from [-3, 2] m/s^2 and the steering angle action signal varies between [-15, 15] deg or [-0.262, 0.262] radian.

4.3 Reward Function

In RL problems, a numerical signal known as the reward is optimized by the agent, which is presented by the environment as a reaction to the actions of the agent. The reward for the agent operates as a mechanism of feedback, notifying it of its efficiency in a specific state and holding its future actions towards its aim. In the DRL algorithm, the agent is a neural network (NN) that obtains the observations of the environment as input and also creates actions as output. According to the rewards it receives from the environment, the network parameters are adjusted by a DRL algorithm. In the DRL method, rewards play a crucial role in the process of agent learning. The agent decision-making process is reinforced by positive rewards, while negative rewards disappoint it from reciting definite actions. As the experience agent gains, it recognizes which actions cause the highest reward value and, to maximize future rewards updates its policy. Hence, to learn and boost its decision-making abilities, rewards suggest a critical feedback signal for the DRL agent.

In the present study, the reward function is described as follows:

$$\mathrm{R_t} = -(0.1 * \mathrm{e_1^2}) - (0.01 * \mathrm{e_V^2}) - (0.5 * \mathrm{u_{t-1}^2}) - (0.1 * \mathrm{a_{t-1}^2}) \tag{14}$$

where e_1 is the lateral deviation, e_v is velocity error, and u_{t-1} and a_{t-1} are the steering input from the previous time step and the acceleration input from the previous time step, respectively.

4.4 Training Process

The neural networks of actor-critic both contain two hidden layers, as demonstrated in Figure 4. Each layer consists of a ReLU activation function with 100 neurons. Remarkably, in the network of critics, the vector of state Interfaces with the first hidden layer, while the action is concatenated before the second hidden layer, following the configuration of the primary algorithm. To bypass the first layer, this mechanism allows the action to develop the performance and stability of the networks [30]. The actor-network final layer is a hyperbolic tangent layer utilized to map the action.

In the present study, the Adam optimizer is used to optimize neural networks with a mini-batch size of 128. Also, the initial acceleration of the car is sampled from a consistent acceleration, with a range of [-3, 2] m/s^2 and a steering angle range of [-0.262, 0.262] rad. Through the initial process of training, the agent selects actions randomly from its action space. Following this, the actor network generates actions based on the observed state, the agent transitions to the next state, computes the reward, and saves this experience as a tuple in the experience replay buffer until the total number of experiences reaches the specified mini-batch size. Once this threshold is reached, the networks are then updated and optimized.

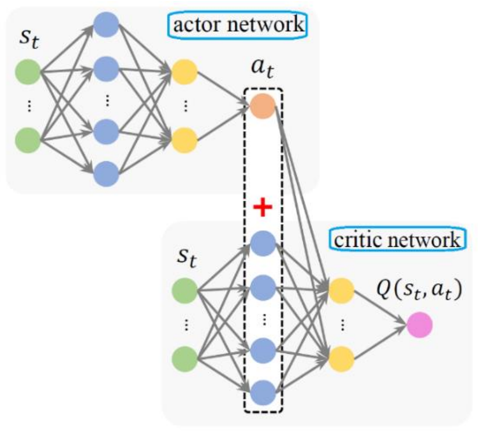


Figure 4. DDPG algorithm structure

The loss function to update the parameters of the actor and critic networks can be determined using equations (15) and (17).

$$L(\theta) = E_{s,a}[(y_t - Q(s_t, a_t|\theta))^2] \tag{15}$$

The variable y_t represents the Temporal Difference (TD) target, which serves as the goal for updating and optimizing the Q-function.

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1})|\theta) \tag{16}$$

Equation (17) illustrates the method for updating the actor parameters θ, by employing the chain rule concerning θ This update utilizes the expected return from the start, denoted as J.

$$\nabla_\theta J = E_S[\nabla_\theta Q(s, a|\theta)|_{s=s_t, a=\mu(s_t|\theta)}] \tag{17}$$

Following the fine-tuning of the original networks, the parameters of the target network undergo adjustment utilizing a soft update technique, as outlined in equation (18). The process includes mixing some of the revised network parameters with the target network's parameters, which improves the consistency of the learning process and avoids oscillations.

$$\acute{\theta} = \tau\theta + (1 - \tau)\acute{\theta} \tag{18}$$

where τ shows how rapidly the update is conducted, and after training the online network, the target network parameter gets updated after each step.

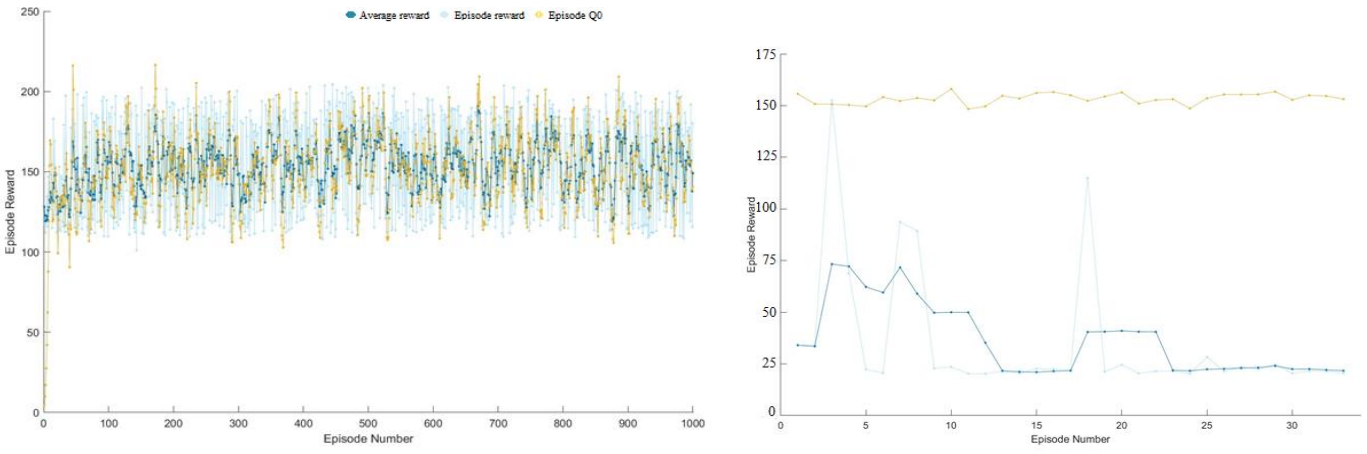
Through the training process, for each episode, a haphazard path is created, and its actions are influenced by exploration noise. The agent's selection is derived from its effectiveness in navigating the environment. In contrast, during evaluation, the agent makes decisions only according to the existing learned policy, with no random exploration noise. Assessment takes place periodically, and the agent with the highest reward performance during these evaluations is chosen. Table 2 outlines the hyperparameters utilized in the training process.

Table 2. Hyperparameters of DDPG-optimized agent

Hyperparameters	Value
Discount factor	0.99
Period	0.1 s
Learning rate actor-network	1e-4
Learning rate critic-network	1e-3
Actor hidden layers	100
Critic hidden layers	100
Training epochs	10
Target soft update rate	0.01
Mini batch size	128
Experience replays buffer size	100,000

4.5 Tools and Simulation Environment

Our simulation was done using the MATLAB R2022a software, which provides a complete framework of reinforcement learning to train agents in the predefined environment using a reinforcement learning toolbox and a wide-ranging deep learning framework for training and constructing neural networks using a deep learning toolbox. In the present study, we implement a designed deep neural network in the RL toolbox and generate a DDPG agent in the RL toolbox to generate our actor-critic networks and for training problems. To update the weights of the networks, we also utilized optimization algorithms, like the Adam optimizer, through training. Also, within a tailored path-following environment constructed utilizing Simulink, the agent was trained, which allowed us to evaluate the performance of the agent under diverse scenarios and simulate a range of conditions. Figure 5 denotes the training process of the DDPG agent for 1000 episodes for the path-following task in MATLAB software.



(a) (b)

Figure 5. DDPG-optimized agent training process: (a) for 1000 episodes; (b) for 30 episodes

Also, Figure 6 represents the DRL-based controller framework for path-following tasks in the Simulink environment.

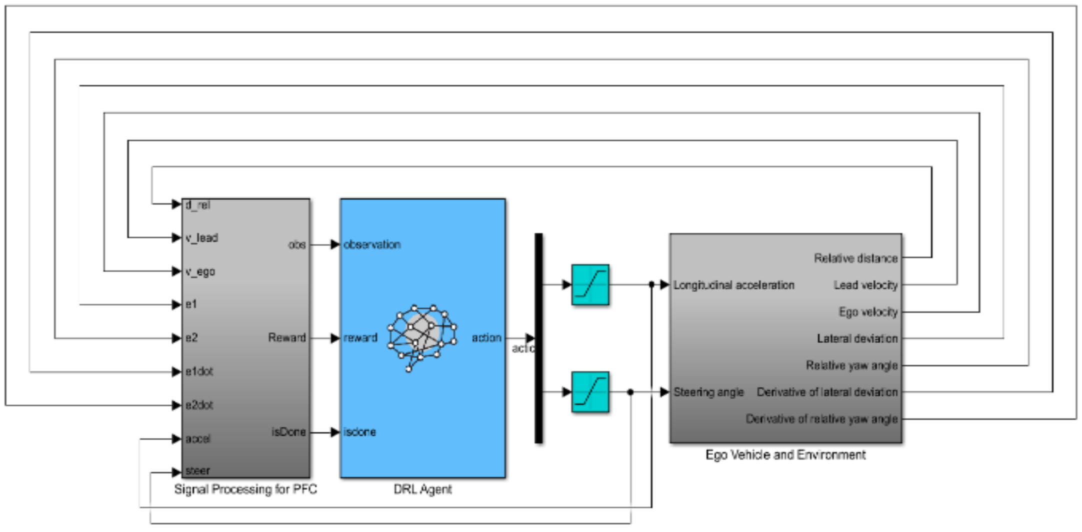


Figure 6. DRL-based control framework for AVs

5. RESULTS AND EVALUATION

In the current section, the DDPG method results for the path-following task are presented, and then the performance of the DDPG-optimized algorithm is compared and evaluated with the DDPG algorithm. Also, Figure 7 demonstrates the path-following and traffic environment in this work. According to Figure 8, the steering angle of the ego vehicle in the DDPG-optimized algorithm is smoother than the DDPG algorithm for the path-following task. The DDPG-optimized algorithm provides more stable control of the autonomous vehicle's steering angle. The optimization techniques employed in the DDPG-optimized agent enhance the learning process, leading to smoother steering angles. These techniques include changes to the network architecture and hyperparameter tuning, all of which contribute to improved convergence and stability during training. Moreover, the optimization process in the DDPG-optimized agent strikes a better balance between exploration and exploitation. By fine-tuning the learning rate, momentum, or other optimization parameters, the agent can effectively explore the action space while exploiting learned policies, resulting in smoother and more refined steering commands. The optimization process involves fine-tuning the policy parameters of the DDPG-optimized agent to prioritize smoother steering behavior. By adjusting the reward function to penalize sharp steering maneuvers, the agent can learn to generate smoother trajectories while still achieving the desired control objectives.

Consequently, the DDPG-optimized algorithm is more reliable than the DDPG algorithm for driving. Also, for lateral maneuvers, the DDPG-optimized agent is more familiar with the driving traffic environment than the DDPG agent.

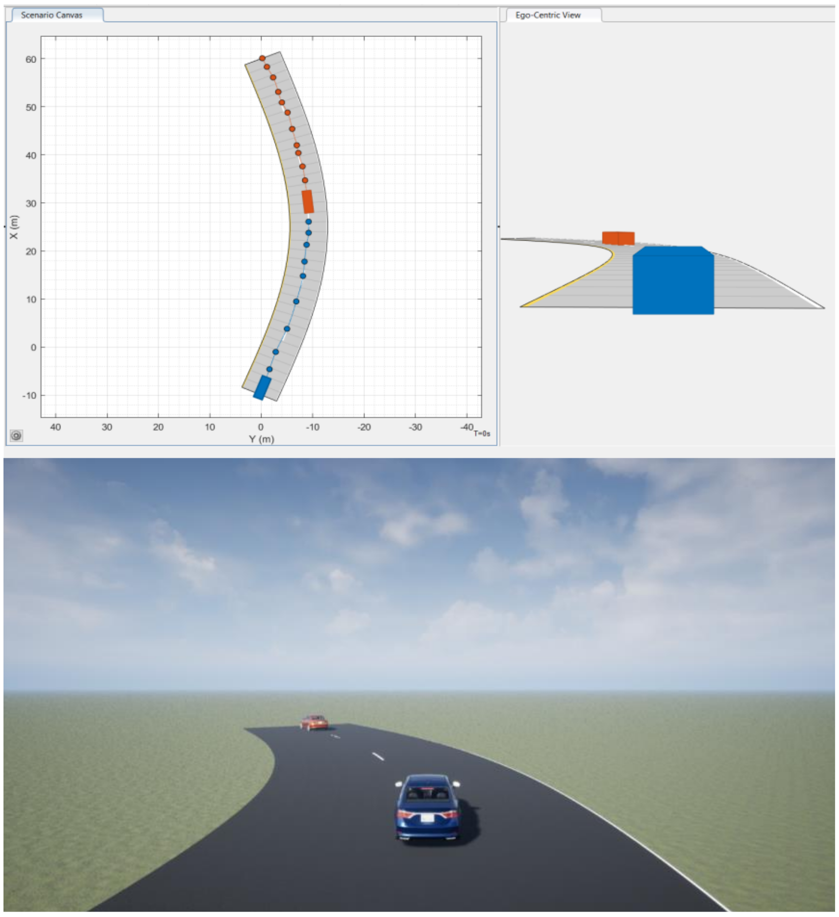


Figure 7. Path-following and traffic environment

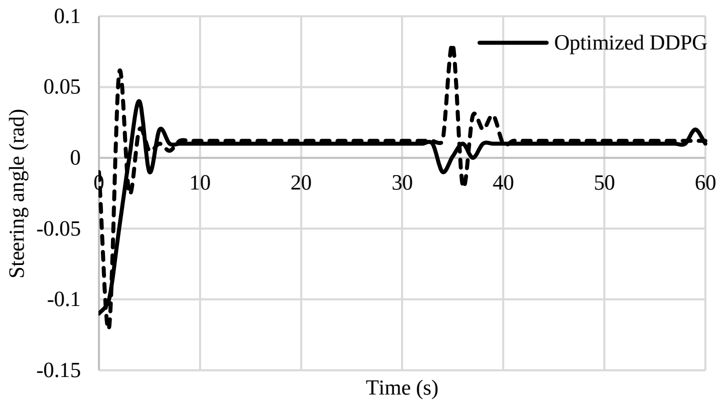


Figure 8. Steering angle- DDPG-optimized and DDPG algorithms

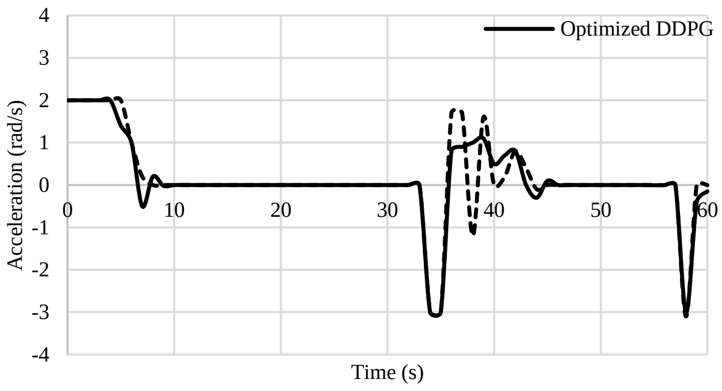


Figure 9. Acceleration- DDPG-optimized and DDPG algorithms

According to Figure 9, based on the acceleration scheme, the DDPG-optimized algorithm is more comfortable and safer than the DDPG algorithm for longitudinal maneuvers. The optimization techniques applied to the DDPG-optimized agent refine the learned control policies to prioritize smoother acceleration profiles. By fine-tuning the neural network architecture and adjusting hyperparameters, the agent can learn to produce more gradual and consistent acceleration commands, resulting in smoother vehicle motion. Furthermore, the optimization process enhances the robustness of the DDPG-optimized agent to environmental disturbances and uncertainties, resulting in smoother acceleration behavior. Through techniques such as regularization or robust optimization, the agent can learn control policies that are more resilient to external factors, leading to smoother vehicle dynamics. Also, smoother acceleration profiles observed in the DDPG-optimized agent contribute to enhanced safety and passenger comfort. By reducing abrupt changes in vehicle velocity, the agent can provide a more comfortable ride experience while minimizing the risk of sudden accelerations that could compromise safety.

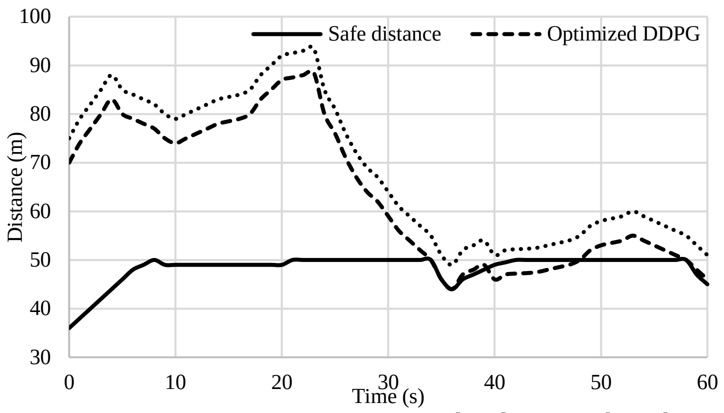


Figure 10. Distance- DDPG-optimized and DDPG algorithms

According to Figure 10, the ego vehicle in the DDPG-optimized algorithm follows the safe distance in the best way, while the DDPG algorithm does not follow the safe distance ideally. Consequently, the DDPG-optimized algorithm controls the ego vehicle in a safe path like an intelligent human driver. Since the optimization techniques enhance the agent's ability to adapt to uncertain or dynamic environments by minimizing the effects of noise or variability in sensor inputs, the agent can learn to adjust its control actions more accurately to maintain a safe distance under varying conditions. Moreover, the optimization techniques used in the DDPG-optimized agent accelerate learning and improve convergence toward safer driving behavior. By addressing issues such as slow convergence, the agent can learn more efficiently and consistently maintain a safe distance from obstacles.

In the present study, for the ego car, the target speed is definite as follows: the ego vehicle follows the lower value between the lead car's velocity and the velocity set by the driver, provided that the relative distance is within the safe threshold. Consequently, the ego car ensures a certain separation from the lead vehicle. The ego car matches the velocity set by the driver when the relative distance exceeds the safe distance. Based on Figure 11, the ego vehicle in the DDPG-optimized algorithm tracks the lead vehicle velocity better than the DDPG algorithm by considering the set-point velocity (30 m/s). According to Figure 2, the lateral deviation is defined as the deviation between the C.G. of the autonomous vehicle and the lane centerline. Therefore, the lateral deviation for the DDPG-optimized and DDPG algorithms is demonstrated in Figure 12.

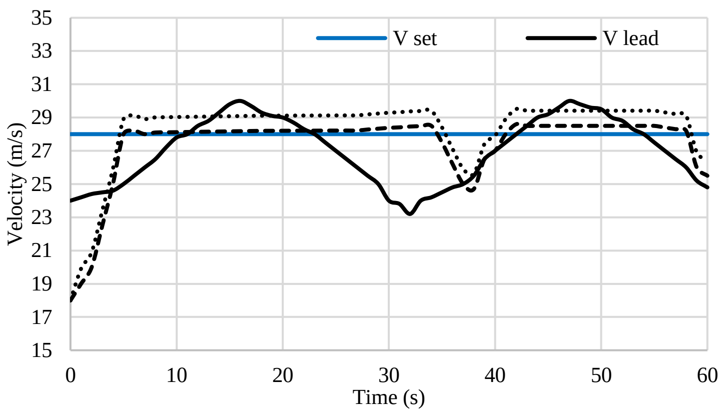


Figure 11. Velocity- DDPG-optimized and DDPG algorithms

According to Figure 12, the lateral deviation in the DDPG-optimized algorithm is less than the DDPG algorithm, hence, the DDPG-optimized agent does the path-following task better than the DDPG agent, and the DDPG-optimized algorithm can control and handle autonomous vehicles for both longitudinal and lateral maneuvers. The optimization process explicitly incorporates safety constraints into the learning objectives of the DDPG-optimized agent. By penalizing deviations from the desired path or rewarding behaviors that prioritize trajectory tracking, the agent can learn to prioritize safety while accomplishing its control objectives, resulting in reduced lateral deviation. Also, the DDPG-optimized agent exhibits improved adaptation to changes in the environment or road conditions. Through continuous learning and policy refinement, the agent can dynamically adjust its steering behavior to compensate for variations in road geometry, surface conditions, and traffic patterns, resulting in reduced lateral deviation.

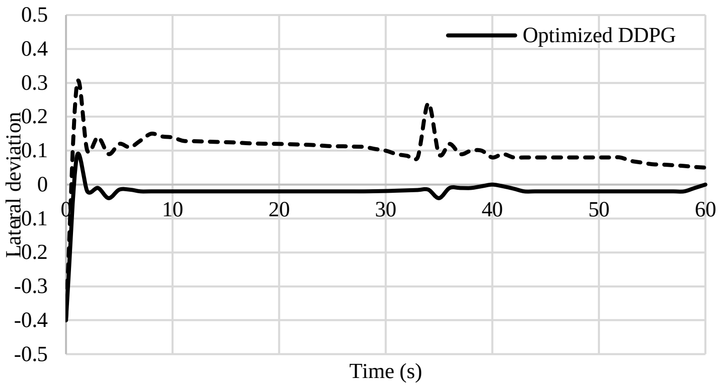


Figure 12. Lateral deviation- DDPG-optimized and DDPG algorithms

The DDPG-optimized agent exhibits smoother steering angles and acceleration profiles, maintains a safer distance, and demonstrates reduced lateral deviation compared to the standard DDPG agent, suggesting a significant improvement in the performance and robustness of the autonomous driving system. The optimization techniques applied to the DDPG algorithm have effectively enhanced various aspects of the agent's behavior, leading to smoother and more precise control actions, better adaptation to environmental uncertainties, and prioritization of safety considerations. By fine-tuning the neural network architecture, adjusting optimization parameters, and integrating safety constraints into the learning objectives, the DDPG-optimized agent demonstrates superior performance in navigating complex and dynamic environments while ensuring safe and comfortable driving experiences. The results highlight the importance of optimization techniques in enhancing the effectiveness and reliability of reinforcement learning-based autonomous driving systems. The DDPG-optimized agent's improved control behavior underscores the potential for optimization strategies to address key challenges in autonomous vehicle control, ultimately contributing to the development of safer, more efficient, and more trustworthy autonomous driving technologies.

6. CONCLUSIONS

In this study, the DRL-based controller has been used to target the path following an autonomous ground vehicle. To accomplish this task, the DDPG-optimized agent was trained and given rewards that were associated with the lateral deviation, the velocity error, the steering angle, and the acceleration. Our agent emphasizes steering angle and acceleration control together to achieve the goal of path-following. While this reduces the action dimension, it also decreases the exploration space. The DDPG-optimized agent successfully confirmed its ability to follow a distance and lead vehicle velocity. Also, the DDPG-optimized agent effectively demonstrated its ability to decrease lateral deviation, and the DDPG-optimized algorithm had a smooth steering angle through driving. Further, it is found to be as effective for tracking trajectory as a classical controller.

Moving forward, the current DRL-based control framework will be improved to include collision and obstacle avoidance, allowing for the optimization of multiple objectives. This will cause the creation of a policy that is both computationally effective and allows implementation in real-time. Multi-objective control configurations will be experimentally tested to gain a better understanding of classical and modern controller’s subtleties. Future works will also explore the efficiency of the controllers under other environmental disturbances like complex traffic environments.

ACKNOWLEDGEMENT

Authors would like to express their gratitude to Faculty of Mechanical Engineering, K.N. Toosi University of Technology, Tehran, Iran for the support and facilities. This study was not supported by any grants from funding bodies in the public, private, or not-for-profit sectors.

CONFLICTS OF INTEREST

We have no conflicts of interest to disclose.

AUTHOR CONTRIBUTIONS

A. Rizehvandi: Conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, draft preparation, review and editing.

S. Azadi: Conceptualization, methodology, validation, resources, review and editing.

REFERENCES

[1] Global Status Report on Road Safety 2018; Technical Report; World Health Organization: Geneva, Switzerland, 2018.

[2] D.J. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations," *Transportation Research Part A: Policy and Practice*, vol. 77, pp. 167–181, 2015.

[3] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey, Technical Report, National Highway Traffic Safety Administration: Washington, DC, USA, 2018.

[4] W.D. Montgomery, R. Mudge, E.L. Groshen, S. Helper, P. Mac Duffie, and C. Carson, "America's workforce and the self-driving future: Realizing productivity gains and spurring economic growth," Available online: (accessed on 8 November 2022).

[5] S. Thrun, W. Burgard, and D. Fox, "Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)," The MIT Press: Cambridge, MA, USA, 2005.

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

[7] Autonomous Vehicles Testing with a Driver, California Department of Motor Vehicles: Sacramento, CA, USA, 2022.

[8] Autonomous Vehicles Testing with a Driver, California Department of Motor Vehicles: Sacramento, CA, USA, 2021.

[9] B. Paden, M. Čáp, S.Z. Yong, D. Yershov and E. Frazzoli "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[10] T. Faulwasser, B. Kern and R. Findeisen, "Model predictive path-following for constrained nonlinear systems," In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) Held Jointly with 2009 28th Chinese Control Conference*, Shanghai, China, pp. 8642–8647, 2009.

[11] E. Yurtsever, J. Lambert, A. Carballo and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020.

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

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

[14] D. Zhang, N.L. Azad, S. Fischmeister and S. Marksteiner, "Zeroth-order optimization attacks on deep reinforcement learning-based lane changing algorithms for autonomous vehicles," In *Proceedings of the International Conference on Informatics in Control, Automation and Robotics*, pp. 665-673, 2023.

[15] O. Amidi and C.E. Thorpe, "Integrated mobile robot control," In *Proceedings Society of Photo-Optical Instrumentation Engineers (SPIE) 1388, Mobile Robots V*, Boston, MA, USA, vol. 1388, pp. 504–523, 1991.

[16] N.H. Amer, H. Zamzuri, K. Hudha and Z.A. Kadir, "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*, vol. 86, pp. 225–254, 2017.

[17] C. Samson, "Path following and time-varying feedback stabilization of a wheeled mobile robot," In *Proceedings of the Second International Conference Control Automation, Robotics and Vision*, Singapore, 1992, vol. 3, pp. 1-7, 1992.

[18] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, et al., "Stanley: The robot that won the DARPA Grand Challenge," Journal of field Robotics, vol. 23, pp. 661–692, 2006.

[19] W. Zhao, J.P. Queralta and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," In *Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence*, Canberra, Australia, pp. 737–744, 2020.

[20] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness and M.G. Bellemare, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[23] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, et al., "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015

[24] X. Cheng, S. Zhang, S. Cheng, Q. Xia and J. Zhang, "Path-following and obstacle avoidance control of nonholonomic wheeled mobile robot based on deep reinforcement learning," *Applied Sciences*, vol. 12, p. 6874, 2022.

[25] Y. Zheng, J. Tao, Q. Sun, X. Zeng, H. Sun and M. Sun, "DDPG-based active disturbance rejection 3D path-following control for powered parafoil under wind disturbances," *Nonlinear Dynamic*, vol. 111, pp. 11205–11221, 2023.

[26] R. Ma, Y. Wang, S. Wang, L. Cheng, R. Wang and M. Tan, "Sample-observed soft actor-critic learning for path following of a biomimetic underwater vehicle," *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 3, pp. 1-10, 2023.

[27] A. Owczarkowski, D. Horla, P. Kozierski and T. Sadalla, "Dynamic modeling and simulation of a bicycle stabilized by LQR control," In *21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, Miedzyzdroje, Poland, pp. 907-911, 2016.

[28] M. Mohri, A. Rostamizadeh and Ameet Talwalkar, "Foundations of machine learning," MIT Press, Second Edition, 2018.

[29] R.S. Sutton and A.G. Barto, "Reinforcement learning: An introduction," MIT Press, 2018.

[30] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra and M. Riedmiller, "Deterministic policy gradient algorithms," In *Proceedings of the International Conference on Machine Learning*, PMLR, pp. 387–395, 2014.