computers

MDPI

Article

Design and Simulation-Based Optimization of an Intelligent Autonomous Cruise Control System

Milad Andalibi [1], Alireza Shourangizhaghighi [2], Mojtaba Hajihosseini [1], Seyed Saeed Madani [3], Carlos Ziebert [3] and Jalil Boudjadar [4,*]

[1] Department of Control and Computer Engineering, University of Zagreb Croatia, 1000 Zagreb, Croatia
[2] Department of Mechanical Engineering, Shiraz University of Technology, Shiraz 71557, Iran
[3] Institute of Applied Materials-Applied Materials Physics (IAM-AWP), Karlsruhe Institute of Technology, Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany
[4] Department of Electrical and Computer Engineering, Aarhus University Denmark, 8200 Aarhus, Denmark
* Correspondence: jalil@ece.au.dk

Abstract: Significant progress has recently been made in transportation automation to alleviate human faults in traffic flow. Recent breakthroughs in artificial intelligence have provided justification for replacing human drivers with digital control systems. This paper proposes the design of a self-adaptive real-time cruise control system to enable path-following control of autonomous ground vehicles so that a self-driving car can drive along a road while following a lead vehicle. To achieve the cooperative objectives, we use a multi-agent deep reinforcement learning (MADRL) technique, including one agent to control the acceleration and another agent to operate the steering control. Since the steering of an autonomous automobile could be adjusted by a stepper motor, a well-known DQN agent is considered to provide the discrete angle values for the closed-loop lateral control. We performed a simulation-based analysis to evaluate the efficacy of the proposed MADRL path following control for autonomous vehicles (AVs). Moreover, we carried out a thorough comparison with two state-of-the-art controllers to examine the accuracy and effectiveness of our proposed control system.

Keywords: autonomous vehicles; cruise control; multi-agent deep reinforcement learning; path following control; artificial intelligence

Citation: Andalibi, M.; Shourangizhaghighi, A.; Hajihosseini, M.; Madani, S.S.; Ziebert, C.; Boudjadar, J. Design and Simulation-Based Optimization of an Intelligent Autonomous Cruise Control System. *Computers* **2023**, *12*, 84. https://doi.org/10.3390/computers12040084

Academic Editor: Paolo Bellavista

Received: 28 February 2023
Revised: 17 April 2023
Accepted: 18 April 2023
Published: 20 April 2023

1. Introduction

With the rapid technological advances, autonomous vehicles have received extensive attention in the past decade [1,2]. Increasing safety by computers has led to considerable benefits, including omitting human errors in critical circumstances, improving occupants' comfort, reducing traffic problems, and reducing environmental impacts, which are among the main impetuses for the automation of driving. Any autonomous driving system consists of several perception level tasks, which must be considered for the design of such a system. Autonomous driving tasks are normally divided into three categories, namely navigation, guidance, and stabilization [3]. Although autonomous vehicles have the potential to revolutionize the transportation industry, there are significant risks associated with their use that must be taken into account, such as cybersecurity concerns, technical failures, and ethical implications. One of the major challenges in self-driving cars is the path following control in which the vehicle keeps the cruise velocity and a safe distance while following another vehicle simultaneously.

Different practical control methodologies have been studied to deliver the capability to keep the cruise velocity and safe distance, such as active steering [3], differential braking [4], integrated chassis control [5] and torque vectoring [6]. In particular, the driver-assist system, which has been extended in [7], enables better lane-keeping and tracking control. In [8], the authors presented a smooth route control for autonomous transport satisfying both

the initial and final circumstances, where the restriction conditions are implemented as a parameterized 6th-order polynomial model [9]. In such a study, the tracking control module along with the model predictive control technique were used. In [10], a developed Kalman filter and linear time-varying model predictive control scheme are applied to predict the future trajectory of an autonomous vehicle (AV), determine the optimal path, and optimize control [9]. A common denominating challenge for the use of machine learning in designing autonomous control applications is the dependency of the decision making no training data, where experiencing learning data that is completely different from training data may lead to inconsistent decisions.

Designing control systems to regulate both throttle and brake is the key part of adaptive cruise control since it ensures the vehicle will keep the speed response of the antecedent vehicle, and consequently retaining a safe inter-vehicle distance under the limitations of driving [11]. In [12], an adaptive cruise control structure considering a variety of funnel controllers are introduced. In [13], the authors presented a least-violating control in the application of cruise control to regulate the system with the properties of safety, uniform reachability, and uniform attractivity. In [14], a varying prediction zone nonlinear model predictive control (NMPC) using a continuation/generalized minimal residual (C/GMRES) optimizer with a dead zone penalty function to guarantee the smoothness and meet the inequalities is designed in the path-following control application. A sliding mode control for speed control in AVs has been applied in [15]. Robust H-infinity control methods are investigated in [16]. A super-twisting sliding mode is presented in [17] based on Lyapunov stability proofed by the backstepping method in the application field of AVs path-following control. These methods demonstrated acceptable outcomes based on analytic control; they require not only complex design but also, are unable to consider unknown uncertainties due to the intense dynamic structure of automobiles. Thus, focusing on the complete mathematical model is impractical. To address this issue and reach high accuracy, reinforcement learning techniques can be adopted to look for optimal controllers for systems with undetermined or highly nonlinear and stochastic dynamics.

Deep reinforcement learning (DRL), which is well-known as an efficient learning framework, is able to train an agent to impressively find the right control command signal by interacting with the system in order to optimize the reward function [18]. Recent significant advances in DRL have prompted the application of this technique in various fields of engineering. DRL algorithms are divided into three categories: 1-continuous; 2-discrete; 3-continuous or discrete. Based on the environment (or system) type, the appropriate algorithm needs to be selected. In the field of transportation, efforts have been devoted to using RL in the AVs path-following control. Unmanned vehicle track control can be divided into three environments: 1-land, 2-water and underwater, and 3-aerial. In [19], a Deep Deterministic Policy Gradient (DDPG) agent was adopted to find a suitable vessel steering policy in the presence of the ocean current. In [20], a neural network (NN)-based RL algorithm was adopted to predict the unknown disturbances, parameter uncertainties, and nonlinearities of autonomous underwater vehicles in trajectory tracking. To obtain adaptive control in AUVs, the study in [21] relies on an actor-critic RL NN-based agent. The authors of [22] presented a strategy for AUV route following by combining the benefits of DRL with interactive RL, which receives a reward from both the environment and the human operator at the same time.

In aerial field path-following, Rubi et al. [23] implemented Q-learning agent for an airship to mitigate the curse of dimensionality problem. In [24], a DDPG agent for a quadrotor was investigated, and its sustainability and performance in the path following in the presence of wind turbulence and other disturbances were probed.

Gabriel et al. proposed a model-based RL (MBRL) for high-speed autonomous driving path tracking [25]. They combined Failure Prediction and Intervention Module (FIM) with MBRL to achieve high performance in a self-driving system. Charles et al. [26] proposed a scheme in which they succeeded in obtaining high-performance longitudinal control by an

NN-based policy gradient algorithm. Wang et al. [27] applied the reinforcement learning approach to learn the automated lane change behavior in an interactive driving environment.

In the aforementioned literature, most of the studies consider a single control feature of the autonomous driving system, for example, either steering control or speed control, or only the issue of constant speed tracking has been addressed. In this paper, we propose a DRL-based solution to control a constellation of driving system features simultaneously, namely speed control, steering control, and safety. Following the acceleration and steering control, in order to control the speed and distance while following the lead vehicle, two DRL agents are considered. The first agent controls the steering wheel of the car, while the second agent manages the acceleration according to speed and distance. Simulation-based experiments are conducted to test the accuracy, efficiency, and response time of the proposed DRL-based control solution. Although the algorithms we propose are formed by the integration of three agents supervising the key features of conventional vehicles, for the same level of complexity, each agent is considered a black box, where, for example, the speed to be applied is computed but the details related to how much fuel to inject and acceleration are omitted as these parameters are dependent on the actual state of the AV and environment.

The rest of the paper Is organized as follows. The dynamic Model of the ground autonomous vehicle is presented in Section 2. The design of the MADRL controller is described in Section 3. Steering, acceleration, and a DQN agent are provided in Section 4, Section 5, and Section 6, respectively. Simulation results are discussed in Section 7. Finally, Section 8 concludes the paper.

2. Two Degree of Freedom Dynamic Model of Ground Autonomous Vehicles

It is important to have a brief overview of the mathematical model to pave the ground for connecting the control agents later. Hence, a schematic representation of the proposed model is depicted in Figure 1. Note that ψ is the yaw angle, $\dot{\psi}$ shows the yaw rate, v_x and v_y determines the velocity according to the vehicle coordination, v_{lf} and v_{cf} are the longitudinal and lateral velocity of the lead vehicle wheel, v_f is their result vector, δ is the front wheel angle, and F_l and F_c are longitudinal and lateral wheel forces. Thus, based on Newton's second law of motion, dynamic equations governing the system are introduced as in [28]:

$$\begin{aligned} m\dot{v}_x &= m\dot{v}_y\dot{\psi} + 2F_{xf} + 2F_{xr} \\ m\dot{v}_y &= -m\dot{v}_x\dot{\psi} + 2F_{yf} + 2F_{yr} \\ I\ddot{\psi} &= 2l_fF_{yf} - 2l_rF_{yr} \end{aligned} \tag{1}$$

where m and I are the mass and inertia, F_x is the lateral force, and F_y is the lateral force at the center of gravity (CoG) of the vehicle. The yaw rate can be calculated as follows:

$$\dot{\psi} = \frac{v_x}{l_f + l_r} tan(\delta) \tag{2}$$

where $\boldsymbol{l_f}$ and $\boldsymbol{l_r}$ are the distances from the CoG. In addition, the position states can be obtained as:

$$\begin{aligned} \dot{X} &= v_x cos(\psi) - v_y sin(\psi) \\ \dot{Y} &= v_x sin(\psi) + v_y cos(\psi) \end{aligned} \tag{3}$$

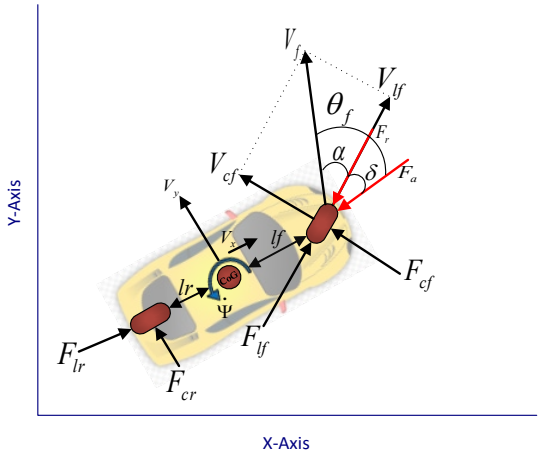

Figure 1. A 2-DoF schematic representation of the vehicle.

$\boldsymbol{F_x}$ and $\boldsymbol{F_y}$ which are the acting forces on the CoG, can be calculated by:

$$\begin{aligned} F_x &= F_l cos(\delta) - F_c sin(\delta) \\ F_y &= F_l sin(\delta) - F_c cos \end{aligned} \tag{4}$$

the longitudinal $\boldsymbol{F_l}$ and lateral, $\boldsymbol{F_c}$ forces are shown as follows :

$$\begin{aligned} F_l &= f(\alpha, \mu, s, F_z) \\ F_c &= f(\alpha, \mu, s, F_z) \end{aligned} \tag{5}$$

As shown in Figure 1, $\boldsymbol{\alpha}$ is the angle between the wheel velocity vector and the wheel direction, and $\boldsymbol{\mu}$ is the road friction coefficient. The difference between ground point velocity and the rotational velocity (slip ratio) is $\boldsymbol{s}$, and $\boldsymbol{F_z}$, which determines the vertical load action on the wheels. Under the assumption of having small $\boldsymbol{\mu}$ values, the lateral tire forces can be obtained as:

$$\begin{aligned} F_{lf} &= C_f \alpha_f \\ F_{lr} &= C_r \alpha_r \end{aligned} \tag{6}$$

where $\boldsymbol{C_f}$ and $\boldsymbol{C_r}$ are the tire stiffness parameters and

$$\begin{aligned} \alpha_f &= \delta - \theta_f \\ \alpha_r &= -\theta_r \end{aligned} \tag{7}$$

where $\boldsymbol{\theta_f} = \boldsymbol{arctan}\left[\frac{v_y + l_f \dot{\psi}}{v_x}\right]$ and $\boldsymbol{\theta_r} = \boldsymbol{arctan}\left[\frac{v_y + l_r \dot{\psi}}{v_x}\right]$.

Assuming that the vehicle is traveling on flat ground, it is not affected by gravitational force but by air drag, $\boldsymbol{F_a} = \frac{1}{2}\boldsymbol{C_D A_a \rho_a (v + v_{wind})^2}$ and rolling resistance, $\boldsymbol{F_r} = \boldsymbol{D_r mgcos(\alpha)}$. $\boldsymbol{C_D}$ is the air drag coefficient, $\boldsymbol{A_a}$ is the maximum cross-sectional area of the vehicle, $\boldsymbol{\rho_a}$ is the air density, $\boldsymbol{v_{wind}}$ is wind velocity, $\boldsymbol{D_r}$ is the roll resistance coefficient, $\boldsymbol{m}$ the vehicle mass, and $\boldsymbol{g}$ the gravitational acceleration. For convenience, $\boldsymbol{\alpha}$ can be ignored because $\boldsymbol{v_{wind}}$ is very small in comparison with the vehicle velocity.

In conclusion, the equivalent dynamic state of the system can be expressed as follows:

$$\begin{aligned} \dot{X} &= v_x cos(\psi) - v_y sin(\psi) \\ \dot{Y} &= v_x sin(\psi) - v_y cos(\psi) \\ \dot{\psi} &= \frac{v_x}{l_f + l_r} tg(\delta) \\ m\dot{v_x} &= F_x + m v_y \dot{\psi} - 2F_{cf} sin(\delta) - F_a - F_r \\ m\dot{v_y} &= -m v_x \dot{\psi} - 2(F_{cf} cos(\delta) + F_{cr}) \\ I\ddot{\psi} &= 2\Big(l_f F_{cf} cos(\delta) - l_r F_{cr}\Big) \end{aligned} \tag{8}$$

It should be noted that δ and F_x are the control input of the vehicle.

3. Design of MADRL Controller

To control the AV following a lead car, velocity and distance stabilization in tracking performance of conventional methodologies like Fuzzy Logic, PID, and model predictive controller (MPC) are restricted from the following regulatory aspects:

1. Since the time intervals in the simulation of vehicles are in microseconds, the computational time for designing the model-based schemes is very definitive in real-time.
2. Due to the destabilization properties and high nonlinear characteristics of AVs, achieving cruise velocity and tracking control simultaneously is one of the main objectives in this field. Thus, further efforts should be considered to not only mitigate the nonlinearities effects on optimal performance control, but also ensure the stability requirements.

As it can be seen in Figure 2, the main control approach is to keep a safe distance and a set velocity while tracking the lead vehicle. In this case, the ego car that is tracking the lead car must follow the following rules: 1-When the distance between the ego car and the lead car is greater than a safe distance, the ego car must speed up to track the set velocity; 2-Otherwise, the acceleration must be reduced to maintain the safe distance.

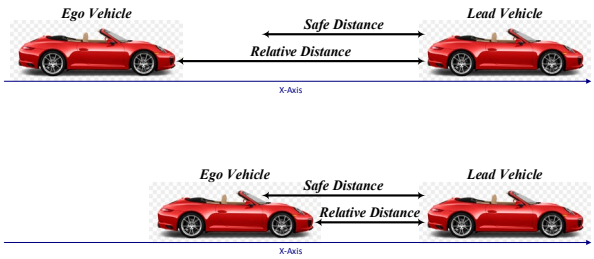

Figure 2. Path tracking from cruise control point of view.

Due to the deficiency in the existing control methodologies, a MADRL-based scheme is proposed in this work in order to find a promising solution for the aforementioned challenges. Moreover, in the Multi Agent RL (MARL) algorithms, agents learn their own distinctive duties, which provides a helpful perspective on control. First, the nonlinear model of the system is employed so that sensors can be considered to identify system states and implement the suitable algorithm. Then, according to the system control inputs, the continuous agent is used for acceleration, while the discrete agent is considered for steering angle. As the acceleration force of the cars is continuous, an actor-critic model-free policy-based agent called Twin Delayed DDPG is used and compared to a DQN agent using a discrete view of acceleration. Yaw angle is set by a stepper motor, which has to be looked at completely in discrete values, and a DQN is utilized for this process.

3.1. Markov Decision Process

In the RL structure, a task can be determined by a Markov Decision Process (MDP) specified by a quintuple $\{S,\ A,\ r,\ p,\ \gamma\}$, where $S\,\epsilon\,\mathbb{R}^n$ indicates the state space, $A\,\epsilon\,\mathbb{R}^m$ shows the action space, $r:\ S\times A\ \rightarrow\ \mathbb{R}$ indicates the reward function, $p:\ S\times A\times S\ \rightarrow\ [0,1]$ is the transition function, which represents the probability of transiting to a new state s_{t+1}, emitting a reward r under the execution of action a_t on the state s_t, and $\gamma\,\epsilon\,[0\ 1]$ indicates the discount factor. With an initial state s_t, the RL is aimed to maximize the obtained rewards $\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r_t\ \right]$.

3.2. The Twin Delayed DDPG (TD3)

A twin-delayed deep deterministic policy gradient agent is an actor-critic RL agent that calculates an optimal policy so that it optimizes the long-term reward and acts on the environment continuously. Note that the TD3 algorithm is off-policy, model-free, and an online RL technique.

The TD3 algorithm, which is an extension of the DDPG algorithm, addresses DDPG function overestimation via learning two Q-values at a time, and during policy updating, it benefits from the minimum value function. Besides, it adds noise to target actions to explore the environment more effectively. To obtain the estimation of the policy and value functions, TD3 uses the following function approximators: (1) The actor $\mu(S)$ observes the system states (or observations) and S correspondingly acts on the environment in a way that maximizes the long-term reward. (2) To increase the stability of the optimization process, the agent $\mu'(S)$ regularly updates the target actor parameters using the received values. (3) State set S and action set A input to the critics, then they give the related expectation of the long-term reward. (4) The previous procedure is periodically done at a determined time to update the target critics. The critics should follow the same structures, but their corresponding targets have to be the same. For better understanding, Figure 3 demonstrates the TD3 algorithm in detail.

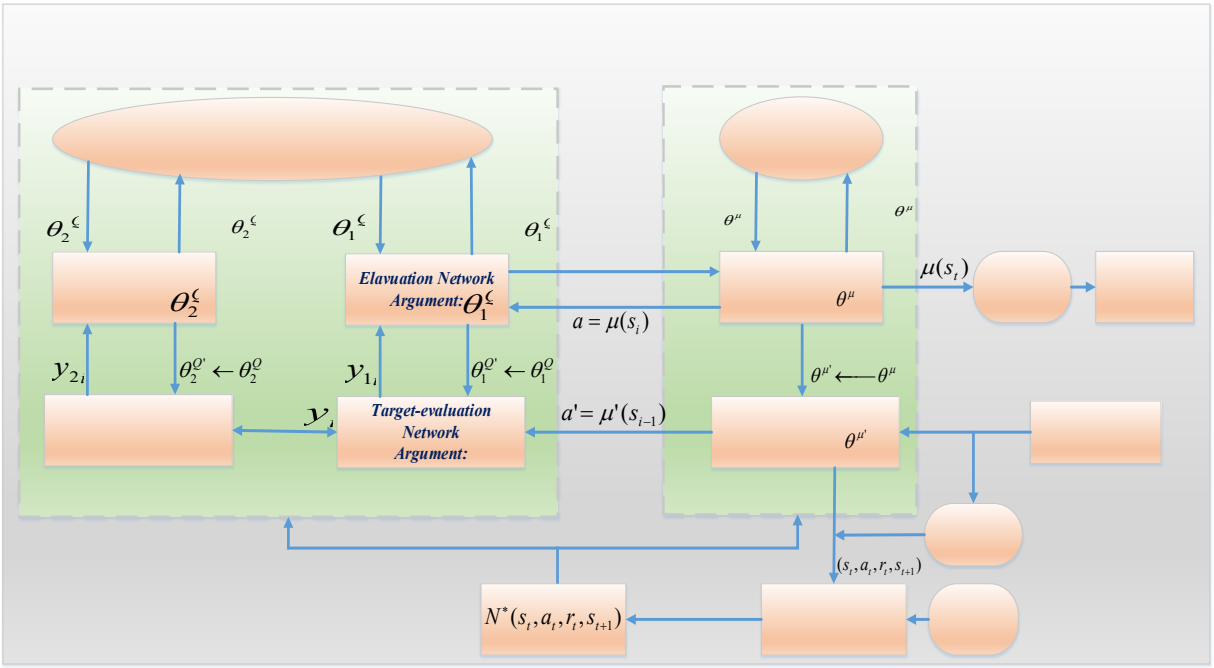

Figure 3. Flowchart of the TD3 algorithm with actor-critic architecture.

4. MADRL AV TD3 Agent for Acceleration Control

Based on the knowledge of the AVs sensors outputs, the exact observations and reward function could be defined. Figure 4 shows that the observation and reward function consist of four parameters: (1) Relative distance (the difference between the lead and ego vehicles positions); (2) Lead vehicle velocity; (3) Ego vehicle velocity; and (4) Last exerted acceleration. It is important to explicitly address a series of calculations before dealing with observations. To do so, the safe distance between the lead car and the ego car, D_{safe}, is defined as:

$$D_{safe} = D_{default} + T_{gap} V_{ego} \tag{9}$$

where $D_{default}$ is the pause default distance in meters, T_{gap} shows the time gap between the vehicles in seconds, and V_{ego} defines the ego vehicle velocity. The error distance is

$$D_{error} = D_{safe} - D_{rel} \tag{10}$$

One should bear in mind that the velocity error is another significant issue, which can be calculated as follows:

$$V_{error} = \begin{cases} \min\{V_{lead}, V_{set}\} - V_{ego} & if \quad \min\{V_{lead}, V_{set}\} \\ V_{set} - V_{ego} & otherwise \end{cases} \tag{11}$$

where V_{lead} is the lead vehicle's velocity, and V_{set} is the velocity at which the ego vehicle is set to drive. Therefore, the observation would be defined as $\{V_{error}, \int V_{error}, V_{lead}\}$.

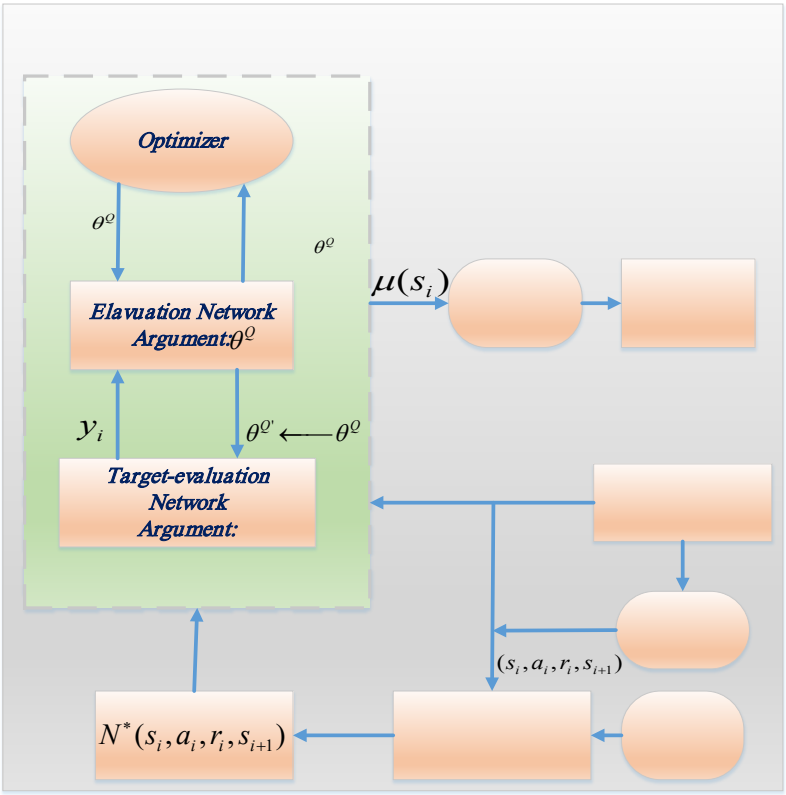

Figure 4. Flowchart of the DQN algorithm.

To compose the reward function, the first step is to calculate the cost function. The cost function consists of the consumed energy, the control action, and the error in velocity with different weights. The cost function is defined as:

$$Cost\ function1 = \sum w_1 V_{error}^2 + w_2 F_x^2 \tag{12}$$

where w_1 and w_2 are the weights of the considered values, and F_x (or a) is the acceleration. The reward function, which is expected to be maximized in the training process, is defined as the minus of the cost function. Throughout this study, the actor-critic TD3 consists of two fully connected hidden layers (HLs) with 50 neurons for both the actor and critic structures. The "rectified linear unit (ReLU)" activation function is applied to all HLs in the network. Readers are directed toward Figure 5 for a schematic configuration representation of the MADRL AV. Additionally, the parameters for the implemented algorithm are given in Table 1.

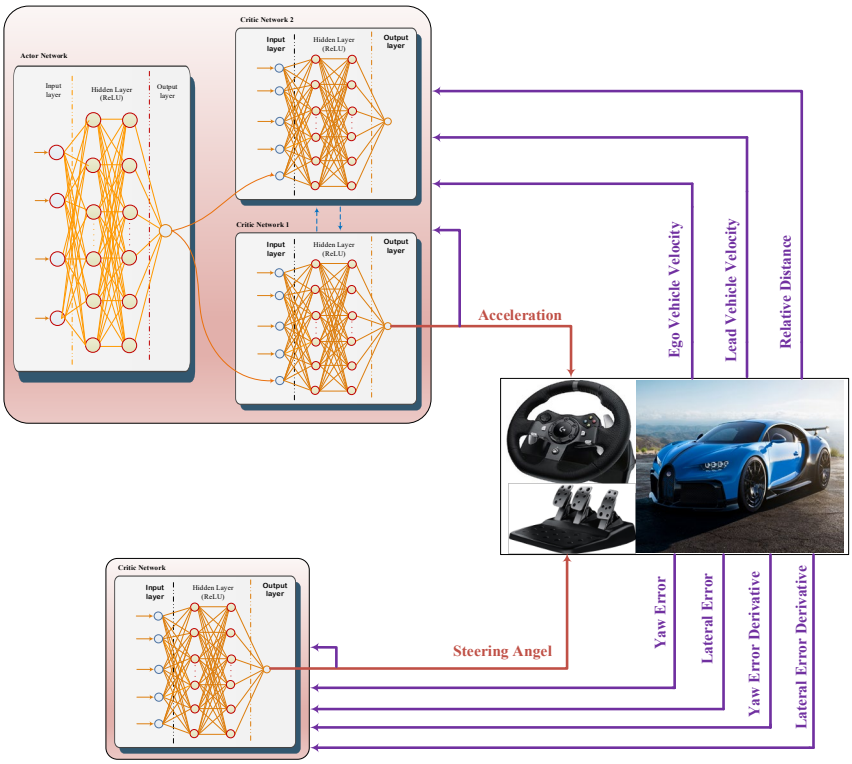

Figure 5. Structure of the implemented MADRL AV.

Table 1. Parameters of the TD3 agent.

Parameter	Value	Parameter	Value
TD3 Training Episode Length	500 ts	Critic2 Learning Rate	1×10^{-3}
Minimum Batch Size	1024	Number of MC cycle	200
Actor Learning Rate	1×10^{-4}	Discount Factor	0.99
Crtitic1 Learning Rate	1×10^{-3}		

5. Deep-Q-Network (DQN)

A DQN agent is an off-policy, value-based RL agent that acts on the environment discretely. Moreover, the DQN algorithm is a model-free, online RL technique. It is challenging in complex state-action spaces to learn from the evaluation of the Q value of both state and action separately. In DRL, several agents' parts, such as policy $\Pi(\mathrm{s}, \mathrm{a})$ or values $q(s, a)$ are given with deep NNs. These NNs parameters are trained to minimize some loss functions through the gradient descent method. In DQN, deep networks and RL estimate values from NNs and the given states S_t. In each step, based on the current state, the agent chooses an action based on the action values $\epsilon - greedily$ and the data $(S_t, A_t, R_{t+1}, \gamma_{t+1}, S_{t+1})$ which have all the preceding data in time t. NNs parameters are then trained using random gradient descent to minimize the following loss function:

$$\left(R_{t+1} + \gamma_{t+1} \underset{a'}{max} q_\theta(S_{t+1}, a') - q_\theta(S_t, A_t)\right)^2 \tag{13}$$

where t is the time step. The cost function gradient for updating θ is done via the back-propagation method. θ defines the target network's parameters and is a copy of the online network over a given time period. Optimization is performed using RMSprop on small, sampled batches from replay memory. The structure of the DQN algorithm is shown in Figure 4.

6. The MADRL AV DQN Agent for Steering Angle Control

Similarly, as defined in the observations and reward function for TD3, for the purpose of having control over the steering angle, some calculations should be done for the DQN agent. As shown in Figure 5, the agent receives some inputs from the environment, including: 1-$\dot{L}_{error}$(Lateral error derivative); 2-$\dot{Y}_{error}$ (Yaw error derivative); 3-L_{error} (Lateral error); 4-, Y_{error} (Yaw error); and 5- θ (Steering angle). Next, the observation set can be calculated as $\left\{ L_{error}\ ,\ \dot{L}_{error},\ \int L_{error},\ Y_{error},\ \dot{Y}_{error},\ \int Y_{error} \right\}$. To obtain the corresponding reward function, first the cost function should be calculated as follows:

$$Cost\ function2 = \sum w'_1 L^2_{error} + w'_2 \theta^2 \tag{14}$$

where w'_1 and w'_2 are weights of the considered values, and θ is the steering angle. The reward function, which is expected to be maximized in the training process, is defined as the minus of the cost function.

In this work, similar to TD3 in part *C*, DQN consists of two fully connected HLs with 50 neurons. For all HLs in the network, the ReLU activation function is used. The configuration is specified in the lower agent of Figure 5, and the parameters for the implemented algorithm are provided in Table 2.

Table 2. Parameters of the DQN agent.

Parameter	Value	Parameter	Value
DQN Training Episode Length	500 ts	Number of MC Cycle	500
Minimum Batch Size	1024	Discount Factor	0.99
Learning Rate	1×10^{-3}		

7. Results and Discussion

In this section, we evaluate the effectiveness of our proposed multi-agent DRL-based control technique and compare it to two of the state-of-the-art control alternatives, namely Holistic Adaptive Multi-Model Predictive Control (HMPC) and Hierarchical Predictive Control (HPC). HMPC [29] has a linear structure, which makes it perform well in real-time. It also has a weight-adaptive mechanism to improve its handling ability and a multi-model adaptive law to account for tire cornering stiffness uncertainties. HMPC benefits from a weight-adaptive structure to control the system in uncertain situation. HPC [30] provides a structure to switch between multiple model predictive controllers in order to decrease the response time. HPC switches between the models at runtime based on a metric called uncontrollable divergence, which reveals the divergence between predicted and true states caused by return time and model mismatch. The relevant parameters of the AV are given in Table 3. In this application, to investigate the reliability and effectiveness of the proposed MADRL controller, the errors of yaw angle, lateral distance error, distance, velocities, and control efforts are investigated. The iterations in which the agents are trained are studied too. Figure 6a plots the training reward against the number of completed epochs. As seen, the training reward improves with every epoch, indicating the algorithm's gradual learning and performance enhancement.

Table 3. Parameters used in the vehicle model.

Parameter	Abbreviation	Value
Vehicle mass	m	1500 kg
Inertia around z-axis	I	1300 mNs^2
Cornering stiffness front wheels	C_f	13,000 N/rad
Cornering stiffness rear wheels	C_r	12,500 N/rad
Distance from front wheels to CoG	l_f	m
Distance from rear wheels to CoG	l_r	m
Cross sectional area	A_a	10.25 m^2
Roll resistance coefficient	D_r	1.5×10^{-3}
Air drag coefficient	C_D	0.6
Air density	ρ_A	1.3 kg/m^3
default spacing between lead and ego cars	$D_{default}$	m
time gap for distance maintaining	t_{gap}	1.5 s
set velocity for ego car	V_{set}	30 m/s

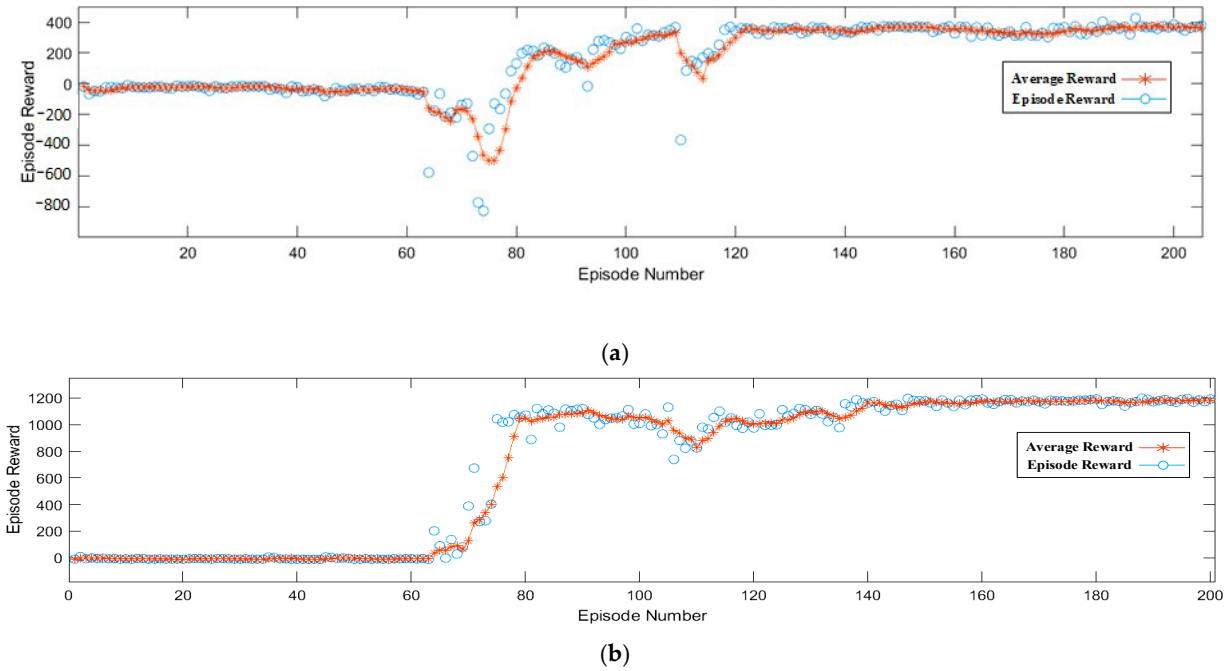

Figure 6. Suggested algorithm training rewards. (**a**) TD3; (**b**) DQN.

To visualize the trend better, Figure 6b also shows the same training progress with an added trendline, which displays a clear improvement in the training reward over time, affirming the algorithm's successful learning and performance enhancement.

As shown in Figure 6a,b, our MADRL based controller outperforms in stabilizing the system under favorable conditions of tracking the lead vehicle at a set velocity and avoiding crossing the safety distance. The progress depicted in Figure 6a,b resulted from the execution of our proposed algorithm on the training data, which was done through multiple epochs. Every epoch entailed iterating through the data set and adjusting the algorithm based on the feedback from the training reward.

The training reward is a metric that reflects the algorithm's performance during training, which is determined by a specific objective function. This function aims to minimize the error between the algorithm's predicted and actual outputs, and it gauges the degree to which the algorithm is learning and enhancing its performance.

To achieve this, as presented in Figure 7, in the early stages of driving to maintain a safe distance, the accelerations fluctuate sharply, and the range of acceleration decreases over time. In the MADRL algorithm, to achieve the set speed, the controller starts moving at

a relatively high acceleration and reduces its acceleration smoothly over time, but in the two HPC and HMPC algorithms, the vehicle accelerates after about 7 s. In the velocity diagram, the MADRL method clearly proves its superiority over the two HPC and HMPC algorithms. It can be seen in Figure 8 that the two alternative algorithms can track a velocity of 30 m/s with overshoots along with some fluctuations. However, the trained neural network with MADRL controls the velocity in the way that it benefits both smooth tracking behavior and fast settling time. This optimized behavior of the system with MADRL in steering angle changes is also shown in Figure 9, while HPC and HMPC methods show oscillating behavior that is not practical and not useful as they definitely cause destructive damages to passengers. In all three compared algorithms, the need to maintain the distances between the two vehicles is another significant issue that should be investigated. Figure 10a–c represents the relative and safe distances of the ego and lead vehicles when MADRL is applied. By adjusting the speed with acceleration, the ego vehicle keeps its distance from the lead vehicle in a way not to cross the safe distance, while sustaining a relative distance to be able to follow it properly. The HPC and HMPC algorithms are shown in Figure 10b,c and they follow the same principle formulated in their constraints in the optimal control problem. As it is obvious in Figure 10a, compared to Figure 10b,c, the safe distance diagram benefits from a smooth behavior, while the HPC and HMPC algorithms experience some oscillating behaviors, for example, at times of 750 and 1080 s for HPC and at times of 420, 760, and 1080 s for HMPC. It can be perceived that with path following and cruise control, all the controllers work in the right way to keep a safe distance and track the trajectory of the leading vehicle. Moreover, the root mean square error (RMSE) is also studied for three algorithms for steering and lateral distance, as depicted in Figure 11. It is evident that the lowest RMSE is obtained using our MADRL controller. As a result, it can be readily seen that the MADRL outperforms the two state-of-the-art algorithms considered earlier.

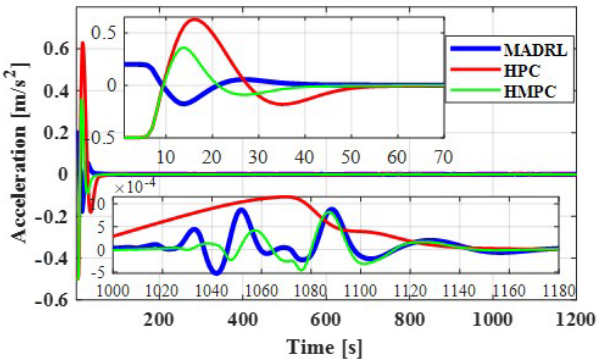

Figure 7. Control effort on acceleration of the ego vehicles with a comparative perspective between the MADRL, HPC, and HMPC algorithms.

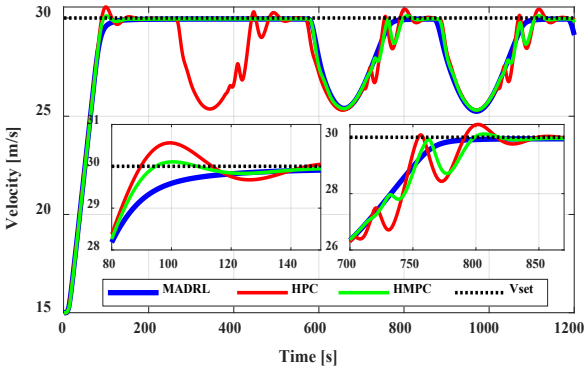

Figure 8. Different ego vehicle velocities in the scenario for the proposed MADRL algorithm and a comparison with HPC and HMPC.

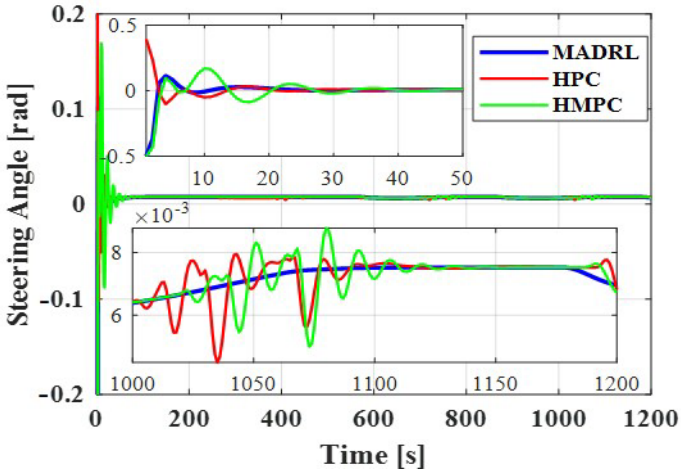

Figure 9. Control effort on steering angle of the ego vehicles with a comparative perspective between the MADRL, HPC, and HMPC algorithms.

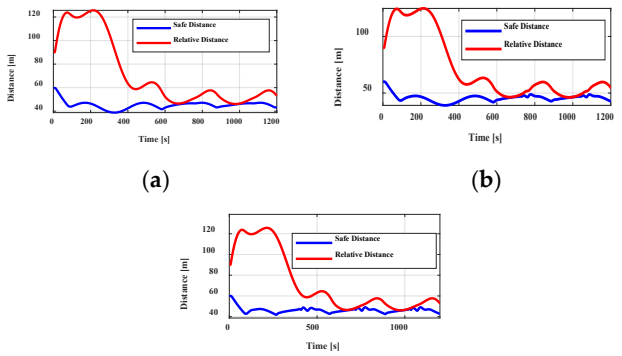

Figure 10. Relative and safe distances between the ego and lead vehicles for MADRL, HPC, and HMPC, respectively, shown in (**a**–**c**).

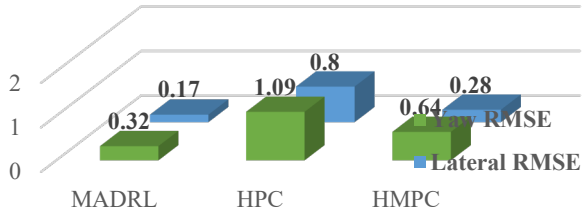

Figure 11. Bar chart comparison of different algorithms RMSE indices.

8. Conclusions

This paper proposes a multi-agent deep reinforcement learning-based method to control both speed and steering (cruise control) of unmanned vehicles using DLR agents, in which the agents learn to select the optimum actions to control steering and acceleration. The proposed method has the potential to enhance the safety and efficiency of autonomous vehicles, particularly in challenging environments due to its reduced computation requirements distributed among agents. The study's findings reveal that the suggested approach surpasses existing state-of-the-art techniques, demonstrating its potential to be applied in real-world situations. To overcome the real-time learning mission, both the DQN and TD3 for the actor and critic sections follow the structure of two hidden layers made up of 50 neurons, with the RELU acting as the activation function. To meet the control requirements, the MADRL technique was used, where one agent is in charge of acceleration

and the other is considered to be steering angle. As a result, the following outcomes were obtained: 1-yaw and lateral errors reached approximately zero in less than 4 s, 2-the ego's velocity reached set point velocity in less than 10 s, while it is intelligent not to pass the safe distance simultaneously, 3-acceleration and steering act in such a way that the smallest amount of energy was acquired. Lastly, the performance of the proposed control method was tested and compared to two state-of-the-art techniques, HPC and HMPC, with the clear outcome that our proposal outperforms the state-of-the-art techniques. A future work would be to consider other challenges and risks related to delays, data loss, and control compromise of AV and propose new mitigation agents to maintain safety.

Author Contributions: Conceptualization, M.A., A.S and M.H.; methodology, M.A., S.S.M. and C.Z and J.B.; software, M.A. and A.S.; validation, A.S., S.S.M., C.Z. and J.B.; formal analysis, M.A., A.S., S.S.M. and C.Z.; investigation, M.A., A.S and C.Z.; resources, all.; data curation, M.A. and M.H.; writing—original draft preparation, all.; writing—review and editing, J.B., S.S.M. and C.Z.; visualization, M.A., J.B.; supervision, C.Z. and J.B.; funding acquisition, J.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

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

References

1. Lv, M.; Peng, Z.; Wang, D.; Han, Q.-L. Event-Triggered Cooperative Path Following of Autonomous Surface Vehicles Over Wireless Network with Experiment Results. *IEEE Trans. Ind. Electron.* **2021**, *69*, 11479–11489. [CrossRef]
2. Jain, R.P.; Aguiar, A.P.; de Sousa, J.B. Cooperative Path Following of Robotic Vehicles Using an Event-Based Control and Communication Strategy. *IEEE Robot. Autom. Lett.* **2018**, *3*, 1941–1948. [CrossRef]
3. Li, W.; Xie, Z.; Wong, P.K.; Mei, X.; Zhao, J. Adaptive-Event-Trigger-Based Fuzzy Nonlinear Lateral Dynamic Control for Autonomous Electric Vehicles Under Insecure Communication Networks. *IEEE Trans. Ind. Electron.* **2020**, *68*, 2447–2459. [CrossRef]
4. Chen, J.; Shuai, Z.; Zhang, H.; Zhao, W. Path Following Control of Autonomous Four-Wheel-Independent-Drive Electric Vehicles via Second-Order Sliding Mode and Nonlinear Disturbance Observer Techniques. *IEEE Trans. Ind. Electron.* **2020**, *68*, 2460–2469. [CrossRef]
5. Zhang, L.; Ding, H.; Guo, K.; Zhang, J.; Pan, W.; Jiang, Z. Cooperative chassis control system of electric vehicles for agility and stability improvements. *IET Intell. Transp. Syst.* **2018**, *13*, 134–140. [CrossRef]
6. Lucchini, A.; Formentin, S.; Corno, M.; Piga, D.; Savaresi, S.M. Torque Vectoring for High-Performance Electric Vehicles: An Efficient MPC Calibration. *IEEE Control Syst. Lett.* **2020**, *4*, 725–730. [CrossRef]
7. Yurtsever, E.; Lambert, J.; Carballo, A.; Takeda, K. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access* **2020**, *8*, 58443–58469. [CrossRef]
8. Kanchwala, H.; Viana, I.B.; Ceccoti, M.; Aouf, N. Model predictive tracking controller for a high fidelity vehicle dynamics model. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019. [CrossRef]
9. Mazzilli, V.; De Pinto, S.; Pascali, L.; Contrino, M.; Bottiglione, F.; Mantriota, G.; Gruber, P.; Sorniotti, A. Integrated chassis control: Classification, analysis and future trends. *Annu. Rev. Control.* **2021**, *54*, 172–205. [CrossRef]
10. Xiang, S.; Gao, H.; Liu, Z.; Gosselin, C. Dynamic transition trajectory planning of three-DOF cable-suspended parallel robots via linear time-varying MPC. *Mech. Mach. Theory* **2020**, *146*, 103715. [CrossRef]
11. Zhang, J.; Feng, T.; Yan, F.; Qiao, S.; Wang, X. Analysis and design on intervehicle distance control of autonomous vehicle platoons. *ISA Trans.* **2019**, *100*, 446–453. [CrossRef] [PubMed]
12. Berger, T.; Rauert, A.-L. Funnel cruise control. *Automatica* **2020**, *119*, 109061. [CrossRef]
13. Girard, A.; Eqtami, A. Least-violating symbolic controller synthesis for safety, reachability and attractivity specifications. *Automatica* **2021**, *127*, 109543. [CrossRef]
14. Guo, N.; Zhang, X.; Zou, Y.; Lenzo, B.; Zhang, T. A Computationally Efficient Path-Following Control Strategy of Auton-omous Electric Vehicles with Yaw Motion Stabilization. *IEEE Trans. Transp. Electrif.* **2020**, *6*, 728–739. [CrossRef]
15. Liang, Z.; Zhao, J.; Liu, B.; Wang, Y.; Ding, Z. Velocity-based path following control for autonomous vehicles to avoid ex-ceeding road friction limits using sliding mode method. *IEEE Trans. Intell. Transp. Syst.* **2020**, *23*, 1947–1958. [CrossRef]
16. Ni, J.; Hu, J.; Xiang, C. Robust Path Following Control at Driving/Handling Limits of an Autonomous Electric Racecar. *IEEE Trans. Veh. Technol.* **2019**, *68*, 5518–5526. [CrossRef]

17. Ao, D.; Huang, W.; Wong, P.K.; Li, J. Robust Backstepping Super-Twisting Sliding Mode Control for Autonomous Vehicle Path Following. *IEEE Access* **2021**, *9*, 123165–123177. [CrossRef]
18. Wu, Y.; Liao, S.; Liu, X.; Li, Z.; Lu, R. Deep Reinforcement Learning on Autonomous Driving Policy with Auxiliary Critic Network. *IEEE Trans. Neural Networks Learn. Syst.* **2021**, *10*, 1–11. [CrossRef]
19. Martinsen, B.; Lekkas, A.M. Curved path following with deep reinforcement learning: Results from three vessel models. In Proceedings of the OCEANS 2018 MTS/IEEE Charleston, Charleston, SC, USA, 22–25 October 2018.
20. Cui, R.; Yang, C.; Li, Y.; Sharma, S. Adaptive Neural Network Control of AUVs With Control Input Nonlinearities Using Reinforcement Learning. *IEEE Trans. Syst. Man, Cybern. Syst.* **2017**, *47*, 1019–1029. [CrossRef]
21. Carlucho, I.; De Paula, M.; Wang, S.; Petillot, Y.; Acosta, G.G. Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning. *Robot. Auton. Syst.* **2018**, *107*, 71–86. [CrossRef]
22. Zhang, Q.; Lin, J.; Sha, Q.; He, B.; Li, G. Deep Interactive Reinforcement Learning for Path Following of Autonomous Underwater Vehicle. *IEEE Access* **2020**, *8*, 24258–24268. [CrossRef]
23. Hung, S.-M.; Givigi, S.N. A Q-Learning Approach to Flocking with UAVs in a Stochastic Environment. *IEEE Trans. Cybern.* **2016**, *47*, 186–197. [CrossRef] [PubMed]
24. Rubi, B.; Morcego, B.; Perez, R. A Deep Reinforcement Learning Approach for Path Following on a Quadrotor. In Proceedings of the 2020 European Control Conference (ECC), St. Petersburg, Russia, 12–15 May 2020. [CrossRef]
25. Hartmann, G.; Shiller, Z.; Azaria, A. Model-Based Reinforcement Learning for Time-Optimal Velocity Control. *IEEE Robot. Autom. Lett.* **2020**, *5*, 6185–6192. [CrossRef]
26. Desjardins, C.; Chaib-Draa, B. Cooperative Adaptive Cruise Control: A Reinforcement Learning Approach. *IEEE Trans. Intell. Transp. Syst.* **2011**, *12*, 1248–1260. [CrossRef]
27. Wang, P.; Chan, C.-Y.; De La Fortelle, A. A Reinforcement Learning Based Approach for Automated Lane Change Maneuvers. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26 June 26–1 July 2018; pp. 1379–1384.
28. Artunedo, A.; Villagra, J.; Godoy, J. Jerk-Limited Time-Optimal Speed Planning for Arbitrary Paths. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 8194–8208. [CrossRef]
29. Liang, Y.; Li, Y.N.; Khajepour, A.; Zheng, L. Holistic Adaptive Multi-Model Predictive Control for the Path Following of 4WID Autonomous Vehicles. *IEEE Trans. Veh. Technol.* **2020**, *70*, 69–81. [CrossRef]
30. Zhang, K.; Sprinkle, J.; Sanfelice, R.G. Computationally aware control of autonomous vehicles: A hybrid model predictive control approach. *Auton. Robot.* **2015**, *39*, 503–517. [CrossRef]