Reinforcement Learning Based Approach for Multi-Vehicle Platooning Problem with Nonlinear Dynamic Behavior

Amr Farag
Mechatronics Department
Faculty of Engineering and Materials Science
German University in Cairo (GUC)
Cairo, Egypt
amr.farag@ieee.org

Omar M. AbdelAziz
Mechatronics Department
Faculty of Engineering and Materials Science
German University in Cairo (GUC)
Cairo, Egypt
omarabdelazizguc@gmail.com

Ahmed Hussein
IAV GmbH
Berlin, Germany
ahmed.hussein@ieee.org

Omar M. Shehata
Mechatronics Department
Faculty of Engineering and Materials Science
German University in Cairo (GUC)
Cairo, Egypt
Omar.shehata@ieee.org

Abstract

One of the most recent research directions in the field of Cooperative Intelligent Transportation System is vehicle platooning. Researchers focused on various ways of tackling the autonomous vehicle control by traditional control strategies as well as state-of-the-art deep Reinforcement Learning (RL) methods. In this study, a detailed nonlinear dynamical vehicle model is reintroduced, in addition to a proposed optimal gap controller based on RL, which is demonstrated by an actor-critic policy with a deep deterministic policy gradient algorithm. The resulting agent is simulated for variable speed and variable gap aggressive scenarios and compared with the Model Predictive Control (MPC) performance. Results show that there exists a trade-off between accuracy and learning time. However, both controllers have a near-optimal performance.

1 Introduction

One of the recent trends in the field of Cooperative Intelligent Transportation System (C-ITS) is vehicle platooning. Various goals are achieved by platooning; such as reduction of fuel consumption, CO_2 emissions and traffic congestion and improvement of road capacity safety [16] [6]. One of the main challenges that face the development of platooning is follower vehicle control. The leader vehicle should follow a speed trajectory and the follower vehicles are expected to have the same trajectory while maintaining a predefined relative inter-vehicular distances [19]. Studies have shown that optimal control strategies can be achieved using RL methods [1, 8].

In this study, the focus is to develop an RL-based optimal controller for the vehicle platooning problem. The motivation behind the focus is to develop a mature controller that can handle various scenarios while having robust performance against model nonlinear dynamics and uncertainties. Thus, enabling further research directed towards other challenges than follower vehicle control.

Machine Learning for Autonomous Driving Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

The paper is organized as follows: Section 2 discusses related work to the problem. Section 3 presents the platoon configuration as well as remodeling of an existing detailed longitudinal dynamical vehicle model to a more realistic model for platooning purposes. Section 4 demonstrates RL as a tool for finding the optimal control law, while section 5 exhibits the environment used for training and simulation. The policy, learning algorithm and the parameters used are shown in section 6. Results and discussion are carried on in section 7. Finally, section 8 summarizes the work done in the study as well as future recommendations.

2 Related Work

The problem of controlling vehicles for autonomous driving purposes is not recent. Extensive researches were targeted to deploy different controllers in various environments. State-of-the-art algorithms suggest using Machine Learning approaches for addressing complex, scenario-based controllers. In particular, RL techniques are explored for various applications of autonomous vehicles. Dai et al. used RL for tuning of the fuzzy controller sets as an approach for autonomous vehicle longitudinal control problem [2]. On the other hand, Desjardins et al. used RL for cooperative adaptive cruise control by using gradient-descent learning algorithms [3], while B. Wang et al. handled adaptive cruise control problem by utilizing supervised actor-critic RL for acceleration controllers and using throttle and brake controllers for the lower level control [17]. Hu et al. used three model-free RL algorithms and a model-based one to keep the vehicle within its lanes with the desired speed and compared it to a fine-tuned traditional proportional–integral–derivative (PID) controller [4]. Other researchers focused more on the various models of multiple vehicles. In particular, a car-follower model where they can apply control techniques in order to maintain target speed and gap while satisfying desired objective functions. Zhu et al. developed a throttle and brake controller based on RL for a follower vehicle to maintain the leader's speed and the desired following distance (gap) [20].

Moreover, Lin et al. compared car-follower models where they employed Deep RL for controlling a point-mass kinematic model versus a longitudinal dynamic model. They showed that the Deep RL agent, which is based on the kinematic model, exhibited a degraded performance for realistic scenarios with acceleration-based dynamics while considering a delayed acceleration effect and vehicle dynamics for the redesigned Deep RL agent resulted in near-optimal control performance. They recommended developing a Deep RL controller, while the vehicle dynamics are considered to overcome more challenging scenarios [12].

From the above-demonstrated literature, there exists a research gap in utilizing RL as a controller for a dynamical vehicle model in a platoon. Some researchers use conventional controllers for the platooning problem. Moreover, [12] mentioned various studies that apply Deep RL based controllers considering only the point-mass kinematic model without the effect of the acceleration delay dynamics. Other researchers focused on using RL based controllers on a single vehicle or a car-follower problem without taking into consideration relative distance control. The aforementioned research gaps are filled by this study as explained in the following sections.

3 Problem Formulation

This section presents the platoon configuration along with remodeling of an existing detailed longitudinal dynamical vehicle model to a more realistic model for platooning purposes.

3.1 Platoon Configuration

In this study, the platoon configuration considered consists of heterogeneous vehicles, i.e. vehicles may vary in mass, length, or most importantly, the drag force coefficients of the vehicle. Moreover, the platoon has four vehicles, a leader and arbitrary number of follower vehicles where the leader is at the first position. The leader's states that describe the motion profile are $\dot{x}_0$ and $\ddot{x}_0$ which represent the speed and the acceleration of the leader, respectively. Furthermore, the gap between two vehicles, $d_g^{(i)}$, is the distance between the front bumper of the ego vehicle at position i and the rear bumper of the preceding vehicle at position $(i-1)$.

In addition, the leader is assumed to follow a speed trajectory that is only predefined for it. The objective of the other vehicles is to follow the trajectory of the preceding vehicles while leaving a defined relative distance, $d_g^{(i)}$. Therefore, it is assumed that there exists a perfect vehicle-to-vehicle (V2V) communication, i.e. all vehicles can observe the states of the preceding vehicle and the position of the i^{th} vehicle, $x^{(i)}$, is measured from the vehicle's centroid. For the remainder of this section, further investigations are carried on for the model that is used to represent the vehicles behavior.

3.2 Vehicle Longitudinal Dynamic Model in a Platoon

In this study, the vehicle model used is based on the nonlinear longitudinal model derived in [13]. However, the drag forces is further developed. Since the intent behind this study is to investigate the behavior in a platoon, it is more realistic to consider the effects of the reduction in the drag forces acting on the ego vehicle as a result of the platoon configuration. Under the mentioned reasons, the dynamics of the single platoon member is modeled by Equation 1.

$$\ddot{x} = \frac{T_t - T_{br} - (F_d + F_r + F_g) \cdot R}{m \cdot R + \frac{I_w}{R}} \tag{1}$$

where T_t is the traction torque that is generated from the engine and causes the vehicle forward motion, T_{br} is the torque produced from the brakes system to control the vehicle's decelration, F_d, F_r, and F_g are the drag, rolling resistance, and gravitational forces, respectively. The drag force is multiplied by a drag reduction ratio that simulates the effect of the preceding vehicle on the ego vehicle's drag, as presented in [15]. Finally, m, R, and I_w are the vehicle mass, wheel radius and wheel moment of inertia, respectively.

4 Proposed Model

RL can be defined as learning to take *Actions* while being in a *State* in order to maximize a numerical *Reward Function* [18] [14]. RL differs from dynamic programming that RL assumes no prior knowledge about the state or the best action to take to impact the reward.

4.1 Reinforcement Learning as Dynamical Model Controller

To understand how to utilize RL for a standard control problem, we must map between main components (concepts) of RL environment and the ones of a standard feedback loop of a dynamical system. The policy (inside RL agent) has the role of a controller. The states of the environment in RL are the same as the observed outputs of the system, while the action produced by the policy is the control input to the plant. The reward used in RL can also be seen as the objective function that the controller trying to maximize as in standard optimal control problems. In this manner, RL can be seen as the method of finding the solution for an optimization problem where the generated optimal policy is the optimal control law for the reward function.

4.2 Updating Policy Algorithms

During the training of the agent, there are plenty of methods that specify how to update the policy based on the feedback of the reward. The majority of these methods falls into the following two methods [9]:

- *Actor* methods which are policy-function based. The policy parameters are updated in the direction of improvement. These methods are considered direct methods
- *Critic* methods which are value-function based. These methods are considered indirect because they aim at learning an approximate solution to the Bellman optimality equation. This leads indirectly to update the policy parameters such that it reaches a near-optimal solution.

Another category that aims at combining the advantages of the previous aforementioned methods, is *Acotr-Critic* methods. The actor-network generates action and controls the agent output. On the

other hand, the critic takes the action from the actor with the current state as inputs and generates the predicted value of the reward function which is also known by policy evaluation. After that, the error between the predicted value and the actual reward is used as feedback for the actor and critic networks. Thus, the actor will be updated towards the optimal value based on the critic feedback of the predicted value, a process known as policy improvement [10]. In addition, the critic takes the pair of the current state and the action taken by the actor instead of evaluating the value of all possible state-action pairs, which could be extremely large or infinite in case of continuous action spaces [5].

5 Environment Setup

In this study, a decentralized unidirectional longitudinal RL-based controller is designed for each follower's vehicle in the platoon to achieve the desired gap between the vehicle and its preceding vehicle. The controller agent observes only the errors in the gap, speed, and acceleration between the two consecutive vehicles. For the leader, a separate speed controller agent is designed to maintain a set speed $\dot{x}_{ref}^{(0)}$. The vehicle model for all vehicles in the platoon is the same as the one derived in Section 3. Furthermore, the controllers are used for controlling the inter-vehicular distances for the platoon.

In this section, the rewards for the leader's speed controller and the followers' gap controller are introduced. As well as a training algorithm that is used for both agents.

5.1 Reward Shaping

For the RL agent, the policy inherited in the agent is updated to get the maximum reward. So, the optimal policy is the one that achieves the maximum for any given scenario. Hence, the reward function is considered an important aspect for the agent because the reward is the feedback from the environment that carries the information of the adequacy and acceptability of the performance based on the agent's policy. Furthermore, shaping the reward function guides the agent towards the direction of the optimum solution during the learning process. However, choosing the reward function is heavily dependent on the system. Therefore, the reward is defined based on the designer and their understanding of how the system behaves.

The formulated reward for the speed controller for the leader is governed by:

$$R_{l,t} = -(u_{t-1}^2 + 0.05\dot{u}_{t-1}^2 + 0.1e_{vl,t}^2) + Q_{l,t} \tag{2}$$

where u_{t-1} is the control effort (acceleration) of the previous time instant, $\dot{u}_{t-1}$ is the derivative of the control effort (acceleration) of the previous time instant. $e_{vl,t}$ is the error in the velocity between the set reference velocity and the current leader velocity ($v_{ref} - \dot{x}_0$), and $Q_{l,t}$ is a positive reward based on the logical equation:

$$Q_{l,t} = |e_{vl,t}| \leq \vartheta_v \wedge t \geq \tau \tag{3}$$

where ϑ_v is the threshold value of the acceptable tolerance in velocity error, τ is the threshold time after which the reward exists.

The squaring in the negative part of the reward is to account for positive or negative values in the described terms. The first term accounts for the minimization of the acceleration, while the second one is to account for the change in acceleration. Thus, minimizing the chattering in the control effort and ensures the smoothness of the signal. The third term eliminates the error in the velocity to achieve the controller's intended behavior. The Q part is to give the agent positive reward around the set-points to have a smother control efforts and not tighten the controller to strict values. The delay condition is important to ensure that the controller only receives positive rewards when it is actually in the right velocity and prevent any fake positive rewards for near-zero error in initial leader velocity from the set velocity

For the follower, the reward of the agent's policy is shaped such that:

$$R_{f,t} = -\left(u_{t-1}^2 + 0.05\dot{u}_{t-1}^2 + \left(\frac{1}{v_{max}} e_{vf,t}\right)^2 + \left(\frac{1}{d_{g_{max}}} e_{g,t}\right)^2 \right) + Q_{f,t} \tag{4}$$

which has the same parameters of the control effort as in equation (2). However, $e_{vf,t}$ is the error in velocity between the current velocity and the preceding vehicle's velocity ($v_i - v_{i-1}$), while $e_{g,t}$ is the error in gap ($gap_{desired} - d_g^i$), and $Q_{f,t}$ is a positive reward based on the logical equation:

$$Q_{f,t} = |e_{v,t}| \leq \vartheta_v \wedge |e_{g,t}| \leq \vartheta_g \wedge t \geq \tau \tag{5}$$

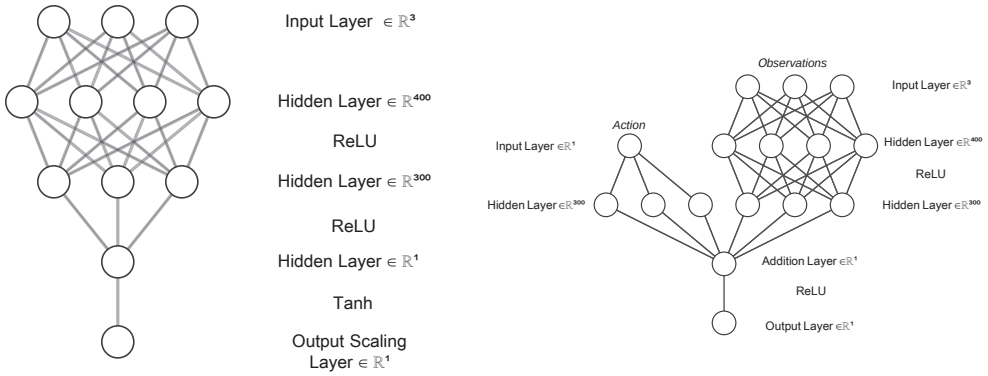
where ϑ_g is the threshold value of the acceptable tolerance in gap error.

The same reasons for terms in equations (2) and (3) are used in equations (4) and (5) with the addition of the error in gap term to eliminate the error achieving the desired controller. The gains of the errors in velocity and gap in equation (4) are used to normalize the values. Thus, the agent can satisfy easier the multi-objective reward function than the non-normalized function.

6 Experimental Work

6.1 Training

For this study, the two networks, the actor and critic networks are utilized with Deep Deterministic Policy Gradient (DDPG) algorithm. The reason behind this choice is the successful implementation in several difficult applications with continuous action spaces as mentioned by [11] and [7]. The actor-network has three fully connected hidden layers with 400, 300, and one neurons, respectively, while for the critic network, the action path has one hidden layer with 300 neurons. The state path has two hidden layers with 400, 300 neurons, respectively. Both paths merge into the output layer through the layer of addition. The RL model run at a sampling time of 0.1 seconds, with a reward discount factor of 0.99, a noise model variance of 0.6. For the training process, the mini-batch size was 128, while the learning rate for the actor and critic networks are 10^{-4} and 10^{-3}, respectively.



(a) Actor Neural Network

(b) Critic Neural Network

Figure 1: DDPG Neural Network Architecture

6.2 Simulation and Testing

The platoon is constructed based on discrete individual vehicle models using Simulink blocks and MATLAB functions. The gap controller agent is trained using RL Toolbox in MATLAB. The hardware platform used for training is a personal computer with a CPU of Intel Core i7-9750H and a GPU of Nvidia GeForce RTX 2060. The resulting agent is simulated with the simple-and-aggressive speed trajectory, the desired *constant* gap and initial conditions that are used in [13] on a platoon of

four vehicles. Furthermore, another more aggressive scenario with variable gaps during the trajectory is examined with the gap controller agent.

7 Results and Discussion

In this section, the outcomes of the training on the gap controller are exhibited. the results of the simulation on the gap controller with the constant gap scenario are presented. Moreover, a scenario with variable gaps is simulated on both controllers proposed and the results are demonstrated as well. Furthermore, a detailed comparison for optimization-based controllers is discussed between the RL agent and the conventional MPC.

The training for the ego vehicle controller took 2163 episodes with a total number of steps of 973,186 for approximately 9.5 hours. The final average reward based on Equation 4 was 326.8 as seen in Figure 2. The initial conditions of the velocities and positions of the vehicles are randomly changed before the beginning of each episode to ensure the reliability of the agent for any given realistic scenario and prevent overfitting of the model to certain scenarios. Thus, each episode has a different initial gap error and different initial speed error.

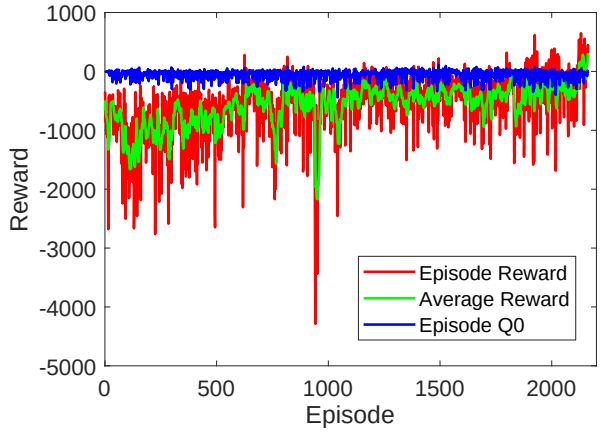


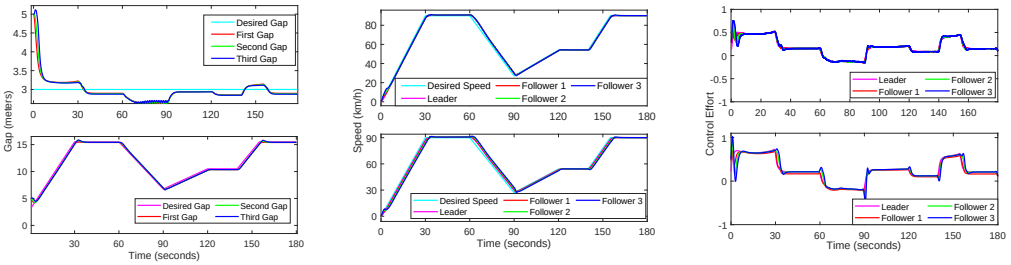
Figure 2: Episodes Reward During The Ego Vehicle Agent Training

Figure 3 shows the response to the constant gap scenario, compared with the variable gap scenario, with the multi-objective reward function agent. The agent can track the speed trajectory with acceptable error tolerance when the speed is varying. On the other hand, the agent maintained the reference speed at the part where the trajectory has a constant speed, with no steady-state error.

Regarding the constant distance gap scenario, the agent achieved the desired gap with acceptable error tolerance ($\sim |e_{g,t}| \leq 0.3$) which fulfils the reward function in Equation 5. The settling time is very close to the response of the MPC which is the fastest of the three controllers in [13]. The response overall is similar to the MPC, shown in Figure 5, which is justified and can be deduced from the fact that RL and MPC both are trying to solve the optimization problem to find the optimal policy π^* or control law u^*, respectively. Furthermore, it is observed that the control efforts are smooth and realistic for the predefined speed trajectory. The gap controller has proven the ability to handle simple and aggressive scenarios in a satisfactory manner with respect to the tolerance specified in the objective reward function.

The variable gap reference trajectory is designed based on the speed trajectory, where the desired gap is equal to a safe distance of 3 meters added to a timed gap which has a magnitude of half of the speed (in m/s). In addition, the response of the agent has a fast reaction against the changes in the gap with smooth speed trajectory as shown by Figure 3b in a satisfactory manner. It must be pointed out that during the parts where the gap is changing, there exists a constant shift in the followers' speed than the reference speed to create the increase or the decrease in the gap.

Regarding control efforts, both scenarios have some heavy changes in the control effort especially when the scenario is considerably aggressive, yet both responses are adequate and realistic enough



(a) Gaps (b) Speeds (c) Control Efforts

Figure 3: Response of Platoon Vehicles for the constant vs variable Gap scenarios, respectively

given the ruggedness in the scenario. However, based on the overall response of both agents, the robustness of the agent is insured by the variable-and-aggressive gap and speed trajectories simulation, which proves that both of them can overcome any changes from the scenarios that they were trained on based on the various initial conditions in the training phase.

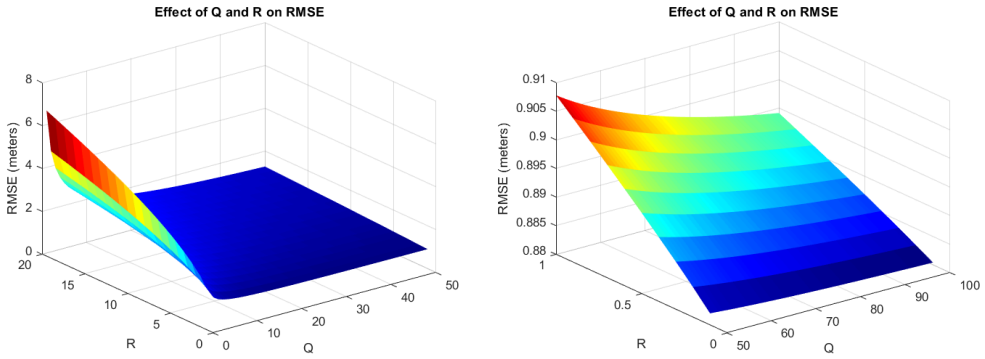
As seen in Table 1, the RL agent gives a similar Root Mean Square Error (RMSE) to the previously discussed controllers in [13]. However, the RL agent simulates significantly faster than the previously proposed optimal control, the MPC, in terms of computational time. In this manner, the RL agent demonstrates a good balance between accuracy and computational time among the proposed optimal controllers.

Table 1: RMSE of RL and MPC for constant and variable Gap scenarios

Controller	**Constant Gap Scenario**			**Variable Gap Scenario**			**Simulation Time**
	RMSE			**RMSE**			
	Gap 1	Gap 2	Gap 3	Gap 1	Gap 2	Gap 3	
MPC	0.2197	0.3032	0.3575	0.25505	0.26641	0.27045	58.83 s
RL	0.2670	0.2979	0.3395	0.31076	0.31182	0.33331	10.9 s

Further analysis can be carried on to investigate the difference in the performance between RL and MPC controllers. It should be pointed out that the comparison is not presented with the same parameters previously used in [13]. Alternatively, a parametric study was conducted on R and Q, the weight matrices for the control input and states, respectively. After that, the best combination of the R and Q matrices is chosen such that it achieves the least cumulative RMSE for the three gaps. The results of the study can be seen in Figure 4. Also, the MPC results were obtained from the model used in [13] without the proposed drag forces model in [15]. From Table 1, the RMSE for the RL agent, in constant gap scenario, is better in all but the first gap. Which can be interpreted that the propagation of the error across consecutive gaps is less than the MPC. In addition, Table 1 presents the RMSE for the variable gap scenario. As the RMSE for the MPC seem slightly better, the computational time was significantly larger because the optimal problem was reformulated at each time step. Thus, the simulation needed several minutes, while the results of the agent were achieved in just 10.4225 seconds justifying the need of RL as a solver for more complex scenarios with the existence of the nonlinear drag forces presented in section 3. Figure 5a shows the gap, speed and control effort for the MPC. In comparison to Figure 3a, settling time is almost the same. The error in tracking is similar in both controllers. The speed reference tracking is fulfilled. However, since we have 3 profiles in this scenario (increasing, constant, and decreasing speed), the control effort is smoother in the RL and less changing within the same speed profile. Furthermore, the analysis is carried on also on the variable gap scenario. Figure 5b also presents the gap, speed and control effort for the MPC in variable gap scenario. Taking into consideration the responses from the RL agent in Figure 3, it can be concluded that both controllers satisfy the given gap and speed profiles in a satisfactory manner with almost no significant difference. On the other hand, the RL agent kept its dominance in the control efforts by achieving smoother trajectories. Thus, it can be concluded that the utilization of RL in the context of controlling dynamic multi-vehicle platooning was successful,

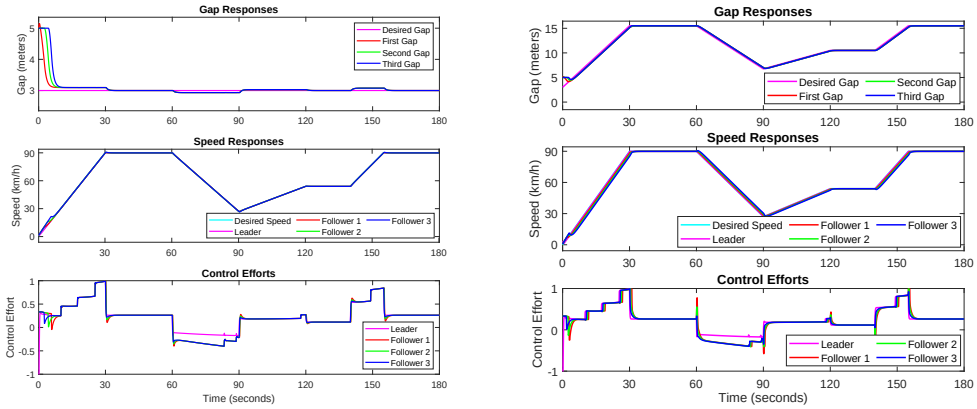
justified by the significant better optimal performance in terms of computational time as well as the control efforts.



(a) Q and R vs RMSE

(b) Zoomed in values from (a)

Figure 4: The Parametric Study on R and Q



(a) Constant Gap Response

(b) Variable Gap Response

Figure 5: Gap, Speed and Control Effort of MPC for both constant and variable gap scenarios

8 Conclusion

The problem of managing the inter-vehicle distances in a platoon of heterogeneous vehicles is addressed in this study, where a more mature model that reduces the air drag for each vehicle respectively is developed on an existing detailed nonlinear longitudinal dynamical model. RL is utilized as a tool to design a controller for the leader and follower vehicles. A gap-and-speed controller with a multi-objective reward function is proposed. This agent is trained based on the DDPG algorithm with actor and critic networks. From the results obtained by the simulation, the RL agent perform satisfactorily with respect to the reward function, the control efforts and the speed trajectory tracking. The analysis of optimal controllers confirms that the RL controller outperform the MPC in terms of computational time and control effort specially in more realistic and complex scenarios while maintaining similar RMSE in the inter-vehicle distances.

Moreover, it is recommended to continue training the model on the tuning parameters to get the best performance with respect to adjustable reward functions. Furthermore, the RL controller training can be carried over to satisfy other objectives without contradicting to the current objectives. Utilizing the controllers in extensive and realistic simulations can be done by Hardware-in-The-Loop simulators to investigate the vehicle's behavior under other scenarios and circumstances.

References

[1] L. Buşoniu, T. D. Bruin, D. Tolić, J. Kober, and I. Palunko. Reinforcement learning for control: Performance, stability, and deep approximators. *Annual Reviews in Control*, 46:8–28, 2018. doi: 10.1016/j.arcontrol.2018.09.005.

[2] X. Dai, C. Li, and A. Rad. An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems*, 6(3):285–293, 2005. doi: 10.1109/tits.2005.853698.

[3] C. Desjardins and B. Chaib-Draa. Cooperative adaptive cruise control: A reinforcement learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1248–1260, 2011. doi: 10.1109/tits.2011.2157145.

[4] B. Hu, J. Li, J. Yang, H. Bai, S. Li, Y. Sun, and X. Yang. Reinforcement learning approach to design practical adaptive control for a small-scale intelligent vehicle. *Symmetry*, 11(9):1139, Jul 2019. doi: 10.3390/sym11091139.

[5] Z. Huang, X. Xu, Z. Sun, J. Tan, and L. Qian. Speed tracking control via online continuous actor-critic learning. *2016 35th Chinese Control Conference (CCC)*, 2016. doi: 10.1109/chicc.2016.7553847.

[6] P. Kavathekar and Y. Chen. Vehicle platooning: A brief survey and categorization. *Volume 3: 2011 ASME/IEEE International Conference on Mechatronic and Embedded Systems and Applications, Parts A and B*, Jan 2011. doi: 10.1115/detc2011-47861.

[7] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez. Deep reinforcement learning for autonomous driving: A survey, 2020.

[8] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis. Optimal and autonomous control using reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 29(6):2042–2062, 2018. doi: 10.1109/tnnls.2017.2773458.

[9] V. R. Konda and J. N. Tsitsiklis. On actor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2003. doi: 10.1137/s0363012901385691.

[10] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis. Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems*, 32(6):76–105, 2012. doi: 10.1109/mcs.2012.2214134.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *International Conference on Learning Representations*, 2016.

[12] Y. Lin, J. Mcphee, and N. L. Azad. Longitudinal dynamic versus kinematic models for car-following control using deep reinforcement learning. *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019. doi: 10.1109/itsc.2019.8916781.

[13] M. K. Shalaby, A. Farag, O. M. Abdelaziz, D. M. Mahfouz, O. M. Shehata, and E. I. Morgan. Design of various dynamical-based trajectory tracking control strategies for multi-vehicle platooning problem. *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019. doi: 10.1109/itsc.2019.8917439.

[14] R. S. Sutton and A. G. Barto. *Reinforcement learning: an introduction*. The MIT Press, 2018.

[15] K. Tadakuma, T. Doi, M. Shida, and K. Maeda. Prediction formula of aerodynamic drag reduction in multiple-vehicle platooning based on wake analysis and on-road experiments. *SAE Int. J. Passeng. Cars - Mech. Syst.*, 9:645–656, 04 2016. doi: 10.4271/2016-01-1596.

[16] S. Tsugawa and S. Kato. Energy its: another application of vehicular communications. *IEEE Communications Magazine*, 48(11):120–126, 2010. doi: 10.1109/mcom.2010.5621978.

[17] B. Wang, D. Zhao, C. Li, and Y. Dai. Design and implementation of an adaptive cruise control system based on supervised actor-critic learning. *2015 5th International Conference on Information Science and Technology (ICIST)*, 2015. doi: 10.1109/icist.2015.7288976.

[18] M. Wiering and M. Otterlo. *Reinforcement Learning State-of-the-Art*. Springer Berlin, 2014.

[19] Y. Zheng, S. E. Li, J. Wang, D. Cao, and K. Li. Stability and scalability of homogeneous vehicular platoon: Study on the influence of information flow topologies. *IEEE Transactions on intelligent transportation systems*, 17(1):14–26, 2015.

[20] Q. Zhu, Z. Huang, Z. Sun, D. Liu, and B. Dai. Reinforcement learning based throttle and brake control for autonomous vehicle following. *2017 Chinese Automation Congress (CAC)*, 2017. doi: 10.1109/cac.2017.8243976.