Deep RL Reward Function Design for Lane-Free Autonomous Driving*

Athanasia Karalakou[1,2][0000−0003−0911−9692], Dimitrios Troullinos[1,2][0000−0003−3228−3888], Georgios Chalkiadakis[1][0000−0002−0716−2972], and Markos Papageorgiou[1][0000−0001−5821−4982]

[1] Technical University of Crete, Chania 73100, Greece
akaralakou@isc.tuc.gr,dtroullinos@dssl.tuc.gr,
gehalk@intelligence.tuc.gr,markos@dssl.tuc.gr
[2] Both authors contributed equally to this research.

Abstract. In this paper we present an application of Deep Reinforcement Learning to lane-free traffic, where vehicles do not adhere to the notion of lanes, but are rather able to be located at any lateral position within the road boundaries. This constitutes an entirely different problem domain for autonomous driving compared to lane-based traffic, as vehicles consider the actual two dimensional space available, and their decision making needs to adapt to this concept. We also consider that each vehicle wishes to maintain a (different) desired speed, therefore creating many situations where vehicles need to perform overtaking, and react appropriately to the behaviour of others. As such, in this work, we design a Reinforcement Learning agent for the problem at hand, considering different components of reward functions tied to the environment at various levels of information. Finally, we examine the effectiveness of our approach using the Deep Deterministic Policy Gradient algorithm.

Keywords: deep reinforcement learning · lane-free traffic · autonomous driving

1 Introduction

Applications of Reinforcement Learning (RL) in the field of autonomous driving are gaining momentum in recent years [1] due to advancements in Deep RL [11, 2], giving rise to novel techniques [4, 1]. Another important reason for this momentum is an increasing interest towards autonomous vehicles (AVs), as the current and projected technological advancements in the automotive industry can enable such methodologies in the real-world [8, 7].

As a result, novel traffic flow research endeavours have already emerged, such as *TrafficFluid* [12], which primarily targets traffic environments with 100% penetration rate of AVs (no human drivers). *TrafficFluid* examines traffic environments with two fundamental principles, namely: (i) *Lane-free* vehicle movement,

* The research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 Research and Innovation programme/ ERC Grant Agreement n. [833915], project TrafficFluid.

meaning that AVs under this paradigm do not consider lane-keeping, but are rather free to be located anywhere laterally;and (ii) *Nudging*, where vehicles may adjust their behavior so as to assist vehicles in their rear that attempt overtake.

In the context of lane-free driving, multiple vehicle movement strategies have already been proposed [12, 15, 17, 6]. To the best of our knowledge, so far there is no work that tackles the problem with RL techniques, while there is an abundance of (Deep) RL applications for conventional (lane-based) traffic environments [1, 8, 4]. In this work, we view the problem of designing an RL agent that learns a vehicle movement strategy in lane-free traffic environments. Specifically, we examine the Deep Deterministic Policy Gradient (DDPG) algorithm, which was designed to handle continuous action domains.

The reward design is crucial and determines the overall efficiency of the resulting policy [8]. Given the nature of the algorithms, their ability to properly learn only with delayed rewards and obtain a (near) optimal policy is uncertain, so we propose a set of different reward components, ranging from delayed rewards to more elaborate and therefore more informative regarding the problem's objectives. The learning objectives in our environment are twofold, and include safety, i.e., collision avoidance among vehicles, and that our agent is able to maintain a desired speed of choice.

2 Background And Related Work

This section presents the theoretical background of the Deep Deterministic Policy Gradient algorithm that we utilised, as well as related work tied to lane-free environments.

2.1 Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG) [10] is an off-policy, actor-critic, deep reinforcement learning algorithm based on the Deterministic Policy Gradient (DPG) [10], developed specifically for continuous action domains.

It uses experience replay and target networks, as in DQN [11], solving the issues of network learning stability. The target networks are two separate networks which are copies of the actor and critic network and track the learned networks with soft target updates.

The update for the critic is given by the standard DQN [11] update by taking targets $y_t = r_t + \gamma Q(s_{t+1}, \mu(s_{t+1}; \theta^{\mu^-}); \theta^{Q^-})$, where $Q(s, a; \theta^{Q^-})$ and $\mu(s; \theta^{\mu^-})$ refer to the target networks for the critic and actor respectively. The actor network is updated according to the sampled policy gradient, as stated in the Deterministic Policy Gradient Theorem [10].

2.2 Related Work

Under the lane-free traffic paradigm, multiple vehicle movement strategies [12, 15, 17, 6] have already been proposed, with approaches stemming from Control

Theory, Optimal Control and Multi-agent decision making. In more detail, [12] introduces a lane-free vehicle movement strategy based on heuristic rules that involve the notion of "forces" being applied to vehicles, in the sense that vehicles "push" one another so as to overtake, or in general to react appropriately. Now, [17] introduces a policy for lane-free vehicles based on optimal control methods, and more specifically *model predictive control*, where each vehicle optimizes its behavior for a specified future horizon, considering the trajectories of nearby vehicles as well. Furthermore, [6] designs a two-dimensional cruise controller for lane-free traffic, with more emphasis on Control Theory. Finally, [15] tackles the problem with the use of the max-plus algorithm, and constructs a dynamic graph structure of the vehicles, considering communication among vehicles as well. In this work, we introduce an alternative movement strategy based on Deep RL, providing various configurations for the reward function.

3 Our Approach

In this section we first present in detail the lane-free traffic environment we consider, and then the various aspects of the MDP formulation, specifically the State Representation, and Action Space, along with the different components proposed for the Reward Function Design.

3.1 The Lane-Free Traffic environment

As a training environment, we consider a ring-road traffic scenario populated with multiple automated vehicles applying the lane-free driving behavior, as outlined in [12]. Our agent is a vehicle that adopts the proposed MDP formulation, learning a policy through observation of the environment. The observational capabilities of our agent includes the position (x, y), speed (v_x, v_y) of nearby vehicles and its own. Both the position and speed are observed as 2-dimensional vectors, consisting of the associated longitudinal (x axis) and lateral (y axis) values. All the observable vehicles share the same dimensions and movement dynamics. Each vehicle selects randomly a desired speed v_d, within a specified range ($[v_{d,min}, v_{d,max}]$), and this information can also be monitored. Our agent controls 2 (continuous) variables, namely the longitudinal and lateral acceleration values (a_x, a_y), and determines the acceleration/deceleration through a_x, and left/right steering through a_y. Fig. 1 illustrates the traffic environment. The examined ring-road scenario is emulated through a highway, by having vehicles reaching the end-point reenter the highway appropriately. Vehicles' observations are adjusted accordingly, so that they observe a ring-road, e.g., vehicles towards the end of the highway observe vehicles in front, located after the highway's starting point.

As mentioned, other vehicles follow the lane-free vehicle movement strategy in [12], which does not involve learning, i.e., other agents follow a deterministic behavior w.r.t. their own surroundings. In addition, we disable nudging [12] for other vehicles—since when enabled, other vehicles move aside whenever we

Fig. 1. Snapshot of the Lane-Free Traffic environment

attempt an overtake maneuver, effectively easing the environment of our RL agent and leading it to learn an unrealistically aggressive driving policy.

3.2 State Representation

The state space describes the environment and must contain sufficient information for choosing the appropriate action. Thus, our observation space contains information about both the state of the agent in the environment and the surrounding vehicles. More specifically, regarding the state of the agent, it was deemed necessary to store its lateral position y, as well as both its longitudinal and lateral speed v_x, v_y. On the other hand, as far as the environment and the surrounding vehicles are concerned, matters are more complicated, due to the nature of our problem. In particular, in lane-based environments, the state space can be defined in a more straightforward manner, as an agent can be trained by utilizing information about the front and back vehicles on its lane, and the position of its previous vehicle on the adjacent lanes, in case it handles lane-changing movement as well. By contrast, in a lane-free environment, we need a more extensive set of information that depicts our environment in its entirety, as the number of surrounding vehicles in the 2D space we consider varies.

Our state needs to include information about a predefined number of vehicles, so only the n closest vehicles are considered in the state space, while 'placeholder' vehicles appearing far away may be included in the event that the number of vehicle is lower than n. This is necessary since the MDP formulation does not handle state vectors with variable size.

We store information about the speed of the surrounding vehicles, both longitudinal and lateral. Additionally, intending to have a sufficient state represen-

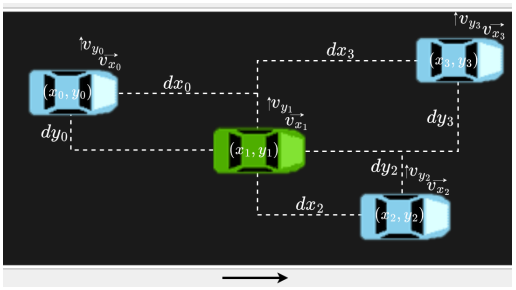

Fig. 2. Observed information of surrounding vehicles

tation, we include information regarding the longitudinal (dx) and lateral (dy) distances of the agent among our car's center and the cars within an at most d meters' (longitudinal) distance—as shown in Fig. 2. Finally, the state space of our experiments contains our agent's desired speed v_d.

3.3 Action Space

In this work, the primary objective is to find an optimal policy that generates the appropriate high-level driving behavior for the agent to move efficiently in a lane-free environment. Hence, our action space consists of two principal actions: one concerning the car's longitudinal movement by addressing braking and accelerating commands; and one relevant to the lateral movement through acceleration commands for acceleration towards the left or right direction. Since the DDPG method, which we used in this work, is developed on to environments with continuous action spaces, we formed our action space into a vector $\mathbf{a} \in \mathbb{R}^2$, coinciding with the two desired types of actions: a longitudinal acceleration and a lateral acceleration.

3.4 Reward Function Design

Finding an appropriate reward function for this problem is quite arduous due to the novel traffic environment. In particular, most of the related work in the literature on RL techniques for traffic is typically based on the existence of driving lanes, which constitutes a different problem altogether. For this reason, a reward function was constructed specifically for lane-free environments.

Several components of reward functions were investigated to explore their mechanism of influence, as well as to find the most effective form. Before presenting the various components, we first determine the agent's objectives within the lane-free traffic environment.

The designed reward function should combine the two objectives of our problem, that is, maintaining the desired speed v_d and avoiding collisions with other vehicles. All of the presented reward components attempt to tackle these two objectives. Some are more targeted only towards the end goal, and do not provide the agent with information for intermediate states, i.e., delayed rewards, while others are more elaborate and informative, and consequently tend to better guide the agent towards the aforementioned goals. Naturally, the more informative rewards aid in the learning process, and for the baseline algorithms examined, we also observe a strong influence in the results.

Longitudinal Target Regarding the desired speed objective, we utilize a linear function that focuses on maintaining the desired speed. In detail, the function is linear with respect to the current longitudinal speed v_x and calculates a reward based on the the deviation from the desired speed v_d of the agent at that specific time-step. To achieve this, the following mathematical formula is used:

$$c_x = \frac{|v_x - v_d|}{v_d} \tag{1}$$

It is evident that this function tends to be minimized at 0 whenever we approach the respective goal. As such, the form of the total reward r_t is a reciprocal function that contains a weighted form of c_x in the denominator. That being the case, we determined that our evaluation function should reverse this, so we select a reciprocal form $(1/x)$, and we put c_x as a denominator.

$$r_t = \frac{\epsilon_r}{\epsilon_r + w_x \cdot c_x} \tag{2}$$

where r_t is the total reward at any time-step t, while ϵ_r is a parameter that allows the reward to be maximized at 1 whenever c_x tends to 0. We choose a small value for ϵ_r, specifically $\epsilon_r = 0.1$, so as to make the minimum reward be close to 0 when c_x is maximized.

Overtake Motivation Term In our preliminary experiments, we determined that our agent tends to stuck behind slower vehicles, as it is deemed a "safer" action. However, this behavior is not ideal, as it usually leads to a greater deviation from the desired speed. To address this particular problem, we created a function that motivates the agent to overtake its surrounding vehicles.

In detail, a positive reward $c_{overtake}$ is attributed whenever our agent overtakes one of its neighboring vehicle. However, this reward is received only in cases that there are no collisions.

$$r_{t,o} = \begin{cases} r_t + c_{overtake} & \text{if agent does not collide \& overtakes a vehicle;} \\ r_t & \text{otherwise} \end{cases} \tag{3}$$

Collision Avoidance Term During the experimental evaluation of the aforementioned methods, we noticed that even though a significant number of collisions seemed to be averted, there were still some occurring that our agent did not manage to avoid. Based on that, the next logical step was to incorporate the collisions into our reward function. In this light, we examined numerous components, with the first being a "simpler" reward, by incorporating the training objective directly into the reward, aiming to "punish" the agent whenever a collision occurs.

This is exclusively based on the collisions between our agent and its surrounding vehicles. Specifically, a negative reward $c_{collide}$ is received whenever a collision occurs. Essentially, provided with a reward r_t according to one of the aforementioned forms, the reward $r_{t,c}$ that the agent receives is calculated as:

$$r_{t,c} = \begin{cases} r_t + c_{collide} & \text{if agent is involved in a collision;} \\ r_t & \text{otherwise} \end{cases} \tag{4}$$

However, this imposes the issue of delayed rewards, as our agent only receives a negative reward due to a collision with another vehicle. Especially in our domain of interest, our agent can be in many situations where a collision is inevitable, even many time-steps before the collision actually occurs, depending on the speed of our agent, along with the speed deviation and distance from the colliding vehicle.

Potential Fields To tackle the problem above, we also employ an alternative, more informative reward component, one that "quantifies" the danger of collision among two vehicles. The use of ellipsoid fields has been already utilized for lane-free autonomous driving as a measurement of collision danger with other vehicles [15, 17]. Provided with a pair of vehicles, the form of the ellipsoid functions calculates a utility that evaluates the danger of collision, taking into account the longitudinal and lateral distances, along with the respective longitudinal and lateral speeds of the vehicles and their deviations, adopted from [15].

Given our agent and a neighboring vehicle j, with longitudinal and lateral distance dx_j, dy_j, and longitudinal and lateral speed deviation $dv_{x,j}, dv_{y,j}$, the form of the ellipsoid functions is as follows: $u_j = E_c(dx_j, dy_j) + E_b(dx_j, dy_j, dv_{x,j}, dv_{y,j})$. Both $E_c(dx_j, dy_j)$ and $E_b(dx_j, dy_j, dv_{x,j}, dv_{y,j})$ contain an ellipsoid function and capture a critical and broad region respectively. Essentially, the critical region is based only on the distance of the two vehicles, while the broader region stretches appropriately according to the speed deviations, so as to properly inform on the danger of collision from a greater distance, and consequently the agent has more time to respond appropriately. The interested reader may refer to [15] for more information on these functions.

Moreover, we also need to accumulate the corresponding values for all neighboring agents, i.e., $u_t = \sum_j u_j$ for each neighboring vehicle j within our state observation at a given time-step t. Finally, to bound the associated reward, we have $c_{fields} = \min\{u_t, 1\}$. We know that each ellipsoid function is bounded within $[0, m_u]$, where m_u is a tuning parameter. As such, each utility u_j is bounded within $[0, 2m_u]$. Therefore, m is set accordingly ($m_u = 0.5$), so as to normalize each u_j values to $[0, 1]$. Thus, the reward $r_{t,fields}$ is adjusted according to r_t in Eq. 2 so as to incorporate this new component, and is calculated as:

$$r_{t,fields} = \frac{\epsilon_r}{\epsilon_r + w_x \cdot c_x + w_f \cdot c_{fields}} \tag{5}$$

Table 1. Hyper-parameters for RL algorithms

Parameter	Value	Parameter	Value
Optimizer	Adam	Learning rate α	0.001
Mini-Batch size	64	Actor Act. Function	ReLU(Hidden), TanH(Output)
Discount factor γ	0.98	Critic Act. Function	ReLU(Hidden), Linear(Output)
Replay Memory size	100000	Actor Layer Size	256,128,2
Soft update parameter	0.001	Critic Layer Size	256,128,1
Training episodes N	625	Noise Process	Ornstein-Uhlenbeck

Notice that $r_{t,fields} = r_t$ whenever there is no captured danger with neighboring vehicles, i.e., the ellipsoid functions for each neighbor j returns $u_j = 0$.

All-Components Reward Function To further improve our agent's performance, we combine all of the previous components in a single reward function $r_{t,all}$, that contains $r_{t,fields}$ (Eq. 5) and the additional terms for overtake (Eq. 3) and collision avoidance (Eq. 4).

4 Experimental Evaluation

In this section we present our experimental results via a comparative study of the different reward functions that we propose, aiming to showcase trade-offs between the two objectives.

4.1 RL Algorithm and Simulation Setup

DDPG was the prevalent choice for testing the proposed design, as it tackles continuous action domains. DQN (and popular variants) were also examined with a discretized action domain, but exhibited inferior performance. The hyperparameters used in our implementation are provided in Table 1. We empirically examined different parameter tunings, and selected the ones that provide the best overall results.

We train and evaluate our method on a lane-free extension of the Flow [16] simulation tool, as described in [15]. Moreover, to facilitate our experiments, we utilized the Keras-RL library [13]. The Keras-RL framework implements some of the most widely used deep reinforcement learning algorithms in Python and seamlessly integrates with Tensorflow and Keras. However, technical adjustments and modifications were necessary to make this library compatible with our problem and environment. Furthermore, the proposed lane-free driving behavior decision-making model was tested in the highway scenario as specified in Sec. 3.1 with the specified parameter choices of Table 2, whereas in Table 3, we provide the parameter settings related to the MDP formulation.

4.2 Results and Analysis

As discussed in Sec. 3.4, we proposed several reward components. Their effectiveness is evaluated based on three metrics. These are: the average reward value,

Table 2. Simulation Parameters

Parameter	Value	Parameter	Value
Highway length	500 m	Vehicles' length	3.5 m
Highway width	10.2 m	Vehicles' width	1.8 m
Types of vehicles	2	Num. of vehicles	35
Agent's length	3.2 m	Time-Interval	0.25 s
Agent's width	1.6 m	Execution Time	200 s

Table 3. Parameter choices related to MDP formulation

Parameter	Value	Parameter	Value
agent's desired speed v_d	$20m/s$	w_x	0.65
other vehicles' desired speed	$[18-22]m/s$	$c_{overtake}$	2
num. of vehicles in state n	5	$c_{collide}$	-2.5
longitudinal observation distance d	80	w_f	1

the speed deviation from the desired speed (for each step, we measure the deviation of the current longitudinal speed from the desired one ($v_x - v_d$), in m/s), and the average number of collisions. All results are averaged from 10 runs.

In Figs. 3, 4 and 5 we demonstrate our agent's average reward, speed deviation, and the number of collisions respectively. In each of these figures, there are five curves that depict the performance of the proposed reward functions. Specifically, in each examined reward function, the Longitudinal Target reward 3.4 is combined with an associated component to tackle the collision avoidance objective. We refer to the reward associated with the Collision Avoidance Term (Eq. 4) as 'Collision Avoidance Reward Functions', while the addition of the overtaking motivation in that specific reward function, (Eq. 3), is labeled as 'Overtake and Avoid Collision Reward Function'. Furthermore, the use of the fields (Eq. 5) for that objective is labeled as 'Fields Reward Function' and 'Fields and Avoid Collision Reward Function' when combined with the Collision Avoidance Term. Finally, the assembly of all components in a single reward function (see Sec. 3.4) is referred to as 'All-Components Reward Function'. All of the aforementioned functions demonstrate how the agent's policy has improved over time.

As evident in Figs. 4 and 5, the 'Collision Avoidance Reward Function' manages to maintain a longitudinal speed close to the desired one. Yet, it does not manage to decrease the number of collisions sufficiently. Moreover, the addition of the overtaking component, in 'Overtake and Avoid Collision Reward Function', achieves a longitudinal speed slightly closer to the desired one, while the collision number is still relatively high. On the contrary, the 'Fields Reward

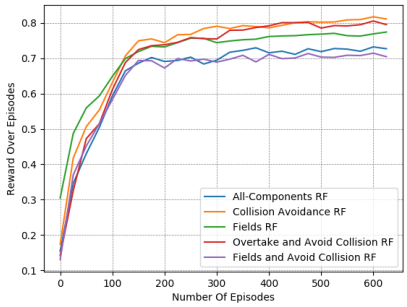

Fig. 3. Reward over time for different reward functions

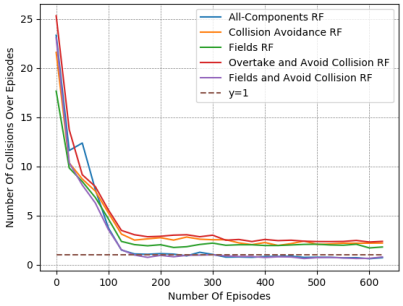

Fig. 4. Collisions over time for different reward functions

Table 4. Comparing the different Collision related components

Function	Coll.	Overtake &Coll.	Fields	Fields &Coll.	All-Components
Collisions	2.26	2.29	1.76	0.72	0.64
Speed Dev. (m/s)	-0.13	-0.05	-0.32	-0.69	-0.61

Function' exhibits a similar behavior to the previous mentioned reward functions, with a slight improvement on collision occurrences. Finally, both the 'Fields and Avoid Collisions Reward Function' and the 'All-Components Reward Function' perform slightly worse in terms of speed deviations. However, they obtain significantly better results in terms of collision avoidance, therefore balancing the two objectives much better. On closer inspection though, the 'All-Components Reward Function' manages to maintain a smaller speed deviation and number of collisions, thus making it the prevalent choice for a more effective policy overall.

To further demonstrate this point, we present in Table 4 a detailed comparison between these 5 reward functions. The reported results are averaged from the last 50 episodes of each variant, where the learned policy has converged in all

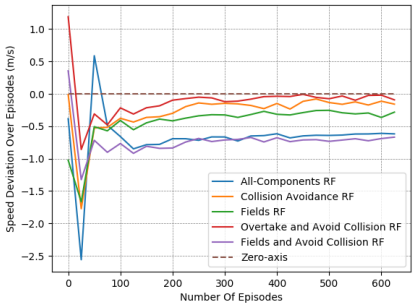

Fig. 5. Speed Deviation over time for different reward functions

cases, as noticeable in Fig. 3. Policies resulting from different parameter tunings that give more priority to terms related to collision avoidance ($c_{collide}, w_f$) do in fact further decrease collision occurrences, but we always observed a very simplistic behavior where the learned agent just follows the speed of a slower moving vehicle in front, i.e., is too defensive and never attempts overtake. Therefore, such policies were neglected.

Evidently, higher rewards do not coincide with fewer collisions, meaning that the reward metric should not be taken at face value, as we compare different reward functions. This is particularly noticeable in the case of the 'All-Components Reward Function' and the 'Fields and Avoid Collisions Reward Function', where there is a reduced reward over episodes, but when observing each objective, they clearly exhibit the best performance (cf. Fig. 4 , Fig. 5 and Table 4). This is expected, since the examined reward functions have different forms. In Table 4 we also observe the effect of the 'Overtake' component. Its influence in the final policy is apparent only when combined with 'Fields and Avoid Collisions Reward Function', i.e., forming the 'All-Components Reward Function'.[3]

Finally, we must point out that a slight deviation from the desired speed in our experiments is to be expected. Maintaining the desired speed throughout an episode is not realistic. Throughout our experiments, it is obvious that the two objectives are countering each other since a vehicle operating with slower speed is more conservative, while a vehicle wishing to maintain higher speed than its neighbors needs to overtake in a safe manner, and consequently has to learn a more complex policy that performs such an elaborate maneuvering.

5 Conclusions and Future Work

In this work, we formulated the problem of single agent autonomous driving in a lane-free traffic environment, and introduced a set of reward functions at various levels of information in order to tackle the two objectives, namely collision avoidance and targeting a specific speed of interest. Moreover, we evaluated our formulation and compared the proposed reward functions, utilizing a popular Deep RL algorithm, DDPG. In future work, we plan on examining different RL algorithms, such as NAF [5], PPO [14], other noteworthy advancements from the Deep RL literature [2], and also methods that do not necessarily involve learning, such as Monte-Carlo tree search (MCTS) [3]. Finally, it would be interesting to utilize Deep RL to explicitly tackle multi-objective problems [9].

References

1. Aradi, S.: Survey of deep reinforcement learning for motion planning of autonomous vehicles. IEEE Trans. on Intelligent Transportation Systems **23**(2), 740–759 (2022)

[3] Videos showcasing a trained agent with 'All-Components Reward Function' can be found at: https://bit.ly/3O0LjJW.

2. Badia, A.P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z.D., Blundell, C.: Agent57: Outperforming the Atari human benchmark. In: Proc. of the 37th Intern. Conf. on Machine Learning. PMLR, vol. 119, pp. 507–517 (2020)
3. Coulom, R.: Efficient selectivity and backup operators in monte-carlo tree search. In: Computers and Games. pp. 72–83 (2007)
4. Di, X., Shi, R.: A survey on autonomous vehicle control in the era of mixed-autonomy: From physics-based to ai-guided driving policy learning. Transportation Research Part C: Emerging Technologies **125** (2021)
5. Gu, S., Lillicrap, T., Sutskever, I., Levine, S.: Continuous deep q-learning with model-based acceleration. In: Proc. of The 33rd International Conference on Machine Learning. PMLR, vol. 48, pp. 2829–2838 (2016)
6. Karafyllis, I., Theodosis, D., Papageorgiou, M.: Two-dimensional cruise control of autonomous vehicles on lane-free roads. In: 60th IEEE conference on Decision and Control (CDC2021). pp. 2683–2689 (2021)
7. Kendall, A., Hawke, J., Janz, D., Mazur, P., Reda, D., Allen, J.M., Lam, V.D., Bewley, A., Shah, A.: Learning to drive in a day. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 8248–8254 (2019)
8. Kiran, B.R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A.A.A., Yogamani, S., Pérez, P.: Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems pp. 1–18 (2021)
9. Li, C., Czarnecki, K.: Urban driving with multi-objective deep reinforcement learning. In: Proc. of the 18th International Conference on Autonomous Agents and MultiAgent Systems. p. 359–367. AAMAS '19 (2019)
10. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: Proc. of the 4th International Conference on Learning Representations, ICLR, (2016)
11. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning (2013)
12. Papageorgiou, M., Mountakis, K.S., Karafyllis, I., Papamichail, I., Wang, Y.: Lane-free artificial-fluid concept for vehicular traffic. Proceedings of the IEEE **109**(2), 114–121 (2021)
13. Plappert, M.: keras-rl. https://github.com/keras-rl/keras-rl (2016)
14. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
15. Troullinos, D., Chalkiadakis, G., Papamichail, I., Papageorgiou, M.: Collaborative multiagent decision making for lane-free autonomous driving. In: Proceedings of AAMAS-2020. p. 1335–1343 (2021)
16. Wu, C., Kreidieh, A.R., Parvate, K., Vinitsky, E., Bayen, A.M.: Flow: A modular learning framework for mixed autonomy traffic. IEEE Transactions on Robotics p. 1–17 (2021)
17. Yanumula, V.K., Typaldos, P., Troullinos, D., Malekzadeh, M., Papamichail, I., Papageorgiou, M.: Optimal path planning for connected and automated vehicles in lane-free traffic. In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). pp. 3545–3552 (2021)