

Design and Experimental Validation of RL-Based Decision-Making System for Autonomous Vehicles



Ana María Gómez Ruiz, Hussam Atoui, and Olivier Sename

Abstract In autonomous driving, different Reinforcement Learning (RL) methods have been implemented to deal with different challenges. One of its advantages is the capability to deal with unexpected situations after an adequate trained environment. The inclusion of RL algorithms is considered as a solution for autonomous driving called “agent” that gathers the environmental information and acts according to this from one state to the next one. This paper proposes a solution for a specific environment that is trained with Deep RL and then is tested in simulation and in on experimental platform.

Keywords Deep reinforcement learning · Decision making · Autonomous vehicles

1 Introduction

Autonomous driving systems have raised a considerable interest in the last decades for several reasons. Initially, it can decrease the majority of lethal accidents that are caused by distracted drivers which will create safer roads. More than 90% of reported traffic accidents are the outcome of human error and caused by issues related to the acquisition of visual information as debated in [10]. Nevertheless, sophisticated

A. M. Gómez Ruiz (✉) · O. Sename
University Grenoble Alpes, CNRS, Grenoble INP, GIPSA-Lab, 38000 Grenoble, France
e-mail: ana.gomez@univ-grenoble-alpes.fr

O. Sename
e-mail: olivier.sename@gipsa-lab.grenoble-inp.fr

H. Atoui
Valeo, Driving Assistance Research (DAR) Team, 94000 Créteil, France
e-mail: hussam.atoui@valeo.com

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2024
G. L. Conte and O. Sename (eds.), *Proceedings of the 11th International Conference on Mechatronics and Control Engineering*, Lecture Notes in Mechanical Engineering,
https://doi.org/10.1007/978-981-99-6523-6_8

autonomous driving can decrease accidents caused by human errors, can redirect driving time into more productive ends and it can lower operating costs per mile finding optimal paths to destination.

The autonomous driving system have been under fast development in the recent years and different approaches have been implemented. Common modules to design autonomous systems include localization, perception, decision making (path planning) and dynamics control [13]. The main task of the environment localization and perception module is to extract useful features from the surroundings and locate the vehicle in the track to establish spatial and temporal relationships among the vehicle [3]. Identifying objects in the road, pedestrians, bicycles among others is classification ability that has raised a great interest with Machine Learning algorithms specially with supervised learning. To get this information from the vehicle environment, the module relies on different kinds of perception sensors such as cameras, radar and lasers [14].

The trajectory planning module aims to plan different longitudinal and lateral vehicle maneuvers which might include lane changing, braking, lane following and obstacle avoidance. There are existing methods that rely on traditional classical planners or machine learning methods. An alternative approach to the classical planners and supervised learning methods is Reinforcement Learning. This framework works on the principle of maximizing reward for a particular action at a given state [5]. RL is the theory of an agent that learns optimal behavior through interaction with its environment. With the aid of Deep RL techniques it is possible to use the benefits of deep learning in conjunction with RL to learn optimal behavior from high dimensional inputs to action outputs as discussed in [11]. In this paper, Actor-Critic methods are used to combine value-based and policy-based algorithms to sample efficiency and stability being effective in high dimensional and stochastic actions.

The general objective of the project is to build, integrate and test different modules of perception and control for a scaled autonomous vehicle in the Robot Operating System (ROS) framework. The car is grouped by Engineers at Gipsa-Lab. Previous work has been made in the vehicle such as identification of the model's vehicle, and its actuators along with the main connections on ROS2. In addition, the design and implementation of robust controllers for the vehicle lateral dynamics using different approaches has been made.

The following work aims to design the decision making module based on deep RL approach. The vehicle must avoid collisions, achieve high driving efficiency by taking an optimal path, and execute smooth maneuvers without veering off the track while maintaining the center-line of a two-lane race track. The RL model is trained in simulation with a Deep Q-Network, and is then validated and tested in an experimental scenario with the scaled RC car.

This paper is organized in five sections. The Sect. 2 aims to explain the theory and main components of the RL Algorithms that are used and the Actor Critic Approach. The Sect. 3 explains the implementation and training of the RL model with some simulation results. The Sect. 2 shows the validation and experimental results, and the Sect. 5 are the conclusions and final remarks of the work made.

2 Reinforcement Learning for Autonomous Driving

For autonomous driving, different Machine Learning (ML) methods have been implemented to deal with different challenges. Some of these algorithms have raised great interest because of the capability to deal with unexpected situations after an adequate training on a large set of sample data. One of the biggest challenges with ML algorithms for autonomous driving is when considering the vehicle in an open context environment to train the model with all possible scenarios in the real world. The variety of context that could happen are infinite and the companies leading this field must solve it by collecting a big amount of data and validating system operation based on the collected data to ensure that a self-driving car has already learned all possible scenarios and with safety scenarios for each case [1]. The inclusion of RL algorithms is being considered as a solution for the car called agent that gathers the environmental information and acts according to this from one state to the next one.

The general idea for implementing RL algorithms is to take the most important aspects of a learning agent that is interacting with its environment to reach a goal. The agent must be capable to perceive the state of the environment described as observation and it must be able to take actions that affect its state; refer to [17]. This agent also has a reward according to the state of the environment and the objective is to obtain the highest value for the sum of rewards over the long run.

The RL algorithms are considered closed-loop because the actions taken by the agent influence its later inputs. As a difference with ML algorithms, the agent is not guided to which action to take but instead to discover which actions will yield to the most reward by exploring them out. In the most complicated cases, actions may affect not only the immediate reward, but also the next situations and all the subsequent rewards. Such characteristic of not having a direct instruction on what action to take, and the consequences of actions are the most important features of the reinforcement learning problems [12]. The goal is to find a sequence of inputs that drive a dynamical system to maximize some objective, beginning with minimal knowledge of how the system responds to inputs.

2.1 Elements of Reinforcement Learning

In order to explain the elements of the RL algorithm some definitions for the interaction to achieve a goal will be explained. The learner and decision-maker is called the *agent*, the ego vehicle. The agent interacts with what is called the *environment* which includes everything outside the agent, i.e., the racetrack, the obstacles and the surrounding vehicles. The agent takes an action which results in a change in the environment. This interaction is received by the agent as a *state* which includes information about coordinates and/or speed of other vehicles, features of the road, among others. Refer to Fig. 1 to visualize the connection between these components.

The main subelements of RL algorithms are:

- Policy: Is a mapping from perceived states of the environment to actions to be taken in those actions. It is sufficient to determine the behavior, policies may be stochastic.
- Reward: The objective of the agent is to maximize the cumulative reward received over the long run. This value depends on the agents current action and the current state of the agent’s environment at any time. The only way the agent can influence the reward signal is through its actions, which can have a direct effect on the total reward, or an indirect effect through changing the environment’s state. The policy may be changed to select the action that will be followed by a higher reward on that situation in the future.
- Value function: Specifies what is good in the long run defined as episode. The value of a state can be described as the total amount of reward an agent can expect to accumulate over the future, starting from that state. Whereas rewards determine the immediate, intrinsic desirability of environmental states, values indicate the long-term desirability of states after taking into account the states that are likely to follow, and the rewards available in those states.
- Model: Is a representation of the behavior of the environment. When an action is made given a state the model might predict the resultant next state and reward due to this action. The model is used for planning and to consider possible future situations before they actually happen.

2.2 Reinforcement Learning theory

The interaction between the agent and the environment occurs at a sequence of discrete time steps t in which it receives some representation of the environment’s state $S_t \in \mathcal{S}$ in the \mathcal{S} set of possible states, and it selects an action $A_t \in \mathcal{A}(S_t)$ where $\mathcal{A}(S_t)$ is the set of actions available in state S_t . One time step later, in part as consequence of its action, the agent receives a numerical reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ and finds itself in a new state S_{t+1} . The Fig. 1 represents the agent-environment interaction. At each time step, the agent implements a mapping from states to probabilities of selecting each possible action. This mapping is called the agent’s policy and is denoted π_t , where $\pi_t(a | s)$ is the probability that $A_t = a$ if $S_t = s$. Reinforcement learning methods specify how the agent changes its policy as a result of its experience. The agent’s goal, roughly speaking, is to maximize the total amount of reward it receives over the long run [12].

An agent can increase the long-term reward by exploiting knowledge learned about the discounted sum of expected future rewards of different state-action pairs. The learning agent has to exploit what it already knows in order to obtain rewards, but it also has to explore the unknown in order to make better action selections in the future [2].

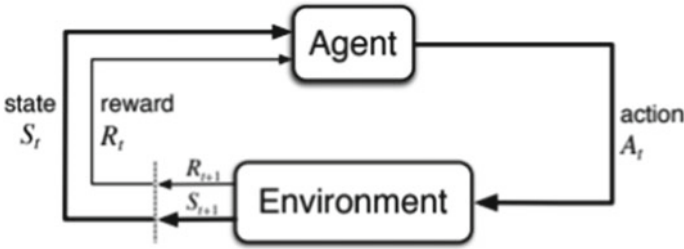


Fig. 1 The agent-environment interaction in RL. Image taken from [12]

For some stochastic control problems when the models for sequential decision making outcomes are uncertain, Markov Decision Processes (MDP) are used. The MDP model consists of decision epochs, states S , actions A , rewards R , and transition probabilities T ; a tuple $\langle S, A, T, R \rangle$. Choosing an action a in a state s generates a reward $R(s,a)$ and determines the state at the next decision epoch s' through a transition probability function $T(s, a, s')$. Policies are instructions of which action to choose under any occurrence at every future decision. The agent look for policies which are optimal [7]. The mathematical representation of the policy which is a mapping from the state space to a probability over the set of actions, and $\pi_t(a | s)$ represents the probability of choosing action a at state s . The goal is to find the optimal policy π^* at time k , defined as:

$$\pi^* = \operatorname{argmax}_{\pi} \mathbb{E}_{\pi} \left\{ \sum_{k=0}^{H-1} \gamma^k R(s_k, a_k) \mid s_0 = s \right\} := \operatorname{argmax}_{\pi} V_{\pi}(s) \quad (1)$$

where γ is the discount factor that controls how an agent consider future rewards. When γ is low the agent will maximize short term rewards, on the contrary with high values of γ the agent will try to maximize rewards over a longer time frame. The Eq. (1) represents the highest expected sum of discounted rewards ([16]) in a time horizon H in the MDP. From the models directly, RL agents may learn value function estimates, policies and/or environment. Finding a policy π that maximizes the expected discounted sum of rewards over trajectories in the state space is what solving a RL task means.

2.3 Reinforcement Learning Components for Autonomous Driving

Some of the most important elements of the RL model are the actions, the state, the observations and rewards.

Actions The actions that the vehicle can perform are driven by the acceleration and the steering control of the vehicle. The actions are considered discrete for the agent to decide which distinct action to perform from a finite action set.

The *DiscreteMetaAction* type adds a layer of speed and steering controllers on top of the continuous low-level control, so that the ego-vehicle can automatically follow the road at a desired velocity. Then, the available meta-actions consist in changing the target lane and speed that are used as set points for the low-level controllers. The actions are listed as:

- 0: Lane left
- 1: IDLE
- 2: Lane right
- 3: Faster
- 4: Slower.

State The state of the vehicle, also named as observations, contains information of the agent and the vehicles around it. The *KinematicObservation* is the default of the library, this is an array of size $nObs \times nF$ where n is the number of nearby vehicles and F is a set of features such as curvature, x , y , v_x , v_y . The number of vehicles n is constant and configured initially by the environment, so that the observation has a fixed size. The curvature of the track has been included as the inverse of the lookahead radius ($\frac{1}{r}$) after several attempts of training the model. Its inclusion is an improvement to consider the approaching curve so that the agent can decrease the speed when getting into a pronounced curve that is 3m in front so it can keep the lane center trajectory.

$$\left[\frac{1}{r} \ x \ y \ v_x \ v_y \right]^T \quad (2)$$

Rewards The final element to be defined are the rewards, the choice of an appropriate reward function yields realistic optimal driving behavior. A reward for collision, zero speed, lane centering and high speed has been defined. The total reward in every step will be determined by the sum of each condition. R_{coll} is the reward if it collides being -10 if it does and 0 if it does not. R_{stop} is the reward given if the vehicle stops, is 0 if the vehicle has some speed and -10 if it stops. R_{lc} is the reward given for lane centering, is maximum when the vehicle is in the center of the lane and it decreases proportionally when it moves away from the center lane as in Eq. (4). Finally, R_{hs} is the high speed reward and its value is a function of the speed of the vehicle as in 4. The total reward R_{total} is given by Eq. (3) and the final tuning of the rewards which resulted on the best simulation results is given in Table 1

$$R_{total} = R_{coll} + R_{stop} + R_{lc} + R_{hs} \quad (3)$$

$$R_{lc} = \frac{1}{1 + r_{lc} * lat\ error^2} - 0.5 \quad (4)$$

Table 1 Values of rewards of the RL model

Reward	Value
Collision r_c	-10
Zero speed r_{v0}	-10
Lane centering r_{lc}	1
High speed r_{hs}	0.7

r_{lc} is the weight of the lane centering, and *lat error* is the difference between the reference trajectory and the position of the vehicle, the second term of Eq. (4) is also tuned. The values of speed are also discrete and could take 6 different values between 0 and 1.3m/s as shown in Eq. (5), when the speed is at the maximum, then the reward will be r_{hs} , if it decreases then the reward will decrease proportionally as the range of speed of the vehicle.

$$R_{hs} = r_{hs} \frac{index_v}{index_{v \max}} \quad (5)$$

After the R_{total} is obtained it is normalized between 0 and 1, and it becomes an input of the RL model.

2.4 Actor Critic Approach

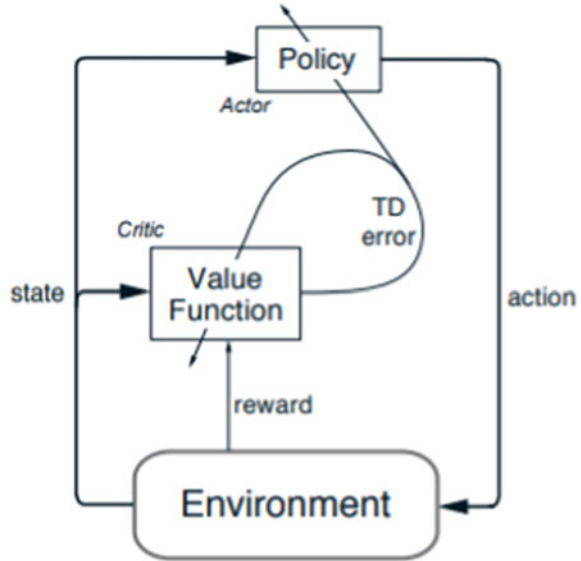
Actor-critic methods are hybrid methods that combine value-based and policy-based algorithms. One actor is the one that selects the actions and this is the policy-structure. After an action is made by the ‘actor’, the estimated value function evaluates the action and this is known as the ‘critic’. The value-based methods are model-free Temporal Difference (TD), are methods that can learn directly from raw experience without a model of the environment’s dynamics and learn estimates of the utility of individual state-action pairs represented in Eq. (6) [15]. This scalar signal is the sole output of the critic and drives all learning in both actor and critic, as shown in Fig. 2.

$$Q_{\pi}(s, a) = \mathbb{E}_{\pi} \left\{ \sum_{k=0}^{H-1} \gamma^k R(s_k, a_k) \mid s_0 = s, a_0 = a \right\} \quad (6)$$

Q-learning will learn (near) optimal state-action values provided a big number of samples are obtained for each pair. Agents implementing Q-learning update their Q values according to the update rule of Eq. (7):

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a' \in A} Q(s', a') - Q(s, a) \right] \quad (7)$$

Fig. 2 The actor-critic architecture. Image taken from [12]



where $Q(s, a)$ is an estimate of the utility of selecting action a in state s ; α is the learning rate which controls the degree to which Q values are updated at each time step [15].

The policy-based methods aim to estimate the optimal policy directly, and the value is a secondary. Typically, a policy π_θ is parameterized as a neural network. Policy gradient methods use gradient descent to estimate the parameters of the policy that maximize the expected reward. The result can be a stochastic policy where actions are selected by sampling, or a deterministic policy. When selecting actions, exploration is performed by adding noise to the actor policy. To stabilize learning a replay buffer is used to minimize data correlation. A separate actor-critic specific target network is also used. Normal Q-learning is adapted with a restricted number of discrete actions the optimal Q-value and optimal action as Q^* and a^* .

$$Q^*(s, a) = \max_{\pi} Q_{\pi}(s, a), \quad a^* = \operatorname{argmax}_a Q^*(s, a) \quad (8)$$

By correcting the Q-values towards the optimal values using the chosen action, the policy is updated towards the optimal action proposition. Thus two separate networks work at estimating Q^* and π^* .

3 Training and Testing of the Model

The aim of the project is to drive a scaled vehicle on a racetrack without veering off track or crashing, and reaching an optimal speed to finish a loop as fast as possible. Initially, the implemented solution has been developed in an existing framework named *highway-env* ([github library](#)), which is an open source Python library with a collection of different environments for autonomous driving and tactical decision-making tasks. This tool has been modified to create a new environment with specific dimensions of the track and the car for the particular environment of the vehicle and available space of the experimental room at GIPSA-Lab, in the next chapter the details of the scaled vehicle will be discussed. After setting the vehicle behavior and the environment, a RL model is used to estimate the action in every step of the trajectory given to the agent.

3.1 Vehicle Behavior

Some of the vehicle parameters of the vehicle are presented in Table 2 with the dimensions of the scaled vehicle. The motion of the vehicle is represented by the modified bicycle model shown in Fig. 3. The vehicle kinematics are presented by the following Eq. [6]:

$$\dot{x} = v \cos(\psi + \beta) \quad (9)$$

$$\dot{y} = v \sin(\psi + \beta) \quad (10)$$

$$\dot{v} = a \quad (11)$$

$$\dot{\psi} = \frac{v}{l} \sin(\beta) \quad (12)$$

$$\beta = \tan^{-1} \left(\frac{1}{2} \tan \delta \right) \quad (13)$$

where (x, y) is the vehicle position; v is the forward speed; ψ is heading angle; a is the acceleration command; β is the slip angle at the center of gravity; and δ is the front wheel angle used as a steering command. Its state is propagated depending on

Table 2 Car parameters of the scaled vehicle

Parameter	Unit	Value
Mass	kg	1.34
Length	m	0.174
Width	m	0.0870
Wheel radius	m	0.0650

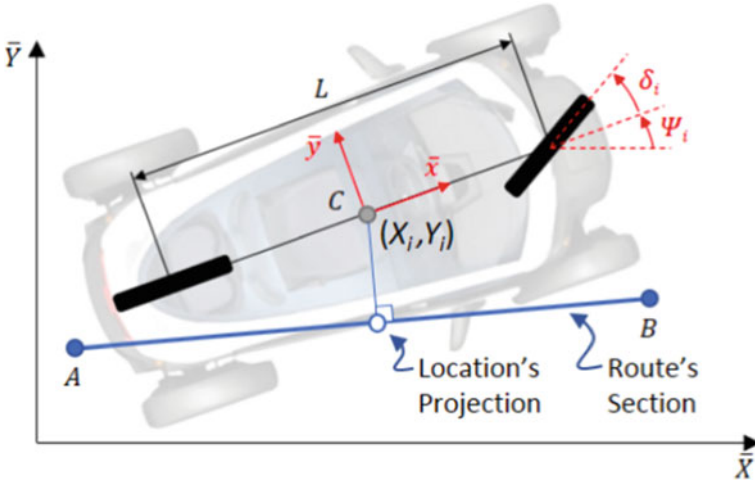


Fig. 3 Lateral vehicle dynamics. Image taken from [4]

the steering and acceleration actions. For the vehicle dynamics, the two degrees of freedom are represented by the vehicle lateral position y and the vehicle yaw angle Ψ . The vehicle lateral position is measured along the lateral axis of the vehicle to the point C which is the center of rotation of the vehicle. The vehicle yaw angle Ψ is measured with respect to the global X axis. The longitudinal velocity of the vehicle at the center of gravity is denoted by V_x . The Eq. (14) represents the lateral translational motion of the vehicle and Eq. (15) represents the moment balance about the z axis.

$$m(\ddot{y} + \dot{\Psi}V_x) = F_{yf} + F_{yr} \quad (14)$$

where F_{yf} and F_{yr} are lateral tire force of the front and rear wheels, respectively.

$$I_z \ddot{\Psi} = l_f F_{yf} - l_r F_{yr} \quad (15)$$

where l_f and l_r are the distances of the front tire and the rear tire respectively from the center of gravity of the vehicle [8].

The controlled vehicle is a low-level controller, allowing to track a given target speed and follow a target lane. The longitudinal controller is a simple proportional controller as shown in the Eq. (16).

$$a = K_p (v_r - v) \quad (16)$$

The lateral controller is a simple proportional-derivative controller, combined with some non-linearities that invert those of the kinematics model. The position and heading control are shown in Eqs. (18)–(21), respectively.

$$v_{lat,r} = -K_{p,lat} \Delta_{lat} \quad (17)$$

$$\Delta\psi_r = \arcsin\left(\frac{v_{lat,r}}{v}\right) \quad (18)$$

where Δ_{lat} is the lateral position of the vehicle with respect to the lane center-line; $v_{lat,r}$ is the lateral velocity command and $\Delta\psi_r$ is a heading variation to apply the lateral velocity command.

$$\psi_r = \psi_L + \Delta\psi_r \quad (19)$$

$$\dot{\psi}_r = K_{p,\psi} (\psi_r - \psi) \quad (20)$$

$$\delta = \arcsin\left(\frac{1}{2} \frac{l}{v} \dot{\psi}_r\right) \quad (21)$$

where ψ_L is the lane heading (at some lookahead position to anticipate turns); ψ_r is the target heading to follow the lane heading and position; $\dot{\psi}_r$ is the yaw rate command; δ is the front wheels angle control; and $K_{p,lat}$ and $K_{p,\psi}$ are the position and heading control gains.

3.2 Environment Setup

The Fig. 4 shows the racetrack that has been used to train the model. Several factors are considered to build the environment, such as two lanes, straight segments, pronounced curves and obstacles in the road, which increase the complexity of the vehicle performance.

Obstacles The ego-vehicle, in green, is surrounded by other vehicles that have speed zero, which can be considered as objects or obstacles for this scenario to decrease the implementation complexity but for future works it is expected to threat as vehicles with different speeds.

Road The dimensions of the track are constrained by the experimental room in GIPSA-Lab with an available space of 4 m × 4 m. The maximum distance in the horizontal axis is 3.6 and 2.8 m in the vertical one. Finally, the track has been built with the union of 11 segments as union of straight lines and segments of different radius circles, the radius of each segment is an important characteristic that is included in the RL model and will be explained in the next section.

Training Procedure The components mentioned in the previous section enter a deep neural network which will estimate the action of the car in every step of the trajectory and can be trained from a Stable Baselines3 (SB3) [github library](#) that is a set of reliable implementations of reinforcement learning algorithms in PyTorch.

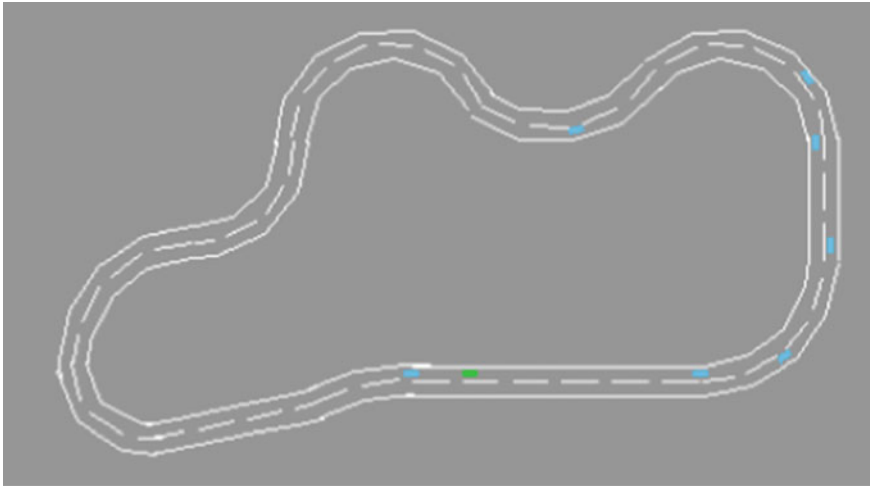


Fig. 4 Racetrack based on the highway-env library

Table 3 Parameters for training

Parameter	Value
Learning rate	0.0005
Discount factor	0.8
Exploration fraction	0.8
Total timesteps	60.000

This RL model is considered as an episodic domain that may terminate after a fixed number of time steps, or when an agent reaches a specified goal state. Also, the implemented policy is a Multilayer Perceptron (MLP) that consist of biased neurons arranged in layers, connected by weighted connections. 2 layers of 64 nodes have been used. Their effectiveness depends on finding the optimal weights and biases that reduce the classification error [9]. Some parameters of the training are displayed in the Table 3.

3.3 Training Results

After the combination of the vehicle behavior, the environment and the RL components, the training model has been performed on a Google Colab service that requires no setup to use, while providing access free of charge to computing resources including GPUs. The total training time is 5.5 h and the model output is obtained as a .zip extension for later use in the experimental results.

After training, the results can be reproduced for one episode by the states, observations, actions and rewards at every step. The duration of one episode is 140 s and

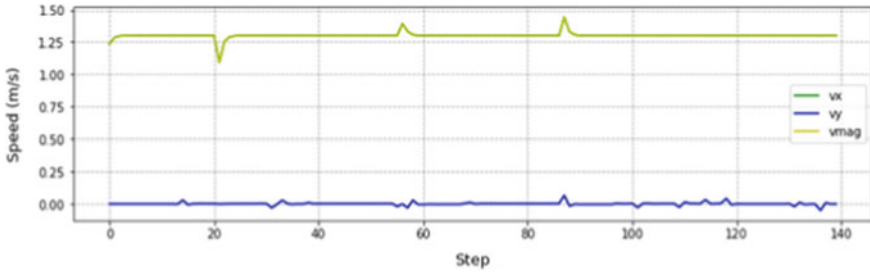


Fig. 5 Speed behavior of the vehicle in one episode

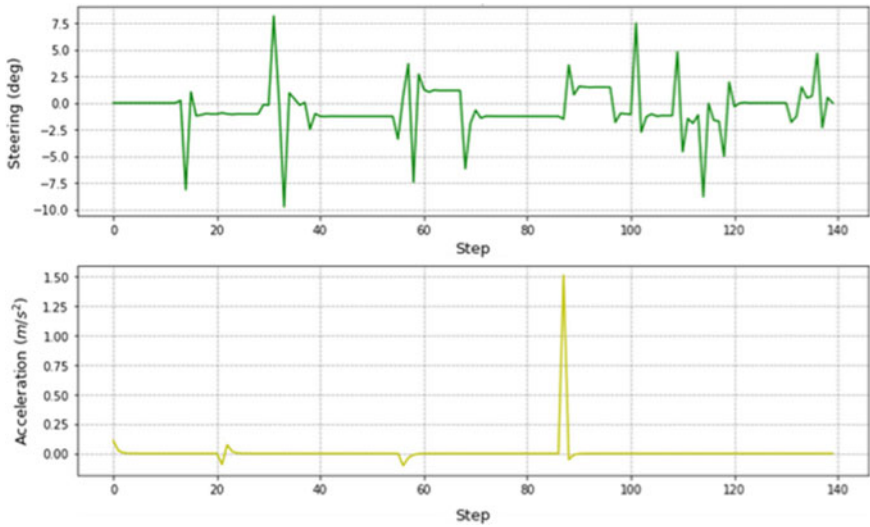


Fig. 6 Control variables of the vehicle in one episode

the frequency is of 2 actions per second. On the other hand, about the behavior of the vehicle, the available speed values are [0 , 0.26, 0.52, 0.78, 1.04, 1.3] m/s, and the speed profile is presented in Fig. 5 for the magnitude, lateral and longitudinal values. Here the lateral speed is lower than 0.1 m/s and the magnitude is very similar to the longitudinal speed. The vehicle tried to complete a loop closer to the maximum available speed, here the functionality of the high speed reward is shown.

Additionally, the control variables are the steering angle and the acceleration. The steering angle that is limited between $[-15 , 15]^{\circ}$. This limit affected the model and restricted the vehicle to behave more conservatively. The longitudinal acceleration is also restricted between $[-2, 2]$ m/s² to avoid aggressive speed changes. The results are shown in Fig. 6.

Lastly, the actions and the output of the RL model is shown in Fig. 7. The predominant action is to accelerate but it also changes lane when facing obstacles and

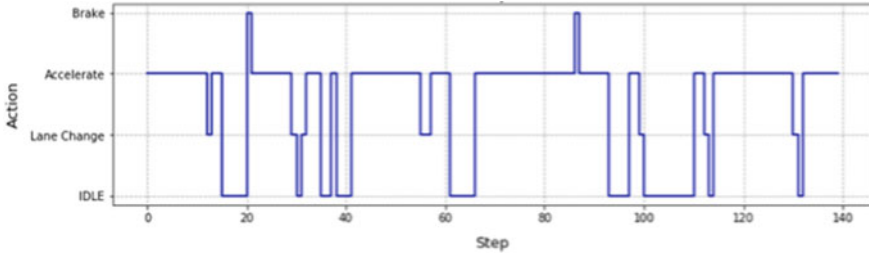


Fig. 7 Actions taken by the agent in one episode

in some pronounced curves. Because the behavior is predominant by the controllers, if the action would be to accelerate and if the speed limit is reached, the action can be ‘ignored’ and the vehicle kept an IDLE action, which for future works it would be preferable for the vehicle to take the available actions and not all of them.

4 Experimental Validation

A new environment is created with specific dimensions within the space of the experimental room, likewise, applied on the scaled vehicle in GIPSA-Lab. Figure 8 shows the experimental scenario where the validation of the results have been carried out, a two-lanes track is displayed as a reference. Table 4 shows the sensors and actuators of the vehicle enumerated in Fig. 9. In addition to the car components, there are high resolution cameras from which the position of the vehicle is measured with high accuracy. This sensor information of the road and the car is required to build the ROS2 architecture, that includes the perception, planning, decision making, and control nodes.

In the ROS2 decision making node, the observations have been programmed with the WiFi communication between the sensors and the computer. The RL model is trained in a track with a higher complexity than the one tested in this experiment. The decision making node includes the file model.zip containing the model, in which

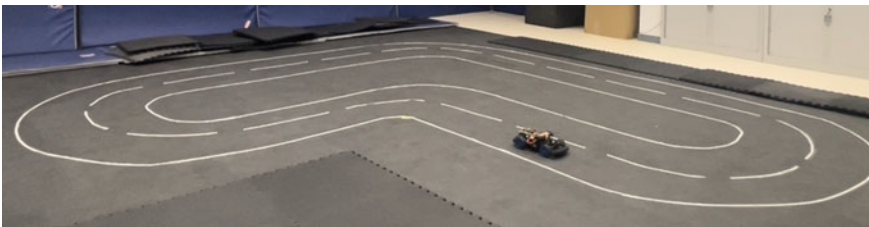


Fig. 8 Experimental scenario setup at GIPSA-Lab

Table 4 RC car components

N	Type	Functionality
1	Switch	Switching car on-off
2	8 mm qualisys super-spherical	Captured by vicon tracker
3	Arduino RP 2040	Micro-controller of the vehicle
4	Spur gears	Increase torque given by BLDC
5	Elastic wheel	2 Rear wheel of the vehicle
6	ACCU NI-MH 3000	Supply power battery
7	MG996R servo motor	Steering actuator
8	Elastic wheel	2 Front wheels of the vehicle
9	BLDC-A2212/13T	Throttle actuator

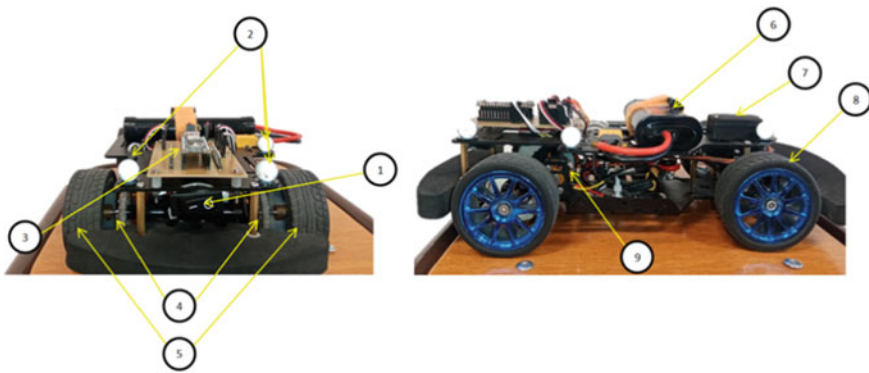


Fig. 9 Front and side pictures of the car

given the observation of the environment to predict the next discrete action to perform. Several scenarios have been tested with different obstacle positions. The first scenario with no obstacles is shown in Fig. 10, the vehicle can maintain the lane where it started. In this figure two loops are displayed with a different starting point of the vehicle in a direction counter clockwise.

The second scenario with one obstacle, the vehicle changes the lane before the curvature of the obstacle to avoid the crash, as shown in Fig. 11. The obstacle is displayed as the black square. The lane change is occurred between 4000 and 4200 ms, shown in orange in the figure.

The third scenario includes two obstacles, see Fig. 12, in which the vehicle changes the lane in advance to avoid collision. The times where the change lane action is performed are between 2000–2200 and 3200–3500 ms. One interesting observation is when the obstacle is located at the end of the curvature, the vehicle has the tendency

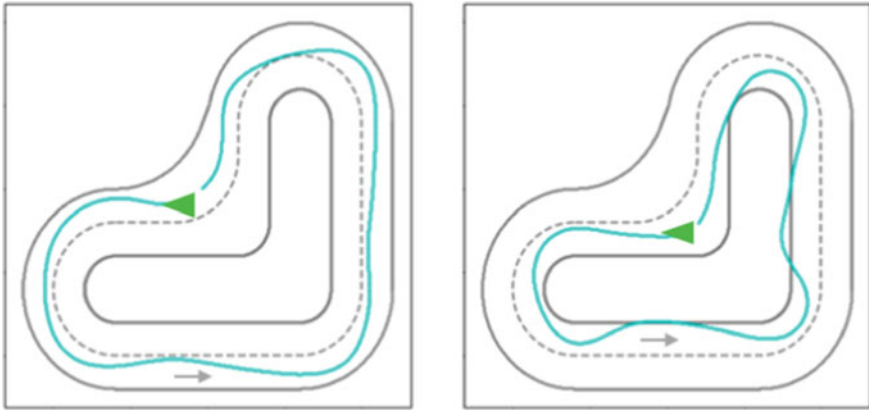


Fig. 10 Experimental results with no obstacles

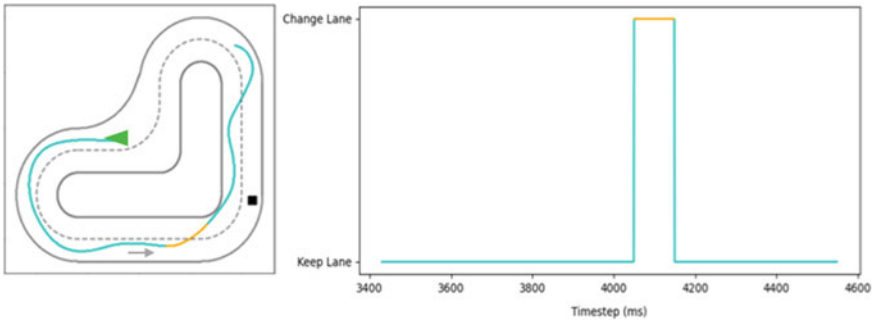


Fig. 11 Experimental results with one obstacle

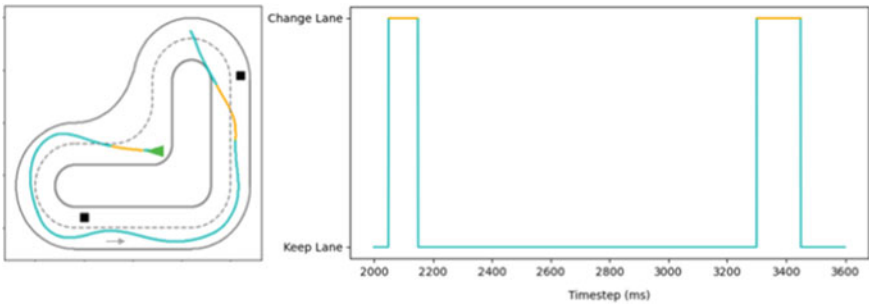


Fig. 12 Experimental results with two obstacles

to change lane in advance earlier than when the obstacle is on the straight path. This behavior has been observed in more scenarios that might be explained after the inclusion of the curvature as one attribute of the state in the training.

5 Conclusion

The performance validation of the RL model has been presented in simulation and in experimental tests. The results showed that scaled vehicle avoided obstacles, achieved high driving efficiency by taking an optimal path and executing maneuvers without veering off the track while maintaining the center-line of the two lane racetrack. Some remarks about the training model are improvement of the results obtained by including the curvature of the next segment in the track and also the influence of the rewards affect drastically the results, here one solution has been presented but infinite options could be implemented.

Finally, there are many scopes for improvement, such as modify the available actions according to the vehicle state so it can choose the feasible actions or it also would be interesting to try to train the model in the experimental scenarios and include some physical constrains that are not considered or neglected in the vehicle's behavior. Also, a longitudinal controller can be implemented to create a speed reference. Some improvements of the low-level controllers can be made to have straighter paths on the track and also while training the model it would be better to include large heading errors for sharp curvatures and replicate this for other tracks.

References

1. Emuna R, Borowsky A, Biess A (2020) Deep reinforcement learning for human-like driving policies in collision avoidance tasks of self-driving cars. [arxiv:2006.04218](https://arxiv.org/abs/2006.04218)
2. Kiran BR, Sobh I, Talpaert V, Mannion P, Sallab S, Yogamani K, Pérez P (2020) Deep reinforcement learning for autonomous driving: a survey. [arxiv:2002.00444](https://arxiv.org/abs/2002.00444)
3. Li D, Zhao D, Zhang Q, Chen Y (2018) Reinforcement learning and deep learning based lateral control for autonomous driving. [arxiv:1810.12778](https://arxiv.org/abs/1810.12778)
4. Matute J, Marcano M, Diaz S, Pérez J (2019) Experimental validation of a kinematic bicycle model predictive control with lateral acceleration consideration. 52:07. <https://doi.org/10.1016/j.ifacol.2019.08.085>
5. Naveed KB, Qiao Z, Dolan JM (2020) Trajectory planning for autonomous vehicles using hierarchical reinforcement learning. [arxiv:2011.04752](https://arxiv.org/abs/2011.04752)
6. Polack P, Althé F, d'Andréa Novel B, de La Fortelle A (2017) The kinematic bicycle model: a consistent model for planning feasible trajectories for autonomous vehicles. In: 2017 IEEE intelligent vehicles symposium (IV), pp 812–818. <https://doi.org/10.1109/IVS.2017.7995816>
7. Puterman ML (2014) Markov decision processes: discrete stochastic dynamic programming. Wiley
8. Rajamani R (2006) Vehicle dynamics and control. ISBN 0-387-26396-9. <https://doi.org/10.1007/0-387-28823-6>
9. Rojas MG, Olivera AC, Vidal PJ (2022) Optimising multilayer perceptron weights and biases through a cellular genetic algorithm for medical data classification. Array 14:100173. ISSN 2590-0056. <https://doi.org/10.1016/j.array.2022.100173>. URL <https://www.sciencedirect.com/science/article/pii/S2590005622000339>
10. Singh S (2015) Critical reasons for crashes investigated in the national motor vehicle crash causation survey
11. Stang M, Grimm D, Gaiser M, Sax E (2020) Evaluation of deep reinforcement learning algorithms for autonomous driving. In: 2020 IEEE intelligent vehicles symposium (IV), pp 1576–1582. <https://doi.org/10.1109/IV47402.2020.9304792>

12. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction. a bradford book. Cambridge, MA, USA, p 0262039249
13. Szoke L, Aradi S, Becsi T, Gaspar P (2020) Vehicle control in highway traffic by using reinforcement learning and microscopic traffic simulation. In: 2020 IEEE 18th international symposium on intelligent systems and informatics (SISY), pp 21–26. <https://doi.org/10.1109/SISY50555.2020.9217076>
14. Vu T-D (2009) Vehicle perception: localization, mapping with detection, classification and tracking of moving objects. Theses, Institut National Polytechnique de Grenoble—INPG. URL <https://tel.archives-ouvertes.fr/tel-00454238>
15. Watkins C, Dayan P (1992) Technical note: Q-learning. Mach Learn 8:279–292. <https://doi.org/10.1007/BF00992698>
16. Wiering M, van Otterlo M (2014) Reinforcement learning: state-of-the-art. Springer Publishing Company, Incorporated, p 364244685X
17. William F, Milliken D-LM (1995) Race car vehicle dynamics. Society of Automotive Engineers. Warrendale, Pa