Proceedings of the 41st Chinese Control Conference
July 25-27, 2022, Hefei, China

Application of Reinforcement Learning in the Autonomous Driving Platform of the DeepRacer

Wenjie Zhu[1], Haikuo Du[2], Moyan Zhu[1], Yanbo Liu[3] *, Chaoting Lin[4], Shaobo Wang[5], Weiqi Sun[6], Huaming Yan[7]

1. Department of Electrical & Automation of SEIEE, Shanghai Jiaotong University, Shanghai, 200240
E-mail: zhuwenjie0406@sjtu.edu.cn, E-mail: 13401839932@sjtu.edu.cn

2.Department of Automation of SEIEE, Shanghai Jiaotong University, Shanghai, 200240
E-mail: duhaikuo1013@sjtu.edu.cn

3.Teaching development and Student Innovation Center of SEIEE, Shanghai Jiao Tong University, Shanghai, 200240
E-mail: liuyanbo1205@sjtu.edu.cn

4.AWS BD Consultant, Amazon Web Service Inc, Beijing, 100015
E-mail: lincarol@amazon.com

5. Chinese Electronics Technology Group 54Th Institute, Shi Jiazhuang, 050081
E-mail: 337978737@qq.com

6.Xin Dong Interactive Entertainment Company Ltd, Shanghai, 200070
E-mail: sunweiqi7777@gmail.com

7.Summer Son Smart Technology Company Ltd, Shanghai, 200240
E-mail: q980013017@gmail.com

Abstract: This article revolves around autonomous driving, mainly introducing the autonomous driving cloud platform based on the reinforcement learning to improve the autonomous driving of the car on the DeepRacer using the AWS (Amazon Web Service). Applying the sample codes provided by the DeepRacer platform, the training and completion of the car requires a long time. Therefore, we applied path planning into DeepRacer based on reinforcement learning. Several breakthroughs have been shown as follows: The formulation and solution of RL are completed on DeepRacer. The model of vehicle is simplified as the bicycle and thus reality gap between perception and joints was narrowed. A novel system framework VNARM(Vehicle Network Autonomous Racing Model) is introduced. Reward functions were set properly for tracing in RL. The vehicle's performance of finishing one lap is increased from nearly 30 seconds to less than 9 seconds, while maintaining a high percentage of completion.
Key Words: Autonomous driving, Amazon's cloud platform, Reinforcement learning, Sarsa, Action space, Path planning

1 Introduction

The demand for automatic means of transportation in industrial production process, transportation and service industry becomes higher and higher with the development of social modernization. To meet these needs, the social environment has provided good basic conditions for the development of automatic driving.

More and more researches emerge based on autonomous driving. Reinforcement learning (RL) is used in *Constrained Policy Optimization Algorithm for Autonomous Driving via Reinforcement Learning*[1], which explored an algorithm followed by policy optimization and Markov modeling. *Training Method of Automatic Driving Strategy Based on Deep Reinforcement Learning*[2] and *Autonomous Driving system based on Deep Q learning*[3] adopted the Q-learning method under RL. To handle the controlling and driving limits of the autonomous racecar in a worldwide competition, Module Predictive Control (MPC) for path planning is adopted in *Dynamic Path Planning for Fomula Autonomous Racing Car*[4] based on the data collected by the sensors of the vehicle; In *The Multi-sensor Fusion Automatic Driving Test Scene Algorithm based on Cloud Platform*[5], the model training speed and development is improved through data collection, processing and analysis on Cloud Platform. How the model is created and trained through RL on the DeepRacer[6] is exhibited in *Easy Learning of Reinforcement Learning with a Gamified Tool*[7]. Combined with RL, raw Cameras are applied for observation and end-to-end policy[8] is used for path planning on the DeepRacer [9].

Since the RL solution can give the state input at any time, and it is more flexible in the training of model, in order to complete the vehicle's racing on the Amazon's cloud platform for autonomous driving, this article aims to propose a relatively simple and feasible path planning method, combined with RL, to gradually improve the completion of the vehicle and its performance on the DeepRacer. The remaining content of the paper is divided into four sections, of which the second part mainly introduces the architecture of the Amazon cloud platform and its deep learning framework; the third part gives a brief introduction to the characteristics of the track and the car; The fourth part puts forward two strategies in path planning; the fifth and sixth parts combine RL with path planning to realize the autonomous driving of the car, and analyze the performance, completion rate and path of the vehicle under four policies; the seventh part is the conclusion.

*This work is supported in part by Cooperative education program of the Ministry of Education (202101001039, 202002110014) and in part by the National Natural Science Foundation of China under Grant Nos. 61873163.

2 'Cloud+terminal' architecture on AWS

Amazon Web Services[10] is the most comprehensive and widely used cloud platform in the world. Compared with Alibaba Cloud and Huawei Cloud that we are familiar with, from infrastructure technologies such as computing, storage, and databases to emerging technologies such as machine learning, artificial intelligence, data lakes and analytics, the services provided by AWS are much more than any other cloud platforms.

2.1 The workflow of 'cloud+terminal' structure in autonomous driving of AWS

The 'Cloud + terminal' structure is divided into two parts: cloud platform and the terminal representing a vehicle. The terminal is responsible for transmitting the data of the vehicle to the cloud platform via LIDAR, on-board camera and speed and acceleration sensors through wireless WiFi; the cloud platform is responsible for processing the data uploaded by the terminal and quickly set up a virtual test and the training environment. In Fig. 1, firstly, the real-time data including the speed, steering angle and its position etc. will be extracted from the terminal through AWS Outposts, and then uploaded to the data lake, where the data is processed and uploaded to the Amazon cloud platform for labeling, map development, modeling and simulation. Finally these results on cloud platform will be re-uploaded to the data lake for arrangement and deployment.

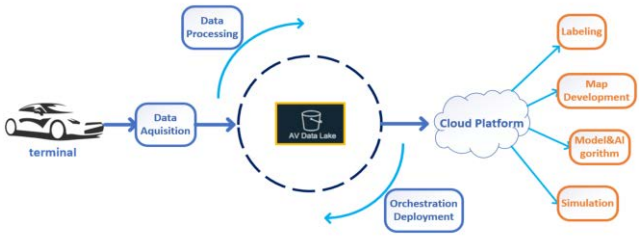


Fig. 1: Amazon's 'cloud + terminal' architecture

2.2 Framework of machine learning for Amazon Web Services' cloud platform

AWS has a large storage and computing capacity and supports ML frameworks such as Apache MXNet, TensorFlow, and PyTorch, which can speed up algorithm training and testing. AWS DeepRacer is an autonomous driving platform based on the AWS 'cloud + terminal' architecture. Users can solve a series of problems in autonomous driving through models, and train, test, and evaluate RL on a racing simulator. The framework is different from ones introduced above, since it uses the ML frameworks of Robomaker[11] and Sagemaker[12].

2.2.1 Robomaker framework

With Robomaker, the Robot Operating System (ROS) which runs the robot can easily navigate, communicate, understand and learn. The advantage of AWS Robomaker is that it is a fully managed service that allows developers to create simulation worlds and conduct simulation training. After developing and testing the application, users can manage the robot more conveniently through the deployment function of Robomaker application in the AWS cloud service, monitoring the status of the robot, and obtaining the performance of the robot.

2.2.2 Process in Sagemaker

AWS Sagemaker integrates the extensive feature library built for ML to help developers build and deploy machine learning models faster. The AWS Sagemaker process is divided into four parts: preparation, construction, training and tuning, deployment and management. Among them, during the preparation stage Sagemaker mainly marks the training data for ML while in the construction stage algorithms are written and models for ML are established on Jupyter Notebook. AWS Sagemaker can capture real-time metrics during training Users can use the various algorithms provided by AWS Sagemaker to deploy training models through Amazon Elastic Compute Cloud (Amazon EC2), and start Sagemaker terminal nodes to host the model.

2.2.3 The architecture of AWS DeepRacer

The AWS DeepRacer service is based on AWS Sagemaker, Amazon Robomaker and other cloud services similar to Amazon S3[13]. It mainly uses Sagemaker based on Robomaker to build and train ML models. In Fig. 2, AWS RoboMaker creates a simulation environment for the agent to drive along a specified track. The agent moves according to the strategy network model trained in SageMaker for a certain period of time. Every run starts from the starting line to the end. The end can be the finish line or off the track, which is called an episode. For each episode, the lesson is divided into a fixed number of steps. In each fragment, the experience is defined as an ordered list of tuples(state, action, reward) associated with each step, which is cached in Redis[14]. After that, it stores the updated model in Amazon S3 for use by SageMaker, and this cycle continues until the training stops.

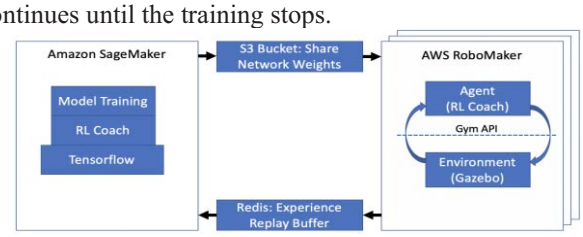


Fig. 2: The architecture of AWS DeepRacer

2.2.4 Overall framework

The camera and lidar is for detection. The detected data and state of vehicle is sent to AWS cloud platform and then Sarsa is applied. Wheels, electronic steering and brake are controlled by the tracking control system, which has just received data from the cloud platform shown in Fig.3.

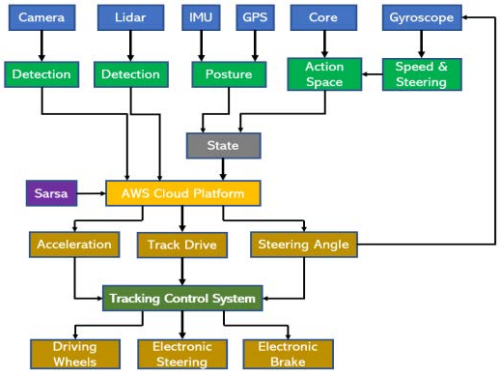


Fig. 3: VNARM system framework

3 Characteristics of the track and the vehicle

3.1 Introduction to 2018 wide track

The 2018 wide track is a relatively simple one among all the tracks in Fig.4 and Fig.5. There are total 120 waypoints in the track, in which a hairpin (23 to 42), two long straights (0 to 18 and 110 to 120, 89 to 110) and three short straights (72 to 80, 56 to 66, 43 to 51) are included. All these sections of the track are connected by several curves. The biggest difficulty of the whole track is the hairpin with a 180° turning angle. If the speed of the car is too fast or the steering angle is not adjusted properly, the probability of running out of track is very high, resulting in low completion percentage and poor performance of the car. It is also the most difficult point in the process of path planning.

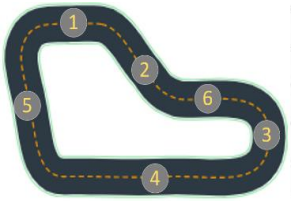


Fig. 4: The sketch of the track

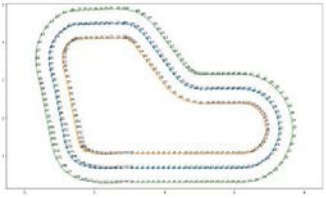


Fig. 5: Waypoints in the track

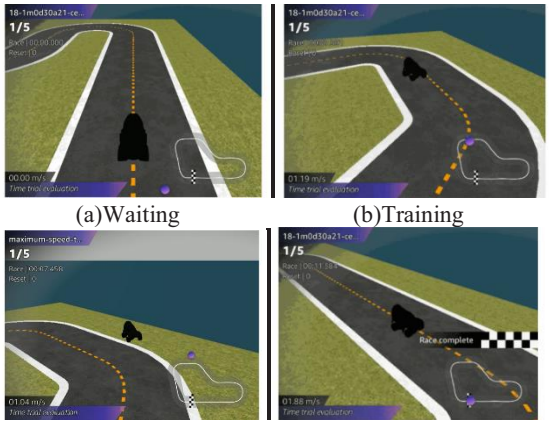
3.1.1 Related parameters

The commonly used parameters of the track are shown in Table 1. Waypoints can be used to segment a specific track; [x,y] represents the current position of the car in the track coordinate system, which can be used to judge the distance between the vehicle and the expected path; Steering_angle can prevent the steering angle of the vehicle from being too large or too small to cause a deviation from the track.

Table 1. Related parameters

Parametres	Explanation
closest_waypoints	The nearest waypoints
[x,y]	Agent's x/y-coordinate in meters
waypoints	List of (x,y) as milestones along the track center
track_width	Width of the track
distance_from_centre	Distance in meters from the track center
is_left_of_centre	Flag to judge if agent is on left side to the track center
speed	Agent's speed in meters per second (m/s)
steering_angle	Steering angle of a vehicle,negative if the vehicle turns right

The simulation scene of the car training under the AWS cloud platform Simulation Video Stream is shown in Fig. 6. The vehicle is waiting for training in state (a); it's training in state(b); the vehicle gets out of track in state(c) and in state(d), it finished one lap.



(a)Waiting (b)Training

(c)Out of track (d)Finished

Fig. 6: States of the Vehicle

3.2 The vehicle in AWS DeepRacer

3.2.1 Model of AWS's vehicle

Fig. 7 shows the side view of the car. The camera is responsible for monitoring the track where the car is located. The dual core is connected to the cloud platform via WiFi wireless, and is responsible for feeding back data such as the action space of the vehicle to the cloud platform; the gyroscope below is responsible for controlling the acceleration and steering of the vehicle.

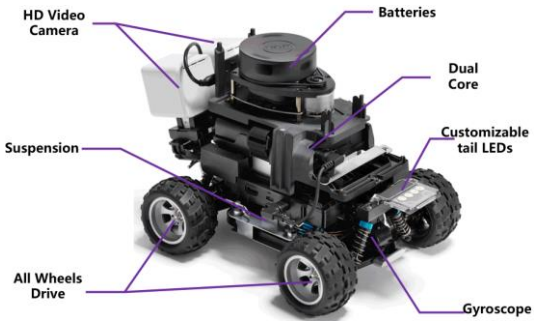


Fig. 7: Side view

3.2.2 Action space of the vehicle

In RL, all effective action sets when the agent interacts with the environment are called action spaces. The action space in AWS deepRacer includes speed and steering angle, and the binding of two parameters constitutes action spaces. The vehicle has an action a(v, θ) at any time, where v is the instantaneous speed of the vehicle, and θ is the steering angle. There are two kinds of action spaces, one of which is discrete, that is, the value of the agent's speed or steering angle is in a finite set. For DeepRacer, this means that in different states, the neural network selects the speed and direction of the vehicle based on the input of the its camera and lidar sensor. Suppose there are m elements arranged in arithmetic sequence in the sets of speed and n in steering angle, the combination of these speeds and steering angles can form a series of m*n actions.

The other kind of action space is continuous, which allows the vehicle to select the speed and steering angle in a continuous range.

The advantage of using a continuous action space is that the speed and steering angle in different states can be continuously changed, which ensures the smoothness of the

model in the action space. In a heavily trained model, the vehicle can choose the best speed and steering angle.

Although the continuous action space seems to be a good choice, it takes a lot of time for the vehicle to learn how to select the speed and steering angle through RL, so the discrete action space is adopted in this research. Also, choosing the discrete type can also better help us analyze the influence of the speed or steering angle on the path trajectory of the vehicle.

3.2.3 Model of the vehicle

Since the kinematics model of the racing car is more complicated and involves multiple degrees of freedom, it is necessary to establish a simplified model and thus narrow the gap between perception and joints. In this paper, the vehicle is simplified to a linear 2-DOF (Degrees Of Freedom) bicycle structure with the yaw and lateral motion, and the forces considered are concentrated on the front and rear wheels It has two degrees of freedom and is supported on the ground by two lateral elastic tires. Following is the simplification of the model.

1) The movement in the vertical direction is dismissed. The vehicle is simplified as a moving object on a two-dimensional plane, that is, the displacement of the vehicle on the z-axis, the pitch angle on the y-axis and the roll angle on the x-axis are omitted.

2) The steering angle is similar to that of the bicycle structure, and all wheels are assumed to have the same steering angle and speed.

3) Similar to a bicycle, suppose the front wheel of the car controls the steering angle.

The modeling of the vehicle is exhibited in the following figure. Formula (1) to (10) are deductions in the field of dynamics.

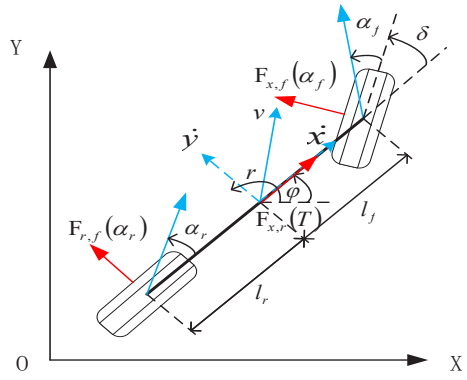


Fig. 8: Model of the vehicle

$$\dot{X} = \dot{x}\cos\varphi - \dot{y}\sin\varphi \tag{1}$$

$$\dot{Y} = \dot{x}\sin\varphi + \dot{y}\cos\varphi \tag{2}$$

$$\ddot{\phi} I_z = -F_{y,r} l_r + F_{y,f} l_f \cos\delta \tag{3}$$

$$\ddot{x} = \dot{y}\dot{\phi} + \frac{F_x}{m} - \frac{F_{y,f}\sin\delta}{m} \tag{4}$$

$$\ddot{y} = -\dot{x}\dot{\phi} + \frac{F_{y,r}}{m} + \frac{F_{y,f}\sin\delta}{m} \tag{5}$$

$$\alpha_f = \frac{F_{y,f}}{C_f} \tag{6}$$

$$\alpha_r = \frac{F_{y,r}}{C_r} \tag{7}$$

$$F_x = -C_{r0} - C_{r2}\dot{x}^2 + C_m T \tag{8}$$

$$\alpha_r = \tan^{-1}\left(\frac{\dot{y} - l_r r}{\dot{x}}\right) \tag{9}$$

$$\alpha_f = \tan^{-1}\left(\frac{\dot{y} + l_f r}{\dot{x}}\right) - \delta \tag{10}$$

(X, Y) is where the gravity of the vehicle is in the global coordinate system; $(\dot{x}, \dot{y})$ stands for the logitudinal and lateral speed; I_z is the rotor inertia; $F_{y,r}$ is the lateral force of the rear axle while $F_{y,f}$ is the longitudinal force of the front axle; l_r and l_f stand for the distance between the rear and front axles of the vehicle; α_f and α_r are the front and rear slip angles ; C_r and C_f are the lateral stiffness of the rear and front wheel; C_{r0} is the RR(rolling resistance), C_{r2} is the resistance coefficient and C_m stands for the maximum longitudinal force; T represents the throttle and r represents the yaw rate.[4]

4 The implementation of path planning

The example on the DeepRacer adopts the 'following the center line' strategy, which as it suggests, encourages the car to move along the centerline of the track. The closer the car is to the centerline of the track, the more rewards it will receive. The default car action parameters on the DeepRacer which is applied in the 'following the centerline strategy' are shown in Table 2.

Table 2. Initial action space

No.	1	2	3	4	5	6
Steering angle(°)	-30	-30	0	0	30	30
Speed(m/s)	0.5	1.0	0.5	1.0	0.5	1.0

The performance of the vehicle is shown in Table 3.

Table 3. The vehicle's performance

Trial	1	2	3	4	5
Performance	Fail	Fail	27.041s	30.087s	25.403s

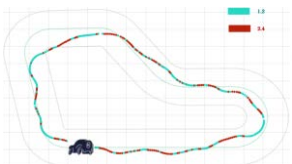


Fig. 9:The vehicle's path

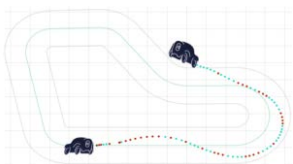


Fig. 10: Path under 20°

The path trajectory is exhibited in Fig. 9. It can be seen that since the speed limit is set lower, the finishing time is close to 30s; the steering angle is only divided into three parts , which makes the car inflexible in steering.

Due to the inflexibility in steering, the path is distorted and is not smooth enough, which will affect the performance of the car. The vehicle is more likely to get out of track when passing through curves. Additionally, the vehicle sometimes accelerates in the curves or decelerates in the straights in Fig. 10. In view of the shortcomings above, the path planning strategy is adopted and related parameters in the action space of the vehicle is adjusted.

4.1 Two strategies for path planning

Path planning is adopted by dividing all 120 waypoints in the 2018 wide track into several sections and approximating the path by using functions in Geogebra mathematical software.

Firstly, the vehicle is encouraged to turn at a larger radius in the hairpin (referring to Fig. 11). In the long straight section (H_1 to D), a linear function is used and those two short straights (E to F, F to G) are also fitted with a linear function. The long straight and a short curve (G to H_1) are fitted with a power function, and we allow the vehicle to pass through in a circular trajectory (D to E) in the hairpin.

Secondly, try another feasible path, that is, as shown in Fig. 12, the vehicle chooses to go through the hairpin in a shorter radius. Different from the previous path, this path is shorter, but considering that the speed of the car needs to be much more lowered when passing the hairpin, training and testing are needed on which path the car will perform better in.

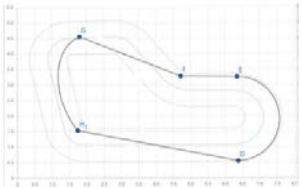


Fig. 11: Strategy 1

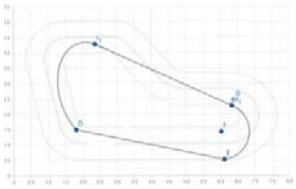


Fig. 12: Strategy 2

4.2 Selection of action space of the vehicle

4.2.1 Selection of the steering angle

After path planning, appropriate action space needs to be selected to improve the performance and completion of the car on the 2018 wide track. The maximum steering angle is adjusted to 20° and 30° respectively. After training and testing, we find that when the maximum steering angle of the vehicle is 30°, it is not easy for the vehicle to leave curves when it is turning. When maximum steering angle is 20°,it's easier for the vehicle to get out of track. It can be seen from Fig. 10 that when the car passes through a hairpin, the steering angle is not enough for the driving direction to be adjusted in time.

4.2.2 Selection of the maximum and types of speed

In the action space of a vehicle, speed also plays a decisive role in the performance and completion. The selection of the speed limit directly affects the time for the vehicle to complete a lap; and it also ensures that the car can pass at a suitable speed in different sections of the track. In Strategy 1, the vehicle is able to select two types of speed. It goes through straights at a higher speed while it passes at a lower speed through curves(including the hairpin). We try to set the limit of speed to 2.4m/s and 2.7m/s respectively, and get Policy 1 and 2. After training and testing, we compare the performance and track completion of these two policies.

For the second strategy, we deal with the hairpin in a specialized way. Since the vehicle needs to pass at a much lower speed in the hairpin—otherwise the vehicle will be more likely out of track. In Strategy 2, the vehicle is able to select three types of speed We encourage the vehicle to pass through straights in the fastest speed, go through the hairpin at the lowest speed and the curves at the moderate speed. In terms of the limit, we also try the limit of 2.7m/s and 2.4m/s and get Policy 3 and 4.

5 Autonomous driving based on reinforcement learning

RL is applied to explore the unknown environment while building a model and learning an optimal strategy. It has the following characteristics: no supervision data, only reward signals--not necessarily real-time.

In RL, no one tells you what to do under what state in advance. Instead, the agent learns only by reflecting whether its action in the previous state is correct. From this perspective, it can be considered that RL is supervised learning with time-delayed labeled information.

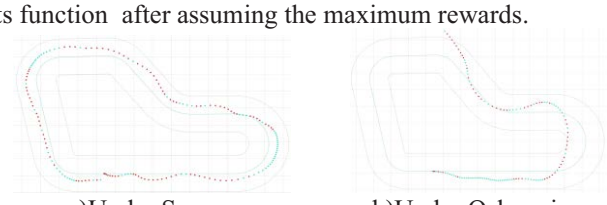
5.1 Autonomous driving combined with RL

All RL training processes are based on Markov Decision Process (MDP)[15]. In MDP, the next state depends on the current action and reward. For autonomous driving, RL functions as follow: the agent interacts with the environment, obtains reward, and adjusts its strategy in order to maximize its rewards, which makes it easier for training and testing. Therefore, we approximate the path by using functions, apply the method of RL in the training of the vehicle and optimize our policy according to training and test results.

Table 4. Key words in RL

Key words	Explanation
Agent	The training subject of RL
Environment	Background
State	Current state of agent and environement
Action	What the agent could act based on its state and reward
Reward	Feedback from the environment

We applied **Sarsa**, which appears to be an on-policy and epsilon-greedy strategy—the state of a vehicle is updated based on rewards after its action, into DeepRacer instead of the commonly-used Q-learning, which otherwise, updates its function after assuming the maximum rewards.



a)Under Sarsa b)Under Q-learning

Fig. 13: Comparison between Sarsa and Q-learning

Compared with Q-learning, which encourages the agent to be adventurous, without consideration of costs, Sarsa, whose strategy is consistent with actual behaviour, expects the vehicle to perform better in racing without getting out of the track.

In this research, at each step ***t***, the vehicle chooses action a_t based on state s_t and reward r_t. After a_t is taken, s_t and r_t are updated to s_{t+1} and r_{t+1}. At the beginning, the action of the vehicle is completely random. The vehicle needs to continuously adjust its action in order to obtain more rewards until it is close to the user's expectation of it.

Following fomulae are the expressions of how RL works in the autonomous driving by applying Bellman Equation. Fomula (11) and (13) reveals the relationship between the state value function $R_\pi(s)$ and state-action value function $Q_\pi(s,a)$, in which $R_\pi(s)$ is the expectation of total rewards the vehicle gets after taking action in state **s** according to the strategy π. What differs $Q_\pi(s,a)$ from $R_\pi(s)$ is that action **a** is taken before the vehicle takes other actions based on π. $\pi(a \mid s)$ means the possibility of taking action ***a*** in state **s**. Formula (13) is the definition of state-action value function discounted by a factor $\gamma \in [0,1]$, in which $P^a_{s->s'}$ represents the probability of shifting from **s** to **s'** after action **a** while K^a_s is the instant reward in state **s**.

Then put formula (13) into (11) and get the integration of state value function (15). Similarly, put (12) into (13) and get the iteration of state-action value function (16).

$$R_\pi(s) = \sum_{a \in A} \pi(a \mid s) Q_\pi(s,a) \tag{11}$$

$$R_\pi(s') = \sum_{a \in A} \pi(a' \mid s') Q_\pi(s',a') \tag{12}$$

$$Q_\pi(s,a) = \sum_{s' \in S} P^a_{s->s'}(K + \gamma R_\pi(s'))$$

$$= K^a_s + \gamma \sum_{s' \in S} P^a_{s->s'} R_\pi(s') \tag{13}$$

$$Q_\pi(s',a') = K^{a'}_{s'} + \gamma \sum_{s'' \in S} P^a_{s'->s''} R_\pi(s'') \tag{14}$$

$$R_\pi(s) = \sum_{a \in A} \pi(a \mid s)(K^a_s + \gamma \sum_{s' \in S} P^a_{s->s'} R_\pi(s')) \tag{15}$$

$$Q_\pi(s,a) = K^a_s + \gamma \sum_{s' \in S} P^a_{s->s'} \sum_{a \in A} \pi(a' \mid s') Q_\pi(s',a') \tag{16}$$

5.2 Setting of reward function

To encourage the vehicle to run faster and get a higher completion rate, it's necessary to set proper reward functions. We first try to prompt the vehicle to run as fast as possible. We set reward as follows: *reward=speed*speed* and find that it accelerates in straights and decelerates in curves. However, without the projected route for racing, the vehicle tends to deviate and get out of the track.

To improve the vehicle's performance through RL, we set rewards to encourage the car to follow the path we envisioned. At the same time, considering the impact of the speed and steering angle of the vehicle on its performance and the completion percentage, we set additional rewards based on speed and steering angle. The total reward is added by these three rewards in the same weight.

5.2.1 Rewards based on routing

This reward is mainly based on the deviation between the actual position of the vehicle and the ideal position that we expected.

According to the path planning, the 120 waypoints are divided into several parts. For any waypoint, the difference between the actual position of the vehicle and the ideal position is calculated, and different rewards are given. The closer the actual position of the vehicle is to the ideal position, the more rewards it will receive.

It is worth noting that, in order to try to make the vehicle close to the inner or outer boundary through the hairpin, the vehicle can be appropriately encouraged to move along the left or right line of the track.

Algorithm 1:Reward based on routes

1: Divide *closest_waypoints[i]* into several sections based on path planning

2: For each section, fit the path with a compatible function, ***Set:*** $y' = \Gamma(x')$

In which $[x', y']$ represents the desired location of vehicle

3: **for** i∈{0,1,2,…,119}

a)Compute the distance between actual points $[x, y]$ and desired points $[x', y']$

b)Set rewards α_i for different distances respectively

c)As for the hairpin in Policy 1 and 2 / 3 and 4, if *Is_left_of_center* equals *FALSE/TRUE* & *distance_from_center*

overrides 0.4**track_width*, Give the extra reward as α_{extra}

d)Compute total rewards α for each *closest_waypoints[i]*

end for

5.2.2 Rewards based on speed

This reward is based on whether the vehicle should accelerate or decelerate and certain rewards will be given depending on how the vehicle performed. To encourage acceleration in straights, we give more rewards for higher speed while the vehicle gains more if it decelerates in curves.

Algorithm 2:Reward based on speed

1: Divide *closest_waypoints[i]* based on types of routes

Type A: fastest Type B: intermediate Type C: slowest

2: **for** i∈{0,1,2,…,119}

if *speed* equals what we encouraged

Set rewards β *for closest_waypoints[i]* in different routes

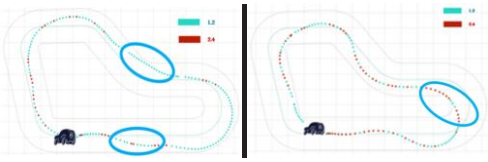
else

β /=2

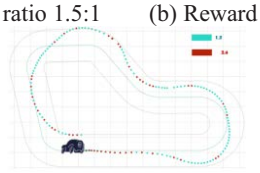
end if

end for

The setting of the ratio of reward is also a point worth paying attention to. For example, suppose that the rewards vehicle gets by passing through straights at the speed of 2.4m/s are 1.5, 2 or 3 times than 1.2m/s; Similarly, in curves, the vehicle gets more rewards when passing by at a lower speed. After training and testing, the speed and track trajectory of the vehicle at each waypoint is shown in Fig. 14.



(a) Reward ratio 1.5:1 (b) Reward ratio 3:1



(c) Reward ratio 2:1

Fig. 14: Paths under different reward ratios

When rewards are set to 1.5:1, since the rewards for the acceleration of the car in straights are not enough, most of the waypoints in straights are passed by at a lower speed of 1.2m/s, which leads to the poorer performance of the vehicle. When the ratio is set to 3:1, the proportion of waypoints where the vehicle accelerates is larger, and thus the probability of running out of track becomes higher.

5.2.3 Rewards based on steering angle

The last reward is based on whether the vehicle should steer Similar to the second reward, the track is divided into type A (straights) and B (curves) in which we encourage the steering angle to be 0° in curves and to be 30° in straights. We also give rewards ω if the vehicle performs as we expected.

5.2.4 The evaluation and improvement of strategies

After we have set the reward function based on the algorithm 1 and 2, we evaluate and improve our strategy based on RL, where $R(s)$ represents the previous rewards that the vehicle gains.

Algorithm 3:Policy evaluation and improvement

1: **Initialization:**

$R(s) \in \mathbb{R}$ and $\pi(s) \in \mathbb{A}$ (set of all action spaces)

2: **Policy evaluation**

Repeat

$\Delta \leftarrow 0$

for $s \in \mathbb{S}$:

for i∈{0,1,2,…,119}:

$K_s^i \leftarrow \alpha_{i,s} + \alpha_{extra}^{i,s} + \beta_{i,s} + \omega_{i,s}$

$r_i \leftarrow R_i(s)$

Update

$$R_i(s) \leftarrow \sum_{a_i \in A}^{i} \pi_i(a_i \mid s) \bullet (K_s^i + \gamma \sum_{s' \in S} P_{s->s'}^{a_i} R_{\pi_i}(s'))$$

$$\Delta \leftarrow \max(\Delta, \mid r_i - R_i(s) \mid)$$

end for

end for

until $\Delta < \varepsilon$ (a small positive value that is set initially)

3: **Policy improvement**

for $s \in \mathbb{S}$:

for i∈{0,1,2,…,119}:

$$Q_{\pi_i}(s, a_i) \leftarrow K_s^i + \gamma \sum_{s' \in S} P_{s->s'}^{a_i} R_{\pi_i}(s')$$

$$\mu \leftarrow \pi_i(s)$$

Update $\pi_i(s) \leftarrow \arg\max_{a_i} (K_s^i + \gamma \sum_{s' \in S} P_{s->s'}^{a_i} R_{\pi_i}(s'))$

if $\mu \neq \pi_i(s)$ return to Policy evaluation

else the policy is stable, we get the best strategy π_i for each *closest_waypoints[i]*

end for

end for

6 Analysis of results of training and evaluation

6.1 Extent of completion during training

The completion percentage of the vehicle under the four policies during training is indicated in Fig. 15. The horizontal axis is the number of iterations during training, and the vertical axis is the proportion of the track completion. After two hours of training, the vehicle has a high track completion percentage in the final test.

In Fig. 15, since the speed limit of Policy 1 is lower than Policy 2, the completion percentage of the former is a little higher than the latter, meaning that it is more stable and conservative than Policy 2; Still, Policy 4 has a slightly higher track completion percentage than Policy 3. It is worth noting that under the four policies, the second has the lowest percentage of completion. Observing the test results of the vehicle, we find that it is easy to run out of track in the hairpin when passing through near the outer boundary due to the higher speed limit(referring to Vehicle B in Fig. 16).

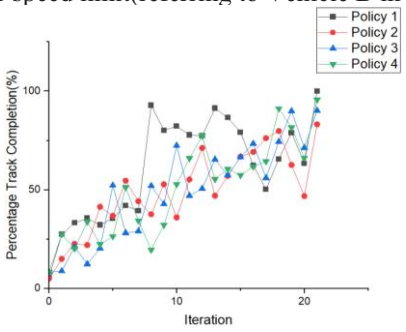


Fig. 15: Comparison of completion percentage

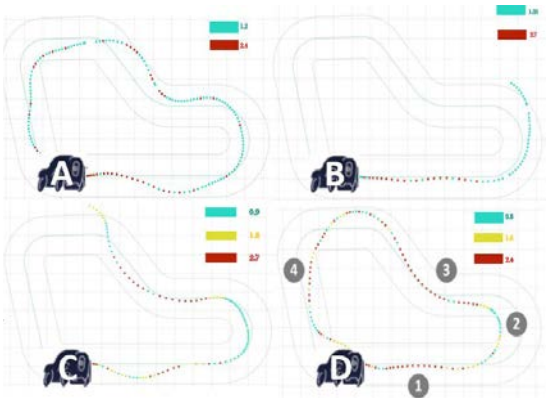
6.2 Analysis of the vehicle's performance and track trajectory

After training, we evaluated the performance under 4 policies. Table 6 shows that the vehicle completes 5 laps under all 4 policies. Policy 2 improves the performance of

the vehicle by increasing the limit of speed, however, the faster it passes through, the more likely it will run out of track. Therefore, in one of five tests, the lap was not completed. In contrast, Policy 1 is more conservative. Also, Vehicle A has more inclination to follow the centerline instead of passing through near the outer boundary in Fig. 16. Comparing with Policy 3 and 4, the vehicle needs to pass through the hairpin at a higher speed and the vehicle appropriately stays away from the outer boundary in case it runs out of track at a high speed, which leads to an extent of deviation from the expected routes in Fig. 11.

Table 5. Performance of the vehicle

Policy	1	2	3	4
Lap1	11.195s	Unfinished	11.023s	10.127s
Lap2	11.866s	10.670s	10.879s	9.834s
Lap3	12.136s	10.299s	Unfinished	9.535s
Lap4	10.803s	9.866s	10.536s	9.195s
Lap5	11.661s	9.984s	10.870s	9.398s



A: Policy 1 B: Policy 2 C: Policy 3 D: Policy 4
Fig. 16: Routes under 4 policies

In order to improve the performance of the vehicle and the completion percentage of the track, we encourage the vehicle to pass through the hairpin in a shorter radius. The speed limit was set to 2.7m/s in Policy 3. However, we find that the path of the vehicle passing through the hairpin (referring to Vehicle C) is somewhat different from what we expected (referring to Fig. 12). This is probably because the minimum speed is still relatively high, making it difficult to steer near the sideline, and thus the result is not even as good as Policy 1. Also, since the vehicle diverges from the routes what we expected, there is one out of five tests in which the vehicle failed to complete.

Then the speed limit was adjusted to 2.4m/s. The completion percentage is improved, which is close to Policy 1, and the overall results are satisfactory.

Vehicle D finished a lap with a shortest period of time(8.736s) under Policy 4. It travels at the fastest speed at most waypoints in Section 1 and Section 3, and passes at a medium or lowest speed when steering. The path trajectory is approximately close to the expected path (referring to Fig. 12). Compared with other policies, path planning in this policy looks more radical in the choice of steering. However, it ensures that the vehicle will not go out of bounds because of over speeding. Therefore, the vehicle can achieve the best results under this policy.

7 Conclusion

This article mainly explores the autonomous driving algorithm based on RL, relying on the AWS cloud platform to complete the virtual simulation test of the automatic driving of the car, and gradually improve the vehicle's performance and completion percentage through the following four steps of planning the path, establishing a suitable car model, setting the reward function and analyzing the log data. Compared with the strategy provided by the AWS cloud platform, we have increased the speed limit of the vehicle and set more types of speed to meet the speed requirements of the car in different sections and also, we increased the types of steering angle to realize the flexible steering of the vehicle and let it pass through more smoothly. Additionally, path planning is adopted to fix the path of the vehicle on the track.

The next aspect that needs to be optimized or broken through is the weight distribution of the reward function. Also, path planning can be used for chasing races and obstacle racing.

References

[1] Qing Kong et al., *Constrained Policy Optimization Algorithm for Autonomous Driving via Reinforcement Learning,* 2021 6th ICIVC, 2021, pp. 378-383.

[2] Xia Wei, Li Huiyun. *Training Method of Automatic Driving Strategy Based on Deep Reinforcement Learning*, Journal of Integration Technology, 2017, pp. 29-35.

[3] Takafumi Okuyama, Tad Gonsalves, Jaychand Upadhay. *Autonomous Driving system based on Deep Q learning*, 2018 ICoIAS, 2018, pp. 201-205.

[4] Y. Liu, W. Sun et al., "Dynamic Path Planning for Fomula Autonomous Racing Car," 2021 40th CCC, pp. 6087-6093.

[5] Y. Liu, L. Shi et al., "The Multi-sensor Fusion Automatic Driving Test Scene Algorithm based on Cloud Platform, "2020 CAC, 2020, pp. 5975-5980.

[6] "DeepRacer", https://aws.amazon.com/cn/deepracer

[7] E. A. Dreveck et al., "Easy Learning of Reinforcement Learning with a Gamified Tool," 2021 Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE), 2021, pp. 360-365.

[8] Y. Liu, W. Sun et al., "Multi-agent Collaborative Adaptive Cruise Control Based On Reinforcement Learning," 2021 China Automation Congress (CAC), 2021, pp. 3388-3393.

[9] B. Balaji et al., "DeepRacer: Autonomous Racing Platform for Experimentation with Sim2Real Reinforcement Learning," 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 2746-2754.

[10] "Amazon Web Services", http://aws.amazon.com/cn

[11] "Robomaker", https://aws.amazon.com/cn/robomaker/

[12] "Sagemaker", https://aws.amazon.com/cn/sagemaker/

[13] "Amazon S3", https://aws.amazon.com/cn/S3

[14] "Redis", https://redis.io

[15] Arghyadip Roy et al., *Online Reinforcement Learning of Optimal Threshold Policies for Markov Decision Process,* IEEE Transactions on Automatic Control, 2021.