Reinforce Learning-Based Collision Avoidance in Network Assisted Automated Driving

yu chen (buptchen@163.com)
Beijing Polytechnic https://orcid.org/0000-0003-1553-0330
Wei Han
Beijing Polytechnic
Qinghua Zhu
Beijing Polytechnic
Yong Liu
Beijing Polytechnic
Jingya Zhao
Beijing Polytechnic

Research Article

Keywords: network assisted, automated driving, path planning, reinforcement learning, obstacle avoidance, path following

Posted Date: January 25th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1226853/v1

Title: Reinforce Learning-Based collision avoidance in Network Assisted Automated Driving

Authors: Yu Chen*, Wei Han, Qinghua Zhu, Yong Liu, Jingya Zhao

Author affiliations: Beijing Polytechnic, Beijing 100015, China

author details:

Associate Professor：Yu Chen

E-mail: chenyu@bpi.edu.cn

Detailed-Address: 9 Liangshuihe 1st Street, Beijing Economic-Technological Development Area

Yu Chen is an Associate Professor at Beijing Polytechnic University and has PhD degree in communication engineering by Beijing University of Posts and Telecommunications. His current research interests focus on radio resource and mobility management, software defined wireless networks, and broadband multimedia transmission technology.

Associate Professor：Wei Han

E-mail: hanwei@bpi.edu.cn

Detailed-Address: 9 Liangshuihe 1st Street, Beijing Economic-Technological Development Area

Wei Han is an Associate Professor at Beijing Polytechnic and has Master degree in electrical automation by Southwest Jiaotong University. Her current research interests focus on Electronic Information Engineering Technology， Big data technology and application， Artificial Intelligence Application Technology.

Name：Qinghua Zhu

E-mail: zhuqinghua@bpi.edu.cn

Detailed-Address: 9 Liangshuihe 1st Street, Beijing Economic-Technological Development Area

Qinghua Zhu received his Bachelor's degree in computer software and application specialty from Beijing University, Beijing, China in 2001. He is currently working in Beijing Polytechnic University. His research interests are in the area of multimedia communication, QoE management, Green energy efficient network and network virtualization (SDN).

Name：Yong Liu

E-mail: liuyong@bpi.edu.cn

Detailed-Address: 9 Liangshuihe 1st Street, Beijing Economic-Technological Development Area

Yong Liu received his Bachelor's degree in communication engineering from Beijing University of Chemical Technology, Beijing, China, in 2004. His research includes green multimedia communication, QoE-centric network and application management and so on.

Name：Jingya Zhao

E-mail: zhaojingya@bpi.edu.cn

Detailed-Address: 9 Liangshuihe 1st Street, Beijing Economic-Technological Development Area

Jingya Zhao received her Master's degree in Beijing Institute of Technology in 2007. She joined the School of Telecommunications Engineering in Beijing Polytechnic University in 2000. Her research includes Open Wireless Networks, QoE management in wireless networks, software defined wireless networks, cross-layer design for mobile video applications and so on.

ABSTRACT

In the field of autonomous driving, obstacle avoidance is of great significance for safe driving. At present, in addition to traditional obstacle avoidance algorithms including VFH algorithm, artificial potential field method, a large number of related researches are focused on algorithms based on vision and neural networks. Researches on these algorithms have achieved some results, and some of which have completed real road tests. However, most of algorithms consider only local environmental information which may cause local optimum in complex driving environments. Therefore, it is necessary to consider the environmental information beyond the sensor's perceptual ability for autonomous driving in complex environment. In the network assisted automated driving system, networked vehicles can obtain road obstacles' and condition information through roadside sensors and mobile network, so as to gain extra sensing ability. Therefore, network assisted automated driving is of great significance in obstacle avoidance. Under this background, this paper presents an automatic driving obstacle avoidance strategy combining path planning and reinforcement learning. At first, a global optimal path is planned through global information, then merge the global optimal path and vehicle information into a vector. Use this vector as input of reinforcement learning neural network and output vehicle control signals to follow optimal path while avoiding obstacles.

KEY WORDS: network assisted, automated driving, path planning, reinforcement learning, obstacle avoidance, path following

1. INTRODUCTION

With the continuous development of science and technology, artificial intelligence has gradually entered all aspects of human life, as well as in the field of transportation. In the last decade, the development of automatic driving has advanced by leaps and bounds. With the participation of many large companies and the continuous improvement of automatic driving related standards, automatic driving technology is developing in a better and better direction. For instance, the projects on automatic driving done by Google and Tesla are single vehicle automatic driving [1]. This automatic driving mode only relies on the sensor equipment carried by the vehicle to sense the surrounding environment and drive according to the planned route. "Networking" and "intelligence" have become a new generation of development direction of China's automobile industry. Specifically, the network connected automatic driving vehicle is equipped with advanced on-board sensors, controllers and other devices. It is connected to the network through the mobile network access point to obtain the automatic driving auxiliary information, the other intelligent information, the vehicle status information and the road condition information sent by the network in real time. The network connected automatic driving technology integrates modern communication and network technologies, realizes the interaction of people, vehicles, roads, networks and other information, and provides safe, comfortable, energy-saving and efficient driving capabilities. It is a new generation of autonomous driving technology with great potential, and it is also the future development trend of autonomous driving [2].

In automatic driving, the first thing to be considered is safety. Obstacle avoidance should be the most basic function of automatic driving. At present, the sensing equipment of the automatic driving car mainly includes camera, laser radar, millimeter wave radar and ultrasonic radar. In terms of obstacle avoidance, the obstacle avoidance strategies are also different according to the vehicle intelligence level. When the vehicle level is L2 or L3, the vehicle has the function of collision warning and emergency braking when the distance is too close. After the vehicle reaches L4 level, it has the functions of emergency braking and bypassing obstacles. Obstacle avoidance is a complex comprehensive decision-making task. At present, the obstacle avoidance methods using traditional methods and reinforcement

learning methods have been studied and practiced.

Although there has been a lot of research on obstacle avoidance algorithms for automatic driving, there are still some problems and challenges:

- The problems falling into local optimization in the process of avoiding obstacles

In the case of single vehicle automatic driving, because the vehicle can only obtain the obstacle information around the vehicle through the on-board sensor, the vehicle control has the characteristics of local optimization. The goal driven obstacle avoidance task includes two parts: moving to the target point and obstacle avoidance. In the vehicle control, these two parts could be considered comprehensively. However, the system with local optimal control characteristics would pay too much attention to the immediate interests and could not understand the long-term interests. Specifically, if there are obstacles near the vehicle, the vehicle will think more about the obstacles around it. Only when there are no obstacles around, more consideration will be given to move to the target point [3]. In order to avoid vehicles falling into local optimization, it is very significant to obtain comprehensive information and carry out path planning.

- Vehicle model needs to be considered in vehicle control

In the simulation environment or the real environment, the vehicle model needs to be considered. According to the finite integral form of whether the position coordinates could be obtained, the vehicle model could be divided into complete constraint model and non-complete constraint model [4]. Compared with nonholonomic constraint model, the solution of holonomic constraint model is relatively simple. In the automatic control theory, when there are the paths to be followed and the surrounding obstacle information, this task [5] could be handled by a variety of vehicle control algorithms, but this involves a lot of automatic control theory knowledge, which is difficult to deploy the algorithms. In order to solve the problem that the automatic control algorithm is too complex, reinforcement learning could be adopted for vehicle control.

- When the vehicle follows the global path, it will be affected by dynamic obstacles

The global path is obtained by considering static obstacles. If there are dynamic obstacles in the environment, the vehicle must try to avoid static and dynamic obstacles when following the global path. If the traditional algorithm is used, the size, distribution and moving speed of obstacles and the relative position of targets will have a great impact on vehicle control [6] [7] [8], and the control mode is more complex. If the reinforcement learning method is used, the reinforcement learning model needs to be considered for the two tasks of "path following" and "obstacle avoidance". Compared with single task reinforcement learning, this multi task reinforcement learning training is more difficult and the weight of each task needs to be comprehensively considered [9].

2. PATH PLANNING AND OBSTACLE AVOIDANCE ALGORITHM

2.1 Path planning algorithm

The path planning algorithm is divided into graph-based path planning algorithm and sampling-based path planning algorithm. The performance of the two algorithms is compared and analyzed, and the algorithm that is most suitable for this project is selected.

2.1.1 Graph-based path planning algorithm

Graph-based path planning algorithm is a complete algorithm. Namely, if there is a feasible path from the starting point to the target point, the solution must be obtained. If there is no solution, there must be no path from the starting point to the target point. This kind of algorithm is usually planned on a grid network map. At present, the commonly used algorithms are depth-first prime search algorithm, breadth-first search algorithm, Dijkstra algorithm and A* algorithm.

2.1.2 Path planning algorithm based on sampling

Being different from map-based path planning algorithm, this type of algorithm does not require rasterization of the map, but randomly sets some sampling points in the map, and uses these sampling points to abstract the actual map to perform path planning. Compared with the graph-based path planning algorithm, the sampling-based path planning algorithm has a faster calculation speed, but the cost of the generated path may be higher, and it would cause a situation of "no solution" ". Such algorithm is generally widely used in high-dimensional planning problems. At present, the commonly used sampling-based path planning algorithms include PRM algorithm, RRT algorithm and their improved algorithm.

2.1.3 Comparison of several common path planning algorithms

A) Calculation efficiency

Here we analyze the calculation efficiency of the worst-case scenario under the optimal solutions of the above-mentioned several path planning algorithms. In the assumption of a path planning, there are V nodes, E edges and X obstacles altogether.

We discuss the calculation efficiency of graph-based path planning algorithms. The time complexity of the Dijkstra algorithm that uses the adjacency matrix and is not optimized is $O(V^2)$. If the small root heap and the adjacency list are used for optimization, the time complexity could be optimized to $O(V\log(V)+E)$. For the A* algorithm, the algorithm efficiency of different heuristic function is different, so the time complexity of the A* algorithm could not be calculated specifically. However, due to the introduction of the heuristic function, the calculation efficiency of the A* algorithm is higher than that of the Dijkstra algorithm. In addition, when the graph-based path planning algorithm is used, the plane needs to be rasterized firstly, and the time complexity of rasterization is $O(V)$. Therefore, the time complexity of the graph-based path planning algorithm is the worst $O(V+V\log(V)+E)$.

We discuss the calculation efficiency of the sampling-based path planning algorithm. For the PRM algorithm, a series of sampling points need to be randomly sampled on the map firstly, and the time complexity is $O(V)$. For each sampling point, the space tree structure is used to obtain the nearest k neighboring sampling points, and the time complexity is $O(\log(V))$. For each neighboring sampling point, the space tree structure is used for collision detection, and the time complexity is $O(\log(x))$. For each shortest path query, the Dijkstra algorithm is used to calculate the path from the starting point to the target point, the time complexity is $O(V\log(V)+E)$. For all the above processes, the time complexity of the PRM algorithm is $O(V\log(V)\log(x)+E)$ approximately. For each sampling of the RRT algorithm, the space tree structure is used to calculate the point on the random search tree that is the closest to the sampling point, and the time complexity is $O(\log(V))$. For the collision detection, the time complexity is $O(\log(x))$.There are V times of sampling, and the total time complexity is $O(V\log(Vx))$.

By comparing the time complexity, the graph-based path planning algorithm is more efficient than the sampling-based path planning algorithm, when the number of nodes is the same. However, when the granularity of rasterization is considered, the node number of the graph-based path planning algorithm is much larger than that of the sampling-based path planning algorithm. Therefore, the actual situation is that the sampling-based path planning algorithm has higher calculation efficiency.

The PRM algorithm is an algorithm based on multiple path queries, while RRT is an algorithm based on a single path query. For the PRM algorithm, in order to adapt to multiple queries, the sampling points need to cover all areas of the space as much as possible. Therefore, the PRM algorithm needs a large number of sampling points. For the RRT algorithm, the path graph is constructed while sampling. The algorithm terminates on the feasible path from the starting point to the target area. Therefore, the sampling point of the RRT algorithm is much less than that of the PRM algorithm. When this project

only needs a single path query, choosing the RRT algorithm could achieve higher calculation efficiency.

B) Obstacle avoidance effect

When the graph-based path planning algorithm is considered and the map in the form of a two-dimensional matrix is rasterized, the vehicle model constraints are not considered during path planning. Therefore, the obstacle avoidance effect is affected by the rasterization granularity and the adopted vehicle model. It is difficult to clearly analyze the obstacle avoidance effect.

When a sampling-based path planning algorithm is considered and a new sampling point is created and connected to the original path point, the connection could be made to follow the constraints of the vehicle model. In addition, the collision detection between the vehicle and the obstacle could can be judged by a variety of algorithms. Compared with the graph-based path planning algorithm, the obstacle avoidance effect is better.

2.1.4 RRT* algorithm

A) Algorithm overview

The RRT* algorithm [10] is a fast path planning method based on random sampling and progressive optimization, which is an improved version of the RRT algorithm. For a long time, sampling-based path planning algorithms (such as Probabilistic Road Map (PRM) and Rapid Random Tree (RRT)) have performed well in practice and have theoretical guarantees such as probabilistic integrity. However, studies have shown that the solutions returned by the above two algorithms could almost certainly converge to a non-optimal value. Therefore, many improvements on the RRT algorithm are dedicated to solving the problem of path optimization, and the RRT* algorithm is one of them. The RRT* algorithm retains many excellent characteristics of the RRT algorithm, such as complete probability, which could be directly applied to complete and incomplete constraints. The difference from the RRT algorithm is that the RRT* algorithm is progressively optimal. The RRT* algorithm could quickly find a path from the start point to the end point, and then continue to optimize as the sampling points increase. The path from the start point to the end point gradually becomes the optimal path. The main reason for the RRT* algorithm to obtain this characteristic is the reselection process of the new parent node and the rewiring process of the potential parent node.

B) Algorithm steps

A directed graph is used to represent the path. A feasible path is a sequence of vertices $(v_1, v_2, v_3, \mathrm{L}\ , v_n)$, $v_1 = x_{init}$,and $v_n = x_{goal}$. At the same time, it represents the edge $(v_i, v_{i+1}) \in E, 1 \le i \le n-1$. The graph is expanded by sampling until a path from the start point to the end point could be found. The steps of the RRT* algorithm are as follows:

Algorithm 1: The main part of the RRT* algorithm

1. $V \leftarrow \{x_{init}\}; E \leftarrow \varnothing; i \leftarrow 0;$
2. **while** $i < N$ **do**
3. $\quad G \leftarrow (V, E);$
4. $\quad x_{rand} \leftarrow Sample(i); i \leftarrow i+1;$
5. $\quad (V, E) \leftarrow Extend(G, x_{rand})$

Algorithm 2: the extended function of RRT* algorithm

1. $V' \leftarrow V; E' \leftarrow E;$
2. $x_{nearest} \leftarrow Nearest(G, x);$
3. $x_{new} \leftarrow Steer(x_{nearest}, x);$
4. **if** $ObstacleFree(x_{nearest}, x_{new})$ **then**
5. $\quad V' \leftarrow V' \cup \{x_{new}\};$

6. $x_{\min} \leftarrow x_{nearest}$;

7. $X_{near} \leftarrow Near(G, x_{new}, |V|)$;

8. **for** *all* $x_{near} \in X_{near}$ **do**

9. **if** $ObstacleFree(x_{near}, x_{new})$ **then**

10. $c' \leftarrow \mathrm{Cost}(x_{near}) + c(Line(x_{near}, x_{new}))$;

11. **if** $c' < \mathrm{Cost}(x_{new})$ **then**

12. $x_{\min} \leftarrow x_{near}$;

13. $E' \leftarrow E' \cup \{(x_{\min}, x_{new})\}$;

14. **for** *all* $x_{near} \in X_{near} \backslash \{x_{\min}\}$ **do**

15. **if** $ObstacleFree(x_{new}, x_{near})$ *and* $\mathrm{Cost}(x_{near}) > \mathrm{Cost}(x_{new}) + c(Line(x_{new}, x_{near}))$ **then**

16. $x_{parent} \leftarrow Parent(x_{near})$;

17. $E' \leftarrow E' \backslash \{(x_{parent}, x_{near})\}$;

18. $E' \leftarrow E' \cup \{(x_{new}, x_{near})\}$;

19. *return* $G' = (V', E')$;

The RRT* algorithm consists of two parts. One is the main part of the algorithm, and the other is the extend function. Here is a brief introduction to the steps.

In the main function, the root node is firstly initialized as x_{init}, and the edge set is E. Enter the while loop and try to add more points to the graph. Generate the graph G according to the point set V and the edge set E. Sample a new point x_{new} and use the new point to expand G.

In the extend function, firstly put V, E in temporary storage. $Nearest(G, x)$ means the closest point to x in the figure. $Steer(x_{nearest}, x)$ means there is a x_{new}. In the case of satisfying $\|x_{new} - x_{nearest}\| < \eta$, minimize $\|x_{nearest}, x_{new}\|$. η is a value set artificially. Steps 5 to 18 are the rewiring steps of the new node x_{new} and the potential parent node. $x_{near} \in X_{near}$.

C) Rewiring of new node and potential parent node

Firstly, reselect the x_{new} parent node to minimize the cost from the root node to x_{new}. Take x_{new} as the center of the circle and make a circle is made with the threshold radius. At this time, all the points in the circle are called potential parent nodes, and all potential parent nodes form a set X_{near}. Try to use all potential parent nodes $x_{near} \in X_{near}$ and x_{new} to connect with each other. That the connection is the trajectory of the car is assumed. If the trajectory does not pass the collision detection, delete the potential parent node from the set of potential parent nodes, as shown in Figure 1:

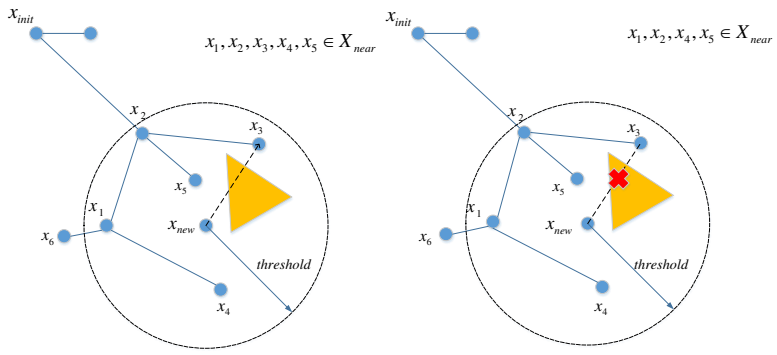

Figure 1 Delete the potential parent nodes that fail the collision detection

Among the remaining potential parent nodes, select the parent nodes according to the steps 8 to 12

in Algorithm 2, as shown in Figure 2-2:

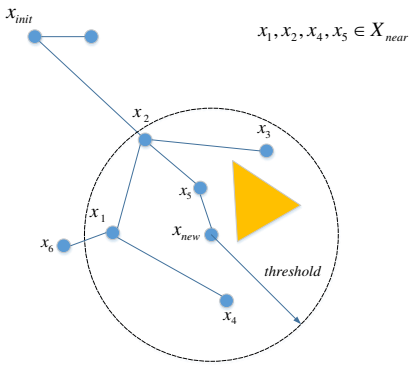

Figure 2 Select the parent nodes of the new nodes

Rewire all potential parent nodes, namely, Steps 13 to 18 in Algorithm 2. Check each potential parent node to determine whether the cost from the root node to the node is greater than the cost from the root node to the node x_{new}. If so, we need to change the parent node of this node to x_{new}, as shown in Figure 3:

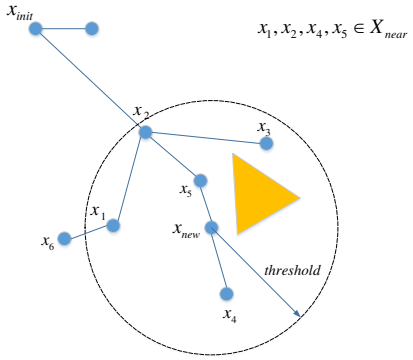

Figure 3 Potential parent node rewiring

2.2 Obstacle avoidance algorithm

The introduction of obstacle avoidance algorithms includes traditional obstacle avoidance algorithms and obstacle avoidance algorithms based on reinforcement learning. The performance of the two types of algorithms is compared and analyzed, and the obstacle avoidance algorithm suitable for this project is selected.

2.2.1 Traditional obstacle avoidance algorithm

A) VFH algorithm

The Vector Field Histogram algorithm [11][12][13] is a real-time motion planning algorithm that uses sensors on the intelligent agent to detect the surrounding environment, generate obstacle distance histograms, and use the histogram to perform target-driven obstacle avoidance. In practice, the VFH algorithm has been proven to be fast and reliable, especially when traversing densely populated obstacle routes. However, the VFH algorithm only considers local obstacle information and belongs to a local path planning algorithm.

The VFH algorithm consists of three main parts: Cartesian histogram construction, polar coordinate histogram construction, and candidate valley selection. The Cartesian histogram is constructed based on the distance of surrounding obstacles detected by the distance sensor on the intelligent agent (such as sonar or laser rangefinder). The more obstacles are detected on the same grid, the greater the possibility of obstacles in the grid is. The histogram grid is updated in real time. With the intelligent agent as the

pole, the polar coordinate histogram is obtained by transforming the Cartesian histogram, describing the distribution of obstacles around the intelligent agent. Then, according to a threshold, the polar coordinate histogram is divided into several continuous areas. The area below the threshold is called a candidate valley, which means that the direction is safer. The direction of the target point is combined to control the turning of the intelligent agent.

B) Artificial Potential Field Method

The artificial potential field method [14] is an algorithm for avoiding obstacles in analogy to the phenomenon of "homogeneous phase exclusion and specific phase absorption " in an analogous electric field. In this algorithm, the target point would generate a gravitational potential field for the intelligent agent. The obstacle would generate a repulsive potential field for the intelligent agent. The two potential fields are scalarly added to obtain the potential field distribution on the two-dimensional plane. According to the principle of "move from a place with higher potential energy to a place with lower potential energy", the intelligent agent moves from the starting point to the target point and avoids surrounding obstacles.

In the general artificial potential field method, the size of the gravitational potential field is proportional to the square of the distance between the intelligent agent and the target point. The size of the repulsive potential field is inversely proportional to the square of the distance between the intelligent agent and the obstacle. There are many improvement methods based on the general artificial potential field method, which will make certain improvements to the gravitational potential field and the repulsive potential field.

2.2.2 Obstacle avoidance algorithm based on reinforcement learning

At present, there are many obstacle avoidance algorithms based on reinforcement learning, which can be roughly divided into the following four categories: obstacle avoidance in a static environment, obstacle avoidance in a dynamic environment, target-driven obstacle avoidance in a static environment [15][16], and target-driven obstacle avoidance in the dynamic environment [17]. The first two do not have the task of "reaching the goal". When setting the reward function, the closer is to the obstacle, the greater the penalty is. The farther is from the obstacle, the smaller the penalty is. The latter two include the task of "reaching the goal". When setting the reward function, it is necessary to weight the distance from the target and the distance from the obstacle. In most papers, goal-driven obstacle avoidance in a dynamic environment involves the problem of multi-agent obstacle avoidance.

2.2.3 Comparison of two types of obstacle avoidance algorithms

Table 1 Comparison of two types of obstacle avoidance algorithms

Obstacle Avoidance Program	Advantage	limitation
Traditional obstacle avoidance method	It has been fully practiced and the obstacle avoidance effect is excellent.	The algorithm that does not consider the vehicle model is not suitable for actual driving scenarios. The algorithm that considers the vehicle model involves a large amount of knowledge in the field of automatic control. It is very difficult to deploy.
avoidance method based on reinforcement	For the vehicle model, only a small amount of knowledge in the field of	Need to constantly try to change the reward function to get better results. Need to be trained, and the training period is longer.

learning	automatic control is required.	

2.2.4 Obstacle avoidance algorithm of goal-driven reinforcement learning in a dynamic environment

A) Algorithm overview

According to the different problems to be solved and the different vehicle models used, the design of the obstacle avoidance algorithm based on reinforcement learning should be adjusted according to the specific situation. Here is a general algorithm for reinforcement learning to solve the target-driven obstacle avoidance problem in a dynamic environment.

The obstacle avoidance algorithm of goal-driven reinforcement learning in a dynamic environment contains three characteristics: dynamic environment, goal-driven, and obstacle avoidance. Dynamic environment means that the vehicle operating environment contains moving objects. These moving objects can be moving obstacles that do not adopt any obstacle avoidance strategy or other intelligent agents that adopt the same or different obstacle avoidance strategies. Target drive means that the vehicle ultimately needs to reach designated target area. Obstacle avoidance requires vehicles to avoid dynamic and static obstacles in the environment when reaching the target.

As a reinforcement learning algorithm, the obstacle avoidance algorithm of goal-driven reinforcement learning in a dynamic environment must have a simulation environment that can simulate vehicle operation and a reasonable reinforcement signal. After that, the intelligent agent runs in the environment according to the current strategy, and learns through the reinforcement signals returned by the environment.

B) Algorithm steps

Here, env represents the environment, and π represents the current strategy of the reinforcement learning agent.

Algorithm 3: obstacle avoidance algorithm of target-driven reinforcement learning in a dynamic environment

```
init(π) and set hyperparameter
for episode = 1,M do
    state = env.reset
    for t = 1,T do
        action = π(state)
        stateNext, reward, done = env.step(action)
        update(π)
        if done = true then
            break
        state = stateNext
```

The intelligent agent equipped with this kind of reinforcement learning algorithm interacts with the environment and combines the reinforcement signals returned by the environment to continuously improve the strategy network, which finally enables the intelligent agent to obtain a relatively complete vehicle control strategy. Different reinforcement learning algorithms update the strategy network differently, but the training steps of the goal-driven obstacle avoidance task in a dynamic environment are all as described above.

C) Markov decision process

In reinforcement learning, the intelligent agent and the environment are always interacting. At each

time t, the intelligent agent makes an action based on the state returned by the environment. The environment transforms to the new state and returns the enhanced signal reward. Therefore, the interaction between the intelligent agent and the environment produces a sequence:

$$state_0, action_0, reward_1, state_1, action_1, reward_2, \mathrm{L}$$

This sequence is called the decision process, and the Markov decision process is a formulation of a typical sequence decision process. Approximating the reinforcement learning training process as the Markov decision process can simplify the problem. Therefore, reinforcement learning algorithms are established in the Markov decision process. Under the assumption of the Markov decision process, the next state is only related to the current state and actions, namely: $P[S_{t+1} \mid S_t, A_t, S_{t-1}, A_{t-1}, S_{t-2} \mathrm{L}\ ] = P[S_{t+1} \mid S_t, A_t]$.

The obstacle avoidance algorithm of goal-driven reinforcement learning in a dynamic environment is a Markov decision process. The observed value of the environment includes the current position of the vehicle, the relative position of the target area, and the distribution of surrounding obstacles. The movement of the vehicle includes the vehicle control information under the current vehicle model. After the current action, the next state is the position of the vehicle at the next moment, the relative position of the target area and the distribution of surrounding obstacles at this time. It can be clearly seen that the next state only needs the current state and the current action to be obtained, and has Markov characteristics. Therefore, it is possible to use reinforcement learning to solve the goal-driven obstacle avoidance problem in a dynamic environment.

3. DESIGN AND IMPLEMENTATION OF TARGET-DRIVEN OBSTACLE AVOIDANCE ALGORITHM BASED ON DDPG

3.1 The establishment of scenarios and models

3.1.1 Vehicle model

In this project, a fully constrained vehicle kinematics model is adopted. The description of the vehicle in the scene includes coordinates, speed magnitude and speed direction. The symbols are shown in Figure 4:

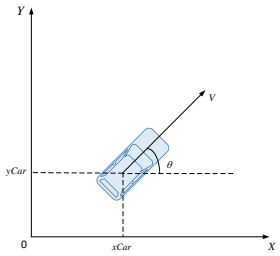

Figure 4 Vehicle status information

It is assumed that the simulation time interval of each step is ΔT. The vehicle moves in a straight line at a constant speed in each time. The control variables of the vehicle are the longitudinal acceleration and the angular acceleration in the velocity direction. Therefore, within the simulation time interval of one step, the following equation is satisfied:

$$\begin{aligned} V_{t+1} &= V_t + \alpha \Delta T \\ \theta_{t+1} &= \theta_t + \beta \Delta T \\ xCar_{t+1} &= xCar_t + V_{t+1} \Delta T \cos(\theta_{t+1}) \\ yCar_{t+1} &= yCar_t + V_{t+1} \Delta T \sin(\theta_{t+1}) \end{aligned} \tag{1}$$

The contour of the equivalent vehicle is approximated according to the encircling circle method.

The 11 laser rangefinders are installed at equal angular intervals in front of the vehicle to measure the distances of obstacles in 11 different directions. The vehicle equivalent model is shown in Figure 5:

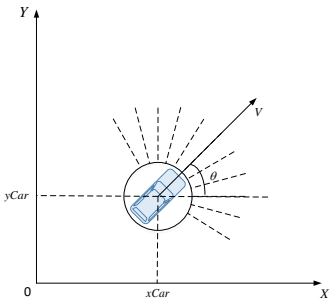

Figure 5 Vehicle equivalent model

3.1.2 Training scenario

The tasks to be completed by reinforcement learning are: reaching the target point and avoiding obstacles. The training scene can include dynamic obstacles and static obstacles. However, because the reinforcement learning algorithm has Markov characteristics, it is impossible to judge the movement direction and speed of the currently detected obstacles from the state information of the vehicle. If dynamic obstacles are contained, the obstacle avoidance behavior of the vehicle will be affected by the movement state of the currently detected obstacle, resulting in different evaluations when the vehicle performs the same action in the same state. Namely, the Q function is unstable.

Therefore, in this project, the training scene could be the scene with only static obstacles, and use the encircling circle method to approximate the contour of the obstacle, as shown in Figure 6:

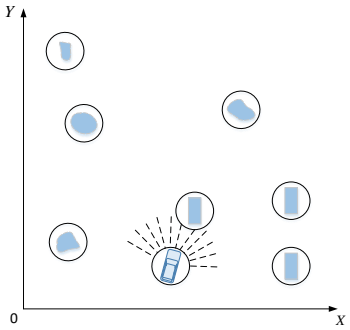

Figure 6 Training scene

3.2 Neural network design based on DDPG

3.2.1 Overview of DDPG algorithm

DDPG (Deep Deterministic Policy Gradient) algorithm [18] is a model free, off-policy reinforcement learning algorithm that can be used to solve continuous state space and continuous action space problems. DDPG is based on actor-critic, which includes both value function network (critic) and policy network (actor).

The DDPG algorithm has many advantages to adopt a deterministic strategy network and update the strategy network using strategy gradients. This enables the network to handle high-dimensional continuous actions. The introduction of experience playback arrays increases the utilization of data to; adopt dual network updates of the target network and the current network. This makes the algorithm easier to converge. The target network adopts the soft update method, which increases the stability of the algorithm.

3.2.2 DDPG algorithm principle

DDPG uses a neural network to approximate the value function. This value function network is also called a critic network. Its input is action and state value $[a, s]$ and its output is $Q(s,a)$. There is also a neural network to approximate the strategy function. This strategy network is also called an actor network. Namely, its input is a state s, and its output is an action a.

The critic network uses the gradient descent method to minimize the loss function and update the network parameters. The specific method is as follows. N in the formula is the size of the mini-batch:

$$\begin{aligned} y_i &= r_i + \gamma Q(s_{i+1}, \mu(s_{i+1} \mid \theta^{\mu}) \mid \theta^{Q}) \\ Loss &= \frac{1}{N}\sum_i (y_i - Q(s_i, a_i \mid \theta^{Q}))^2 \end{aligned} \tag{2}$$

The actor network follows the chain rule and updates the network parameters by analogy with the strategy gradient algorithm. The specific methods are as follows:

$$\begin{aligned} \nabla_{\theta^{\mu}} J &\approx \mathbf{E}_{s_t \sim \rho^{\beta}}[\nabla_{\theta^{\mu}} Q(s, a \mid \theta^{Q})\big|_{s=s_t, a=\mu(s_t \mid \theta^{\mu})}] \\ &= \mathbf{E}_{s_t \sim \rho^{\beta}}[\nabla_a Q(s, a \mid \theta^{Q})\big|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^{\mu}} \mu(s \mid \theta^{\mu})\big|_{s=s_t}] \end{aligned} \tag{3}$$

DDPG requires samples to be independently and identically distributed, so it is necessary to use experience playback arrays for mini-batch learning. In addition, for reference to DQN's dual network technology, DDPG divides both the critic network and the actor network into the target network and the current network. And the soft update method is used to greatly increase the stability of the network.

3.2.3 DDPG algorithm steps

Algorithm 4: DDPG algorithm steps

1. Randomly initialize the weights θ^{Q} and θ^{μ} of the critic network and the actor network,
2. Initialize the weights $\theta^{Q'} \leftarrow \theta^{Q}$ and $\theta^{\mu'} \leftarrow \theta^{\mu}$ of the target networks Q' and μ',
3. Initialize the experience playback array R
4. **for** $episode = 1, M$ **do**
5. Initialize a random process N as an exploration
6. Initialization state s_1
7. **for** $t = 1, T$ **do**
8. Select the current action $a_t = \mu(s_t \mid \theta^{\mu}) + \mathrm{N}_t$
9. Perform action a_t to get the reward r_t and the next state s_{t+1}
10. Save (s_t, a_t, r_t, s_{t+1}) into the experience replay array R
11. Randomly sample N elements (s_i, a_i, r_i, s_{i+1}) from the experience playback array
12. Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1} \mid \theta^{\mu'}) \mid \theta^{Q'})$
13. Update the critic network to minimize the loss function $Loss = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i \mid \theta^{Q}))^2$
14. Update the actor network: $\nabla_{\theta^{\mu}} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a \mid \theta^{Q})\big|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^{\mu}} \mu(s \mid \theta^{\mu})\big|_{s=s_t}$
15. Update the target network: $\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$, $\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$

3.2.4 Design of neural network

A) Neural network structure

Figure 7 shows the actor network built in this article, including 3 hidden layers, which are all fully connected networks. We set the fully connected layer Fc1 to 300 neurons, set the fully connected layer

Fc2 to 400 neurons, and set Fc3 to 300 neurons. The number of nodes in the output layer is the same as the action dimension. The activation function of the hidden layer selects the relu function, and the activation function of the output layer is tanh. The network input is the state s, and the output is the action a.

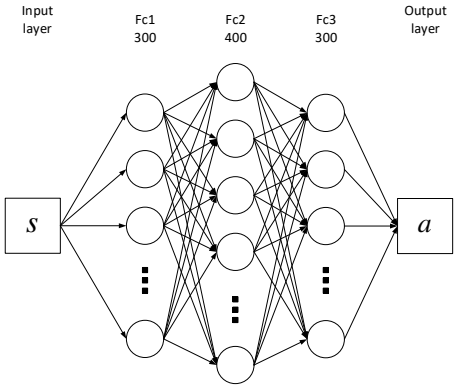

Figure 7 actor network structure

Figure 8 shows the critic network built in this article, including 2 hidden layers, which are all fully connected networks. The activation functions are all relu. The fully connected layer Fc1 is connected to the input state s and contains 300 neurons. The fully connected layer Fc2 is connected to the input action a and contains 300 neurons. The Fc1 output and the Fc2 output are added to the fully connected layer Fc3 and Fc3 contains 100 Neuron. The output layer contains 1 neuron to represent $Q(s,a)$ for linear activation.

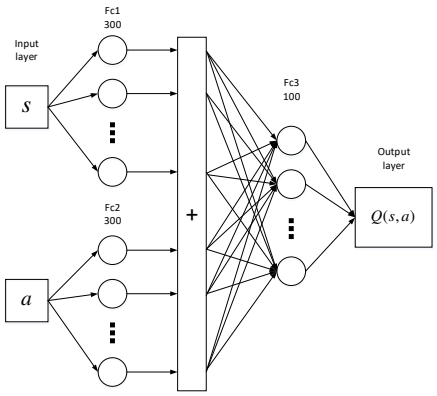

Figure 8 critic network

B) Network parameter setting

The size of the experience replay array is 10^5. Generally speaking, the larger the experience replay array is, the longer the training data could be used. The data utilization rate would be improved. However, if the reinforcement learning network generates more garbage data during the training process, an excessively large experience playback array will cause the garbage data to be learned multiple times, thereby affecting the final effect. In this project, because the obstacles in the training environment are randomly generated, it is possible that the target point cannot be reached. Such data is garbage data. In order to reduce the impact of garbage data, we set the experience playback array size to be smaller.

The discount factor is set to 0.98. The empirical formula $1/1-\gamma$ represents the number of steps considered in the future. There are some special situations in this project. There are obstacles near the target point, or the target point is reached through obstacles. At this time, approaching obstacles is able

to reach the target point in the future. Therefore, during training, we need to consider the impact of the next few steps as much as possible. However, if γ is too large, it may make the algorithm difficult to converge. Therefore, γ is set to 0.98 to consider the impact of the next 50 steps.

The batch size is set to 32. The reason is similar to the setting of the experience playback array. If more garbage data will be generated during training, too large batch size will increase the impact of garbage data on the result. Therefore, the bath size should be small he. However, too small batch size may cause the algorithm to converge to the local optimum, so the batch size is set to 32 here.

We take Gaussian random process as the exploration noise. The actions must be normalized. The standard deviation of the initial Gaussian noise is 1. After each iteration, the standard deviation of the Gaussian noise becomes 0.99 times of the original. Generally speaking, continuous and inertial motions are all explored by adding Gaussian noise. Here we take the general method.

The soft update coefficients τ of the actor network and the critic network are both 0.01. In the DDPG algorithm, in order to ensure the stability of the algorithm, the value τ is generally small and its value is 0.1 or 0.01. In some cases, the value τ of critic will be slightly larger than the value τ of actor. Here, the values τ of the two networks are set to be the same.

Both the actor network and the critic network are updated with the Adam optimizer. The learning rate of the actor network is 0.0001, and the learning rate of the critic network is 0.0002. The learning rate of the Adam optimizer is generally recommended to be 0.0001. In this project, the actor network is set according to the recommended value. The critic network needs to converge as soon as possible, so the learning rate can be set slightly larger.

4. METHODS AND EXPERIMENTAL

4.1 Simulation environment

There are only stationary obstacles in the environment of this experiment. It is assumed that the equivalent circle radius of all obstacle contours is the same as the equivalent circle radius of the vehicle $rCar = rObstacle = 0.5m$. The test scene is a two-dimensional plane of 25m*25m, with the abscissa $x \in [0,25]m$ and the vertical coordinate $y \in [0,25]m$ .The target area is a circle, and the radius is $endR = 0.1m$. In the initial state, the vehicle position and direction are randomly generated, and the speed is 0. The target area is randomly generated. The obstacles are randomly generated without overlap. The simulation time interval is $\Delta T = 0.01s$. The number of training rounds is 300, and each round has a maximum of 1000 time steps. If $done = true$, then the round is stopped.

(1) Status. The description of the state in this project includes the data received by the vehicle-mounted sensor, the positional relationship of the vehicle relative to the target area, and the speed and direction information of the vehicle. It is composed of these three parts. There are 11 vehicle-mounted laser rangefinders, which are evenly distributed within 180° in front of the car. The maximum detection distance of each rangefinder is $maxSensorDis = 4m$. The 11 laser rangefinders form the sensor vector $sensors = (sensor_1, sensor_2, \mathrm{L}\ , sensor_{11})$ in the order from the front left side of the car to the front right side of the car. The position relationship of the vehicle relative to the target is expressed in the form of polar coordinates, with the center of the vehicle as the pole and the x axis of the two-dimensional plane as the direction of the polar axis, as shown in Figure 9. Thereinto, the polar diameter of the vehicle is dg and the polar angle is $cgAngle \in (-\pi, \pi]$. The speed of the vehicle is $vCar \in [0,10]$ and the speed direction is $vthetaCar \in (-\pi, \pi]$. The above three parts are spliced into a 15-dimensional vector, and there into the values are normalized to obtain the state vector:

$$s = (\frac{dg}{maxSensorDis}, \frac{cgAngle}{\pi}, \frac{vCar}{vMax}, \frac{vthetaCar}{\pi}, \frac{sensors}{maxSensorDis}) \tag{4}$$

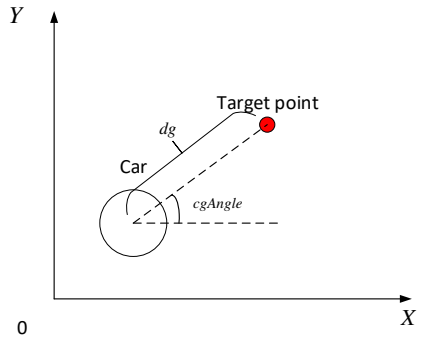

Figure 9 The position of the vehicle relative to the target

(2) Action a. According to the previous introduction to the vehicle model, the control quantity is a two-dimensional row vector composed of longitudinal acceleration $\alpha \in [-\frac{vMax^2}{2*rCar}, \frac{vMax^2}{2*rCar}]$ and angular acceleration $\beta \in [-\frac{\pi}{18}, \frac{\pi}{18}]$ in the direction of velocity. We use the 2-dimensional normalized action vector outputted by the neural network to generate the vehicle control quantity, namely:

$$(\alpha, \beta) = a \begin{pmatrix} \frac{vMax^2}{2*rCar} & 0 \\ 0 & \frac{\pi}{18} \end{pmatrix} \tag{5}$$

(3) Training is terminated $done$. There are three types of training termination situations, which are reaching the target area, colliding with an obstacle, and exceeding the boundary of the area. When training is terminated, set $done = true$, otherwise $done = false$,.

(4) Reward r. The overall reward function consists of three parts, which is distance reward from the target point, distance reward from obstacles, and training termination reward. The overall formula is shown in Equation 5. $reward_1(s,a)$, $reward_2(s,a)$ and $reward_3(s,a)$ are shown in formulas 6, 7, and 8 respectively. Among them, $dgOld$ is the value dg before the action a.

$$r = reward_1(s,a) + reward_2(s,a) + reward_3(s,a) - 1 \tag{6}$$

$$reward_1 = \begin{cases} -3, & dg >= dgOld \\ 0, & dg < dgOld \end{cases} \tag{7}$$

$$reward_2(s,a) = -\sum_{i=1}^{11} \min(\frac{10}{sensor_i} - \frac{10}{maxSensorDis}, 15) \tag{8}$$

$$reward_3(s,a) = \begin{cases} 500, \text{ Reach the target area} \\ -100, \text{ Collision with obstacles} \\ -100, \text{ Cross the border} \end{cases} \tag{9}$$

4.2 Performance analysis

The obstacle avoidance performance is tested under 10 obstacles, 20 obstacles and 30 obstacles. The test process starts from initializing vehicle, obstacles, and target location until the vehicle reaches the target area. The position of the car is displayed at each moment on the map, and the final vehicle

obstacle avoidance effect is shown in Figures 10, 11 and 12.

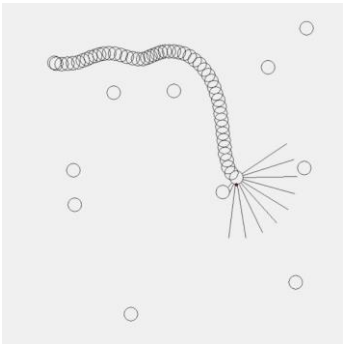

Figure 10. 10 obstacles are contained

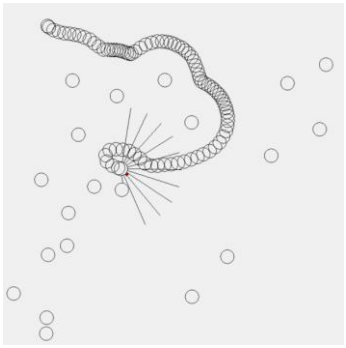

Figure 11. 20 obstacles are contained

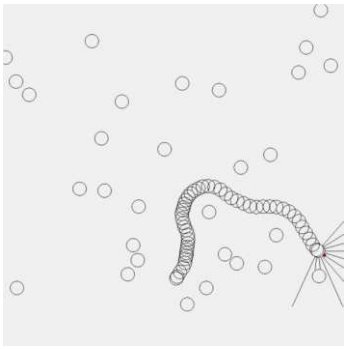

Figure 12. 30 obstacles are contained

The target points in the picture are red dots, the connected circles are the vehicle's trajectory, and the separated circles are stationary obstacles. The 11 lines in front of the car are the distances detected by the laser rangefinder. When there is an obstacle blocking the laser rangefinder, the detected distance becomes shorter.

From the test results in the above three scenarios, it can be seen that the number of obstacles increases, and the obstacle avoidance behavior of the vehicle is more complicated. However, in the end, it can reach the target area smoothly. The obstacle avoidance effect based on reinforcement learning in a static environment is quite excellent, and it can complete the two tasks of obstacle avoidance and reach the target at the same time. However, the above obstacle avoidance method only takes for the local obstacle information and yet does not take for the global obstacle information. In some special scenarios, it will fall into the local optimal value and cannot reach the end point, as shown in Figure 13.

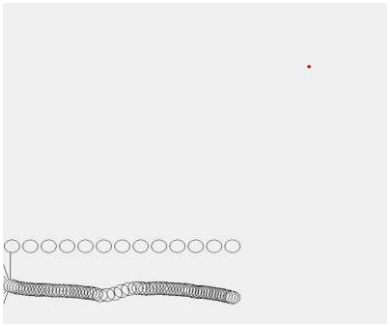

Figure 13: Falling into a local optimum

In order to solve the problem of falling into local optimum in the obstacle avoidance algorithm of traditional reinforcement learning, this paper will propose a dynamic path-following automatic driving scheme combined with path planning in the fourth chapter. By the global obstacles and the target area information being considered, the vehicle is prevented from falling into the local optimum, and the vehicle is guided to drive along the global optimum path.

5. DESIGN AND IMPLEMENTATION OF AUTOMATIC DRIVING OBSTACLE AVOIDANCE SCHEME BASED ON DYNAMIC PATH FOLLOWING

5.1 Path planning realization

For the path planning algorithm in this paper, the RRT* algorithm is selected and has been introduced in detail in Chapter 2. The extend function of the RRT* algorithm includes the design of the cost function and the collision detection function. The specific selection scheme of these two functions will be introduced below.

5.1.1 The choice of cost function

The meaning of the cost function is to represent the cost from one point to another point. It is represented by symbol c in the extend algorithm. The choice of the cost function can directly affect the performance of the RRT* algorithm path planning. However, setting a cost function is very appropriate to the actual problem. The difficulty is no less than that of designing a path planning algorithm. In addition, the cost function will also have a great impact on the convergence rate of the algorithm. Common cost functions include the Euclidean distance between two points, the weighted sum of different terrain lengths, and the cost function for energy consumption.

The simulation environment of this project is a fully constrained car model under the same terrain. There is no need to consider energy consumption. Therefore, choosing Euclidean distance as the cost function can accelerate the convergence of the algorithm and is easy to understand, as shown in Equation 10. The total cost from the starting point to a certain point is derived as Equation 11.

$$c(Line(x_1, x_2)) = \left\| x_1 - x_2 \right\| \tag{10}$$

$$Cost(x) = \begin{cases} Cost(xParent) + c(Line(xParent, x)), & x \neq root \\ 0 \quad , & x = root \end{cases} \tag{11}$$

5.1.2 Vehicle and obstacle collision detection

In the RRT* algorithm, collision detection between cars and obstacles is required. A reasonable collision detection algorithm is very important to ensure the safety of car driving. Common collision detection methods include AABB enclosing box method and enclosing circle method [19]. This project uses the relatively simple calculation of the enclosing circle method. The specific implementation steps are as follows:

If the irregular object is converted to a circle with a slightly larger area, the collision detection of two objects will be converted to the collision detection of two circles. If the distance between the centers of the two circles is less than or equal to the sum of the radii of the two circles, the two circles collide. Otherwise, there will be no collision.

On this basis, an additional safety distance d is maintained between the car and the obstacle. The boundary of the obstacle needs to be expanded. The radius of the obstacle is added to the original basis and the safety distance is added as the new radius of the obstacle. Furthermore, the coverage of the car can also be expressed as a circular area. If the radius of the car is also added to the radius of the obstacle, the car can be assumed to be a point, as shown in Figure 14:

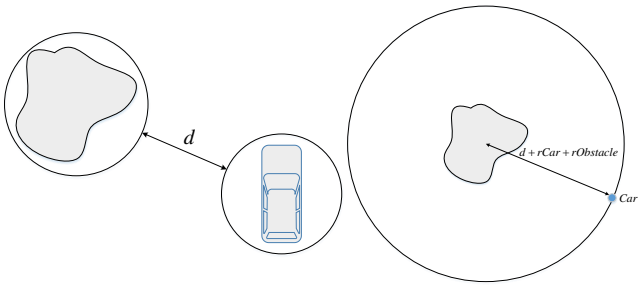

Figure 14. Equivalent radius of obstacle

The RRT * algorithm requires to detect whether the vehicle trajectory collides with the surrounding obstacles. In the case of the encircling circle method being used for collision detection, the question is converted to whether the trajectory of the car at time Δt and the circular range covered by the obstacles could intersect, as shown in Figure 15:

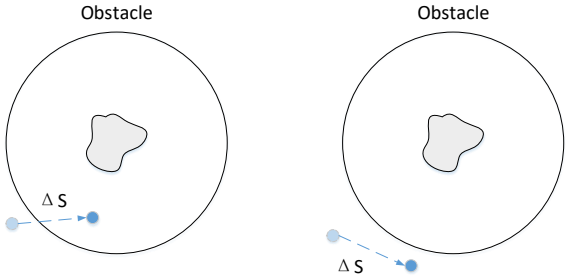

Figure 15 Vehicle trajectory at time Δt

To determine whether there is an intersection between a line segment of length ΔS and a circle, the following methods are used:

If the line unit direction vector of the line segment is $\vec{e}$, the unit normal vector is $\vec{n}$. The vector from the center of an obstacle to the left end of the line segment is $\vec{p}_1$, and the vector to the right end is $\vec{p}_2$. The equivalent radius of the obstacle is r , as shown in Figure 16:

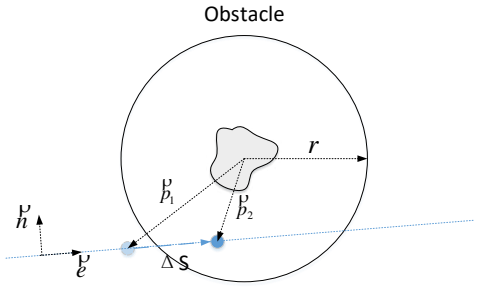

Figure 16 Determine whether trajectory collides with obstacles

If $\left|\vec{p_1} \bullet \vec{n}\right| \le r$, at this time, there is an intersection point between the line of the line segment and the circle, but there may not be an intersection point between the line segment and the circle. If one end point of the line segment is inside the circle, the line segment must intersect with the circle. If both end points of the line segment are outside the circle and if $(\vec{p_1} \bullet \vec{e}) \times (\vec{p_2} \bullet \vec{e}) > 0$, it means that the line segment does not intersect with the circle. Otherwise, the line segment intersects with the circle.

5.1.3 Setting of safety distance sum

The setting of the safety distance d : The safety distance is the additional distance between the obstacle and the vehicle that needs to be maintained on the basis that the vehicle and the obstacle do not collide. The setting of this distance needs to consider the ability of the reinforcement learning algorithm to avoid static and moving obstacles. If the reinforcement learning algorithm is more sensitive to the approach of obstacles, the safe distance should be as large as possible. If the reinforcement learning algorithm avoids closer obstacles with its strong ability, the safety distance can be appropriately reduced. The setting of this distance also has a great influence on the vehicle's optimal path selection. If the value is too large, the path planning algorithm will choose a "safer" path, rather than a shortest path. Here is the final choice $d = 3m$ based on the experimental situation.

η reflects the maximum di stance between the new point generated each time and the closest point on the random search tree. The larger the value η is, the faster the tree grows. However, if the value η is too large, the search will end soon. The sampling points are sparse and difficult to get the optimal path. This project uses reinforcement learning methods for path following, and comprehensively

considers the growth speed of the random search tree and the optimality of the path to select $\eta = 1m$

5.2 Automatic driving obstacle avoidance scheme based on dynamic path following

5.2.1 Program overview

The automatic driving obstacle avoidance scheme based on dynamic path following consists of two parts, which are global path planning and obstacle avoidance algorithm of reinforcement learning. Global path planning has been introduced in 4.1 and 2.1.4. The obstacle avoidance algorithm of reinforcement learning does not need to be retrained in a dynamic environment, and the network trained in Chapter 3 can be directly used for vehicle obstacle avoidance control.

The program firstly performs path planning based on the global static obstacles, and then uses the strategy network trained by reinforcement learning to dynamically follow the global path and avoid static and dynamic obstacles that may be encountered.

5.2.2 Dynamic following algorithm

The dynamic follow algorithm is used to dynamically update the target point that the vehicle is currently following, so that the vehicle can roughly move along the global planned path, thereby avoiding the problem of falling into the local optimum. There are many traditional paths following algorithms, including pure tracking method, Stanley method, dynamic model tracking, optimal predictive control, etc. This project refers to the way to determine the preview point in the pure tracking method to firstly judge the path point closest to the vehicle. The path point is used as the center of the circle, and the preview distance l_d is the radius to make a circle. The first point in front of the path point that is not in the circle, is taken as the next target point to follow. In this experiment, the value η of the RRT* algorithm is 1m. If $l_d = \eta = 1m$ is set, select the next point on the path as the point to follow. This preview distance is greater than the wheelbase of the car. There is a reasonable preview distance. Meanwhile, this project does not require the vehicle trajectory to strictly conform to the global path, so the setting here can be looser. The specific algorithm steps are as follows:

(1) Select the point p closest to the current vehicle position on the global path.

(2) If p is not the end point, set the p next node as the target point to be followed by the vehicle. If p is the end point, set p as the target point to be followed by the vehicle.

After executing this algorithm, the target points the vehicle will follow is the end point or the second closest point on the path ahead, as shown in Figure 17.

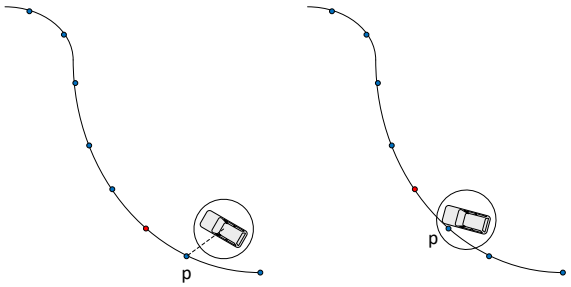

Figure 17 Dynamic path following

5.3 Simulation environment and performance analysis

The simulation environment of the final test is a dynamic environment, including static and dynamic obstacles. The static obstacles are the same as those in the simulation environment in Chapter 3. Dynamic obstacles are newly added here.

5.3.1 Dynamic obstacle model

The dynamic obstacle adopts the bicycle model [20]. The kinematic bicycle model assumes that the

description object is shaped like a bicycle, and its control can be simplified as (acc, δ). Thereinto, acc is acceleration, stepping on the accelerator pedal means positive acceleration, and stepping on the brake pedal means negative acceleration. δ is the steering wheel angle, because the front wheel angle is approximately proportional to the steering wheel angle. So, it can be assumed that this steering wheel angle is the current angle of the front tires. In this experiment, the speed of the dynamic obstacle is constant at 2m/s, so the control amount for the dynamic obstacle is δ only. The trajectory of the dynamic obstacle in each time is approximately an arc, as shown in Figure 18. Thereinto, R is the curvature radius of the trajectory point where the rear wheel is located. L is the wheelbase, which is set to 0.8m in this project.

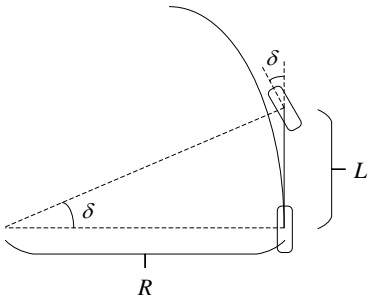

Figure 18 Dynamic obstacle model

The collision of dynamic obstacles obeys the completely elastic collision without rotation. The dynamic obstacle may collide with another moving obstacle, a stationary obstacle or boundary during the movement. In this project, it is assumed that all collisions obey the non-rotational fully elastic collision theorem.

6 PERFORMANCE ANALYSIS AND RESULTS

A) Global path planning with reference to stationary obstacles

The environment contains 15 static obstacles, which are filled with gray. By the RRT* algorithm, the path planning is carried out in the case of considering the global stationary obstacles only. The result is shown in Figure 19. Thereinto, the point in the lower left corner of the path is a red point, which represents the target point. The point in the upper right corner of the path is a yellow point, which represents the starting point. The points on the path are cyan points. The path connects a route from the starting point to the target point.

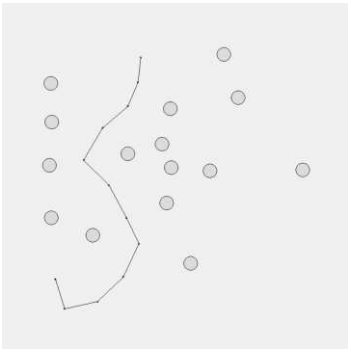

Figure 19 Global path planning

B) Dynamic path following and obstacle avoidance

Dynamic obstacles and vehicles could be added on the basis of static obstacles. 6 dynamic obstacles are taken as an example. In order to facilitate the distinction, dynamic obstacles are marked with colored

outlines, as shown in Figure 20. After that, the neural network is used to control the vehicle to complete dynamic path following and obstacle avoidance tasks. The trajectory of the vehicle and the dynamic obstacle is shown in Figure 21.

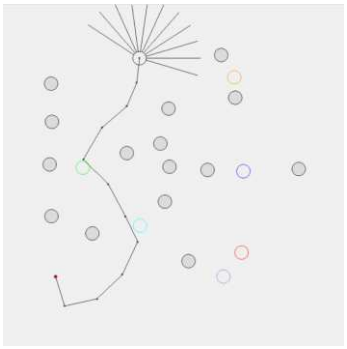

Figure 20 Initial obstacles and vehicle positions

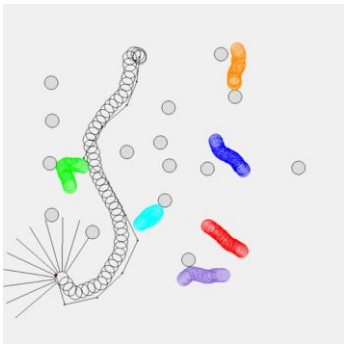

Figure 21 Trajectory diagram

As shown in Figure 21, the vehicle successfully avoids the green and cyan obstacles and successfully reached the target area. It could be seen that in the presence of dynamic obstacles, the automatic driving obstacle avoidance scheme with dynamic path following could accomplish the obstacle avoidance task excellently.

C) Method comparison without considering global path planning

In the third chapter, there is a situation of falling into the local optimum. The test is carried out in the same scene, and the effect comparison is shown in Figure 22 and Figure 23. It can be seen that the two use the same neural network to control the vehicle, but the effect is completely different. The dynamic path following method takes into account the global optimal path and could guide the vehicle away from the local optimal path.

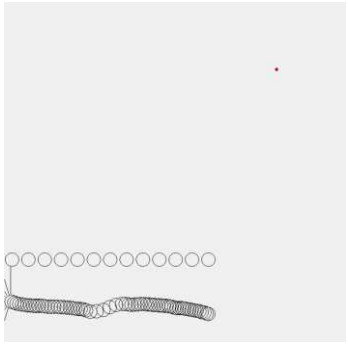

Figure 22 Traditional reinforcement learning method

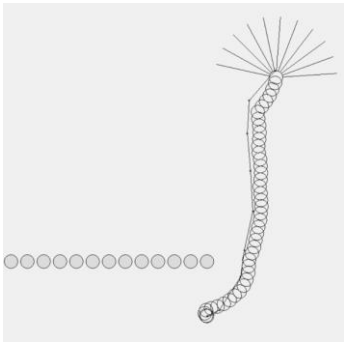

Figure 23 Dynamic path following method

7. CONCLUSION AND DISCUSSION

This paper describes the design and implementation of automatic driving scheme based on dynamic path following, including some implementation details of RRT * path planning algorithm, and the description of the final scheme based on road strength planning and reinforcement learning design. At the end of this paper, through the actual test of the algorithm proposed in this project, the effectiveness of completing the goal driven obstacle avoidance task in the dynamic environment is verified; Compared with the traditional reinforcement learning obstacle avoidance method, the scheme proposed in this project can guide the vehicle away from the local optimal solution and has good target arrival ability.

The obstacle avoidance scheme proposed in this paper still needs to be improved.

(1) The vehicle model used is relatively simple. The vehicle model used in this paper is a vehicle model with complete constraints, but the kinematics model of the real vehicle belongs to the vehicle model with incomplete constraints. If you want to test the algorithm performance in the real scene, you must use this vehicle model. However, it is difficult to use reinforcement learning to control the vehicle with incomplete constraint motion model, and how to solve it remains to be considered.

(2) Multi-agent obstacle avoidance is not considered. In this paper, the dynamic obstacle adopts the bicycle model and the random motion strategy, which leads to the non-active collision problem that cannot be avoided by the vehicle in the process of training and testing, that is, the vehicle has completed the obstacle avoidance behavior, but the moving obstacle actively collides with the vehicle. In the scenario of networked automatic driving, vehicles in a certain area will adopt the same obstacle avoidance strategy, which can solve the problem of non-active collision. Therefore, based on the discussion in this paper, the problem of multi-agent obstacle avoidance needs to be considered.

ABBREVIATIONS

VFH	Vector Field Histogram
PRM	probabilistic roadmap
RRT	Rapidly exploring random tree
DDPG	Deep Deterministic Policy Gradient

References

[1] Y. Zhang, H. Zhang, K. Long, Q. Zheng and X. Xie, "Software-Defined and Fog-Computing-Based Next Generation Vehicular Networks," in IEEE Communications Magazine, vol. 56, no. 9, pp. 34-41, Sept. 2018, doi: 10.1109/MCOM.2018.1701320.

[2] Ali, E. S. , et al. "Machine Learning Technologies for Secure Vehicular Communication in Internet of Vehicles: Recent Advances and Applications." Security and Communication Networks 2021.1(2021):1-23.

[3] Millan, Jose Del R. . "Reinforcement learning of goal-directed obstacle-avoiding reaction strategies in an autonomous mobile robot." Robotics & Autonomous Systems 15.4(1995):275-299.

[4] Snider, J. M. . "Automatic Steering Methods for Autonomous Automobile Path Tracking." [Dissertation]. Pittsburgh. Carnegie Mellon University:2009.

[5] X. Zhang, J. Wang, H. Zhang, L. Li, M. Pan and Z. Han, "Data-Driven Transportation Network Company Vehicle Scheduling with Users' Location Differential Privacy Preservation," in IEEE Transactions on Mobile Computing, doi: 10.1109/TMC.2021.3091148.

[6] D Hsu, et al. "Randomized Kinodynamic Motion Planning with Moving Obstacles." The International Journal of Robotics Research. 21(3). 2002, pp. 233-255.

[7] E. Owen and L. Montano, "Motion planning in dynamic environments using the velocity space," 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005, pp. 2833-2838, doi: 10.1109/IROS.2005.1545110.

[8] Fox, D. , W. Burgard , and S. Thrun . "The Dynamic Window Approach to Collision Avoidance." IEEE Robotics & Automation Magazine 4.1(2002):23-33.

[9] H. W. Jun, H. J. Kim and B. H. Lee, "Goal-Driven Navigation for Non-holonomic Multi-Robot System by Learning Collision," 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 1758-1764, doi: 10.1109/ICRA.2019.8793810.

[10] Islam, F. , et al. "RRT-Smart: Rapid convergence implementation of RRT towards optimal solution." In. IEEE. 2012 IEEE International Conference on Mechatronics and Automation. Chengdu.

2012:1651-1656.
[11] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," in IEEE Transactions on Robotics and Automation, vol. 7, no. 3, pp. 278-288, June 1991, doi: 10.1109/70.88137.
[12] Ulrich Iwan, Borenstein Johann. “VFH/sup */: local obstacle avoidance with look-ahead verification”. In. IEEE. Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065). San Francisco. 2000:2505-2511.
[13] Ulrich Iwan, Borenstein Johann. “VFH+: reliable obstacle avoidance for fast mobile robots”. In. IEEE. Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146). Leuven. 1998:1572-1577.
[14] Khatib Oussama. “Real-time obstacle avoidance for manipulators and mobile robots”. In. IEEE. Proceedings of the 1985 IEEE international conference on robotics and automation. Saint Louis. 1985:500-505.
[15] Caianiello Pasquale, Presutti Domenico. “A Case Study on Goal Oriented Obstacle Avoidance”. In. Napoli Claudia di. Proceedings of the 16th Workshop "From Objects to Agents". Naples. 2015:142-145.
[16] P. X. Hien and G. -W. Kim, "Goal-Oriented Navigation with Avoiding Obstacle based on Deep Reinforcement Learning in Continuous Action Space," 2021 21st International Conference on Control, Automation and Systems (ICCAS), 2021, pp. 8-11, doi: 10.23919/ICCAS52745.2021.9649898.
[17] H. W. Jun, H. J. Kim and B. H. Lee, "Goal-Driven Navigation for Non-holonomic Multi-Robot System by Learning Collision," 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 1758-1764, doi: 10.1109/ICRA.2019.8793810.
[18] Lillicrap Timothy P., Hunt Jonathan J., Pritzel Alexander, et al. “Continuous control with deep reinforcement learning”. Computer ence. 8(6). 2015,9:A187.
[19] Y. Liu, H. Zhang, K. Long, A. Nallanathan, and V. C. M. Leung, “Energy-efficient Subchannel Matching and Power Allocation in NOMA Autonomous Driving Vehicular Networks,” IEEE Wireless Communications, vol. 26, no. 4, pp. 88-93, Aug. 2019.
[20] Laumond J.P., “Robot Motion Planning and Control”. Springer, Berlin, Heidelberg. 2005:171-253.

>> METHODS/EXPERIMENTAL SECTION

This paper does not involve special experiments

>>DECLARATIONS

- ETHICS APPROVAL AND CONSENT TO PARTICIPATE

this Manuscripts don’t involve human participants, human data or human tissue:

- Include a statement on ethics approval and consent

this Manuscripts not involving ethical approval and consent

- Include the name of the ethics committee that approved the study and the committee's reference number if appropriate.

the Studies not involving animals must include a statement on ethics approval

- CONSENT FOR PUBLICATION

Dear Editors:

We would like to submit the enclosed manuscript entitled "**Reinforce Learning-Based collision avoidance in Network Assisted Automated Driving**", which we wish to be considered for publication. No conflict of interest exits in the submission of this manuscript, and manuscript is approved by all authors for publication. I would like to declare on behalf of my co-authors that the work described was original research that has not been published previously, and not under consideration for publication elsewhere, in whole or in part. All the authors listed have approved the manuscript that is enclosed.

Yu Chen, Wei Han, Qinghua Zhu, Yong Liu, Jingya Zhao

- AVAILABILITY OF DATA AND MATERIAL

Please contact author for data requests."

- COMPETING INTERESTS

The authors declare that they have no competing interests.

- FUNDING

This work was supported by the Beijing City Board of education project (NO. KM202110858001).

- AUTHORS' CONTRIBUTIONS

Yu chen and Wei Han complete the overall algorithm design

Qinghua Zhu and Yong Liu complete the simulation design

Jingya Zhao complete the establishment of experimental environment and data sorting

- ACKNOWLEDGMENT

The authors would like to thank the associate professor Zhaoming Lu who is working at Beijing University of Posts and Telecommunications (P.R China) in particularly for the support in thesis examination and guidance. The authors are also grateful to the teachers of OAI WORKSHOP. (http://www.opensource5g.org/) for their invaluable support in deploying the system and in providing experimental data and technical documentation.

- AUTHORS' INFORMATION

Associate Professor：Yu Chen

E-mail: chenyu@bpi.edu.cn

Detailed-Address: 9 Liangshuihe 1st Street, Beijing Economic-Technological Development Area

Yu Chen is an Associate Professor at Beijing Polytechnic University and has PhD degree in communication engineering by Beijing University of Posts and Telecommunications. His current research interests focus on radio resource and mobility management, software defined wireless networks, and broadband multimedia transmission technology.

Associate Professor：Wei Han
E-mail: hanwei@bpi.edu.cn
Detailed-Address: 9 Liangshuihe 1st Street, Beijing Economic-Technological Development Area
Wei Han is an Associate Professor at Beijing Polytechnic and has Master degree in electrical automation by Southwest Jiaotong University. Her current research interests focus on Electronic Information Engineering Technology, Big data technology and application, Artificial Intelligence Application Technology.
Name：Qinghua Zhu
E-mail: zhuqinghua@bpi.edu.cn
Detailed-Address: 9 Liangshuihe 1st Street, Beijing Economic-Technological Development Area
Qinghua Zhu received his Bachelor's degree in computer software and application specialty from Beijing University, Beijing, China in 2001. He is currently working in Beijing Polytechnic University. His research interests are in the area of multimedia communication, QoE management, Green energy efficient network and network virtualization (SDN).
Name：Yong Liu
E-mail: liuyong@bpi.edu.cn
Detailed-Address: 9 Liangshuihe 1st Street, Beijing Economic-Technological Development Area
Yong Liu received his Bachelor's degree in communication engineering from Beijing University of Chemical Technology, Beijing, China, in 2004. His research includes green multimedia communication, QoE-centric network and application management and so on.

Name：Jingya Zhao
E-mail: zhaojingya@bpi.edu.cn
Detailed-Address: 9 Liangshuihe 1st Street, Beijing Economic-Technological Development Area
Jingya Zhao received her Master's degree in Beijing Institute of Technology in 2007. She joined the School of Telecommunications Engineering in Beijing Polytechnic University in 2000. Her research includes Open Wireless Networks, QoE management in wireless networks, software defined wireless networks, cross-layer design for mobile video applications and so on.