Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 2458-9403
Vol. 11 Issue 4, April - 2024

Enhancing Creo's Assembly Process Through Virtual Simulation And Q-Learning Algorithm

WU Hongyu
School of Mechanical Engineering, University of Shanghai for Science and Technology
Shanghai 200093,China
15534411086why@sina.com

ZHANG Sheng
School of optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology
Shanghai 200093, China
zhangsheng@usst.edu.cn

NI Jing
Business school, University of Shanghai for Science and Technology
Shanghai 200093, China
Nijin501@126.com

LI Shuai
DFH SATELLITE CO.LTD
Beijing 10089, China
ls0387@163.com

MIAO Tao
School of Mechanical Engineering, University of Shanghai for Science and Technology
Shanghai 200093, China
miao010126@163.com

ZHONG Liangwei
School of Mechanical Engineering, University of Shanghai for Science and Technology
Shanghai 200093,China
zlvcad@126.com

ZHANG Peng
School of Mechanical Engineering, University of Shanghai for Science and Technology
Shanghai 200093, China
hefengzxp@163.com

Abstract: In the design of equipment's overall assembly process, traditional approaches utilizing two-dimensional process cards fail to provide intuitive descriptions of the assembly process and are incapable of automatically generate assembly processes based on three-dimensional models, resulting in inadequate guidance for assembly work on the assembly site. To address this issue, a method that combines rigid body kinematics with the reinforcement learning Q-Learning algorithm has been proposed. This method automatically extracts assembly information from equipment models and plans the optimal assembly path. Furthermore, an assembly path planning plugin has been developed within the Creo software system. This plugin integrates virtual assembly technology to visualize the assembly process, thereby enhancing the efficiency and accuracy of the process design.

Keywords: Virtual assembly; interference detection; Q-Learning; Creo/TOOLKIT;Creo Secondary Development

I. INTRODUCTION

Traditional two-dimensional process cards are increasingly insufficient to meet modern production demands, prompting researchers to urgently seek an efficient and precise method for automated assembly. Virtual assembly, a process that utilizes computer technology for product design and manufacturing, enables rapid simulation of product design and manufacturing processes. This approach reduces trial and adjustment times, thereby improving production efficiency [1]. Many experts at home and abroad have conducted research in this area and have achieved certain results. Michael Grieves proposed a product lifecycle management method based on virtual assembly, which can help companies better manage and optimize the entire product lifecycle [2]. Wu Lingling developed a virtual assembly system based on VR virtual reality technology, which improves the interactivity and efficiency of virtual assembly through VR immersive simulation of the assembly process [3]. Zhang Peng explored a virtual assembly path planning technology for robotic arms based on Solid Works, reducing the debugging work of the robotic arm and improving the final product quality [4].

Based on these studies, the characteristics of products based on Creo assembly design are analyzed, and a functional module for automatically generating assembly processes and virtual assembly based on Creo assembly files is proposed. This method is based on Creo's overall assembly virtual assembly technology, using MFC programming technology and the Creo/TOOLKIT secondary development package under Creo, to simulate assembly routes and detect assembly collisions in components of the Creo assembly, finding the optimal assembly path to ensure the rationality and feasibility of the assembly process.

Assembly information acquisition:

Before simulating the assembly of a product, it is essential to identify the parent assembly to which the components belong, the assembly relationship information, and the posture information. These three

types of assembly information correspond to the hierarchical relationship information, constraint information, and position/posture information of the components. The extraction of these three types of assembly information is mainly achieved by traversing the model feature tree structure: the hierarchical relationship information can be obtained step by step through a top-down traversal, the posture information can be obtained from the component information, and the constraint information can be obtained under the assembly.

A. Extraction of BOM Information Hierarchy and Pose Relationships

After the product design is completed, a corresponding three-dimensional model is generated within the Creo system. This model contains features, and from this feature tree, the design Bill of Materials (BOM) can be extracted. After the design BOM is extracted, it is combined with supporting information to generate the process BOM. Finally, the design BOM and the process BOM are saved to the database.

B. Extraction of Hierarchy and Pose Relationships

Before conducting the simulated assembly, the components destined for assembly must be integrated into their respective higher-level assemblies and positioned to engage in constraint formation. Therefore, it is necessary to obtain the hierarchical relationships and pose information of the components.

In Creo, the hierarchical relationships of assemblies are expressed using a structure tree. As shown in Figure 1, the lower-level feature relationships in Creo are represented by feature numbers. In an assembly, the same component may participate in multiple assemblies, such as a standard part being involved in the installation of multiple devices. Although these standard parts correspond to a single physical object, they need to be distinguished during assembly. At this time, feature IDs are introduced for differentiation. In Figure 2, the feature ID table for component A is num=3, table[0]=1, table[1]=5, table[2]=8[5].

The pose information of components is encapsulated in a 4x4 matrix, which includes the translation and rotation information of the component relative to the origin. These information serve as important guidance for virtual path planning.

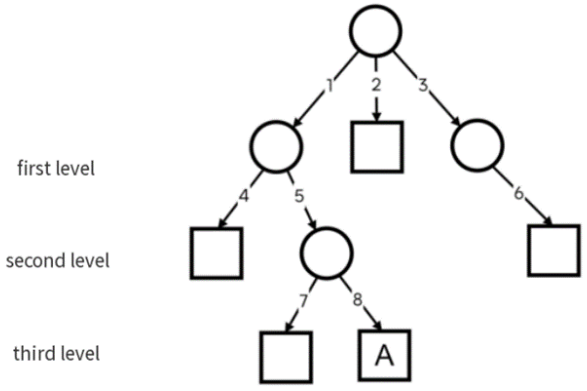

Fig. 1. Tree hierarchy of an assembly composition

(The ○ symbol represents a component and the □ symbol represents a part)

To obtain the hierarchical relationships of components, it is necessary to traverse all components starting from the top level of the assembly. The process and relevant APIs for extracting model hierarchy relationships and pose information are shown in Figure 2. The specific process is as follows:

Use the pfcGetProESession function under OTK to obtain the current session.

Obtain the current model pointer through the current session. The model pointer class contains functions for retrieving model information and properties. For example, these functions can be used to retrieve basic information such as the current model name, model path, and model pose matrix. Use the GetPosition function to retrieve the current model's pose matrix.

Use the GetType function under the model pointer to determine the current model type. If the current component is a part, retrieve basic information about the model using the model pointer. If it is an assembly, traverse the assembly and extract information.

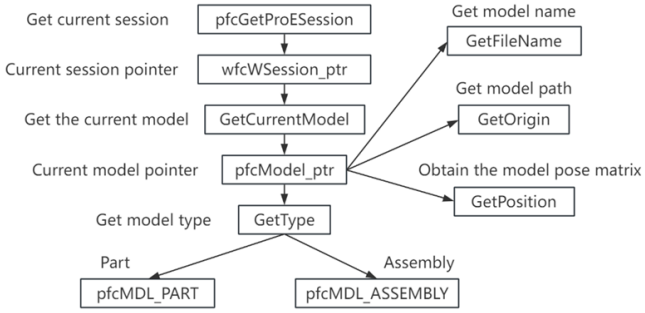

Fig. 2. Model hierarchy extraction process and related APIs

After extracting the model hierarchy information, the model hierarchy information will be associated with the product compatibility information imported when creating a new product, generating the final process BOM (Bill of Materials). This BOM is used for further process file design and displayed in the product structure tree.

C. Extraction of Assembly Relationships

Assembly relationships encompass the constraint information of components relative to reference features, including types such as coincidence, alignment, distance, and tangency, reflecting the constraint relationships between components and references. By moving components to determined positions based on hierarchical relationships and pose information, assembly relationships are established between components and references, completing the assembly process.

For assembly information within the assembly, simply add assembly information extraction methods to the next step of pfcComponentFeat_ptr in Figure 3.

Under the traversed subcomponent features of the assembly, obtain the array of constraint features for

the component, iterate through the array of constraint features, and obtain the constraint pointer. Using the constraint pointer function, obtain the constraint type and references of the assembly, as well as references of the assembly component. Convert the two constraints into model object pointers, and obtain their parent model pointers, thereby completing the extraction of constraint types and reference information in the constraint information. The extraction process of assembly information and the involved APIs are shown in Figure 4.

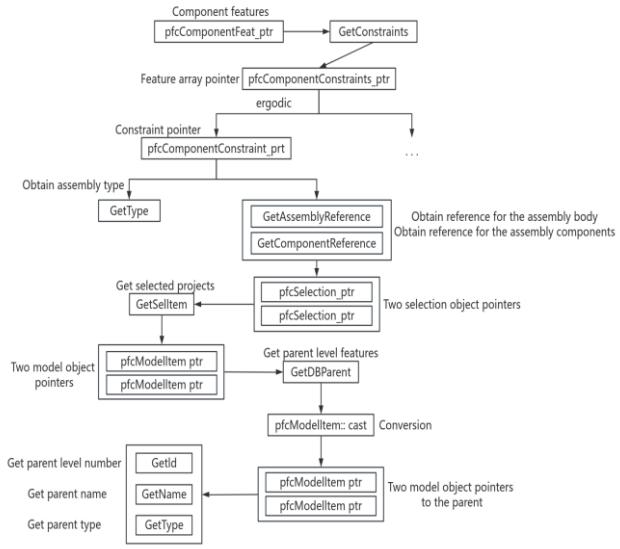

Fig. 3. Extraction process of assembly information

II. COMPONENT POSITION TRANSFORMATION AND INTERFERENCE DETECTION

After the design personnel place the component to be assembled at the starting point, the plugin can obtain its pose based on the corresponding OTK API. The starting and ending poses of the component to be assembled are known. To avoid interference during the translational and rotational motion of the component and continuously approach the final assembly position, the motion of the component needs to be differentiated to ensure smooth and continuous movement. The pose information of the component is a 4X4 numerical matrix, and its motion can introduce matrix transformations. Sequentially recording the pose information of each step in order to guide the completion of virtual assembly without interference.

A. Representation of Component Poses in Assemblies

In Creo assemblies, each component participating in the assembly within the assembly structure has its corresponding pose matrix. The pose matrix reflects the position and orientation of the component's own coordinate system relative to the assembly coordinate system.

B. Creation of Component Assembly Constraints and Assembly Interference Detection

For aligning the component's pose with the target pose, assembly constraints can be effectively established using the OTK C++ library provided by Creo during the final continuous pose transformation. The process is as follows:

Create a constraint group using the static method Create of pfcComponentConstraints.

Use the pfcCreateModelItemSelection method to obtain the assembly features of the component to be assembled and the assembly features on the assembly.

Use the static method Create of the pfcComponentConstraint class to create a constraint, specifying the type according to the actual constraint.

Use SetAssemblyReference to assign the obtained two assembly features to the created constraint.

Add the constraint to the constraint group created earlier, apply the constraint group using the SetConstraints method, and then refresh the model display to complete the assembly. If there are multiple constraints, add them to the constraint group and then apply the constraint group.

Interference detection is crucial in path planning, as it largely determines the outcome of the entire process. In Creo, interference detection is divided into two types: static global interference detection and real-time dynamic collision detection.

Static global interference detection is implemented when components within the assembly are not being manipulated. It detects interference among all components in the assembly. Dynamic collision detection, on the other hand, detects real-time interference only between the dragged components and other components in the assembly.

Comparing the two detection methods, because the transformation of component poses is not continuous motion, collision detection is not applicable. Therefore, static global interference detection is performed. This method utilizes ProFitGlobalinterferenceCompute to calculate all interference data within the assembly, thereby obtaining the interference status information of the current assembly components. The process of interference detection is illustrated in Figure 4.

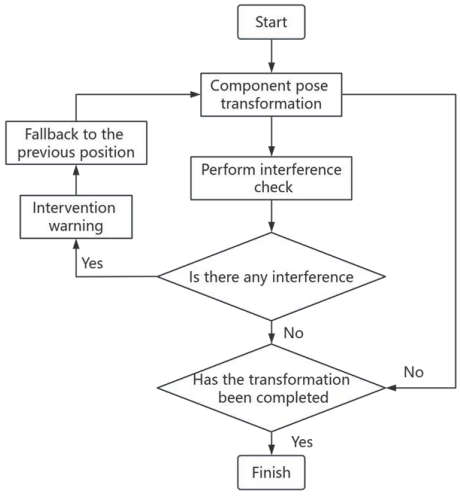

Fig. 4. Flow chart of Interference Detection Algorithm

III. Q-LEARNING BASED ASSEMBLY PATH PLANNING ALGORITHM

Automatically arrange the components involved in assembly before and after according to the sequential order of process steps and procedures in the process document. Then, combined with pose transformation matrices and interference detection technology, and utilizing Q-learning algorithm to plan paths, it is possible to achieve the planning of component assembly paths and create a complete simulation video of the assembly process.

A. Q-Learning Algorithm

The Q-learning algorithm is a classic algorithm in reinforcement learning, commonly used for path planning problems in complex environments. The 'Q' value, denoted as Q(s,a), signifies the expected reward associated with taking a specific action within a given state. When an agent takes an action from a certain state, the environment provides corresponding feedback in terms of a reward. The essence of this algorithm lies in creating a Q_Table, a matrix of expected rewards, that archives Q-values for each unique state-action combination. Then, actions that lead to maximum rewards are selected based on these Q-values. The rows of the Q_Table represent states, and the columns represent actions [8]. Figure 5 illustrates the process of maintaining the Q_Table within the context of this algorithm.

The training formula for the Q-learning algorithm can be expressed as follows: New estimate ← Old estimate + Step size * [Target - Old estimate]. In mathematical terms, it can be represented as follows:

$$\mathrm{Q(s,a)} \leftarrow \mathrm{Q(s,a)} + \alpha(\mathrm{R(s,a)} + \gamma \max_{\mathrm{a}} \mathrm{Q}(S', \mathrm{a}) - \mathrm{Q(s,a)})$$

In this equation, α represents the update step size, which ranges from $\alpha \in (0,1]$ and indicates the degree of influence of feedback on the current policy. R(s,a) denotes the reward function, typically represented in the form of a reward table. γ is the discount factor, indicating the importance of future rewards. In reinforcement learning algorithms, the discount factor helps avoid infinite loops when calculating rewards. $\max_a Q(S', a)$ represents the potential maximum expected reward of the state after executing the action. After each action is executed, Q(s,a) is calculated and updated in the Q_Table. Through continuous execution of actions and updating the Q-value table, the values of Q(s,a) in the table may vary significantly and tend to follow certain patterns after a certain number of training iterations.

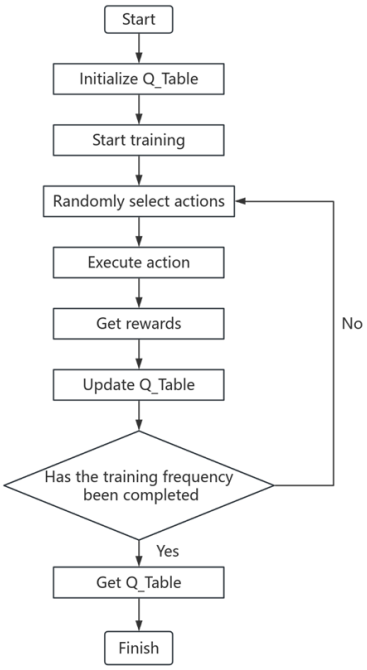

Fig. 5. The flow of the Q-learning algorithm to maintain the Q_Table

B. Algorithm Environment Modeling

Given that the Q-learning algorithm is contingent on environmental factors, it is imperative to conduct an environmental analysis and establish a model prior to problem-solving [9].

The Q-learning algorithm is based on states and actions, and during execution, immediate factors when executing actions are not considered. Therefore, in a narrow sense, it can be viewed as a nonlinear solving process. Creo, a three-dimensional software suite, offers a virtual space akin to real-world conditions, facilitating spatial modeling within its environment. The assembly process is simplified as follows: the assembly space is discretized into a grid, parts are represented as points, and motion changes are characterized by the pose transformations of these points, while rotational transformations are ignored. Considering the worst-case scenario for positions, the assembly space is gridified into an M × M × M three-dimensional array model, with each point being assigned a reward corresponding to its position, completing the virtualization of the assembly process.

The gridification of the assembly space is shown in Figure 6, where the start and end points of the assembly process are represented by startP and endP, respectively. Except for graphite points on the grid, every other point is represented by stepP. In the process of gridification, for each segment on the line connecting startP and endP, planes are generated at intervals of SegDistance to serve as square spatial partitioning planes. For each partitioning plane, the side length SegSideLength is chosen as the maximum value between the diagonal distance of the assembly body's bounding box and the distance between the assembly body and the part to be assembled. SegDistance is the partitioning distance. If the final segment distance to reach the end point is not equal to

SegDistance, the last partitioning plane passes directly through the end point. Gridification is then performed on each partitioning plane, with UnitD as the size of the grid unit. During the algorithm execution process, both SegDistance and UnitD affect the speed of the algorithm and the final generated assembly space. Therefore, careful consideration is required when selecting them. Here, UnitD is chosen as half of the diagonal distance of the bounding box of the part to be assembled, and SegDistance is chosen as the integer part of the diagonal distance of the bounding box. In Figure 10, UnitD is 10, and SegDistance is 20.

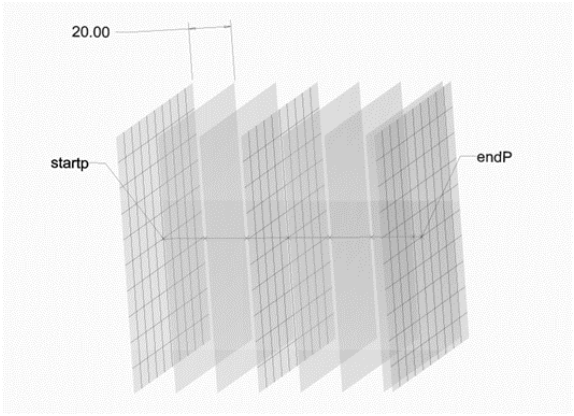

Fig. 6. Schematic diagram of assembly space grid

The movement of parts in the grid space is only specified by their movement on the current partitioning plane and the movement towards the next partitioning plane. There are 26 directions of displacement for the three-dimensional model in the grid space, and backward movement of the model is not allowed. Therefore, there are 17 actions for the movement of parts in the grid.

C. Selection of Algorithm Parameters and Specific Steps

To elucidate the algorithm more effectively, we initially define key variables, as shown in Table I

TABLE I. ALGORITHM KEY VARIABLES

Parameter Name	Parameter Size	Parameter Meaning
Move Direction i	1-17	Movement direction for the next action of the component
Algorithm Update Step Size α	0.8	Magnitude of the influence of the next state on the current state (0~1)
Algorithm Discount Factor γ	0.8	Expectation for the future (0~1)
Greedy Coefficient greedy	0.2	Avoiding falling into local optima
Training Times	5,000	Number of times the algorithm is trained to obtain the final Q_Table

With the greedy coefficient set at 0.2, Q-learning is essentially a greedy algorithm. However, by consistently selecting actions with the highest expected rewards, the algorithm risks not exploring other potential actions during training, potentially becoming trapped in a "local optimum" and failing to achieve the desired outcome. Therefore, by using the greedy coefficient, components have a probability of taking the optimal action and also a certain probability of exploring new paths.

The algorithm follows these specific steps:

Step 1: Obtain the components to be assembled in the current process and their corresponding assembly bodies based on the contents of the manual, open them, and move the components to be assembled to their assembly starting point.

Step 2: Get the assembly start point (startP) and assembly end point (endP) for the components, calculate the distance between the start and end points, as well as the diagonal distance of the assembly body bounding box to determine the side length of the cutting plane (SegSideLength), cutting plane distance (SegDistance), and grid cell size (UnitD).

Step 3: Based on the cutting plane side length (SegSideLength), number of cutting planes, and grid cell size (UnitD), establish the Q_Table matrix Q[x][y][z][i], and then establish the reward matrix R[x][y][z][i]. Perform interference detection for all positions in the grid. If interference is detected, set the reward for actions leading to that point as -100 to prevent the algorithm from considering this point. After interference detection, refine the reward matrix based on displacement distance and whether it moves towards the next cutting plane.

Step 4: Enter the training phase. During training, each grid that the components pass through receives a reward from the environment as the score for that position. Perform 5,000 training iterations to obtain the final Q_Table matrix.

Step 5: Retrieve the optimal path route from the Q_Table, move the components accordingly, and record the animation.

IV. RESULTS PRESENTATION

For instance, in the assembly scenario depicted in Figure 7, the internal structure of a product's box is illustrated, with red highlighting the already assembled components. The next step involves assembling the blue sensor bracket into the box. For some reasons, the initial position of the sensor bracket is fixed. If assembled directly in a straight line, it will collide with other components. The blue polyline delineates the final assembly path, with the user interface on the left displaying the critical nodes of this trajectory.

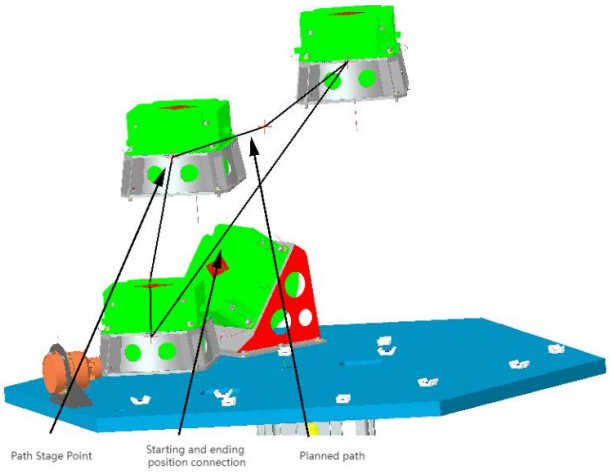

Fig. 7. Sensor bracket assembly path demonstration

V. CONCLUSION

This paper focuses on the assembly phase within the production process, integrating Creo 3D software and secondary development technologies to extract assembly model information. By employing reinforcement learning algorithms, the paper actualizes the simulation and optimization of the assembly path, thereby improving the efficiency and accuracy of the assembly phase in the product manufacturing process. The plane segmentation method, which employs an end-to-end connection approach, can effectively reduce the error rate of interference detection. The integration of the Q-learning algorithm facilitates the planning of a more rational assembly path, offering a solid foundation for the actual assembly process.

REFERENCES

[1] Chen Yaoyao, Liu Yongxia and Fu Chunming. Analysis of the development status of virtual assembly technology. mechanical engineering and automation, 2020, (06): 220-222. https://doi.org/10.3969/j.issn.1672-6413.2020.06.089.

[2] Grieves M. Digital Twin Certified: Employing Virtual Testing of Digital Twins in Manufacturing to Ensure Quality Products[J].Mach-ines,2023,11(8):808. https://doi.org/10.3390/MACHINES11080808

[3] Wu Lingling. Research and implementation of 3D immersive learning system for engineering drawing based on VR technology . South China University of technology, 2019.https://doi.org/10.27151/d.cnki.ghnlu.2019.001582

[4] Zhang Peng, Zhong Liangwei, Zhang Zenan. Research on virtual assembly path planning based on SolidWorks . software engineering, 2022,25 (03): 17-22. https://doi.org/10.27398/d.cnki.gxalu.2023.001284

[5] Zhao Jiaqi. Research on virtual assembly path planning technology based on Creo . aerospace manufacturing technology, 2016, (01): 61-67.https://kns.cnki.net/kcms2/article/abstract?v=fsvnL9wA1q0ZOGleRlfYTbjkLlR-79ImTyFcztoS6p3wvpOUJ31zFocU_l8wBTRiA8l0_TtakJ60HrrJJr8D4OqcYTcWKjFjP0QJlZRESVmvxreXo1YFxzCnxcZDJb4K23BRb6DYl2r1Hxl1Xp6zQ==&uniplatform=NZKPT&language=CHS

[6] Zhang Wenbin, Shen Jinghu, Jiang Zhaokang. Parametric variant design of parts based on secondary development of Creo. microcomputer applications, 2018,34 (02): 48-50+54. https://kns.cnki.net/kcms2/article/abstract?v=fsvnL9wA1q0lO9d9dFk00H36tQOjVVG41RehcLHaNqhc5MqwTPcD5BDaZMh6yVoDOp4aWS4ChFQgGk2SV8D3RLsodtpSGD6tvqHpkjkPTWXhgzVMKXcNTjUSkrP6T_NjUgLvQ4fY32MC1dLJyDBi9A==&uniplatform=NZKPT&language=CHS

[7] Han Xue. Research and application of robot motion function pose matrix . China new technology and new products, 2013, 21:2-3. https://doi.org/10.13612/j.cnki.cntp.2013.21.050

[8] Mao Guojun, Gu Shimin. Improved Q-learning algorithm and its application in path planning . Journal of Taiyuan University of technology, 2021,52 (01): 91-97. https://doi.org/10.16355/j.cnki.issn1007-9432tyut.2021.01.012

[9] Song Lijun, Zhou Ziyu, Li Yunlong. Research on path planning algorithm based on improved Q-learning . small microcomputer system, 2023:1-8. http://kns.cnki.net/kcms/detail/21.1106.tp.20230218.2208.008.html.

[10] Wei Ning. Guidelines for the implementation of deep reinforcement learning . Electronic Industry Press, 2021.