Adaptive Disassembly Sequence Planning for VR Maintenance Training via Deep Reinforcement Learning

Haoyang Mao
State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310027, China
Zhenyu Liu (liuzy@zju.edu.cn)
State Key Laboratory of CAD & CG, ZheJiang University
Chan Qiu
State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310027, China

Research Article

Keywords: disassembly sequence planning, deep reinforcement learning, genetic algorithm, VR maintenance training

Posted Date: May 14th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-497853/v1

Version of Record: A version of this preprint was published at The International Journal of Advanced Manufacturing Technology on November 17th, 2021. See the published version at https://doi.org/10.1007/s00170-021-08290-x.

[Title Page]

Adaptive Disassembly Sequence Planning for VR Maintenance Training via Deep Reinforcement Learning

Haoyang Mao, Zhenyu Liu*, Chan Qiu

State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310027, China

*Corresponding author. E-mail address: liuzy@zju.edu.cn.

Abstract:

Given the great inconvenience caused by the randomness of the fault to the maintenance work, it is necessary to perform on-site and efficient disassembly planning for the faulty parts and present them in combination with virtual reality (VR) technology to achieve rapid repair. As a promising method in solving dynamistic and stochastic problems, deep reinforcement learning (DRL) is adopted in this paper for the solution of adaptive disassembly sequence planning (DSP) in the VR maintenance training system, in which sequences can be generated dynamically based on user inputs. Disassembly Petri net is established to describe and model the disassembly process, and then the DSP problem is defined as a Markov decision process (MDP) that can be solved by the deep Q-network (DQN). For handling the temporal credit assignment with sparse rewards, the long-term return in DQN is replaced with the

fitness function of the genetic algorithm (GA). Meanwhile, the update method of gradient descent in DQN is adopted to speed up the iteration of the population in GA. A case study has been conducted to prove that the proposed method can provide better solutions for DSP problems in terms of VR maintenance training.

Keywords: disassembly sequence planning; deep reinforcement learning; genetic algorithm; VR maintenance training

Declarations:

Funding This study is supported by the National Natural Science Foundation of China (52075480, 51875517)，Key Research and Development Program of Zhejiang Province（2021C01008）and High-level Talent Special Support Plan of Zhejiang Province (2020R52004).

Conflicts of interest The authors declare that they have no conflict of interest.

Availability of data and material The data that supports the findings of this study are available from the corresponding author upon reasonable request.

Adaptive Disassembly Sequence Planning for VR Maintenance Training via Deep Reinforcement Learning

Abstract: Given the great inconvenience caused by the randomness of the fault to the maintenance work, it is necessary to perform on-site and rapid disassembly planning for the variable faulty parts and present them in combination with virtual reality (VR) technology to achieve an efficient repair. As a promising method in solving dynamistic and stochastic problems, deep reinforcement learning (DRL) is adopted in this paper for the solution of adaptive disassembly sequence planning (DSP) in the VR maintenance training system, in which sequences can be generated dynamically based on user inputs. Disassembly Petri net is established to describe and model the disassembly process, and then the DSP problem is defined as a Markov decision process (MDP) that can be solved by the deep Q-network (DQN). For handling the temporal credit assignment with sparse rewards, the long-term return in DQN is replaced with the fitness function of the genetic algorithm (GA). Meanwhile, the update method of gradient descent in DQN is adopted to speed up the iteration of the population in GA. A case study has been conducted to prove that the proposed method can provide better solutions for DSP problems in terms of VR maintenance training.

Keywords: disassembly sequence planning; deep reinforcement learning; genetic algorithm; VR maintenance training

1. Introduction

In view of the high proportion of maintenance costs in the lifecycle costs of products, seeking efficient maintenance methods has become an urgent problem in the manufacturing industry [1]. Due to the randomness of fault and the uncertainty of faulty parts, current maintenance methods rely heavily on the accurate memory and rich experience of workers, which has great instability and low efficiency. In order to improve this situation, VR training is a good attempt, in which many successful cases have emerged [2–5]. But for on-site maintenance training, the real-time disassembly sequence planning (DSP) based on user inputs is of great importance, which determines whether the worker can learn how to disassemble the faulty part as soon as possible so as to complete the fault repair.

Over the past decade, the meta-heuristic methods have been widely developed for solving DSP problems for its good ability in searching optimal or near-optimal solutions and generalization that does not depend on specific problems [6], e.g., genetic algorithm (GA) [7–10], artificial bee colony (ABC) [11–13], ant colony optimization (ACO) [14, 15], particle swarm optimization (PSO) [16, 17], simulated annealing (SA) [13], tabu search (TS) [18], and so forth. In order to integrate the advantages of different algorithms, some improved and new methods have been proposed in recent years to make up for the shortcomings of classic algorithms in

local search ability and convergence speed. Tao et al. [19] presented a TS-based hyper-heuristic algorithm with an exponentially decreasing diversity management strategy, which is proven more efficient when solving DSP for complex products. Inspired by the regenerative properties of the flatworm, Tseng et al. [20] proposed a novel Flatworm algorithm that has great mechanisms to avoid trapping into a local optimum of the DSP problem. Moreover, some fantastic algorithms such as artificial fish swarm (AFS) [21], fireworks algorithm (FWA) [22], firefly algorithm (FA) [23], and flower pollination algorithm (FPA) [24] are successively proposed for the solutions of DSP problems. Most of the swarm intelligence and evolutionary algorithms above are inspired by some random phenomena in nature and used for the DSP of goal-determined tasks. When it comes to on-site maintenance, due to the uncertainty of the disassembly targets caused by the randomness of the fault, the traditional meta-heuristics seem to become incompetent.

To overcome the above problems, Tuncel et al. [25] applied a reinforcement learning method based on Monte-Carlo for settling flexible disassembly problems, which extends new approaches to solving DSP problems in stochastic environments. However, similar to Xia et al. [26] who employed the Q-learning method for the selective disassembly of waste electrical and electronic equipment, as the complexity of the product increases, it is easy to cause a memory explosion. By introducing deep neural networks to achieve the approximation of the action-value function [27], deep reinforcement learning (DRL) methods have witnessed significant breakthroughs in several fields with dense rewards such as games [28–31], path planning [32–35], and

dynamic scheduling [36–38]. Even so, when applied to DSP problems, since the reward is only observed after a series of disassembly operations, it is quite difficult to associate actions with returns, which is referred to as the temporal credit assignment problem on sparse rewards.

Considering the problem of DSP with uncertain disassembly targets, this paper presents an improved deep Q-network (DQN) optimized by genetic algorithm (GA), namely GO-DQN, for the solution of adaptive disassembly sequence planning (ADSP) in VR maintenance training system. First, disassembly Petri net is used to model the disassembly process. Then, by defining the DSP problem as a Markov decision process, we introduce an improved DQN for adaptive sequence planning. Specifically, we replace the long-term return in DQN with the fitness function of GA to address temporal credit assignment in sequence planning. Finally, a VR maintenance training system is established to verify the effectiveness of the proposed method based on a case study.

The contributions and novelty of this paper are as follows:

(1) From the method perspective, GO-DQN inherits GA's ability to handle temporal credit assignment by its adoption of a fitness metric that integrates the return of an entire episode. Meanwhile, the update method based on gradient descent in DQN speeds up the iteration of the population in GA.

(2) From the technical application, the VR maintenance training system equipped with GO-DQN method can generate optimal sequences dynamically according to user

inputs, thereby meeting the requirements for cost-saving and efficient repair under the on-site maintenance environment with uncertain fault.

2. DRL method for adaptive disassembly planning

Taking into account the uncertainty of the targets to be disassembled at the maintenance site, this section represents an approach of ADSP based on DQN.

2.1 Modeling for the disassembly process

To describe and model the disassembly process during training, the method of Disassembly Petri Net (DPN) is introduced to this paper. A nine-tuple DPN is defined as follows:

$$DPN = (P, T, I_{\mathrm{n}\times m}, O_{m\times n}, u_0, G, C, D, W) \tag{2.1}$$

where

1) P is the set of places representing each disassembly state.

2) T is the set of transitions representing the disassembly operators, $P \mathrm{I}\ T = \varnothing$ and $P \mathrm{U} T \neq \varnothing$.

3) $I_{\mathrm{n}\times m} : P \times T \to N$ is the Matrix of $n \times m$ dimensions which indexes all the directed arcs from places to transitions, where n denotes the number of places, m denotes the number of transitions.

4) $O_{m\times n} : P \times T \to N$ is the Matrix of $m \times n$ dimensions which indexes all the directed arcs from transitions to places.

5) u_0 is a vector of n dimensions representing the initial state of disassembly.

6) G is the time gradients representing the time spent on each disassembly operation. To make the process easy to calculate, we divide it into 16 grades from 30 s to 30 mins. G is defined as follows:

$$G=\{g_1,g_2,...,g_n\},g_i\in\{0,1,2,...,15\}$$

7) C is the disassembly cost, including tools to use, change of directions. Similar to the time, the costs are divided into 4 gradients.

$$C=\{c_1,c_2,...,c_n\},c_i\in\{0,1,2,3\}$$

8) D is the degree of difficulty for disassembly. For convenience, we divide the difficulty into 5 levels, including very easy, easy, normal, hard, very hard.

$$D=\{d_1,d_2,...,d_n\},d_i\in\{1,2,3,4,5\}$$

9) W is the set of weights on the above three indexes.

Let E be the evaluation of the disassembly process, then it can be computed as:

$$E=\sum_{i=1}^{n}(w_1\cdot g_i+w_2\cdot c_i+w_3\cdot d_i)/i,w_1+w_2+w_3=1,w_i\in W \tag{2.2}$$

If there are transitions selected and fired, the next disassembly state u can be defined as:

$$u=u_0+f\cdot(O_{m\times n}-I^T_{n\times m}) \tag{2.3}$$

where f is the binary vector of m dimensions indicating whether the transition is fired.

A simple example of a product is shown in Fig.1, and the DPN can be expressed as follows:

$$I_{8\times5}=\begin{pmatrix}1&1&0&0&0\\0&0&1&1&0\\0&0&0&1&0\\0&0&0&0&1\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\end{pmatrix},\qquad O_{5\times8}=\begin{pmatrix}0&1&0&0&0&0&0&1\\0&0&1&0&1&0&0&0\\0&0&0&1&1&0&0&0\\0&0&0&1&0&0&0&1\\0&0&0&0&0&1&1&0\end{pmatrix},\quad u_0=(1,0,0,0,0,0,0,0)$$

If we disassemble the product in the order $A\to D\to B\to C$, that is $f=(0,1,0,1,1)$, then the new disassembly state can be computed by (2.3):

$$u=(0,0,0,0,1,1,1,1)$$

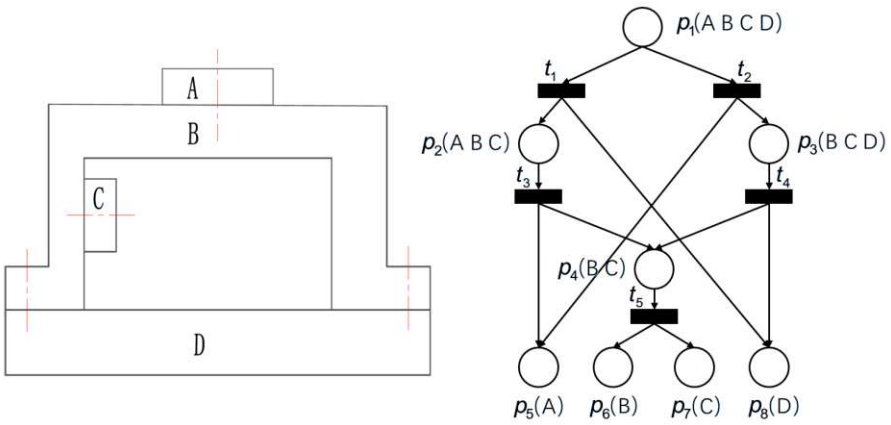

Fig.1 The 2D model and DPN for a simple product

2.2. Markov decision process

According to the theory of reinforcement learning, it is a process of constantly trying and discovering policy that maximizes reward. Since the probability and reward of transforming to the next disassembly state only depend on the current disassembly operation and state, having nothing to do with historical operations and actions, the disassembly task can be described as a Markov decision process (MDP).

MDP is defined by the 5-tuple $\langle \mathrm{S}, \mathrm{A}, \mathrm{P}, \mathrm{R}, \gamma \rangle$, where S indicates the finite state space, A indicates the finite action space, P indicates the probability of a state transition, R indicates the reward, and γ indicates the discount factor. Describe all the possible states of the agent's environment as a set S, where the current state is represented by $\mathrm{S}_t \in \mathrm{S}$ due to the temporality of the interaction process. During the interaction process, the action A_t is selected from the set of all the possible actions $\mathrm{A}(\mathrm{S}_t)$ in the current state S_t. After the agent executes the action A_t in the current state S_t, the environment will return a reward value $\mathrm{R}_{t+1} \in \mathrm{R}$ and reach a new state S_{t+1}.

During the MDP, two concepts are the focus of learning tasks: one is the strategy of selecting actions, which is also the ultimate learning goal of reinforcement learning; the other is the cumulative reward of evaluating the quality of the action policy, which is the indicators to adjust policy for reinforcement learning. Therefore, we define the policy by which the agent chooses actions at any time and state as the state-to-action mapping $\pi(a \mid s)$. Also, we define the cumulative reward for evaluating the quality of the policy as the return G_t, which represents the cumulative sum of the reward R from the current moment until the end of the task. Since the impact of reward on action policy will gradually weaken over time, G_t is defined as follows:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.4}$$

The discount rate $\gamma (0 < \gamma \leq 1)$ denotes the present value of future rewards. Now the problem of MDP is transformed into finding the policy $\pi(a \mid s)$ that maximizes the return G_t at any time and state.

2.3 Sequence Planning using DQN

Q-learning is a kind of reinforcement learning method that interacts with the environment and optimizes learning through real feedback from the environment. The agent randomly selects an action to act on the environment, then the environment executes the action to enter the next state and gives the agent feedback.

As to the disassembly problems in the VR training system, the disassembly states can be expressed by the set of places in DPN as well as the states in Q-learning, and the disassembly operations can be expressed by the transitions in DPN as well as the actions in Q-learning. Therefore, the problem of solving the optimal disassembly sequence in the VR system is transformed to find the optimal policy in Q-learning.

To effectively make a trade-off between exploration and exploitation during the disassembly process, $\varepsilon - greedy$ algorithm is adopted to obtain the optional policy. When the action a is selected by $\varepsilon - greedy$ and executed in the state s, a new state s' will be explored, which can be expressed as:

$$s' = s + (O_{m\times n}[a] - I^{T}_{n\times m}[a]) \tag{2.5}$$

where $O_{m\times n}[a]$ denotes the line a of the matrix $O_{m\times n}$ with $m\times n$ dimensions.

And the reward R_i of the action a that represents the $i-th$ operation in the state s can be computed:

$$R_i = w_1 \cdot g_i + w_2 \cdot c_i + w_3 \cdot d_i \tag{2.6}$$

Then the action-value function $Q(s,a)$ can be updated by the reward R_i:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left[R_i + \gamma \max_{a'\in A(s')} Q(s',a')\right] \tag{2.7}$$

where $\alpha(0<\alpha\le 1)$ is the learning rate, $\gamma(0\le\gamma\le 1)$ is the discount factor.

Since Q-learning algorithm directly learns the optimal strategy, it reduces the exploration of the environmental space. However, it is greatly affected by the samples, making it easier to get trapped into a local optimum. Furthermore, the updating method based on Q-table has certain limitations on the magnitude of the data, especially for complex assemblies with too many disassembly sequences, which is likely to cause a memory explosion.

To solve the above problems, we introduce two neural networks to replace the Q-table, one of which generates the value function approximation of the current state, called *Q_evaluate_net*, while the other generates the value function approximation of the next state, called *Q_target_net*, which is used to calculate the target approximation of the current value function. Both networks can be expressed as:

$$Q(s,a,\theta) \approx Q(s,a) \tag{2.8}$$

where θ is the parameter of the fitting function, representing the weight of the connection between two layers of neurons in the network.

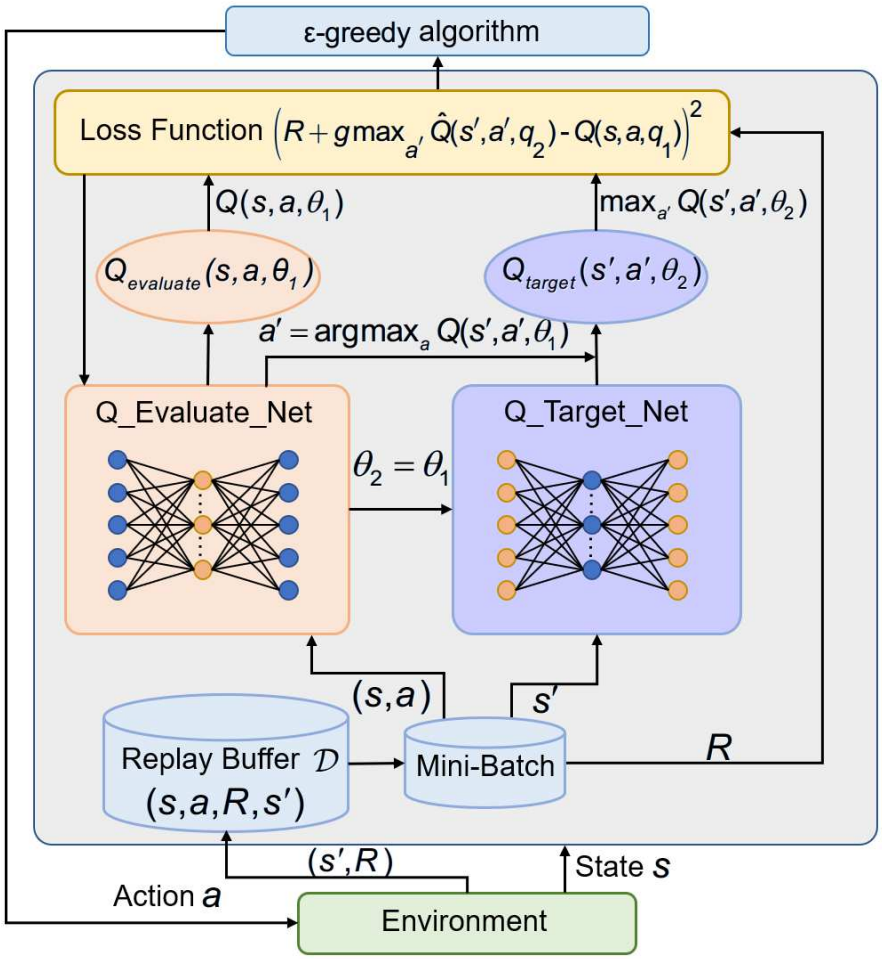

Fig.2 The framework of DRL method for ADSP

As shown in Fig.2, the *Q_evaluate_net* outputs $Q_{evaluate}(s,a,\theta_1)$. For the *Q_target_net,* it chooses the action for the next state that yields $\arg\max_a Q(s',a',\theta_1)$ in the *Q_evaluate_net*, which products the Q-value of the next state-action pair in the target network, i.e., $Q_{target}(s',a',\theta_2)$. Based on this Q-value, the action a is selected by $\varepsilon-greedy$ algorithm and executed in the state s, receiving the reward R and then transferred to the next state s', all of which is stored in the replay memory D as a tuple (s,a,R,s'). During this process, T mini-batch samples are randomly extracted from the replay buffer D to update θ_1. After that, the fitting action-value $y_i=r+\gamma\max_{a'}\hat{Q}(s',a',\theta^-)$ can be calculated, and then the loss is required by:

$$L(\theta) = \mathrm{E}\left[\left(R + \gamma \max_{a'} \hat{Q}(s', a', \theta_2) - Q(s, a, \theta_1)\right)^2\right] \tag{2.9}$$

Then the gradient descent method is applied to update θ_1 of the *Q_evaluate_net*. Further, θ_2 is updated as $\theta_2 = \theta_1$ after a fixed interval.

3. Optimization of DQN based on GA

In the traditional DRL method, the agent learns the experience continuously through the immediate reward acquired by interacting with the environment. But when the reward is sparse, associating actions with returns becomes quite difficult, as mentioned at the beginning. Besides, due to the high sensitivity to hyperparameters for DRL, it is still a hard task to adjust these hyperparameters such as discount factors, learning rates, etc.

The method of crossover and mutation in GA makes the adaptation degree continuously increase, which is similar to the idea of approximation in the neural network. The use of fitness indicators that merge the returns of the entire episode makes it indifferent for GA to the sparsity of reward distribution and robust to long-term horizons[39]. Based on the above thought, we propose an approach of improved DQN optimized by GA named GO-DQN for the solution of optimal sequence in VR training system, which uses the return value in RL as the fitness function:

$$F_{fitness} = E_N(G_t) = E_N(\sum_{k=1}^{l} \gamma^k R_{t+k+1}) \tag{3.1}$$

where N denotes the number of episodes, l denotes the length of an episode, G_t denotes the total return of each episode.

The architecture of GO-DQN is shown in Fig.3, and the algorithm mainly includes two parts: one is the DQN training process optimized by the GA, the other is the evolution process of the GA guided by the gradient descent of DQN.

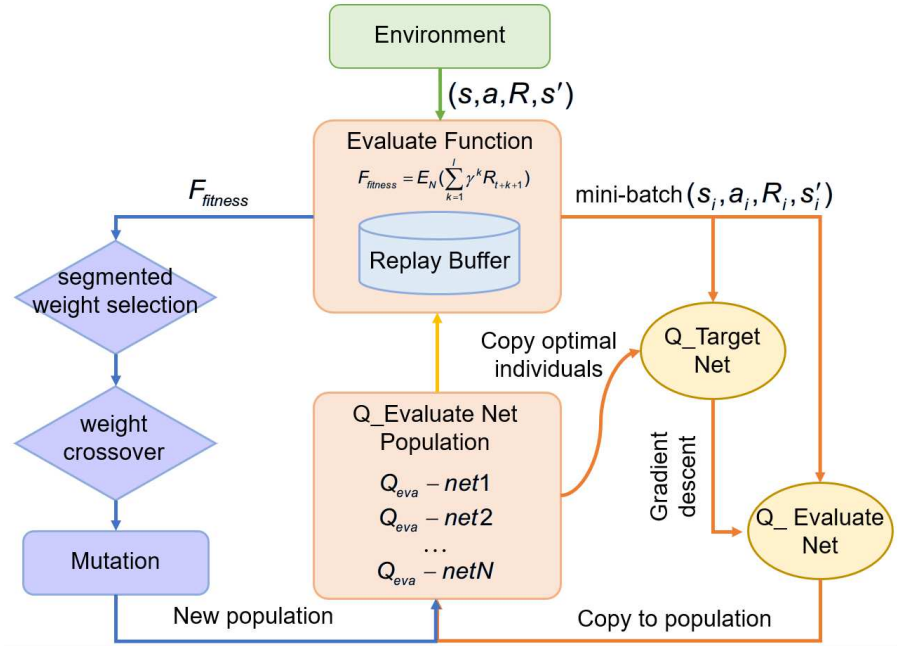

Fig.3 The architecture of GO-DQN

Step 1: Initialization. A population of *Q_evaluate_nets* is initialized as pop_k, as well as an empty cyclic replay buffer. Moreover, one additional *Q_evaluate_net* alongside *Q_target_net* is initialized as Q_{eva} and Q_{tag} with weight $\theta^{Q_{eva}}$ and $\theta^{Q_{tag}}$, respectively.

Step 2: Elites selection. Recycle the interactions between individuals in the population and the environment, and use the cumulative reward as the fitness function value of the individual to select elites.

Step 3: Crossover and mutation. Select individuals from the population by segmented weight selection, from which randomly select individuals as parents with elites for crossover and mutation to create the next generation of *Q_evaluate_nets*.

Step 4: Networks update. Periodically, we sample a random mini-batch of T transitions (s_i, a_i, R_i, s_i') from the replay buffer D , and acquire the action-value function:

$$y_i = R_i + \gamma Q_{tag}(s_i', a_i', \theta^{Q_{tag}}) \tag{3.2}$$

Then the loss function is computed as follows:

$$L = 1/T \sum_i (y_i - Q_{eva}(s_i, a_i, \theta^{Q_{eva}}))^2 \tag{3.3}$$

Therefore, Q_{eva} can be updated by minimizing the loss, and the weights can be updated as well.

Finally, we copy the *Q_evaluate_net* Q_{eva} into the population pop_k:

for optimum $\theta^{Q_{eva}} \in pop_k$: $\theta^{Q_{eva}} \Rightarrow \theta^{Q_{tag}}$

Considering the feature of Q networks in DQN, a method of segmented weight selection is proposed to realize the selection of parents. By segmenting the sorted populations, and then selecting reproduction individuals in each segment according to the weight calculated by the fitness value, both the excellent and inferior groups are ensured to have individuals to achieve reproduction.

4. System implementation and case study

4.1 System design and implementation

Embedding the GO-DQN method, the overall architecture of the training system consists of three major layers: data layer, function layer, and interaction layer, as is shown in Fig.4. 3D models and interference matrix of the industrial products to be disassembled are prepared in advance and stored in the data layer, which are used for training RL models. Once the sequences are generated in the function layer, the

system will create disassembly animations for demonstration automatically and stored them in the data layer. And VR guidance instructions for training and evaluation will be created as well. The interaction layer supplies an interactive interface to allow users to specify their input preferences and accept corresponding feedback. Other VR modules, such as collision detection, haptic feedback, tracking, and rendering, have been developed to realize immersive training.

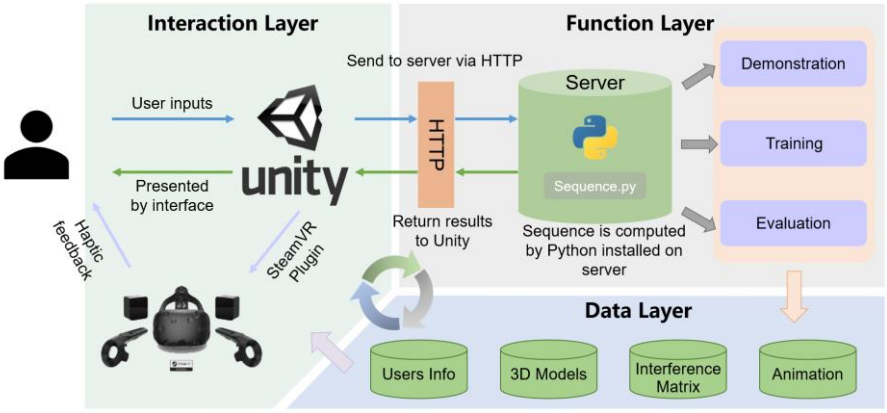

Fig.4 Architecture and workflow of the training system

In order to realize the data transmission between function layers, a server-client communication module has been established to integrate the disassembly sequence planning module with the VR application. A local server with Python installed is set up to handle requests from Unity.

When Unity receives the information of user inputs, it will send requests to the server, which relies on the UnityWebRequest system. Then the server will call the Python scripts for sequence calculation and return the results to Unity in JSON format with the help of a web framework called Flask. Unity presents the results to the user through the user interface (UI) and communicates with HTC VIVE at the same time to achieve multi-channel information feedback.

In order to obtain the best experience, multi-sensory feedback is realized in this system integrating visual, auditory, and haptic feedback. Visual feedback includes the feedback of part status and text information. The selected part highlighting algorithm is developed to realize part status feedback. When the virtual hand touches the part to be disassembled, the outline of the part will be highlighted to indicate that the selection is correct. When the disassembly operation is completed, the part will return to its original state, and the panel will prompt the user for the next operation. If the user makes a wrong choice, the handle will vibrate to remind the user, and the correct operation will be voiced at the same time, as shown in Fig.5.

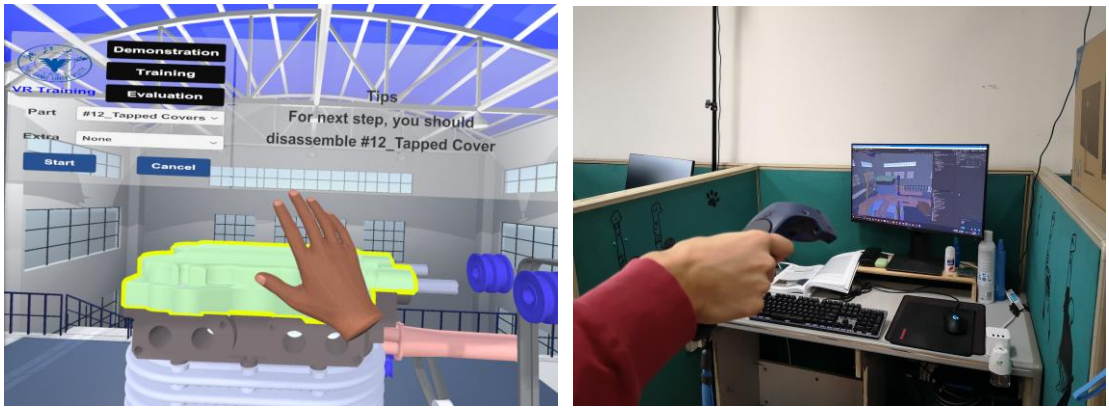

Fig.5 Feedback of vision and haptics during operation

4.2 Case study on aircraft engine

The case model of the aircraft engine used in this study is shown in Fig.6, which consists of 25 parts. Since the complete aircraft engine involves lots of parts, to facilitate subsequent calculations, we have omitted the connectors between parts such as bolts, nuts, etc. Similarly, some trivial parts that do not affect the overall disassembly sequence are ignored as well. Moreover, at any time during the entire

training process, this product structure diagram can be called up for a quick overview by pulling the menu key on the handle.

To train the RL model based on the proposed method, a modified MountainCar-v0 environment has been developed on the basis of OpenAI Gym. Each step is defined as an instance where the agent takes an action and receives a reward back from the environment, and all the steps taken by all *Q_evaluate_nets* in the population are cumulative. During each training generation, the *Q_evaluate_net* with the highest fitness is selected as the best and then tested on 5 task instances, and the average score is logged as its performance.

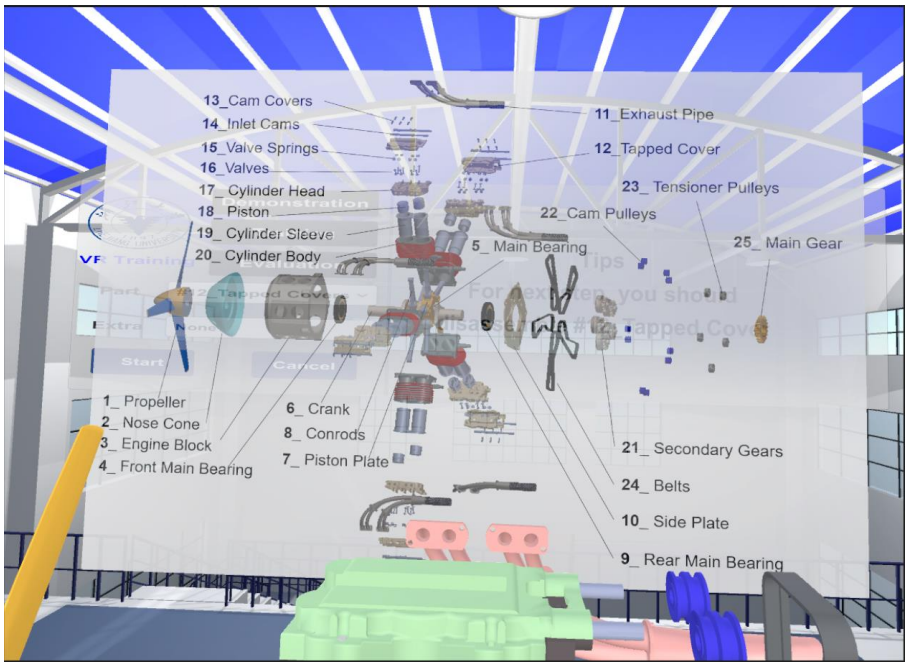

Fig.6 The structure of the aircraft engine

Then three experiments have been designed to verify the validity of our method. In experiment I，we select the rear main bearing(#9) as the only disassembly target. For experiment II，we select the nose cone (#2), the rear main bearing(#9), and tapped covers(#12) as the disassembly targets with no additional preferences. As to

experiment III，we choose additional preferences which are to exclude cam pulleys (#22) and the exhaust pipes(#11).

4.3 Results and discussion

As for the on-site maintenance, it is usually more desirable to use a shorter time and less cost to complete the training so as to fix the fault as soon as possible. Therefore, we set the weights of each index in the formula (2.6) as: $w_1 = 0.5, w_2 = 0.4, w_3 = 0.1$. The size of the population is set to 10, and the elite fraction is set to 0.1. The mutation probability is set to 0.5 with the expectation for more searches in the solving space, and the mutation fraction is set to 0.1 as well as the mutation strength corresponding to a 10% Gaussian noise. Adam optimizer with a learning rate of 0.0001 is used while the discount rate is set to 0.99. The training process is shown in Fig.7, and the results for the above three experiments are represented in Fig.1 and Fig.8.

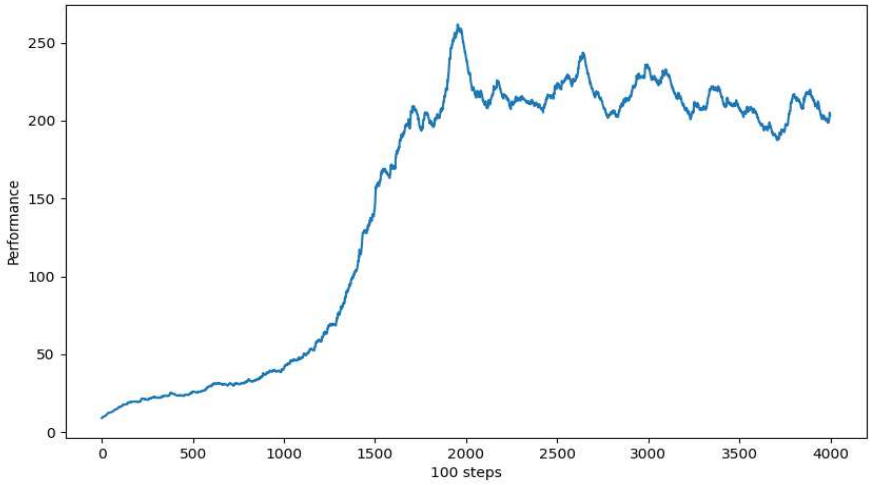

Fig.7 The training process of RL model based on GO-DQN

Table 1. Experimental results

Experiment I	Target(s)	#9
	Sequence	25-24-23-22-21-11-10-9
Experiment II	Target(s)	#2, #9, #12
	Sequence	1-2-25-24-23-22-21-11-10-9-12
Experiment III	Target(s)	#2, #9, #12(Avoid #11, #22)
	Sequence	No sequence available

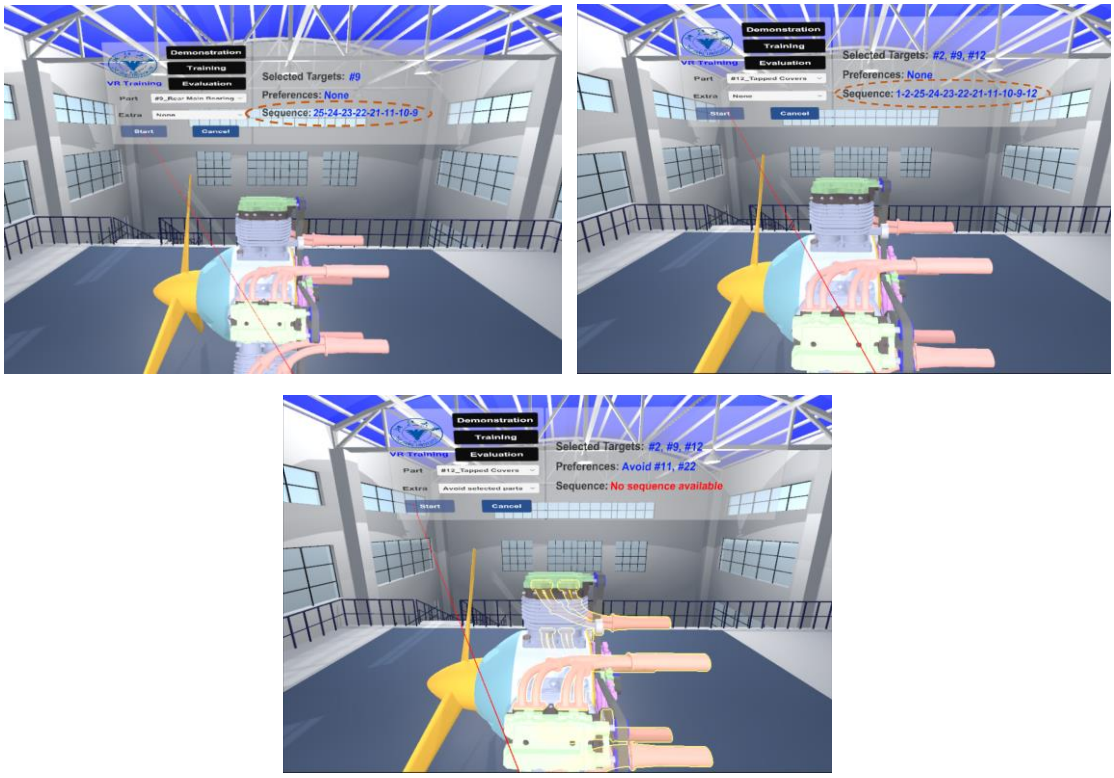

Fig.8 Running results of the three experiments in the training system

As shown in Table 1, the developed system is able to automatically generate the optimal disassembly sequence according to the different inputs of the trainee. Compared with the previous RL method[25, 26], the proposed GO-DQN algorithm can handle the temporal credit assignment in DSP tasks quite well, which attributes to the following two improvements: (1) replace the return in DQN with the fitness

function of GA, avoiding the dependence on the immediate reward, and (2) update the optimal individuals of the population in each iteration by gradient descent to speed up the convergence. Since the states experienced by individuals with high fitness have a higher long-term return, their experience will be more likely to be stored in the replay buffer, which is a form of implicit prioritization effective for domains with long-term horizons and sparse rewards.

In Experiment I, the system calculates the optimal disassembly sequence 25-24-23-22-21-11-10-9 and converts it into operable instruction after the user selects the target part #9, the whole process of which takes 0.7 seconds on average for 50 attempts. When the targets increase to three, as set in Experiment II, the system is still able to immediately calculate the optimal sequence, which only takes 0.2s longer than Experiment I. In Experiment III, after the user chooses additional preferences which are to exclude #22 and #11, the system is unable to acquire any feasible sequence, this is because they are the parts that must be removed before the disassembly of part #2, #9, #12，which is determined by the constraint relationships between the parts, as shown in Fig.6.

As a summary of the above analysis, GO-DQN inherits GA's invariance to sparse rewards with long-term horizons as well as DQN's ability to leverage gradients for higher sample efficiency. Benefiting from these advantages, the constructed system can generate optimal disassembly sequences within a short time after receiving the user inputs.

In most VR systems for training or maintenance, the disassembly sequence of the product is usually authored just for a specific sequence and stored in the database[5, 40, 41]. When the faulty part is not in the specific sequence, the system will not work normally. In this research, for any input part of the specified product, the system can quickly generate the optimal disassembly sequence according to the designated requirements of the user for guidance. Admittedly, with the increase in the complexity of the input products, the training cycle of the RL model will extend. How to shorten the training process is one direction of our future exploration. In addition, the system can also be used in different production lines and maintenance workshops by importing the models and disassembly matrices of other new products.

5. Conclusion and future work

In order to solve the high cost and giant threshold of equipment maintenance caused by the randomness of faults and the uncertainty of faulty parts in the maintenance task, this paper presents an improved DQN optimized by GA, named GO-DQN, for the solution and optimization of adaptive disassembly sequence planning in an on-site interactive VR maintenance training system. Disassembly Petri net is used to describe and model the disassembly process, and then the DQN method is introduced for solving the DSP problem defined by the MDP. To avoid the dependence on the immediate reward, we replace the return in DQN with the fitness function of GA. Based on a case study of the aircraft engine, the system equipped with the proposed method is verified to have better performance for the on-site maintenance when the fault is uncertain.

In summary, the main contributions of our work are as follows:

(1) From the method perspective, GO-DQN inherits GA's ability to handle temporal credit assignment by its adoption of a fitness metric that integrates the return of an entire episode, as well as DQN's update method based on gradient descent for speeding up the iteration.

(2) From the technical application, the VR maintenance training system equipped with GO-DQN method can generate optimal sequences dynamically based on user inputs, thus meeting the needs for cost-saving and efficient repair under the on-site maintenance environment with uncertain fault.

For future work, we will explore the following aspects:

1) Combine Petri nets with graph neural networks to seek more efficient expressions for the disassembly process.

2) Introduce heuristic knowledge such as expert experience for guidance and cloud server for training, so as to accelerate the training process.

3) Verify the robustness and efficiency of the system in much more complex and changeable environments.

Declarations

Funding This study is supported by the National Natural Science Foundation of China (52075480, 51875517)，Key Research and Development Program of Zhejiang Province（2021C01008）and High-level Talent Special Support Plan of Zhejiang Province (2020R52004).

Conflicts of interest The authors declare that they have no conflict of interest.

References

1. Guo Z, Zhou D, Zhou Q, et al (2020) Applications of virtual reality in maintenance during the industrial product lifecycle: A systematic review. J Manuf Syst 56:525–538. https://doi.org/10.1016/j.jmsy.2020.07.007
2. Berg LP, Vance JM (2017) Industry use of virtual reality in product design and manufacturing: a survey. Virtual Real 21:1–17. https://doi.org/10.1007/s10055-016-0293-9
3. Roldán JJ, Crespo E, Martín-Barrio A, et al (2019) A training system for Industry 4.0 operators in complex assemblies based on virtual reality and process mining. Robot Comput-Integr Manuf 59:305–316. https://doi.org/10.1016/j.rcim.2019.05.004
4. Abidi MH, Al-Ahmari A, Ahmad A, et al (2019) Assessment of virtual reality-based manufacturing assembly training system. Int J Adv Manuf Technol 105:3743–3759. https://doi.org/10.1007/s00170-019-03801-3
5. Yang Y, Yang P, Li J, et al (2019) Research on virtual haptic disassembly platform considering disassembly process. Neurocomputing 348:74–81. https://doi.org/10.1016/j.neucom.2018.05.120
6. Gao KZ, He ZM, Huang Y, et al (2020) A survey on meta-heuristics for solving disassembly line balancing, planning and scheduling problems in remanufacturing. Swarm Evol Comput 57:100719. https://doi.org/10.1016/j.swevo.2020.100719
7. Giudice F, Fargione G (2007) Disassembly planning of mechanical systems for service and recovery: a genetic algorithms based approach. J Intell Manuf 18:313–329. https://doi.org/10.1007/s10845-007-0025-9
8. Tseng Y-J, Kao H-T, Huang F-Y (2010) Integrated assembly and disassembly sequence planning using a GA approach. Int J Prod Res 48:5991–6013. https://doi.org/10.1080/00207540903229173
9. Tseng H-E, Chang C-C, Lee S-C, Huang Y-M (2018) A Block-based genetic algorithm for disassembly sequence planning. Expert Syst Appl 96:492–505. https://doi.org/10.1016/j.eswa.2017.11.004

10. Ren Y, Zhang C, Zhao F, et al (2018) An asynchronous parallel disassembly planning based on genetic algorithm. Eur J Oper Res 269:647–660. https://doi.org/10.1016/j.ejor.2018.01.055
11. Ren Y, Tian G, Zhao F, et al (2017) Selective cooperative disassembly planning based on multi-objective discrete artificial bee colony algorithm. Eng Appl Artif Intell 64:415–431. https://doi.org/10.1016/j.engappai.2017.06.025
12. Liu J, Zhou Z, Pham DT, et al (2018) An improved multi-objective discrete bees algorithm for robotic disassembly line balancing problem in remanufacturing. Int J Adv Manuf Technol 97:3937–3962. https://doi.org/10.1007/s00170-018-2183-7
13. Tian G, Ren Y, Feng Y, et al (2019) Modeling and Planning for Dual-Objective Selective Disassembly Using and/or Graph and Discrete Artificial Bee Colony. IEEE Trans Ind Inform 15:2456–2468. https://doi.org/10.1109/TII.2018.2884845
14. Luo Y, Peng Q, Gu P (2016) Integrated multi-layer representation and ant colony search for product selective disassembly planning. Comput Ind 75:13–26. https://doi.org/10.1016/j.compind.2015.10.011
15. Tseng H-E, Chang C-C, Lee S-C, Huang Y-M (2019) Hybrid bidirectional ant colony optimization (hybrid BACO): An algorithm for disassembly sequence planning. Eng Appl Artif Intell 83:45–56. https://doi.org/10.1016/j.engappai.2019.04.015
16. Tseng Y-J, Yu F-Y, Huang F-Y (2011) A green assembly sequence planning model with a closed-loop assembly and disassembly sequence planning using a particle swarm optimization method. Int J Adv Manuf Technol 57:1183–1197. https://doi.org/10.1007/s00170-011-3339-x
17. Li WD, Xia K, Gao L, Chao K-M (2013) Selective disassembly planning for waste electrical and electronic equipment with case studies on liquid crystaldisplays. Robot Comput-Integr Manuf 29:248–260. https://doi.org/10.1016/j.rcim.2013.01.006
18. Alshibli M, El Sayed A, Kongar E, et al (2016) Disassembly Sequencing Using Tabu Search. J Intell Robot Syst 82:69–79. https://doi.org/10.1007/s10846-015-0289-9
19. Tao F, Bi L, Zuo Y, Nee AYC (2018) Partial/Parallel Disassembly Sequence Planning for Complex Products. J Manuf Sci Eng 140:011016. https://doi.org/10.1115/1.4037608

20. Tseng H-E, Huang Y-M, Chang C-C, Lee S-C (2020) Disassembly sequence planning using a Flatworm algorithm. J Manuf Syst 57:416–428. https://doi.org/10.1016/j.jmsy.2020.10.014
21. Zhang Z, Wang K, Zhu L, Wang Y (2017) A Pareto improved artificial fish swarm algorithm for solving a multi-objective fuzzy disassembly line balancing problem. Expert Syst Appl 86:165–176. https://doi.org/10.1016/j.eswa.2017.05.053
22. Lu C, Li J-Y (2017) Assembly sequence planning considering the effect of assembly resources with a discrete fireworks algorithm. Int J Adv Manuf Technol 93:3297–3314. https://doi.org/10.1007/s00170-017-0663-9
23. Zhu L, Zhang Z, Wang Y (2018) A Pareto firefly algorithm for multi-objective disassembly line balancing problems with hazard evaluation. Int J Prod Res 56:7354–7374. https://doi.org/10.1080/00207543.2018.1471238
24. Wang K, Li X, Gao L (2019) A multi-objective discrete flower pollination algorithm for stochastic two-sided partial disassembly line balancing problem. Comput Ind Eng 130:634–649. https://doi.org/10.1016/j.cie.2019.03.017
25. Tuncel E, Zeid A, Kamarthi S (2014) Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning. J Intell Manuf 25:647–659. https://doi.org/10.1007/s10845-012-0711-0
26. Xia K, Gao L, Li W, et al (2014) A Q-Learning Based Selective Disassembly Planning Service in the Cloud Based Remanufacturing System for WEEE. In: Volume 1: Materials; Micro and Nano Technologies; Properties, Applications and Systems; Sustainable Manufacturing. American Society of Mechanical Engineers, Detroit, Michigan, USA, p V001T04A012
27. Mnih V, Kavukcuoglu K, Silver D, et al (2013) Playing Atari with Deep Reinforcement Learning. Comput Sci
28. Moravčík M, Schmid M, Burch N, et al (2017) DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. Science 356:508–513. https://doi.org/10.1126/science.aam6960
29. Jaderberg M, Czarnecki WM, Dunning I, et al (2019) Human-level performance in 3D multiplayer games with population-based reinforcement learning. Science 364:859–865. https://doi.org/10.1126/science.aau6249

30. Silver D, Hubert T, Schrittwieser J, et al (2017) Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. ArXiv171201815 Cs
31. Sutton RS, Barto AG (2018) Reinforcement Learning: An Introduction (2nd Edition). The MIT
32. Fan T, Long P, Liu W, Pan J (2020) Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. Int J Robot Res 39:856–892. https://doi.org/10.1177/0278364920916531
33. Kim M, Han D-K, Park J-H, Kim J-S (2020) Motion Planning of Robot Manipulators for a Smoother Path Using a Twin Delayed Deep Deterministic Policy Gradient with Hindsight Experience Replay. Appl Sci-Basel 10:575. https://doi.org/10.3390/app10020575
34. Jiang L, Huang H, Ding Z (2020) Path planning for intelligent robots based on deep Q-learning with experience replay and heuristic knowledge. IEEE-CAA J Autom Sin 7:1179–1189. https://doi.org/10.1109/JAS.2019.1911732
35. Liu Q, Shi L, Sun L, et al (2020) Path Planning for UAV-Mounted Mobile Edge Computing With Deep Reinforcement Learning. IEEE Trans Veh Technol 69:5723–5728. https://doi.org/10.1109/TVT.2020.2982508
36. Luo S (2020) Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. Appl Soft Comput 91:106208. https://doi.org/10.1016/j.asoc.2020.106208
37. Shi D, Fan W, Xiao Y, et al (2020) Intelligent scheduling of discrete automated production line via deep reinforcement learning. Int J Prod Res 58:3362–3380. https://doi.org/10.1080/00207543.2020.1717008
38. Kim YG, Lee S, Son J, et al (2020) Multi-agent system and reinforcement learning approach for distributed intelligence in a flexible smart manufacturing system. J Manuf Syst 57:440–450. https://doi.org/10.1016/j.jmsy.2020.11.004
39. Khadka S, Tumer K (2018) Evolution-guided policy gradient in reinforcement learning. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, pp 1196–1208

40. Jiang W, Zheng J, Zhou H, Zhang B (2016) A new constraint-based virtual environment for haptic assembly training. Adv Eng Softw 98:58–68. https://doi.org/10.1016/j.advengsoft.2016.03.004
41. Ahmad A, Al-Ahmari AM, Aslam MU, et al (2015) Virtual Assembly of an Airplane Turbine Engine. IFAC-Pap 48:1726–1731. https://doi.org/10.1016/j.ifacol.2015.06.335

Figures

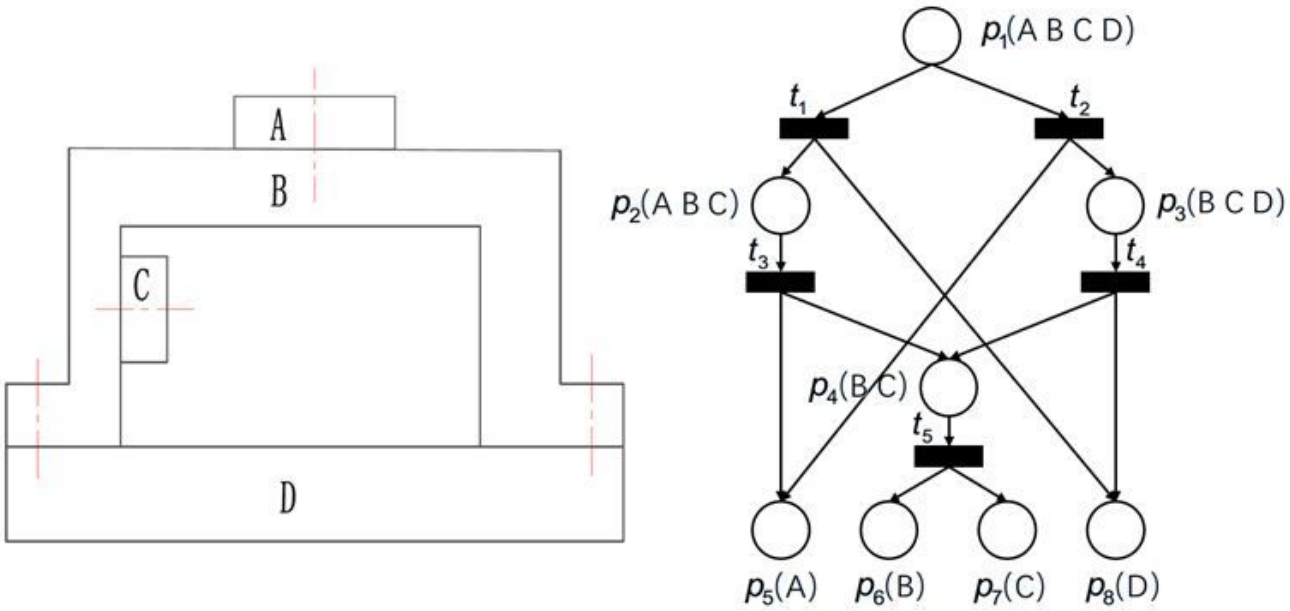

Figure 1

The 2D model and DPN for a simple product

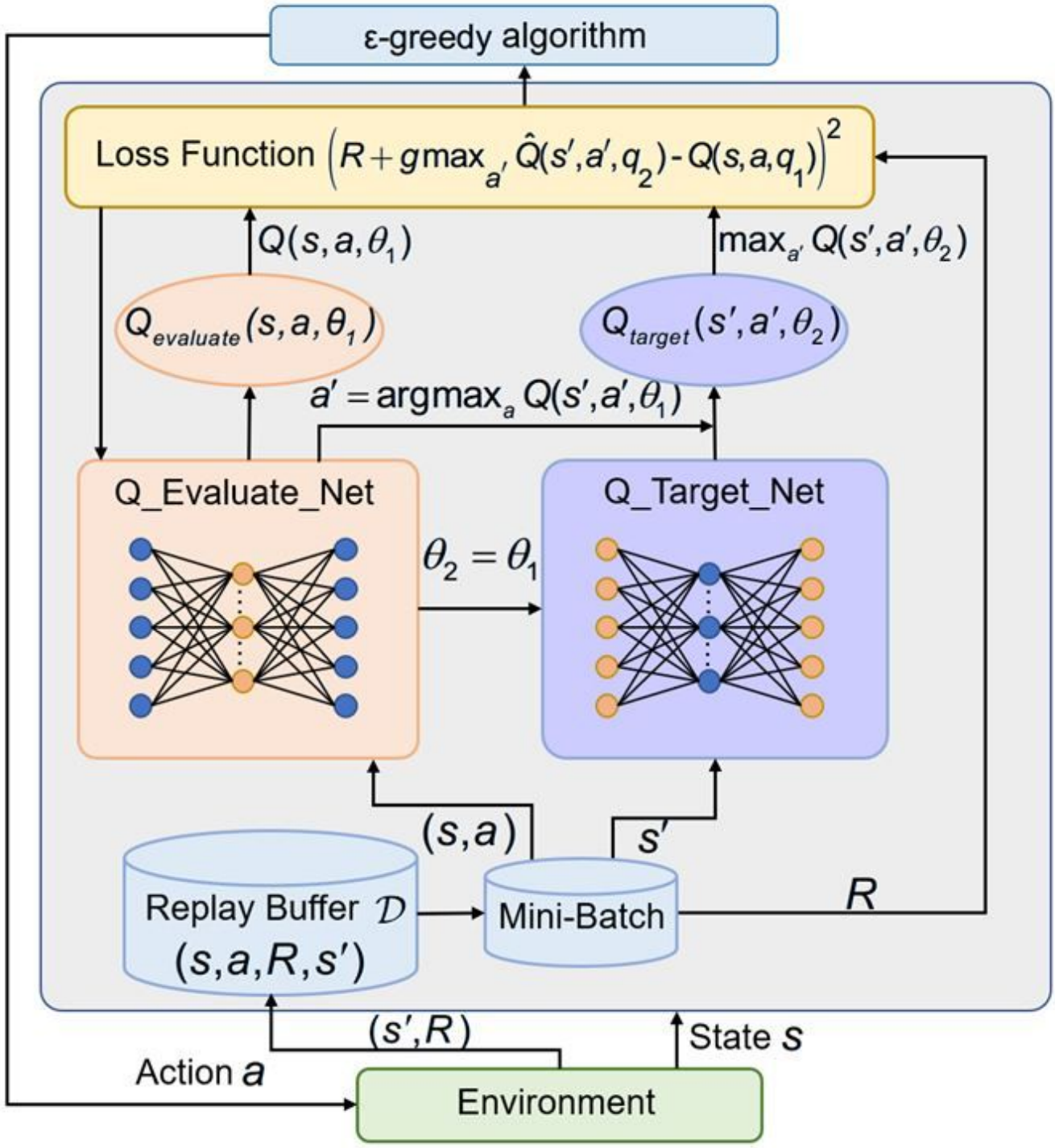

Figure 2

The framework of DRL method for ADSP

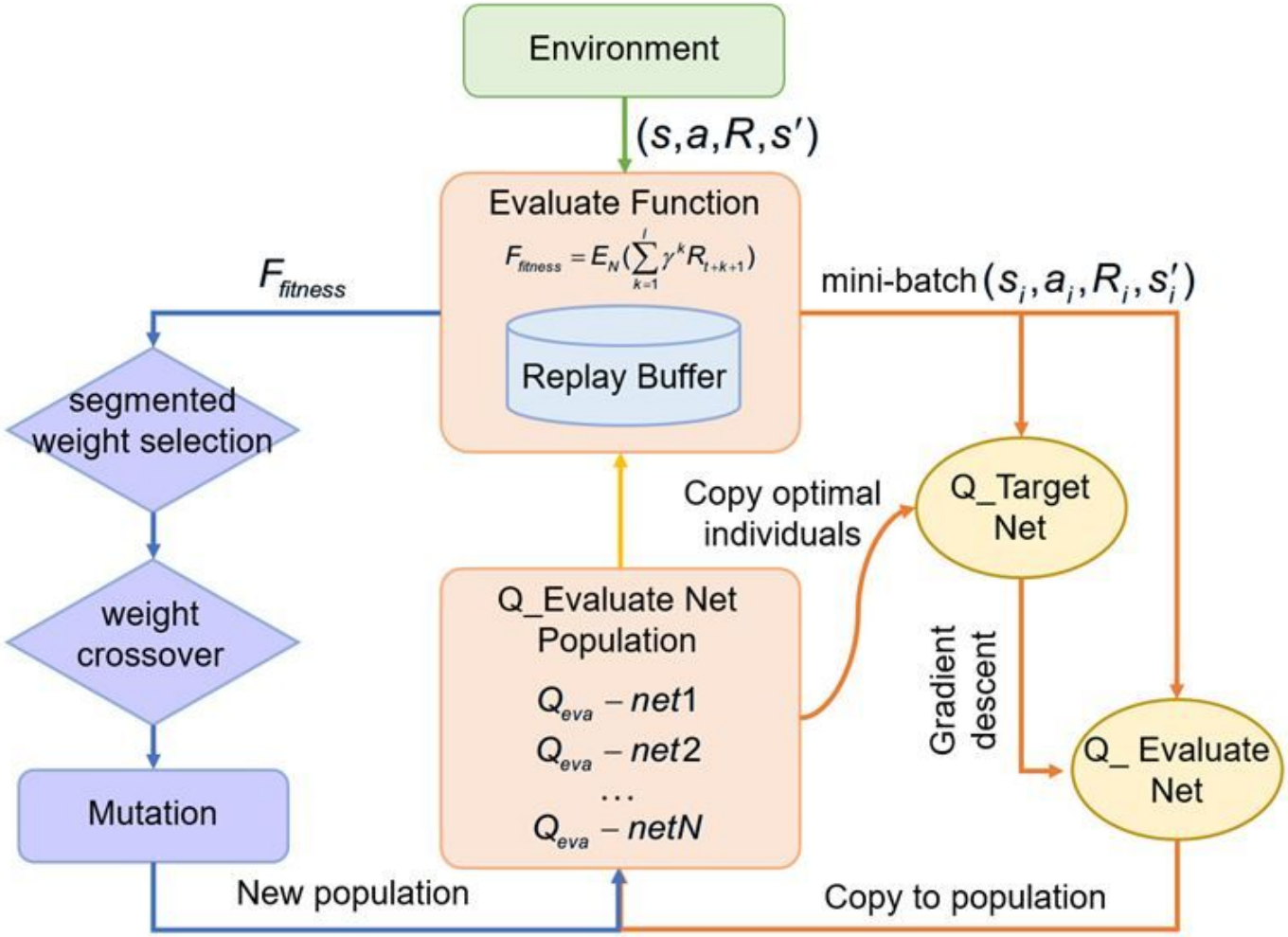

Figure 3

The architecture of GO-DQN

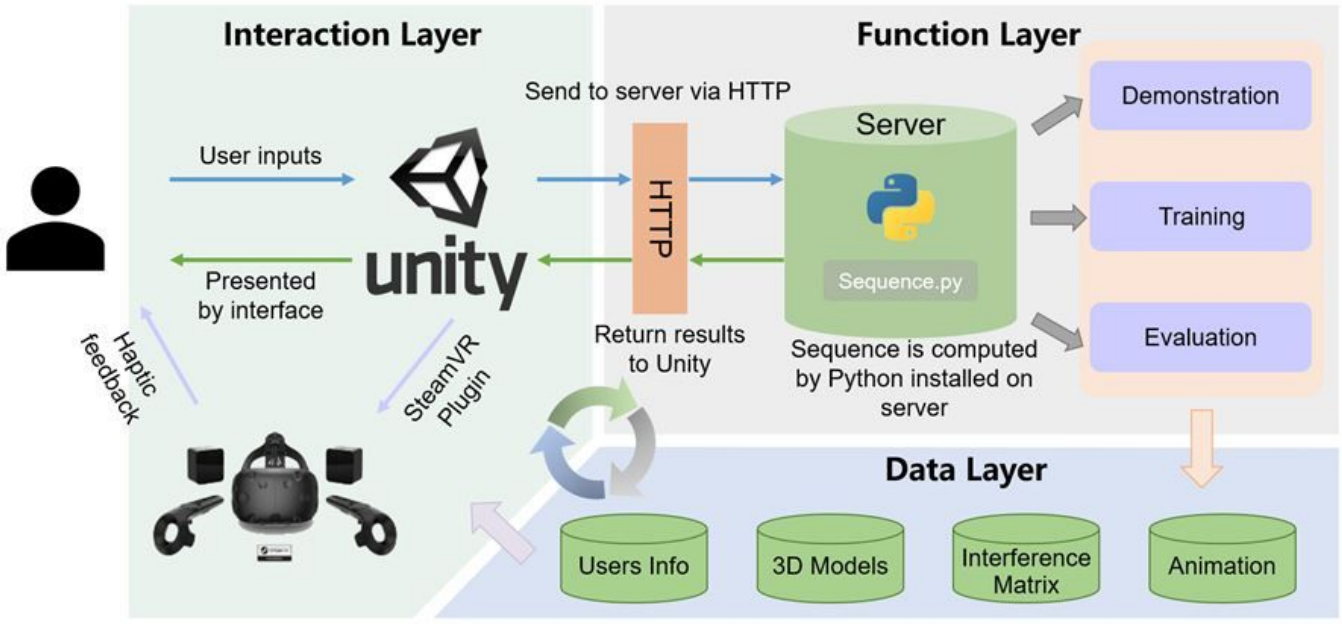

Figure 4

Architecture and workflow of the training system

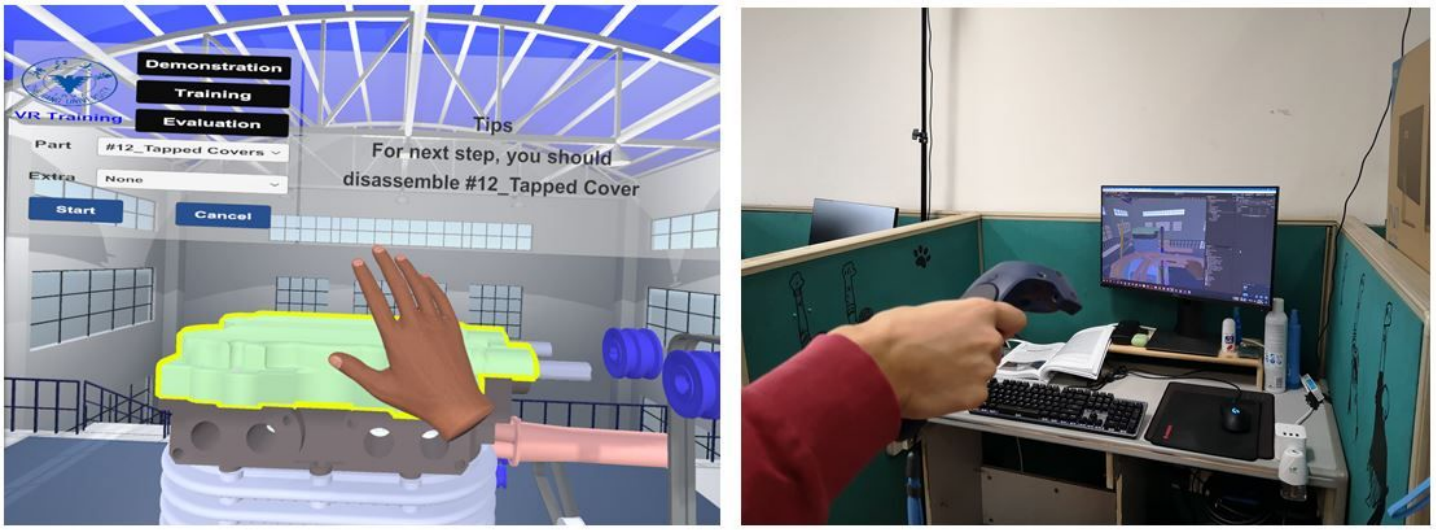

Figure 5

Feedback of vision and haptics during operation

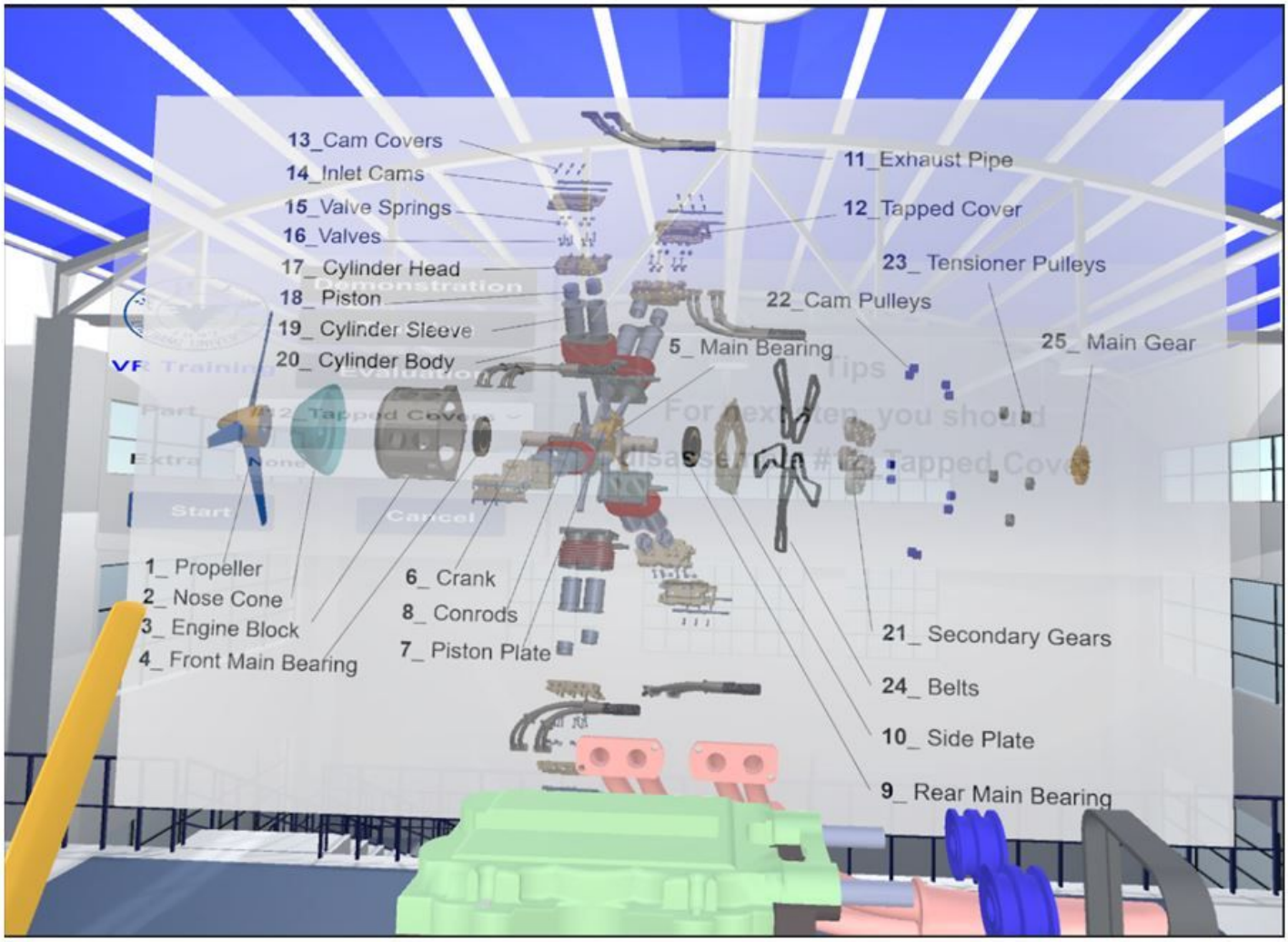

Figure 6

The structure of the aircraft engine

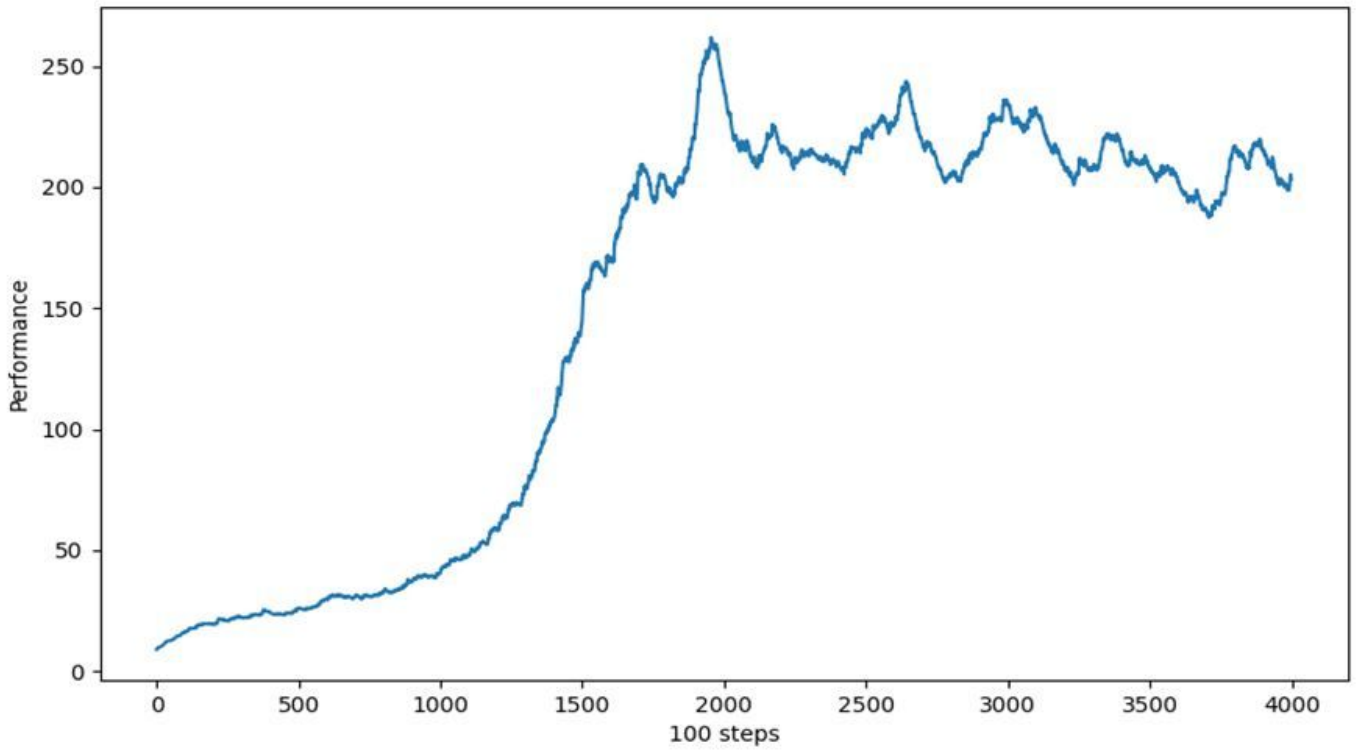

Figure 7

The training process of RL model based on GO-DQN

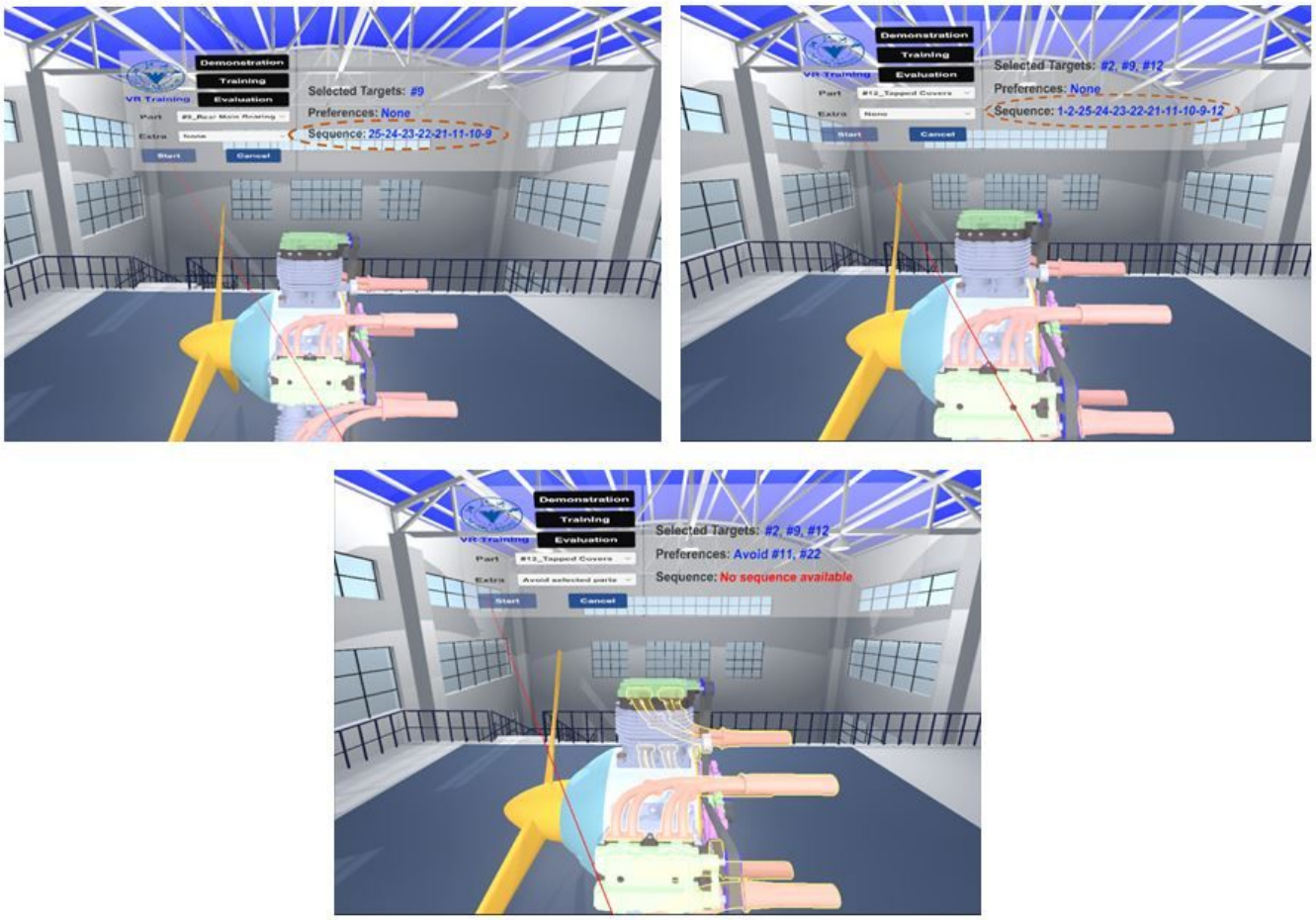

Figure 8

Running results of the three experiments in the training system