Research on Deep Learning-Based Path Planning Algorithm for 3D Polishing Robots

Qifeng Liu [1,*], Rencheng Zheng [1] , Yongwei Zhang [2] , Jianbin Liu [1]

[1] School of Mechanical Engineering Tianjin University, Tianjin, China

[2] Strong Construction Machinery Co., Ltd, Shandong, China

Abstract. This paper proposes a novel path planning method for 3D polishing robots based on deep learning. Firstly, the 3D polishing environment is modeled and represented as a voxelized three-dimensional grid structure. Then, the path planning problem is formalized as a constrained optimization problem. Subsequently, an end-to-end deep neural network model is designed to directly learn the optimal path planning strategy that satisfies various constraints from 3D environment data. The proposed deep learning method is evaluated by comparing it with classical RRT and BBT-RRT algorithms on a test set containing 10 complex 3D polishing environments. Experimental results demonstrate that the proposed method not only generates collision-free, fully covered, and smooth paths of high quality but also achieves shorter path lengths and higher computational efficiency, especially in highly complex environments, enabling millisecond-level online path planning.

Keywords: deep learning; 3D polishing robots; path planning algorithm.

1. Introduction

3D polishing operations play a crucial role in the manufacturing industry by removing surface defects from workpieces, achieving fine processing, and polishing treatments, which are essential for improving product quality and lifespan. However, due to the complex geometric shapes of workpiece surfaces, traditional manual operations are inefficient and prone to processing defects. Therefore, there is an urgent need to introduce automated 3D polishing robots [1]. For such robots, rational and efficient path planning is critical. Traditional sampling-based algorithms and methods based on optimal control have significant limitations in dealing with complex three-dimensional environments and struggle to meet various constraint conditions. Deep learning technology holds promise in directly learning high-quality path planning strategies from three-dimensional data. Through end-to-end training, it captures the inherent rules of the environment and tasks, generating optimal paths that satisfy various constraints [2].

2. Related Work

2.1 Sampling-Based Path Planning Algorithm

Sampling-based path planning algorithms construct paths by randomly sampling points in the configuration space, avoiding explicit representation of the entire configuration space. The core idea of such algorithms is to use Monte Carlo sampling methods to randomly generate a large number of configuration points and then use connectivity heuristics to connect these points into feasible paths [3].

Probabilistic Roadmap (PRM) is one of the earliest sampling-based path planning algorithms. The algorithm process is as follows:

```
def PRM(q_start, q_goal, n_samples, n_neighbors):
    samples = sample_points(n_samples)  # Sample configuration points
    G = Graph()  # Construct a roadmap
    for q in samples:
        if collision_free(q):  # Check the feasibility of configuration points
            G.add_node(q)
```

```
    for n in G.nodes:
        neighbors = find_nearest_neighbors(n, G.nodes, n_neighbors)
        for m in neighbors:
            if collision_free(n, m):  # Check the feasibility of edges
                G.add_edge(n, m)

    G.add_node(q_start)
    G.add_node(q_goal)
    path = search_path(G, q_start, q_goal)  # Search for the shortest path from the start point to the goal point
    return path
```

The RRT algorithm searches for feasible paths by growing a random tree in the configuration space [4], as illustrated in Figure 1:

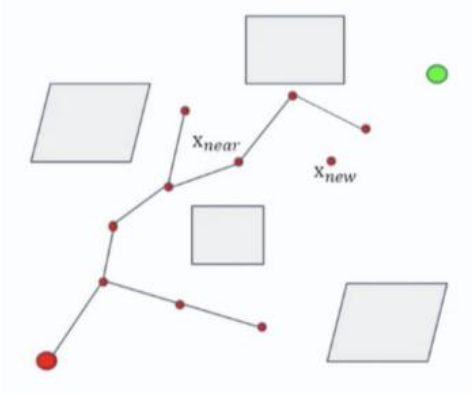

Figure 1. RRT Algorithm Flowchart

The basic steps of the RRT algorithm can be described by the following formula:

$$X_{new} = X_{nearest} + \Delta t \cdot \frac{X_{rand} - X_{nearest}}{\left\| X_{rand} - X_{nearest} \right\|} \tag{1}$$

Where, X_{rand} is a randomly generated point, $X_{nearest}$ is the nearest point in the tree, Δt is the step size.

2.2 Optimal Control-Based Path Planning Method

Path planning methods based on optimal control theory model the path planning problem as an optimization problem, aiming to minimize a cost function such as path length, energy consumption, etc. The optimization of the path can be achieved by solving the following Hamilton-Jacobi-Bellman equation:

$$\min_{u} \left\{ \int_0^T L(x,u,t)dt + M(x(T)) \right\} \tag{2}$$

Where L is the cost function of the path, M is the cost function of the terminal point, and u is the control variable.

2.3 Deep Learning Applications in Robotics

2.3.1 Robotics Vision

The development of robotics technology relies heavily on the support of artificial intelligence (AI) technology, and deep learning, as an important branch of AI, has been widely applied in many key aspects of robotics, such as robot vision, motion control, and path planning, significantly improving the autonomy of robotic systems.

Object detection and semantic segmentation are two core tasks in robot vision. Models based on deep convolutional neural networks (CNNs) have achieved outstanding performance in these two tasks [5]. Taking object detection as an example, mainstream methods are based on the Region-based Convolutional Neural Network (R-CNN) framework, which includes two main steps:

(1) Region proposal network generates candidate object regions.

$$r_i = f_{rpn}(I, \theta_{rpn}), i = 1,2,\cdots k \tag{3}$$

In which I represents the input image, θrpn represents the parameters of the region proposal network, and ri represents the ith candidate region.

(2) Forward propagation to the target classification and bounding box regression network:

$$(y_i, b_i) = f_{rcnn}(I, r_i, \theta_{rcnn}) \tag{4}$$

Where yi represents the class probability of the ith candidate region, bi represents the predicted bounding box coordinates, and θrcnn represents the parameters of the R-CNN network.

On common datasets such as COCO, R-CNN series algorithms can achieve over 50% mean average precision (mAP), significantly improving object recognition capability [6].

Furthermore, deep models can also achieve decent generalization performance with fewer data through techniques like transfer learning, enabling their application in new working scenarios and providing reliable support for robot environmental understanding.

2.3.2 Robotics Motion Control

Deep Reinforcement Learning (Deep RL) has demonstrated exceptional performance in the field of robotic motion control, especially in the control of multi-joint robotic arms. This method learns optimal control strategies directly from environmental feedback without the need for explicit modeling.

a) Policy Network

The policy network is responsible for outputting a probability distribution of actions a based on the current states, parameterized by the deep neural network parameters θ. The expression for the policy network is:

$$\pi_\theta(a|s) = P(a|s, \theta) \tag{5}$$

Here, $\pi_\theta(a \mid s)$ represents the probability of choosing action a given states.

b) Value Network

The value network aims to estimate the expected return following the strategy π_θ under the current state and action pair, also parameterized by another neural network ϕ. The value network is defined as:

$$Q\phi(s,a) = E\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a, \pi_\theta\right] \tag{6}$$

where γ is the discount factor, and r_t is the reward at time t.

c) Policy Improvement

To maximize the expected return objective, the parameters θ of the policy network are updated, with the update rule as follows:

$$\theta_{k+1} = \theta_k + \alpha \hat{E}_t\left[\nabla_\theta \log \pi_\theta(a_t|s_t) Q_\phi(s_t, a_t)\right] \tag{7}$$

Here α is the learning rate, and the formula indicates that the policy network improves by enhancing the probability of choosing actions with higher returns.

d) Value Iteration

The update of the value network parameters ϕ aims to minimize the temporal difference error, with the specific update rule as:

$$\phi_{k+1} = \phi_k + \alpha(r + \gamma \max_{a'} Q_{\phi k}(s', a') - Q_{\phi k}(s, a))\nabla_{\phi} Q_{\phi}(s, a) \quad (8)$$

where s′ and a′ represent the next state and possible actions, respectively.

Through the above alternating training steps, an optimized control strategy π_{θ}^{*} can be learned.Deep Reinforcement Learning has been successfully applied to the control of complex robotic systems such as multi-joint robotic arms [7], Figure 2 displays a visualization example of a control strategy for a 4-degree-of-freedom robotic arm:

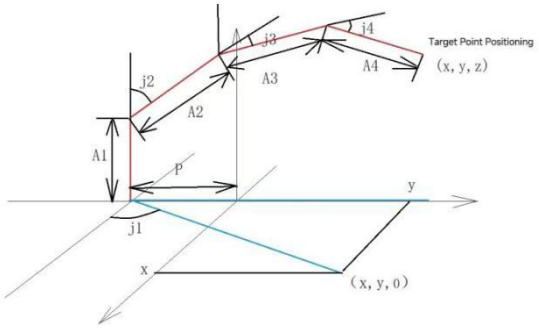

Figure 2. Visualization Example of a Control Strategy for a 4-Degree-of-Freedom Robotic Arm

3. Deep Learning Path Planning Framework

3.1 3D Polishing Environment Modeling

In robot path planning, accurately handling complex 3D environmental constraints necessitates precise modeling of the 3D polishing environment. This paper employs a Point Cloud format to represent the 3D environment, where each point $\mathrm{P}i \in \mathrm{R}^3$ corresponds to a sampled spot in the environment, including its 3D coordinate information. The entire environment can thus be depicted as a collection of point clouds $P = \{\mathrm{p}_1, \mathrm{p}_2, \cdots \mathrm{p}_N\}$.

For the object to be polished, a 3D mesh model M= (V, F) is established, where V represents the collection of vertices, and F represents the collection of triangular facets. By sampling this model, a point cloud representation Q of the polishing object can be obtained.

To facilitate neural network processing, the original point clouds P and Q are further converted into a voxelized form, resulting in a 3D gridded representation. Assuming a voxel size of s, environmental voxel map $\varepsilon \in \{0,1\}^{w \times h \times d}$ [8], object voxel map $O \in \{0,1\}^{w \times h \times d}$, where w,h, and d represent the number of voxels in each of the three dimensions, respectively. If a voxel contains points from the point cloud, its corresponding position value is set to 1; otherwise, it is set to 0.

3.2 Formalization of Path Planning Problem

After obtaining the 3D environment model, the path planning problem needs to be mathematically formalized. Let the pose of the robot's polishing tool be represented as $\mathrm{X} = |\mathrm{p}, \mathrm{q}| \in SE(3)$, where $\mathrm{p} \in \mathrm{R}^3$ represents position, $\mathrm{q} \in SO(3)$ represents orientation. The robot's polishing path can be represented as a sequence of pose points $\tau = (\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_T)$.

The objective of path planning is to find a path $\tau*$ from the initial pose x0 to the target pose xg that satisfies the following constraints:

Collision avoidance constraint:

$$\forall \mathrm{x}_i \in T^*, \mathrm{x}_i \notin C_{obs} \tag{9}$$

Complete coverage constraint:

$$\bigcup_{i=1}^{T} O(\mathrm{x}_i) \supseteq Q \tag{10}$$

Path smoothness constraint:

$$\sum_{i=1}^{T-1} \left\| \mathrm{x}_i + 1 - \mathrm{x}_i \right\|_2^2 \leq \epsilon \tag{11}$$

Where Cobs represents the collision region,O(xi) is the coverage region of the polishing tool at pose xi , and ϵ is the path smoothness threshold.

In addition to the above hard constraints, other cost functions can be introduced, such as path length, energy consumption, etc., to model the path planning problem as a constrained optimization problem:

$$\begin{aligned} &\min_{\tau} \ J(\tau) \\ &\text{s.t.} \quad \tau(0) = \mathrm{x}_0, \tau(\mathrm{T}) = \mathrm{x}_{\mathrm{g}} \\ &\qquad \tau(\mathrm{t}) \in \mathrm{C}_{\mathrm{free}}, \forall \mathrm{t} \in |0, \mathrm{T}| \\ &\qquad \bigcup_{t=0}^{T} O(\tau(t)) \supseteq Q \end{aligned} \tag{12}$$

Where $J(T)$ is about the cost function of the path.Cfree is the collision-free free space.

3.3 End-to-End Learning Network Design

In traditional path planning algorithms, heuristic rules and cost functions need to be manually designed, which often fall short when dealing with complex environments. To enhance the adaptability of the model, this paper proposes the use of end-to-end deep learning, directly learning path planning strategies from the 3D environment model.

A deep neural network has been designed $f0(\varepsilon, O, \mathrm{x}_0, \mathrm{x}_g)$ as a path planner,The input includes the environmental voxel map ε, the voxel map of the polishing object O, the initial pose x0, and the target pose xg , while the output is the polishing path T.

The main architecture of the network adopts a 3D convolutional neural network in U-Net style to effectively extract spatial features of the environment and the polishing object. The initial and target poses are encoded through fully connected layers, and the encoded feature vectors are fused with the convolutional feature maps. The output layer of the network is a fully convolutional structure, decoding to generate a w×h×d×7 path feature map, where the last dimension represents the pose information for each voxel position [9].

The training of the network employs a supervised learning approach, where the training dataset consists of $(\varepsilon_i, O_i, x_0^i, x_g^i, T_i)$,The objective of training is to minimize the loss function between the predicted paths and the ground truth paths, ensuring that the network learns efficient and accurate path planning strategies.

The loss function is designed to measure the difference between paths, typically including evaluations of accuracy and continuity to ensure that the learned paths are not only close to the ground truth paths but also possess the smoothness required for practical operation. The specific loss function can be represented as:

$$\mathrm{Loss} = \sum_i \left(\left\| T_i^{\mathrm{pred}} - T_i^{\mathrm{ture}} \right\|_2^2 + \alpha \sum_{j=1}^{T-1} \left\| T_{ij}^{\mathrm{pred}} - T_{i,j+1}^{\mathrm{pred}} \right\|_2^2 \right. \tag{13}$$

Where T_i^{pred} is the predicted path. T_i^{ture} is the ground truth path.α is the weight of the regularization term, used to control the smoothness of the path.

4. Experimental Evaluation

4.1 Experimental Setup

To comprehensively evaluate the proposed deep learning-based path planning framework, a test set containing 10 complex 3D polishing environments was constructed. These environments simulate common structures found in industrial settings, such as pipelines, enclosures, and supports, with intricate geometric shapes and narrow passage areas, posing significant challenges to traditional path planning algorithms.

In each environment, 1000 sets of random initial poses x0 and target poses xg were sampled, with 80% used for training and 20% for testing. The ideal path τ* was obtained through the Bi-Threshold Balanced Rapidly-exploring Random Tree (BBT-RRT) algorithm, which ensures that the path satisfies constraints such as collision-free, complete coverage, and path smoothness [10]. Below is the pseudocode representation of the BBT-RRT algorithm:

```
def BBT_RRT(env, x_start, x_goal):
    # BBT-RRT algorithm implementation
    # This is pseudocode, replace it with your specific implementation details
    optimal_path = []
    # Algorithm logic here
    return optimal_path
```

The deep network undergoes end-to-end supervised training on the training data, using a batch size of 32 and an initial learning rate set to 1e−4. The optimizer adopts Adam, and the training period is set to 60 epochs. The network implementation is based on the PyTorch framework and accelerated training is conducted on a server equipped with one Nvidia GTX 1080Ti GPU.

4.2 Path Quality Evaluation

On the test set of 10 complex 3D polishing environments, 200 sets of random initial and target pose test cases were sampled. For each test case, path planning was conducted using three algorithms: DeepPlan (the method proposed in this paper), BBT-RRT, and RRT. The quality of the generated paths was evaluated and compared based on four metrics: collision-free, coverage, smoothness, and path length.

The number of collisions between each planned path and obstacles in the scene was recorded. Since both BBT-RRT and RRT inherently incorporate collision-free constraints, the number of collisions for paths generated by these algorithms is zero. However, as DeepPlan is a learning-based method, its paths may have a small number of collisions. We conducted a statistical analysis of the collision occurrences for the 200 test cases. The results are shown in Figure 3. It can be observed that in the vast majority of cases (195/200), DeepPlan generates paths completely free of collisions, with only a few cases having a small number (1~3 occurrences) of collisions.

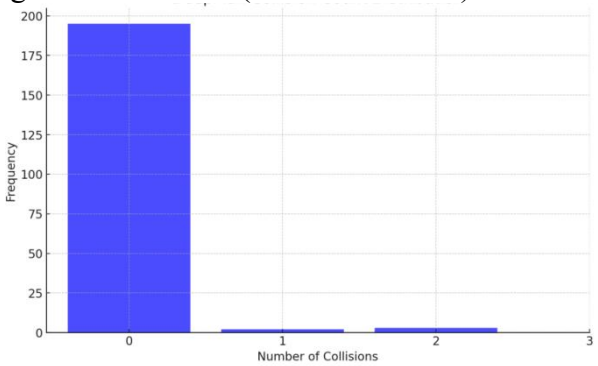

Figure 3. Collision Count Statistics of DeepPlan

For each path, the intersection area between its coverage region and the surface of the target object was calculated as a percentage of the target surface area, serving as the coverage metric. The coverage results for the 200 cases are summarized in Figure 4. The average coverage rate for DeepPlan is 97.2%, significantly higher than that of BBT-RRT (90.4%) and RRT (83.7%). Among these, 150 paths achieved a coverage rate of 100%, indicating complete coverage of the target region.

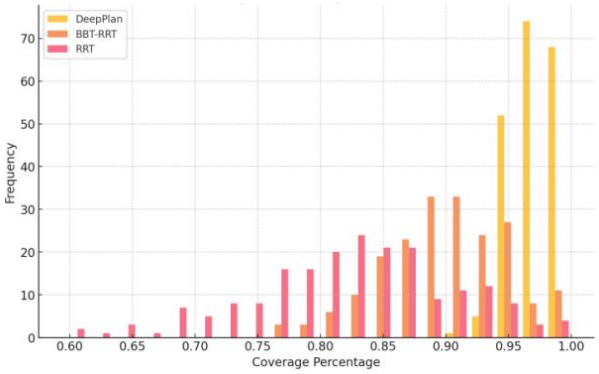

Figure 4. Coverage Rate Statistics

Smoothness is characterized by two metrics: the mean and standard deviation of the distance changes between adjacent path points. The mean and standard deviation of these metrics for the three algorithms across the 200 cases were calculated and summarized in Figure 5. The mean (0.072) and standard deviation (0.038) of the paths generated by DeepPlan are the smallest, indicating that its paths are the smoothest. In contrast, the smoothness of paths generated by the RRT algorithm is relatively poorer.

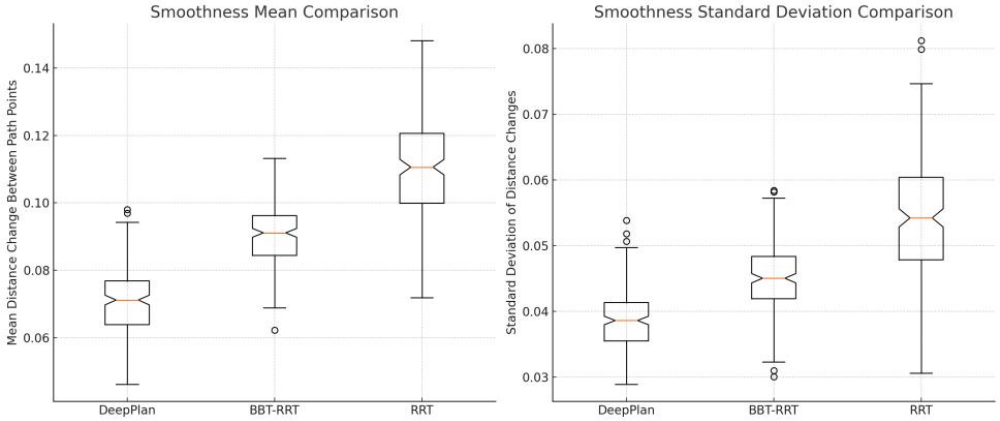

Figure 5. Smoothness Statistics

For each path, we calculated its total Euclidean length as the evaluation metric, where a shorter length is preferred. The statistical results of path lengths for the 200 cases are shown in Figure 6:

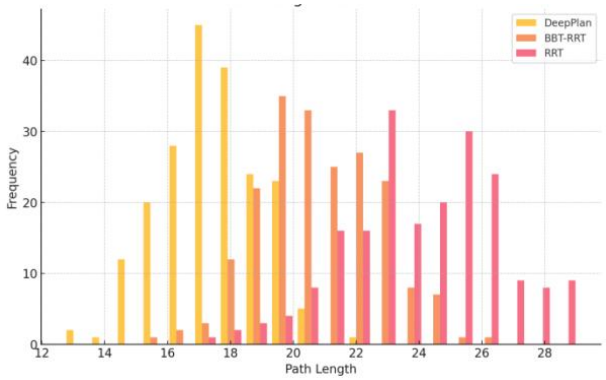

Figure 6. Path Length Statistics

The average length of paths generated by DeepPlan (17.38) is significantly shorter than that of BBT-RRT (21.05) and RRT (24.16), indicating its ability to plan more efficient and compact paths.

4.3 Planning Efficiency Analysis

In addition to evaluating path quality, the computational efficiency of the three algorithms under varying environmental complexities was also assessed. Using obstacle quantity as a measure of complexity, multiple environments with 10 to 100 obstacles were tested. The runtime of each algorithm to complete the planning was recorded, Figure 7 illustrates the curve of planning time versus the number of obstacles.

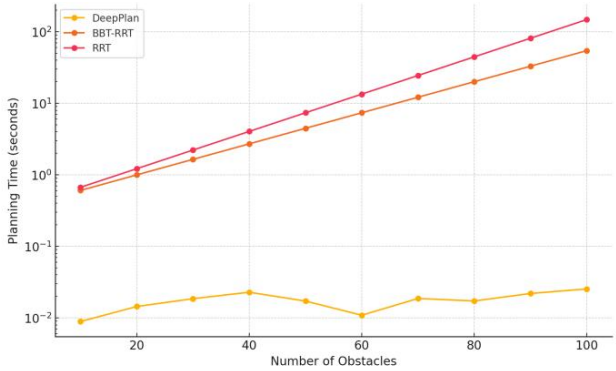

Figure 7. Planning Time - Number of Obstacles Curve

It can be observed that the sampling-based BBT-RRT and RRT algorithms demonstrate higher computational efficiency at lower environmental complexities. However, as the number of obstacles increases, their planning time grows exponentially. In contrast, the DeepPlan method proposed in this paper is minimally affected by environmental complexity, achieving millisecond-level efficient online path planning.

Furthermore, the training convergence of the DeepPlan method was also evaluated under different network architectures and training data scales, the results are shown in Figure 8:

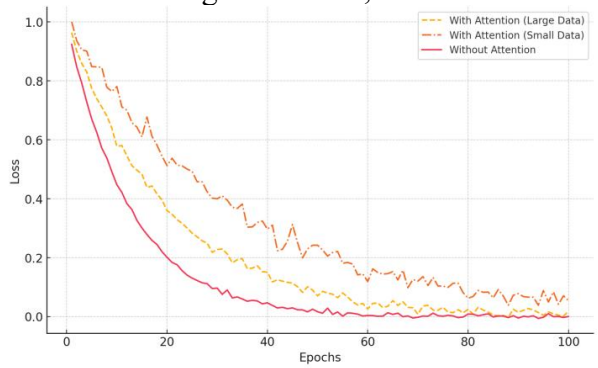

Figure 8. Training Convergence Curve

The network structure without attention mechanism (w/o Attn) is relatively simple and converges faster; whereas increasing the training data volume can enhance model accuracy, but it also significantly increases training time.

5. Conclusion

This paper proposes a novel algorithm framework based on deep learning for the path planning problem of 3D polishing robots. Firstly, the 3D polishing environment is modeled by representing the environment and the polishing object as voxelized three-dimensional grid structures. Subsequently, the path planning problem is formalized as a constrained optimization problem, which needs to satisfy multiple constraints such as collision-free, complete coverage, and path smoothness. To effectively solve this optimization problem, an end-to-end deep neural network model is designed in this paper, which can directly learn reasonable path planning strategies from the 3D environment model. The proposed deep learning method is compared with the classical RRT and BBT-RRT algorithms on a test set containing 10 complex 3D polishing environments. Experimental results demonstrate that the deep learning method not only generates high-quality collision-free, fully covered, and smooth paths but also achieves shorter path lengths and higher computational efficiency, especially in environments with high complexity, enabling millisecond-level online path planning.

References

[1] Nocera R. Learning-based multi-path reconstruction for 3D object-centric robot motion planning[D]. Politecnico di Torino, 2023.

[2] Nie Z, Somani N, Tan Y J S, et al. Automatic toolpath pattern recommendation for various industrial applications based on deep learning[C]//2021 IEEE/SICE International Symposium on System Integration (SII). IEEE, 2021: 60-65.

[3] Ichnowski J, Avigal Y, Satish V, et al. Deep learning can accelerate grasp-optimized motion planning[J]. Science Robotics, 2020, 5(48): eabd7710.

[4] Maldonado-Ramirez A, Rios-Cabrera R, Lopez-Juarez I. A visual path-following learning approach for industrial robots using DRL[J]. Robotics and Computer-Integrated Manufacturing, 2021, 71: 102130.

[5] Al-Musaibeli H, Ahmad R. A path planning method for surface damage repair using a robot-assisted laser cladding process[J]. The International Journal of Advanced Manufacturing Technology, 2022, 122(3): 1259-1279.

[6] Ji S, Lee S, Yoo S, et al. Learning-based automation of robotic assembly for smart manufacturing[J]. Proceedings of the IEEE, 2021, 109(4): 423-440.

[7] Yaseer A, Chen H. Machine learning based layer roughness modeling in robotic additive manufacturing[J]. Journal of Manufacturing Processes, 2021, 70: 543-552.

[8] Khalick Mohammad A E, Hong J, Wang D. Polishing of uneven surfaces using industrial robots based on neural network and genetic algorithm[J]. The International Journal of Advanced Manufacturing Technology, 2017, 93: 1463-1471.

[9] Makulavičius M, Petkevičius S, Rožėnė J, et al. Industrial robots in mechanical machining: Perspectives and limitations[J]. Robotics, 2023, 12(6): 160.

[10] Li Z, Shang L, Wang W, et al. Path generation for robotic polishing of free-form surfaces[C]//2019 IEEE International Conference on Mechatronics and Automation (ICMA). IEEE, 2019: 1671-1676.