actuators

Article

A SAC-Bi-RRT Two-Layer Real-Time Motion Planning Approach for Robot Assembly Tasks in Unstructured Environments

Qinglei Zhang [1], Siyao Hu [1], Jianguo Duan [1], Jiyun Qin [1] and Ying Zhou [2,*]

[1] Logistics Engineering College, Shanghai Maritime University, Pudong, Shanghai 201306, China; qlzhang@shmtu.edu.cn (Q.Z.); hsy1483293225@163.com (S.H.); jgduan@shmtu.edu.cn (J.D.); jyqin@shmtu.edu.cn (J.Q.)

[2] China Institute of FTZ Supply Chain, Shanghai Maritime University, Pudong, Shanghai 201306, China

* Correspondence: zhouying@shmtu.edu.cn

Abstract: Due to the uncertainty and complexity of the assembly process, the trajectory planning of a robot needs to consider the real-time obstacle avoidance problem when it completes the assembly in the unstructured workspace. To realize the safe assembly of assembly robots in dynamic and complex environments, a dynamic obstacle avoidance trajectory planning method for robots combining traditional planning algorithms and deep reinforcement learning algorithms is proposed to improve the robot's agent and obstacle avoidance ability in dynamic and complex environments. The Bidirectional Rapidly-exploring Random Tree (Bi-RRT) method is utilized as a global planner to plan the global optimal path quickly; considering the real-time nature of the assembly process, the Soft Actor-Critic (SAC) is used as a local obstacle avoider to avoid obstacles more accurately and to find the nearest node generated by the Bi-RRT during the planning process, which is regarded as the goal during the local obstacle avoidance to reduce the model's complexity. By training and testing in the simulation engine and comparing with SAC, DDPG and DQN algorithms, the method can avoid obstacles in dynamic and complex environments more efficiently, which verifies that the proposed hybrid method can accomplish the high-precision planning task with a high success rate.

Keywords: Soft Actor-Critic; Bidirectional Rapidly-exploring Random Trees; dynamic obstacle avoidance; motion planning; robot

check for updates

Academic Editor: Zhuming Bi

Received: 23 December 2024

Revised: 22 January 2025

Accepted: 23 January 2025

Published: 26 January 2025

Citation: Zhang, Q.; Hu, S.; Duan, J.; Qin, J.; Zhou, Y. A SAC-Bi-RRT Two-Layer Real-Time Motion Planning Approach for Robot Assembly Tasks in Unstructured Environments. *Actuators* **2025**, *14*, 59. https://doi.org/10.3390/act14020059

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In modern manufacturing, intelligent assembly and the advancement of Industry 5.0 are driving a comprehensive upgrade of production methods. Intelligent assembly realizes efficient, precise and flexible production by integrating automation and information technology. Industry 5.0 emphasizes the deep collaboration between humans and intelligent robots, combining the creativity of workers with the efficient production capacity of intelligent machines to achieve a more personalized and humanized production model. As the core equipment of intelligent assembly, the performance of the robot directly affects the efficiency and reliability of the whole production system. In complex and dynamic environments, robots face many obstacles and uncertainties, so real-time obstacle avoidance motion planning becomes a key technology. By integrating multiple sensors and advanced path planning algorithms, the robot can sense environmental changes in real-time and dynamically calculate the optimal obstacle avoidance path to ensure the safety and continuity of the assembly process. Therefore, how to realize efficient and safe motion planning in the robot assembly process has become an urgent problem to be solved.

Although the motion planning problem for robots has been studied for many years, most of the traditional motion planning methods target static obstacles. For example, Qi [1] et al. implemented probabilistic-based optimal path planning for robot obstacle avoidance for robots. Kong [2] et al. proposed an adaptive Fuzzy Neural Network (FNN) and impedance learning-based control strategy for coordinated multi-robot operation under the conditions of an unknown environment and time-varying constraints in the work interval. However, for environments with cluttered static and dynamic obstacles, the computation of path search increases dramatically with the number of obstacles and spatial dimensions, and the traditional algorithms need to go through a large number of computations in order to find an effective path, meaning they are unable to meet the real-time requirements in practical applications.

When a collision-free path needs to be planned in an environment with complex obstacles, global path planning methods are more suitable for generating avoidance routes and calculating the optimal path for the robot based on the environmental information. The Rapidly-exploring Random Tree (RRT) proposed by LaValle [3] et al. is a powerful tool for finding collision-free paths in cluttered environments. The RRT algorithm has always been the focus of research on global motion planning techniques for robots in the past, and many researchers have proposed a series of improved algorithms that have been extended in several directions. Shen [4] et al. proposed an adaptive maneuverability-based path-planning strategy for industrial robot manipulators. The method utilizes an improved RRT algorithm that combines path length and maneuverability metrics to construct a bi-objective cost function to find the optimal path from the starting point to the goal point. To improve the path planning efficiency and assembly accuracy of the robotic arm during the assembly process, Li [5] et al. proposed a multi-objective point motion planning method for the assembly robotic arm based on the Improved Potential Quick Rapidly-exploring Random Tree (IPQ-RRT*) algorithm. Ferguson [6] et al. proposed an improved RRT algorithm (D-RRT) for fixing the configuration space when the configuration space has changed Rapidly-exploring Random Trees (RRTs). Li [7] et al. modified the initial path when dynamic obstacles blocked the path by using the robot's deviation angle to replan the blocked local path. Some other researchers have also proposed other methods. Han [8] et al. proposed a dynamic obstacle avoidance method based on distance computation and discrete detection, which realized dynamic obstacle avoidance for the manipulator by acquiring dynamic obstacles of arbitrary shapes captured by the Kinect-V2 camera in real-time and converting them into convex packets to compute the distances. Chen [9] et al. designed a dynamic obstacle avoidance based on the improved D* algorithm path planning method to generate collision hazard areas in dynamic environments and apply a real-time path adjustment strategy to accomplish the task. Mukadam [10] et al. proposed a continuous-time Gaussian process motion planning method for robot path optimization through probabilistic reasoning. Zhu [11] et al. proposed a dynamic window to change the speed of the robot to avoid unexpected dynamic obstacles.

Although the above algorithms have obvious advantages in avoiding collisions and generating effective paths, they still face limitations in dynamic and complex environments. First of all, the above algorithms have high computational complexity, especially in high dimensional spaces where the time overhead of path search is large. Although some improved algorithms can reduce the search time by bidirectional search, they are still prone to fall into local optimal solutions in complex environments. Secondly, the above algorithms are slow to react to dynamic obstacles, leading to a decrease in the reliability of path planning when obstacles change rapidly. In addition, these methods are usually used for global paths and lack the ability to adapt to real-time changes in dynamic environments, and thus often fail to effectively respond to real-time obstacle avoidance requirements in practical applications. In recent years, deep reinforcement learning has had a great advantage in handling large amounts of data, which provides ideas for some complex environment planning and can achieve high online planning speed at the cost of long training time. Wang [12] proposed an approach that combines global path planning and local obstacle avoidance. First, a global path planning algorithm is used to plan the optimal path; during the movement of the mobile robot, a local RL-based planner utilizes the surrounding environment information to avoid the moving obstacles. Yin et al. [13] used the structure of deep Q-network architecture to design a comprehensive reward function for planning the unmanned vessel's paths in unknown dynamic environments with obstacle avoidance, target approach, attitude correction, etc., as a part of the reward function. Bing [14] et al. proposed an algorithm called Bbox-HGG, which combines Hindsight Goal Generation (HGG) and Bounding box Encoder (Bbox-Encoder) that is used to implement robot manipulation in dynamic environments. Abdi [15] and others used a hybrid path planning approach for a 2D workspace, integrating Q-learning and neural networks. In addition, they extended their research to cover more complex scenarios such as a 3D workspace [16]. Zhang [17] et al. introduced 3D bounding boxes to represent obstacles and target objects and optimized the control strategy using the Soft Actor-Critic (SAC) algorithm to achieve efficient obstacle avoidance and path planning in dynamic scenarios. Zhang [18] et al. achieved efficient obstacle avoidance and path planning in dynamic scenarios by introducing a multi-objective optimization strategy, combining forward and inverse kinematic models, and utilizing the Soft Actor-Critic (SAC) algorithm to achieve optimal trajectory planning of the robotic arm under the premise of ensuring trajectory accuracy, smoothness, and minimizing energy consumption.

However, the long training time is a non-negligible problem. In order to obtain efficient path planning strategies, DRL usually requires a large amount of interaction data, which leads to a very time-consuming training process. Especially in complex dynamic environments, the training process requires a large number of iterative computations, which can seriously affect the response time of the system. Secondly, DRL is trained by historical data, but due to the dynamic changes in the environment, the trained model may not perform stably in new environments, resulting in large performance fluctuations. In addition, high consumption of computational resources is also a real problem faced by DRL in global path planning. Deep reinforcement learning alone is still not enough to overcome the difficulty of training high-precision tasks, and traditional motion planning algorithms can be utilized to compensate for the shortcomings of deep reinforcement learning algorithms by combining deep reinforcement learning with traditional motion planning to improve the performance. B. Sangiovanni [19] et al. proposed a hybrid control approach combining traditional motion planning algorithms and deep reinforcement learning (DRL) that enables robots to adaptively avoid obstacles and complete tasks in dynamic environments. Xia [20] et al. proposed a new framework, ReLMoGen, that combines motion generation and reinforcement learning (RL), which is a better solution for complex, long-term tasks by elevating the action space to subgoals. J. Yamada [21] et al. proposed a hybrid approach called MoPA-RL, which combines a motion planner with reinforcement learning (RL) to augment the action space of RL agents with long-term planning capabilities. The method smoothly transitions between directly executing actions and invoking the motion planner depending on the size of the action. G.P. Kontoudis [22] et al. obtained a continuous-time Q-learning based dynamics motion planning framework by combining the asymptotically optimal Rapidly Exploring Random Tree (RRT*) algorithm and model-free Q-learning to optimize the robot's motion paths in a dynamic environment online.

Although the previous methods can compensate for their respective shortcomings in some respects, they face a series of challenges. It is difficult to balance the advantages of both dynamic environments and easy to fall into local optimality. Algorithm coordination is a major challenge in hybrid methods, and in this paper, we combine the Soft Actor-Critic (SAC) algorithm with Bidirectional Rapidly-exploring Random Trees (Bi-RRTs) to achieve safer and more reliable real-time obstacle avoidance in dynamic and high-dimensional environments. Unlike existing hybrid approaches, we encourage exploration by introducing the SAC algorithm, which enhances the flexibility and adaptivity of the algorithm, while greatly improving the planning efficiency and real-time responsiveness through the bidirectional fast exploration of Bi-RRT.

Specifically, Bi-RRT is used as a global planner, which is responsible for planning the global path between the robot's current position and the grasping target, and quickly re-planning a new trajectory when needed. Meanwhile, SAC is used as a local obstacle avoider and targets a close node in the planned path of Bi-RRT to handle the robot's obstacle avoidance in real-time environments and task execution in dynamic environments. An integrated reward function modeling of dynamic obstacle avoidance and goal approach is developed for avoiding moving obstacles in the environment and for real-time planning. Compared with existing research, our contribution to this combined approach is that it effectively balances the advantages of deep reinforcement learning and traditional motion planning and solves the problems of local optimality and computational inefficiency of traditional methods in dynamic environments, which allows for more flexible, safe, and reliable real-time obstacle avoidance and task execution for robotic arms in dynamic and complex environments.

In summary, the contributions and highlights of the present study are as follows:

1. The proposed joint motion planning method based on SAC-Bi-RRT can quickly plan a collision-free trajectory in a randomized dynamic environment and can be re-planned in real-time when obstacles appear to finally reach the target position;
2. A collision detection algorithm is designed for the detection of a collision between two irregular objects—a high degree of freedom robot and an obstacle—and the method greatly reduces the amount of computation and improves the real-time performance;
3. The robot's DRL-based control module is specially designed to use the nodes of the Bi-RRT global planning as the targets of the local obstacle avoider with comprehensive reward functions so that the gripper jaws can approach objects and avoid moving obstacles in real-time according to the results of global planning;
4. The global camera can be used, as a visual servo to guide the robot in the workspace, to avoid the existence of the dead angle of the global camera, when there is a moving obstacle in the workspace and the depth camera mounted on the hand of the robotic arm can be activated and assist the robot in obstacle avoidance and trajectory planning through the visual servo.

2. A Logical Framework for Dynamic Obstacle Avoidance Based on Soft Actor-Critic and Bidirectional Rapidly-Exploring Random Tree Joint Planning

The general framework of the SAC-Bi-RRT motion planning method proposed in the present study is shown in Figure 1. In the present study, a hybrid strategy combining Bi-RRT global path planning and SAC local obstacle avoidance is adopted. The RRT algorithm adopts a tree structure to store the robot configuration nodes obtained by random expansion, and the paths between neighboring nodes satisfy the requirements of robotic arm dynamics, kinematics, and collision avoidance, so RRT is suitable for global motion planning in the dynamic obstacle avoidance process of the robot. Bi-RRT adopts a bidirectional searching strategy relative to RRT and expands from both the starting point and the target point, which can find the path faster. Therefore, the Bi-RRT algorithm was chosen as the global planner in the motion planning process in the present study.

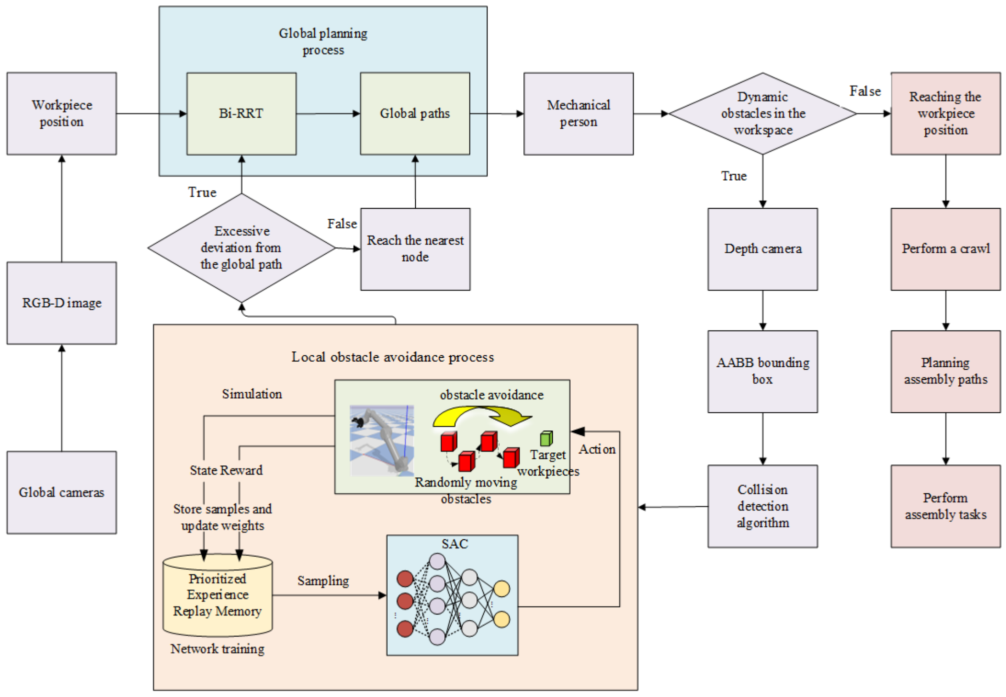


Figure 1. SAC-Bi-RRT motion planning framework (a logical framework for dynamic obstacle avoidance based on the Soft Actor-Critic algorithm with bidirectional fast exploratory random tree joint planning is shown in the figure. The information in the assembly environment is obtained from global and local cameras, and the robotic arm is planned to accomplish the assembly task in the complex environment using a hybrid strategy combining Bi-RRT global path planning and SAC local obstacle avoidance).

Since most of the current robot motion planning algorithms are traditional trajectory planning algorithms and generally target static obstacle avoidance in the work area, motion planning based on the SAC reinforcement learning algorithm proposed by Pei [23] et al. is an approach that is not limited by the planning dimensions and can achieve high online planning speed at the cost of long offline training time. SAC, as a localized obstacle avoidance device, can learn the optimal motion strategy based on the state of the real-time environment and feedback signals. By interacting with different environments, it adjusts the robot's actions in real-time to avoid obstacles and replan the path, which is highly adaptive. Compared with other reinforcement learning algorithms, the SAC algorithm is relatively simple, easy to implement and debug, and suitable for application in practice. Therefore, the SAC algorithm is chosen as the local obstacle avoider in present study.

In present study, the global camera is first used as a guide to determine the location of the artifact, and the Bi-RRT algorithm generates two RRTs in free space based on the environmental information acquired by the camera, which is extended in both directions to find the global path from the start point to the target point. The generated paths are subsequently smoothed to eliminate redundant nodes and make the paths smoother and more efficient. When a moving obstacle is detected in the operation area, the depth camera on the robot is turned on to capture the depth information of the local environment in real-time and simplify it with an AABB (Axis-Aligned Bounding Box) bounding box. The collision detection algorithm is utilized and the obstacle avoidance action is generated based on the real-time updated environment information to adjust the trajectory of the robot arm to ensure that it safely avoids obstacles. In obstacle avoidance, the robot always searches for the nearest global path node as a local target point and moves toward it. If the robot deviates too far from the global path during this process, global path planning is retriggered. At this point, the current robot position is used as a new starting point, and the Bi-RRT algorithm is used to recalculate to generate a new global path and continue the above process. In this case, the robot, the global camera, and the depth camera are communicated through the computer.

3. Methods

The first part of this section explains the global planning method used in the motion planning approach of this paper; the second part transforms the collision detection problem into a problem of determining whether a line segment intersects with a space cube by simplifying the linkage of the robotic arm into a spatial line segment and utilizing the AABB bounding box; the third part describes the local planner of the robotic arm for the dynamic obstacle avoidance path planning on the basis of the previous section, and in the last part, it is attached to the pseudo-code of the SAC-Bi-RRT motion planning method in this paper.

3.1. Global Planner

The Bi-RRT algorithm improves the RRT algorithm, inherits the advantages of the RRT algorithm and the idea of dual-tree improves the efficiency of the search tree expansion to a certain extent, accelerating the convergence of the algorithm. Bi-RRT expands the randomized tree T_i and T_g at the same time for the initial point q_i and the target point q_g, so that the two trees grow in the direction of each other to carry out a rapid search and improve the efficiency. The specific algorithm flow is as follows: take the initial point q_i and the target point q_g as the two root nodes, construct the random tree T_i and T_g, respectively, where the random expansion point of the tree T_i is noted as q_{irand} and the random expansion point of the tree T_g is noted as q_{grand}, randomly select a posture point q_{irand} in the free space C_{free}, select the closest point q_{near} to q_{irand} from the tree T_i, and carry out the collision detection algorithm by the step size ρ to find the closest point q_{new} to q_{irand}, and if q_{new} is in the middle of C_{free}, the tree T_i is increased with a new node q_{new}; and then, thinking of the target point q_{new}, expand the tree T_g according to the q_{new}. If the distance between the new nodes of the two trees is small enough, the two trees are considered to be connected and a path is found. Otherwise, repeat the above process until the two trees are connected and a path is found. Figure 2 shows the schematic diagram of the search process of the Bi-RRT algorithm.

However, in the process of the Bi-RRT search, because the selection of q_{irand} and q_{grand} is random and not oriented, the search efficiency is low and the convergence speed is slow. Therefore, the present study improves the Bi-RRT algorithm by applying a heuristic search approach in the construction process of the random tree. By comparing the distance between the random expansion point and the target point, the expansion point is screened, and the random search point closest to the target point is retained and expanded, which in turn generates a new node. Using this algorithm, the tree expands towards the target point during the construction of the random tree, increasing goal orientation and improving search efficiency. The specific implementation process of the improved heuristic Bi-RRT algorithm is as follows:

Step 1 Initialize the search path, which contains only the q_i and q_g

Step 2 Calculate the distance between q_i and q_g. If D_0 is less than the given threshold, the target point is considered reached and the path is searched. Otherwise, go to step 4.

Step 3 If T_i and T_g are connected, save the searched path.

Step 4 For random tree T_i, when $q_{i_{rand}}$ is selected for the first time, calculate the distance from $q_{i_{rand}}$ to q_g, denoted as D_i, and compare it with D_0. If $D_i \leq D_0$, keep the random point, calculate the current step and continue to expand to obtain a new node q_{new}, and save D_i. Otherwise, discard the random point, and continue sampling.

Step 5 For random tree T_g, when $q_{g_{rand}}$ is selected for the first time, calculate the distance from $q_{g_{rand}}$ to q_i, denoted as D_g, and compare it with D_0. If $D_g \leq D_0$, keep the random point and continue to expand it to obtain a new node q_{new} and save D_g. Otherwise, discard the random point and continue sampling.

Step 6 T_i and T_g are expanded alternately; during T_i expansion, when a new $q_{i_{rand}}$ is selected, calculate its distance q_g to $D_{i_{new}}$, compare it with D_i, and if $D_{i_{new}} \leq D_i$, keep the random point, calculate the direction vector $\vec{d}_i$ from the current node q_i to the random point $q_{i_{rand}}$, normalize it, expand the new node using that direction vector $q_{i_{new}} = q_i + step\_size \times \vec{d}_i$, and assign the value of $D_{i_{new}}$ to D_i; otherwise, discard the random point, and continue sampling; during T_g expansion, when a new q_{grand} is selected, calculate its distance q_i to $D_{g_{new}}$ and compare it with D_g. If $D_{g_{new}} \leq D_g$, keep the random point, calculate the direction vector $\vec{d}_g$ from the current node q_g to the random point $q_{g_{rand}}$, normalize it, extend the new node using this direction vector $q_{g_{new}} = q_g + step\_size \times \vec{d}_g$, and assign the value of $D_{g_{new}}$ to D_g; otherwise, discard the random point and continue sampling.

Step 7 Return to step 3.

To make the algorithm controllable, the number of cycles is limited in step 7, and if T_i and T_g are still not connected within the limited number of cycles, the algorithm returns a failure, indicating that the path search has failed.

$D_0 = dis(q_i, q_g)$ is the distance from the starting point to the target point. $D_i = dis(q_{irand}, q_g)$ is the distance between the random point selected for the first time in the expansion process of T_i and the target point q_g. $D_g = dis\left(q_{grand}, q_i\right)$ is the distance between the random point selected for the first time in the expansion process of T_g and the starting point q_i. The specific process is shown in Figure 3.

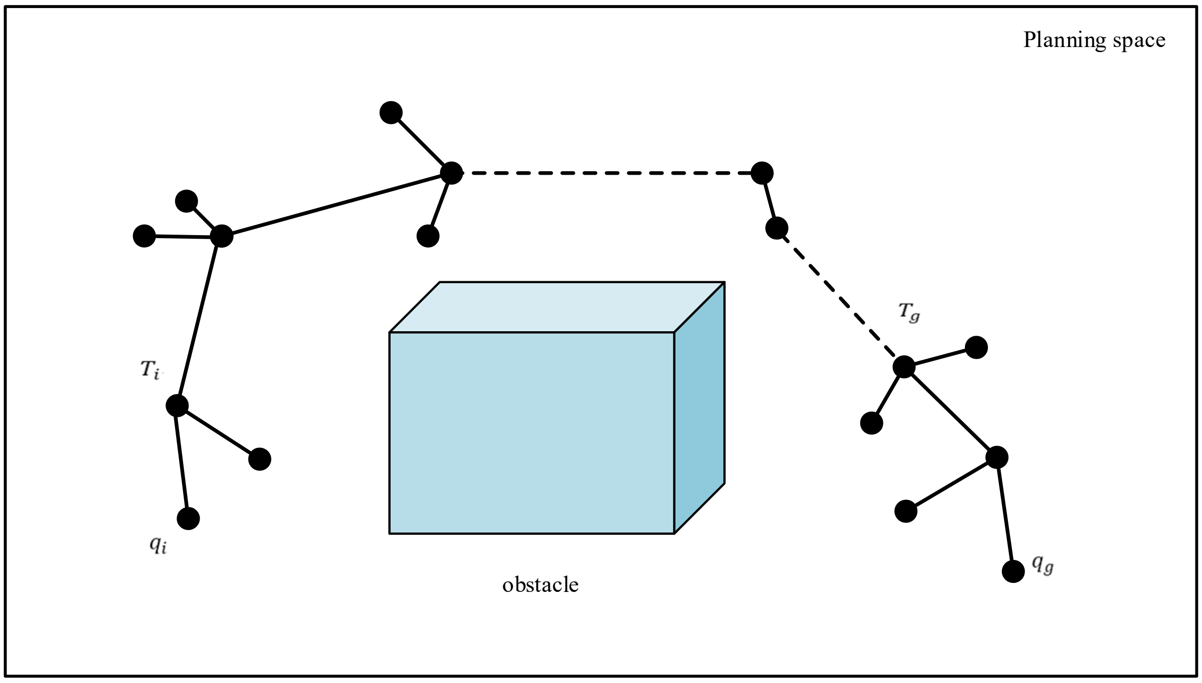


Figure 2. Schematic diagram of Bi-RRT algorithm planning.

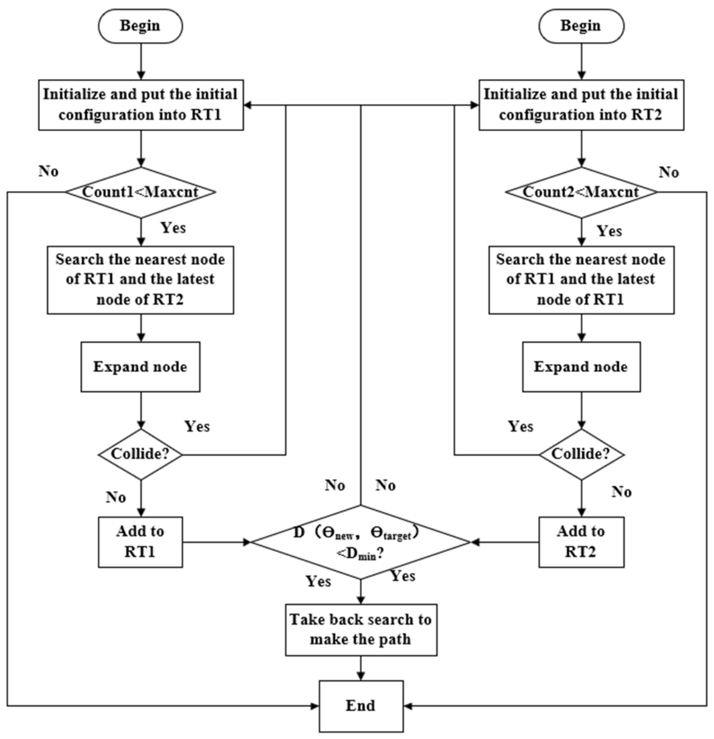

Begin
Initialize and put the initial configuration into RT1
Count1<Maxcnt
No
Yes
Search the nearest node of RT1 and the latest node of RT2
Expand node
Collide?
Yes
No
Add to RT1
Begin
Initialize and put the initial configuration into RT2
Count2<Maxcnt
No
Yes
Search the nearest node of RT1 and the latest node of RT1
Expand node
Collide?
Yes
No
Add to RT2
No
No
D (Θnew, Θtarget) <Dmin?
Yes
Yes
Take back search to make the path
End

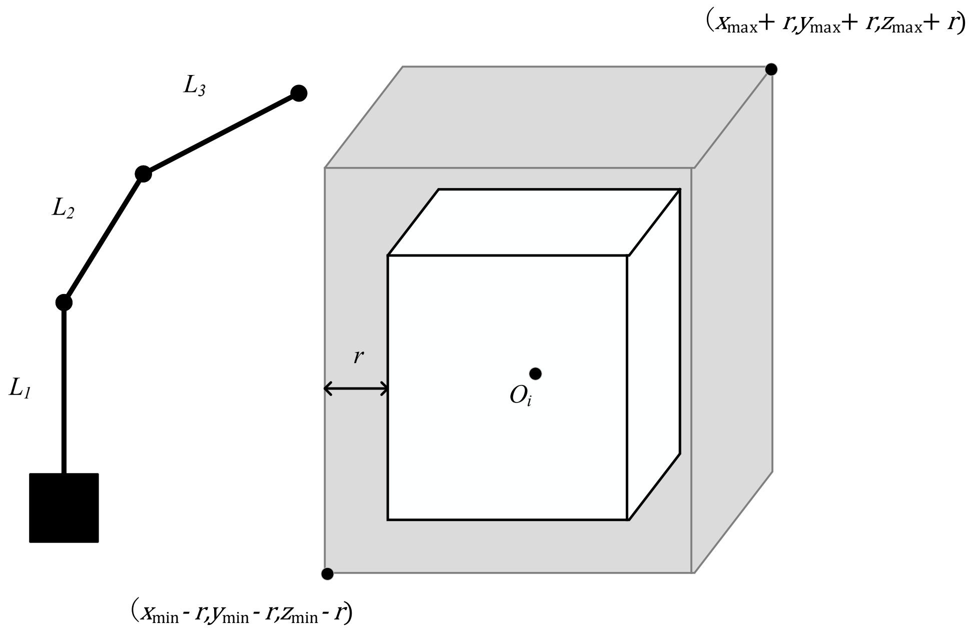
Figure 3. Flowchart of Bi-RRT algorithm.

3.2. Collision Detection

Collision detection can account for up to 90% of the time used in the motion planning process, so designing an appropriate collision detection method can greatly improve the planning efficiency. Envelope box detection method is a commonly used method in robotic arm motion planning, which transforms the collision detection problem into an interference problem of spatial geometry by enveloping the detected object with a simple geometry. There are many kinds of collision detection methods based on the envelope box, and the traditional methods include axis parallel (AABB), enveloping ball, OBB and other envelope methods. In the present study, the AABB bounding box method is selected according to the structural form of the robotic arm and the geometry of the obstacle, where $(x_{min}, y_{min}, z_{min})$ denotes the smallest point of the AABB-type envelope in the spatial Cartesian coordinate system, and $(x_{max}, y_{max}, z_{max})$ denotes the largest point to define an envelope. For further simplification, the radial radius r of the robot linkage is enlarged on the original volume of the obstacle, so as to transform the collision detection into the relative position judgment of the spatial line and plane. The obstacle is defined under the spatial Cartesian coordinate system as follows:

$$\mathrm{Obstacle}\{(\mathrm{x_{min}} - \mathrm{r}, \mathrm{y_{min}} - \mathrm{r}, \mathrm{z_{min}} - \mathrm{r}) \leq (\mathrm{x}, \mathrm{y}, \mathrm{z}) \leq (\mathrm{x_{max}} + \mathrm{r}, \mathrm{y_{max}} + \mathrm{r}, \mathrm{z_{max}} + \mathrm{r})\} \quad (1)$$

As shown in Figure 4, the spatial linear equations of the connecting rods of the enveloping UR5 robotic arm are $\{L_1, L_2, L_3\}$, and the hexahedron of the enveloping obstacle is O_i. If there exists an intersection point between $\{L_1, L_2, L_3\}$ and the facets of O_i, the robotic arm collides with the obstacle, and the path needs to be re-planned, or else the robotic arm moves in accordance with the search trajectory.



$(x_{max}+r, y_{max}+r, z_{max}+r)$
L_3
L_2
L_1
r
O_i
$(x_{min}-r, y_{min}-r, z_{min}-r)$

Figure 4. Simplified model of AABB envelope.

Collision detection algorithms are needed in the planning process of the robot. As can be seen in Figure 4, the obstacle is simplified into an AABB bounding box, and the three joints of the robot are simplified into spatial line segments L_1, L_2 and L_3. The collision detection problem can be transformed into the problem of whether the three line segments intersect with the spatial cube or not, and the collision detection between the robot connecting rod and the obstacle is shown in Figure 5. For ease of description, we assume that the radius of the robot linkage part has been superimposed on the thickness of the bounding box, and the coordinates on the spatial obstacle are denoted as follows:

$$\text{Obstacle}\{(x_{min}, y_{min}, z_{min}) \leq (x, y, z) \leq (x_{max}, y_{max}, z_{max})\} \tag{2}$$

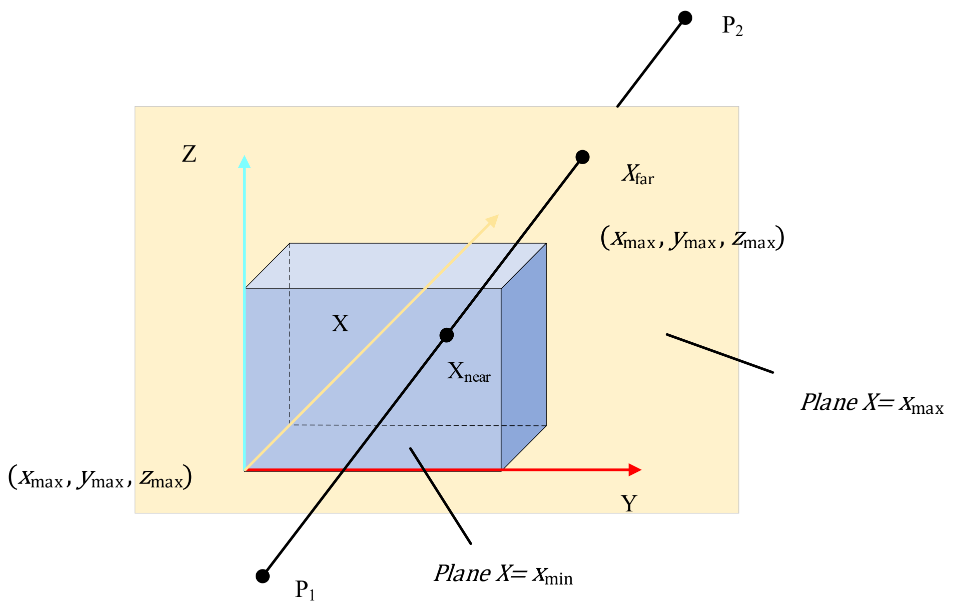


Figure 5. Collision detection between robot linkage and obstacles.

For the link L_1, the intersections of the link L_1 with the two planes $X = x_{min}$ and $X = x_{max}$ in the obstacle bounding box are x_{near} and x_{far}, respectively:

$$\begin{cases} P_1 + \lambda_{x_{near}} \cdot (P_2 - P_1) = [x_{min}, 0, 0]^T \\ P_2 + \lambda_{x_{far}} \cdot (P_2 - P_1) = [x_{max}, 0, 0]^T \end{cases} \tag{3}$$

where P_1 and P_2 are the two ends of the robot linkage, corresponding to the R_2 and R_4 joint coordinates of the robot, and in calculating their coefficients, only the x-component part is required, as shown below:

$$\begin{cases} x_1 + \lambda_{x_{near}} \cdot (x_2 - x_1) = x_{min} \\ x_2 + \lambda_{x_{far}} \cdot (x_2 - x_1) = x_{max} \end{cases} \tag{4}$$

The results obtained are as follows:

$$\begin{cases} \lambda_{x_{near}} = \frac{x_{min} - x_1}{x_2 - x_1} \\ \lambda_{x_{far}} = \frac{x_{max} - x_2}{x_2 - x_1} \end{cases} \tag{5}$$

Similarly, for the AABB bounding box, the intersections of the connecting rod with the two Y-planes and the two Z-planes can be obtained as $\lambda_{y_{near}}, \lambda_{y_{far}}, \lambda_{z_{near}}, \lambda_{z_{far}}$, respectively. The definitions are provided below:

$$\lambda_{near} = \max\left(\lambda_{x_{near}}, \lambda_{y_{near}}, \lambda_{z_{near}}\right) \tag{6}$$

$$\lambda_{far} = \min\left(\lambda_{far_x}, \lambda_{far_y}, \lambda_{far_z}\right) \tag{7}$$

If $\lambda_{near} \leq \lambda_{far}$, the link L_1 will collide with the bounding box of the obstacle; otherwise, no collision will occur. Similarly, this method can be used to determine whether L_2 and L_3 will collide with the bounding box of the obstacle, respectively.

In the obstacle avoidance process, in order to ensure that the robot will not collide with obstacles and can avoid obstacles in real time, it is usually required to maintain a certain distance between the robot and the obstacles. Therefore, the distance detection between each linkage and the obstacle is very important.

Assuming that P_1 and P_2 are the two endpoints of the connecting rod L_1, the linear equation of the space segment L_1 can be obtained from P_1 and P_2. If $m = x_2 - x_1$, $n = y_2 - y_1$, and $p = z_2 - z_1$, then the equation of the link L_1 is as follows:

$$\frac{x - x_1}{m} = \frac{y - y_1}{n} = \frac{z - z_1}{p} = t \tag{8}$$

where t is a parameter of the equation.

According to Equation (4), the center point of the AABB bounding box is defined as follows:

$$O(x_0, y_0, z_0) = O\left(\frac{x_{min} + x_{max}}{2}, \frac{y_{min} + y_{max}}{2}, \frac{z_{min} + z_{max}}{2}\right) \tag{9}$$

The straight line distance from the center point $O(x_0, y_0, z_0)$ of the AABB bounding box to the linkage L_1 can be found by calculating the distance from that point to the nearest point $C(x_1, y_1, z_1)$ on the linkage:

$$dis_1 = \sqrt{(x_0 - (m \cdot t + x_1))^2 + (y_0 - (n \cdot t + y_1))^2 + (z_0 - (p \cdot t + z_1))^2} \tag{10}$$

Similarly, the distance between the center point of the AABB bounding box and the connecting rod L_2 and the distance between the connecting rod L_3 can be calculated separately. Using this method, the distance between each link of the robot and the obstacle can be judged, thus ensuring that the robot can avoid collision with the obstacle during its movement.

3.3. Localized Obstacle Avoider

3.3.1. Soft Actor-Critic

In the present study, the SAC is implemented to solve the problem of localized motion planning for robots during assembly tasks. The SAC algorithm proposed by Haarnoja [24] et al. is a model-free deep reinforcement learning algorithm based on maximum entropy, which is suitable for real-world robots to learn the skills. The SAC algorithm is very efficient, and it solves the problem of reinforcement learning in both discrete action space and continuous action space. Compared with other deep reinforcement learning algorithms, the SAC algorithm introduces the concept of entropy, which can be understood as the degree of randomness, chaos or disorder of the action. The higher the entropy value, the more chaotic the action is, the richer the sampled action information is, and thus the higher reward value can be obtained. The framework of SAC is shown in Figure 6 below.

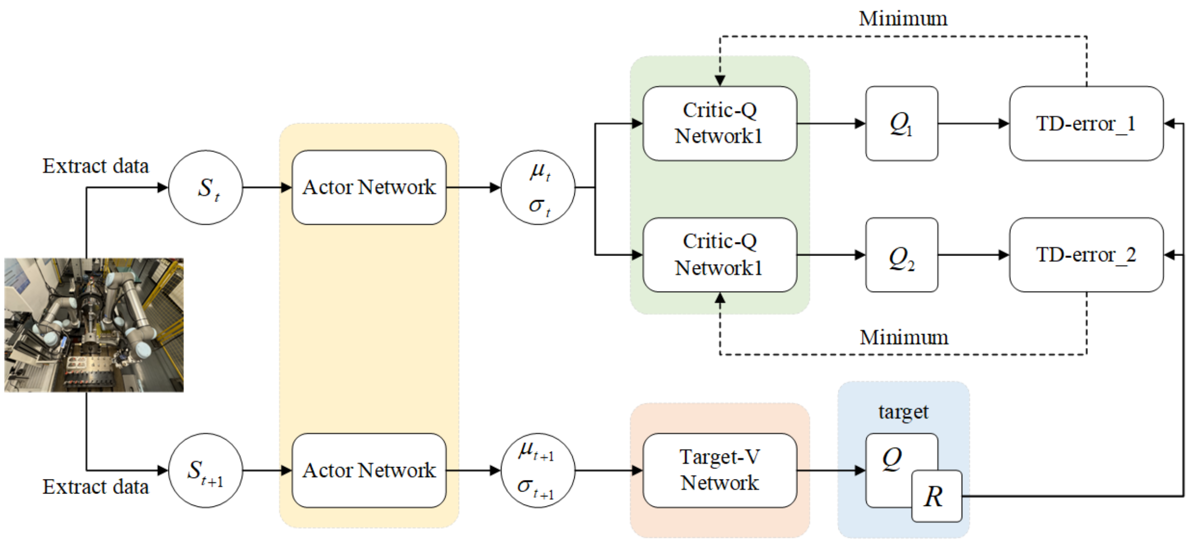


Figure 6. SAC framework diagram.

Introducing entropy into reinforcement learning allows the strategy to be more randomized, thus increasing the chance for the robot to explore the state space. This not only avoids the strategy from falling into local optimal solutions, but also allows the strategy to find more feasible solutions and improves the robustness of the final strategy. Therefore, the optimal strategy formulation of SAC is defined as follows:

$$J(\pi) = \underset{\Pi}{argmax} E_{s_t, a_t \sim \pi(\cdot|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) + \alpha H(\pi(\cdot|s_t)) \right] \tag{11}$$

$$H(\pi(\cdot|s_t)) = E[-log\pi(\cdot|s_t)] \tag{12}$$

where π is used to update the strategy to find the maximum total reward; α is the entropy regularization coefficient, which is used to control the importance of entropy; $H(\pi(\cdot|s_t))$ stands for the entropy value. The larger the entropy value, the more the intelligent body explores the environment, and the more efficient the strategy found, which helps to speed up the learning of the strategy.

The Q-value of SAC can be calculated using Bellman's equation based on entropy improvement with the value function defined as follows:

$$Q(s_t, a_t) = E_{s_{t+1} \sim D}[r(s_t, a_t) + \gamma V^{\pi}(s_{t+1})] \tag{13}$$

where *s*_(*t*+1) is sampled from the empirical playback pool *D*. The state value function is defined as follows:

$$V(s_t) = E_{a_t \sim \pi}[Q(s_t, a_t) - \alpha log\pi(\cdot|s_t)] = E_{a_t \sim \pi}[Q(s_t, a_t) + H(\pi(\cdot|s_t))] \tag{14}$$

$V(s_t)$ denotes the reward that is expected to be received in a given state.

The SAC contains five neural networks: the policy network $\pi_\phi(s_t, a_t)$, the value network $V_\psi(s_t)$, the objective value network $V_{\overline{\psi}}(s_t)$, and two Q-value network numbers $Q_{\theta_1}(s_t, a_t)$ and $Q_{\theta_2}(s_t, a_t)$. In order to find the optimal strategies, stochastic gradient descent is applied to their objective functions:

$$J_V(\psi) = E_{s_t \sim D}\left[\frac{1}{2}\left(V_\psi(s_t) - E_{a_t \sim \pi_\phi}\left[\underset{i=1,2}{min} Q_{\theta_i}(s_t, a_t) - \alpha log\pi_\phi(a_t|s_t)\right]\right)^2\right] \tag{15}$$

In addition, using a form similar to a dual-Q network, the soft Q takes the minimum of two Q-value functions parameterized by θ_1 and θ_2, which helps to avoid overestimating inappropriate Q-values in order to improve training speed. The soft Q-value function is updated by minimizing the Bellman error:

$$J_Q(\theta) = E_{(s_t,a_t) \sim D}\left[\frac{1}{2}\left(Q_{\theta_{i=1,2}}(s_t, a_t) - \left(r(s_t, a_t) + V_{\overline{\psi}}(s_{t+1})\right)\right)^2\right] \tag{16}$$

The objective function of the strategy network can be rewritten as follows:

$$J_\pi(\phi) = E_{a_t \sim \pi, s_t \sim D}\left[log\pi_\phi(s_t, a_t) - \underset{i=1,2}{min} Q_{\theta_i}(s_t, a_t)\right] \tag{17}$$

where the parameter values for adjusting the target soft state are $\overline{\psi} \leftarrow \tau\overline{\psi} + (1-\tau)\overline{\psi}$, $\tau \in [0,\ 1]$.

The SAC algorithm also supports automatic tuning of the temperature coefficient entropy α. When the initial temperature coefficient of the algorithm is large, the agents are encouraged to explore; when the agents are slow to converge, the temperature coefficients can be adaptively decayed. In addition, since SAC is based on the off-policy approach, we train the network by sampling from the empirical playback buffer D. The network can be trained by the off-policy approach, which is based on the off-policy approach.

3.3.2. Prioritized Experience Replay

When drawing samples from the experience replay buffer for training, if random sampling is used, the probability of drawing high value samples is very low, which leads to inefficient learning by the agent. The idea of Prioritized Experience Replay (PER) is to repeat at a higher frequency those experiences that are of higher value to the learning process, and these high-value experience samples are usually either very successful or very unsuccessful samples.

In Deep Reinforcement Learning, Temporal Difference Error (TD Error) represents the degree to which an intelligent body needs to learn. TD Error measures the value of the experience. The larger the TD Error is, the larger the gap between the current Q value and the target value, and the more updating is needed. By frequently learning these important experiences, we can avoid negative behaviors more quickly. By learning these important experiences more often, the agent can learn positive experiences faster and avoid negative behaviors.TD Error serves as a criterion for distinguishing the importance of experiences, and our goal is to make the TD Error smaller.

In order to minimize over-estimation and improve the stability of training, the SAC algorithm employs two Q-networks and uses smaller Q-values for gradient computation. In addition, SAC also includes the strategy network π, so the TD Errors of all three networks need to be considered simultaneously. The TD Error ξ_i is chosen as a metric to evaluate the empirical value and is calculated as follows:

$$\xi_i = |T_e(Q_1)| + |T_e(Q_2)| + \beta|T_e(\pi)| \tag{18}$$

where β is a tuning parameter that balances the TD Error weights computed by the policy network and the Q network. To prevent network overfitting, experiences are extracted probabilistically to ensure that even experiences with a TD error of 0 are sampled. The prioritization value for each experience is defined as follows:

$$P(i) = \frac{\delta_i}{\sum \delta_i} \tag{19}$$

where $\delta_i = |\zeta_i + \varepsilon|$ and ϵ denotes a very small value that prevents sampling with probability 0.

3.3.3. Reward Function

In order for the robot to avoid the moving obstacles in the environment, the intelligent body must reward the robot for the process of avoiding the obstacles. During motion planning, the robot will prioritize the dynamic obstacle avoidance task before the planning task. When a dynamic obstacle appears in the robot's workspace and is in a non-safe area, i.e., the distance between the robot's linkage and the obstacle is less than the safe distance d_s, the obstacle avoidance operation should be prioritized to ensure that the robot does not collide with the obstacle.

$$R_a = \sum_{i=1}^{n=3} \lambda_i \cdot [dis_i(t) - dis_i(t-1)] \tag{20}$$

where dis_i is the distance between each linkage and the obstacle described in Equation (12) above and the center point. When $dis_i(t)$ is smaller than the previous $dis_i(t-1)$, it means that the distance between the robot and the dynamic obstacle in the environment is shrinking, so it receives a negative reward; when $dis_i(t)$ is larger than $dis_i(t-1)$, it means that the distance between the robot and the dynamic obstacle in the environment is expanding, so it receives a positive reward.

When the robot is in the safe zone, i.e., the distance between the robot's linkage and the obstacle is greater than the safe distance d_s, the reward value for this part is defined as follows:

$$R_p = k \cdot dis_t \tag{21}$$

where dis_t is the distance between the target node and the end-effector of the robotic arm in the working interval:

$$dis_t = \sqrt{(x_g - x_t)^2 + (y_g - y_t)^2 + (z_g - z_t)^2} \tag{22}$$

where (x_t, y_t, z_t) are the 3D coordinates of the target node location and (x_g, y_g, z_g) are the 3D coordinates of the end-effector; in this case, the end-effector will be located closer towards the target node in the safe area.

When any of the three linkages collide with a moving obstacle in the environment, a considerable negative reward R_c is generated, and when the robotic arm reaches the target node, a considerable positive reward R_g is obtained. Therefore, the reward function for the motion planning task in present study has the following form:

$$r = \begin{cases} R_g, & \text{Getting to the target} \\ R_c, & \text{Collision} \\ \sum_{i=1}^{n=3} \lambda_i \cdot [dis_i(t) - dis_i(t-1)], & dis_i < d_s \\ R_p, & \text{Other} \end{cases} \tag{23}$$

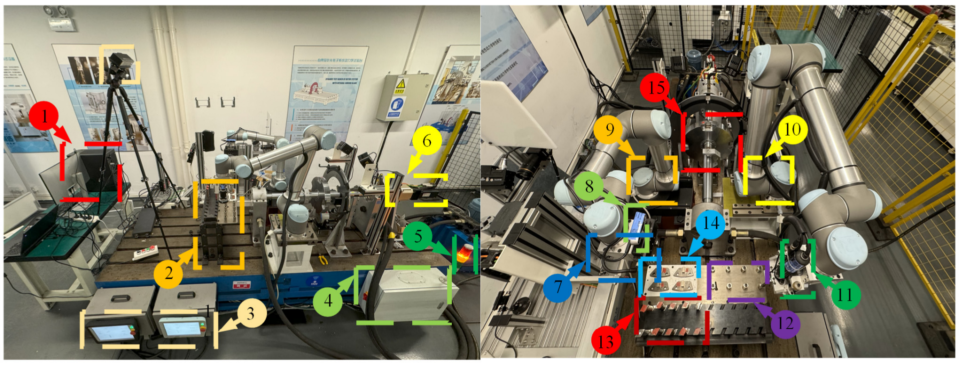
where λ_i is the corresponding weight. Workflow of the proposed algorithm is shown in Algorithm 1.

Algorithm 1 Pseudocode of SAC-Bi-RRT Motion Planning

```
1: Initialize parameter vectors $\phi, \psi, \overline{\psi}, \theta$ and experience replay buffer D with size S
2: Initialize Xbox camera at global position and depth camera on the robot arm
3: Use camera to get static obstacles and initialize Bi-RRT for global path planning
4: Generate initial global path using Bi-RRT from start to goal
5: for each episode do
6:     for each environment step do
7:         Use camera to detect dynamic obstacles
8:         if dynamic obstacles are detected then
9:             Activate depth camera to get precise local obstacle information
10:            According to the state $s_t$ and the policy $\pi_\phi$, sample to get action $a_t$
11:            Execute $a_t$ in the environment
12:            Obtain next state $s_{t+1}$, reward r, and whether done signal d
13:            Store $s_t, a_t, r, s_{t+1}, d$ in the experience replay buffer D
14:            Calculate the TD-error $\Delta$ for the sample and update the priority $P_i$ in D
15:            Find the nearest node on the global path to the current position
16:        end if
17:        if deviation from global path is too large then
18:            Replan the global path using Bi-RRT from the current position to the goal
19:        end if
20:    end for
21:    for each gradient step do
22:        Sample a minibatch of N experiences from D based on priorities
23:        Update Q-network $Q_\theta(s,a)$: $\theta_i \leftarrow \theta_i - \eta_Q \nabla J_Q(\theta_i)$
24:        Update value network $V_\psi(s)$: $\psi \leftarrow \psi - \eta_V \nabla J_V(\psi)$
25:        Update policy network $\pi_\phi(s,a)$: $\phi \leftarrow \phi - \eta_\pi \nabla J_\pi(\phi)$
26:        Update target value network $V_\psi(s)$: $\overline{\psi} \leftarrow \tau\overline{\psi} + (1-\tau)\overline{\psi}$
27:        Calculate the new TD-error for each sample in the minibatch and update their priorities in D
28:    end for
29: end for
```

4. Case Studies

This study analyzes an actual robotic arm assembly scenario as shown in Figure 7, which mainly consists of two robotic arms (UR5 and UR10), a servo-motor-controlled rotor vane assembly module, a material table for placing a variety of workpieces to be gripped, and multiple inspection devices (RealSense d435i and Kinect V2) (Intel RealSense D435i, Intel Corporation, Santa Clara, CA, USA; Microsoft Kinect V2, Microsoft Corporation, Redmond, DC, USA). The end fixtures of the collaborative robotic arms UR5 and UR10 are both self-designed parallel grippers with a motorized Allen wrench at the end of UR10, where UR5 is used for gripping the blades and covers, and UR10 is used for gripping the bolts and tightening the bolts. The servo motor controls the rotation of the leaf disk. After completing the assembly of one set of blades, the servo motor controls the leaf disk through a 1:40 reducer to realize a low-speed and high-precision rotation, rotating 90° clockwise to the next assembly position. The robotic arm, global camera and depth camera are connected to the host computer through Ethernet.



1	Kinect V2	2	material platform	3	teaching aid	4	control box	5	warning light
6	servo motor	7	gripper	8	RealSense d435i	9	UR5	10	UR10
11	screwdriver	12	screw	13	blade	14	cover	15	leaf disk

Figure 7. Lab bench assembly scene (the left picture shows the working scene of the whole experimental bench; the right picture shows the global camera view, the bottom part is the material table where the workpieces are to be gripped, the center is the two UR robotic arms, and the top part is the rotor carousel to be assembled). The assembly task is to grasp various workpieces on the material table by the robotic arms and assemble them on the rotor carousel above.

During this robotic arm assembly process, multiple dynamic obstacles may appear in the work environment, which mainly include operators involved in the human–machine collaboration, other robotic arms, parts and components, temporary tools and equipment, and cables and pipelines. As shown in Figure 8, to cope with these dynamic obstacles, we use a Kinect V2 industrial camera placed in a position where the entire workspace can be observed as a global camera; at the same time, a RealSense d435i depth camera is placed on the robot arm as a localized camera. These sensors are able to collect point cloud data in real time within the work area and generate AABB bounding boxes for each obstacle. The data quality is ensured by denoising and filtering the point cloud data. Using the collision detection algorithm described above, combined with Bi-RRT global path planning and SAC local obstacle avoidance algorithms, dynamic obstacles are detected and processed in real time, and the motion trajectory of the robotic arm is dynamically adjusted to ensure its safe and efficient operation in complex environments.

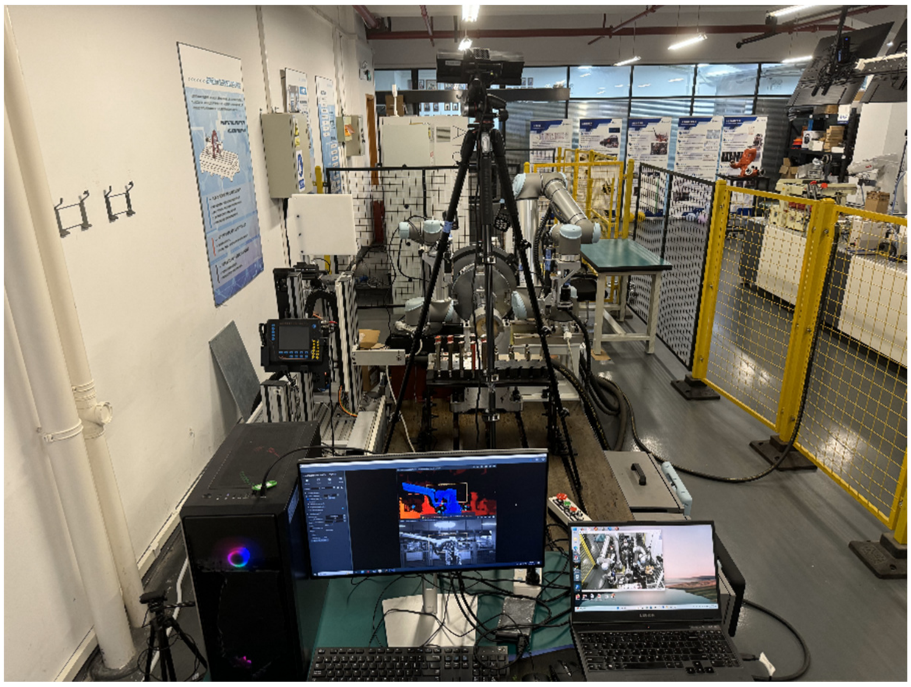


Figure 8. Inspection view during assembly (the left display in the figure shows a localized view of what the RealSense d435i camera can see, and the right display shows the global view of the Kinect V2 camera).

Considering the slow speed of training a real robot and the easy damage of the robot, the present study first constructed a dynamics and kinematics model of the robot in the physics simulation engine, simplified the trajectory planning problem in this assembly process to be realized in the simulation environment, and represented the obstacles with AABB bounding boxes. During the simulation process, the following aspects were focused on: first, evaluating the real-time response capability of the system in the face of dynamic obstacles to ensure that the robotic arm can adjust its trajectory in time to avoid a collision; second, verifying the accuracy of the Bi-RRT global path planning and SAC local obstacle avoidance algorithms to ensure that the robotic arm can accurately avoid obstacles and arrive at the target location; in addition, testing the Kinect V2 and the RealSense d435i camera's environmental perception stability under different environmental conditions to ensure the stability and accuracy of point cloud data acquisition. Finally, the robustness of the system is tested under dynamic obstacle interference to ensure that the robotic arm can still operate safely in complex environments.

Based on the techniques and methods described above, we validate the effectiveness of the methods by simplifying and modeling the simulation environment of the key problem. The simulation environment is shown in Figure 9, with a robotic arm and a simple dynamic obstacle, and using virtual global and local cameras for real-time data acquisition. The obstacle data are processed by a collision detection algorithm and a path planning algorithm that dynamically adjust the robotic arm path to ensure obstacle avoidance.

The experiment is simulated using a UR5 collaborative robotic arm, where the manipulator needs to grasp a randomly initialized object and place it at a specified location. There is a random moving obstacle in the path, which the robotic arm needs to avoid. In addition, the robotic arm needs to avoid collisions with the tabletop as well as its own links. During each training session, the initial configuration of the robotic arm, the target location, and the location of the obstacle are chosen randomly. The positions of the obstacles and the target do not change during the training process, but the position of the obstacles and the target position can change dynamically during the testing process.

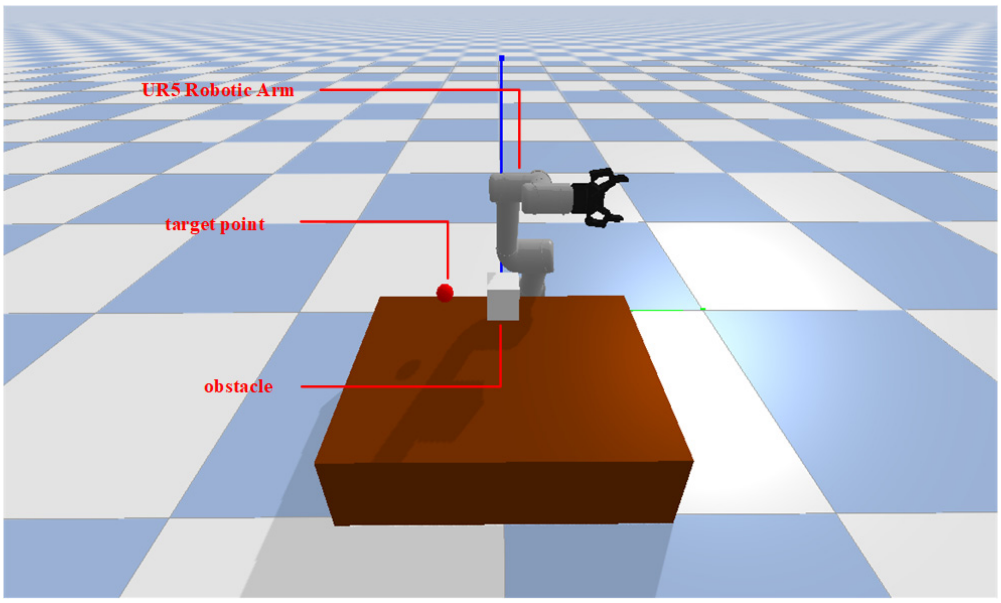


Figure 9. Simulation of the robotic arm during the training process (the UR5 robotic arm present in the figure takes the position of the workpiece to be gripped or the position to be assembled as the target point, and the various obstacles in the complex environment are assumed to be the obstacles of the square in the figure, so that the robotic arm moves from a random position to the target point position and avoids the dynamic obstacles in the environment in the process).

In the present study, the training process uses the improved SAC algorithm to train the network. For each round of training, the current round is ended when the robot successfully reaches the intermediate goal point or collision occurs. The rewards for successfully reaching the target node and bumping into the obstacle are set to $r_g = +4, r_c = -5$, respectively, along with the corresponding reward weights given during the exploration process: $k = -2, \lambda_1 = +1, \lambda_2 = +2, \lambda_3 = +3$ and the parameters inside are $d_t = 0.05m,\ d_s = 0.2m$, and each sample generated by interacting with the environment is stored in the experience playback buffer, and then the sampling probability of each sample is updated. The SAC algorithm has a large number of hyperparameters that affect the effectiveness of the algorithm training. The parameters of the network used in the training are updated by sampling from the experience playback buffer D. The basic parameters used for the training are referenced in other articles in the same field on hyperparameter settings [25,26] and are given in Table 1 with determinable hyperparameter settings.

Table 1. Table of reinforcement learning parameters.

Parameter	Value
Actor learning rate	0.0001
Critic learning rate	0.0001
Discount factor	0.99
Maximum training steps	1,000,001
Batch size	256
Replay buffer size	1,000,000

5. Results and Discussion

Success rate and training speed are two main indicators to evaluate the training effect of a neural network motion planner. In the experimental process, the improved SAC algorithm is used to train the network 100,000 times. Each round of training is randomly given the position where the target appears and the starting position of the robot, and the end of the robot is controlled to move to the target position. During training, we save the model every 500 times, and then test the saved model, calculate the success rate and plot the success rate curve. In addition, for the improved SAC algorithm, we compare several other deep reinforcement learning algorithms (DDPG [27] and DQN [28]) and plot the data in graphs. Figure 10 shows the success curves of the three different algorithms for this problem. Meanwhile, the reward values during training were recorded and calculated, and the three deep reinforcement learning reward value curves were plotted according to the definition of the reward function, as shown in Figure 11. Comparison with the above figure shows that the improved SAC algorithm is more stable and performs better. From the figure, it can be seen that with the increase in the number of training points, the reward convergence is very fast, and the reward values are all gradually stabilized. When the number of training points reaches a certain number of times, the improved SAC algorithm can complete the path planning of the dynamic obstacle avoidance task with a high success rate.

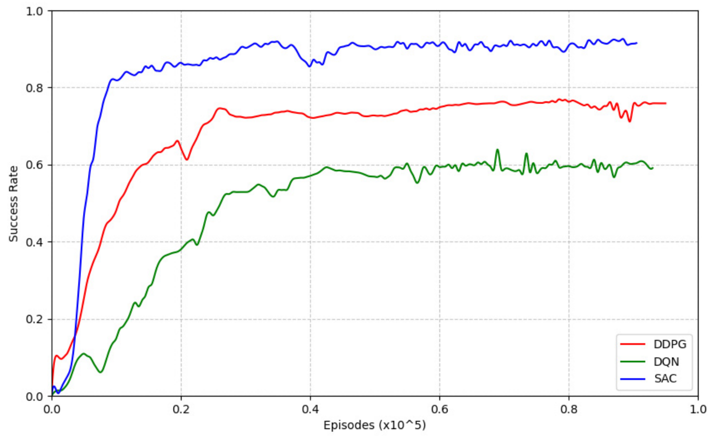


Figure 10. Success rate graphs during training (SAC, DDPG, DQN).

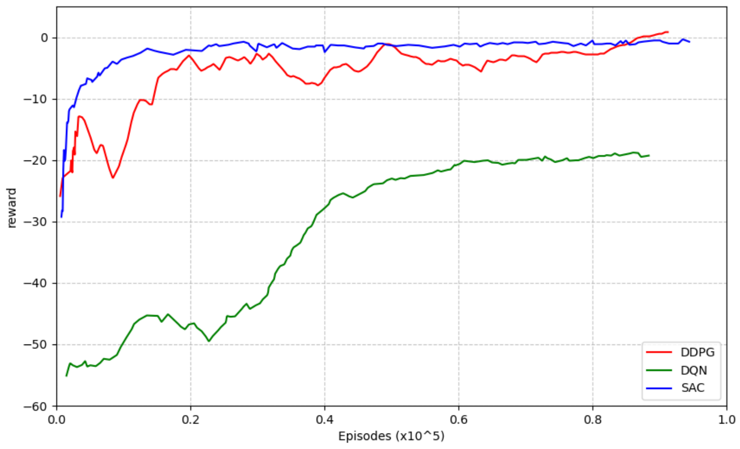


Figure 11. Plot of reward values during training (SAC, DDPG, DQN).

In the present study, the improved SAC is chosen as the main algorithm for local trajectory planning and compared with the Deep Deterministic Policy Gradient (DDPG) and Deep Q Network (DQN). Among them, DDPG is a reinforcement learning algorithm based on actor-critic architecture for continuous action space, which combines the advantages of offline policy learning and policy gradient methods of DQN to achieve efficient policy optimization. The DDPG is able to select the optimal obstacle avoidance action in continuous action space to adapt to the complex motion requirements of the robotic arm. The DQN (Deep Q Network) is a deep reinforcement learning network. As an important algorithm for discrete action spaces, the DQN approximates the Q-value function through a neural network and uses empirical replay and a goal network to stabilize the training process. DQN performs well when dealing with high-dimensional state spaces, and although it was originally used for discrete actions, discretizing the continuous action space of a robotic arm enables the DQN to be applied to solving dynamic obstacle avoidance problems as well. Nevertheless, the training process of DQN is relatively simple and stable, which makes it more suitable for tasks where the actions can be discretized efficiently and the dimensionality of the state space is moderate.

Comparing the plots of the success rate and reward function of the three algorithms, we can see the trend of SAC rising rapidly and stabilizing gradually. The DDPG shows greater volatility during the training process, but eventually reaches a high success rate and reward value. The DQN algorithm, although stable during the training process, does not perform as well as SAC and DDPG in the continuous action control task, with a lower success rate and reward value are lower. The rationality and effectiveness of choosing the SAC algorithm for local trajectory planning of a robotic arm are further verified.

In the simulation environment, Figure 12 shows the whole process of robotic arm path planning and dynamic obstacle avoidance. First, in Figure 12a, the robotic arm is located at the initial position and global path planning is performed using the Bi-RRT algorithm. Then, in Figure 12b, the robotic arm starts to move towards the target position (indicated by the red sphere) where the white squares indicate the obstacles. Figure 12c,d show the robotic arm performing an obstacle avoidance process after detecting an obstacle, and adjusting the local path to avoid the obstacle by the SAC algorithm. Then, in Figure 12e, the robotic arm continues to move towards the target point after successfully bypassing the obstacle. Finally, in Figure 12f, the robotic arm reaches the target point and is ready to perform the grasping task. The whole process demonstrates in detail the path planning ability of the robotic arm in a dynamic obstacle environment, moving from the initial position to the target point and successfully avoiding obstacles.

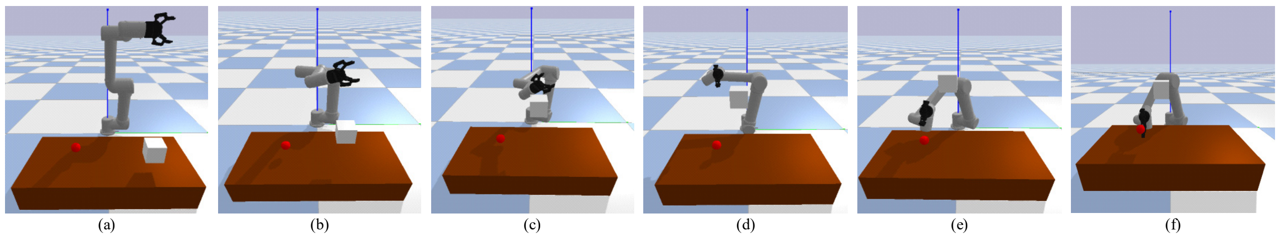


Figure 12. The whole path planning and dynamic obstacle avoidance process of the robotic arm. ((**a**) Robot initial position. (**b**,**c**) Avoiding dynamic obstacles. (**d**,**e**) Movement towards the target. (**f**) Target location).

This next section tests the performance of the SAC-Bi-RRT in the presence of dynamic obstacles. First, an arbitrary pose of the obstacle is randomly generated in space, and then the obstacle is made to move in one direction at a uniform speed while the manipulator arm moves. In the experiments of trajectory planning, the obstacle and the manipulator move simultaneously. A comparison of three different path planning algorithms, SAC-Bi-RRT, DDPG-Bi-RRT, and SAC, is shown in Figure 13, where the blue trajectories represent the paths generated using Bi-RRT when there is no obstacle in the working interval, and the green trajectories represent the paths planned by SAC-Bi-RRT, DDPG-Bi-RRT, and SAC algorithms, respectively. The RRT algorithm and the SAC-Bi-RRT algorithm are compared in Figure 13a, the RRT algorithm and the DDPG-Bi-RRT algorithm are compared in Figure 13b, and the Bi-RRT algorithm and the SAC algorithm are compared in Figure 13c. The start position and target position in each figure are marked with different colors, and the red and green squares indicate the start position and final position of the obstacle, respectively. It can be seen that as the obstacle moves, the path of the manipulator gradually deviates from the initial path, reflecting its real-time obstacle avoidance capability. In contrast, the SAC-Bi-RRT algorithm outperforms the DDPG-Bi-RRT and SAC algorithms in complex dynamic environments. This further validates the effectiveness and adaptability of SAC-Bi-RRT in dealing with dynamic obstacle environments.

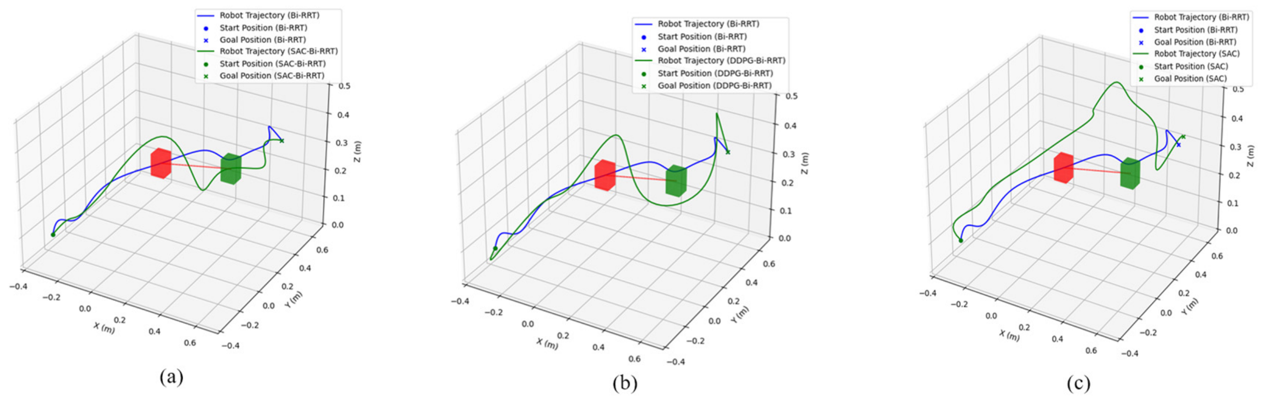


Figure 13. Planned path of the robotic arm in the absence and presence of dynamic obstacles. (**a**) The RRT algorithm and the SAC-Bi-RRT algorithm are compared. (**b**) The RRT algorithm and the DDPG-Bi-RRT algorithm are compared. (**c**) The Bi-RRT algorithm and the SAC algorithm were compared.

In a further validation process, 100 new planned tasks were used as a set of validation sets and the already trained network model was tested based on them. The test results are shown in Table 2, demonstrating the success rate, average path length, and average time taken for the 100 tests. The Bi-RRT algorithm is a path planned in the absence of obstacles in the environment, and in order to observe the results of the improved algorithm, a comparison of the paths versus the time has been included, but its success rate is not listed. In the same environment, combining the SAC algorithm with the Bi-RRT algorithm to jointly realize robotic arm motion planning is more effective than other algorithms to realize motion planning. Specifically, the SAC-Bi-RRT algorithm has a significantly higher success rate than the other algorithms, which indicates that the algorithm is more robust and adaptable in dynamic obstacle avoidance tasks. The improved success rate not only reflects the optimization ability of the algorithm in path planning but is also affected by a combination of factors. For example, the number of obstacles and their motion characteristics in a dynamic environment can significantly affect the difficulty of the task, and the SAC-Bi-RRT algorithm is able to adapt to these changes more flexibly. In addition, sensor accuracy and data latency can have an impact on the perception of obstacle locations, thus indirectly affecting the accuracy of planning; in this experiment, the algorithm effectively reduces the negative impact of these external disturbances through a more efficient perception and response mechanism.

Table 2. Performance comparison of algorithms.

	Success Rate	Path Length	Average Time
Bi-RRT	\	1.41	4.93
SAC	0.87	1.88	6.63
SAC-Bi-RRT	0.95	1.73	6.14
DDPG-Bi-RRT	0.79	1.95	7.06

In addition to the improved success rate, the SAC-Bi-RRT algorithm generates shorter paths, and its efficiency in path optimization is significantly improved. This not only reduces the time for task execution, but also reduces the complexity of the paths and possible deviations during the movement of the robotic arm. At the same time, the algorithm has a shorter average planning time, reflecting its computational efficiency in dynamic environments. Taken together, the SAC-Bi-RRT algorithm outperforms other algorithms in three key metrics: success rate, path length and average planning time.

Although the SAC-Bi-RRT algorithm performs well in the simulation environment, there are still some potential limitations. First, the dynamic characteristics of obstacles in the simulation environment are simplified, and future research needs to consider more complex and dynamic obstacles, especially in real scenarios with many obstacles. Second, the algorithms may face errors and delays in the processing of real sensor data, which have an impact on the effectiveness of obstacle avoidance. Finally, although we consider partial control and sensor assumptions, the real-time and computational efficiency of the algorithms may become a bottleneck in more complex systems, such as multiple robotic arm collaboration or large-scale industrial applications. Therefore, in future work, we will continue to explore how to improve the scalability and robustness of the method, especially in complex and dynamic environments.

6. Summary and Outlook

In the present study, a visual assembly method based on SAC-Bi-RRT with two-layer motion planning is proposed for complex operational task requirements in industrial scenarios to improve the agent of robotic arm assembly in dynamic environments. The SAC algorithm and Bi-RRT algorithm are combined for motion planning, and a global camera is used to monitor the entire operation interval of the robotic arm and obtain environmental information throughout this interval. The method combines the advantages of Bi-RRT for fast global exploration and the SAC algorithm for handling complex problems, enabling the assembly task of the robotic arm to be applied to more complex and randomized scenarios. The SAC deep reinforcement learning algorithm is used for training and testing in simulation. To address the problem of low sample learning efficiency, prioritized experience replay (PER) is used to adjust the sample weights and improve sampling efficiency. To enhance the efficiency and quality of path planning, a heuristic search approach is applied in the Bi-RRT construction process. It is verified that SAC-Bi-RRT demonstrates better performance in terms of success rate, stability, and completion time in the robotic arm motion planning task.

Despite the good experimental results achieved by the method in the simulation environment, there are still some limitations, especially when applying the algorithm to real-world environments. To address these issues, future research will focus on improving the practical applicability of the algorithm; specifically, we plan to introduce multi-sensor fusion technology to improve the accuracy and robustness of sensor data and reduce the impact of environmental complexity and dynamic changes on path planning. Meanwhile, to further improve the real-time and computational efficiency of the algorithms, future research will explore the use of strategies such as hyperparameter optimization, migration learning, and online learning to speed up the training process and reduce the algorithm's response time. In addition, with the increasing scale of multi-robot systems, the algorithms may face larger computational overheads and coordination scheduling challenges; therefore, future work will enhance the scalability and adaptability of the algorithms in multi-robot collaborative scenarios by means of distributed computing and collaborative control strategies.

Although satisfactory experimental results have been achieved in this study, transitioning the approach from a simulation environment to a real robotic arm system still faces many challenges, including issues of sensor accuracy, real-time data processing, and adaptability of the control algorithm. Future research will focus on addressing these issues and further validating the algorithm's performance in real industrial environments, especially its applicability and performance in complex and dynamic environments. In summary, future work will focus on the scalability of the algorithm, training efficiency, multi-robot collaboration, and feasibility in real-world applications, aiming to promote the success of the method in a wider range of practical applications.

Author Contributions: Conceptualization, Q.Z. and S.H.; methodology, S.H.; software, Q.Z. and J.D.; validation, Q.Z. and J.Q.; formal analysis, S.H. and J.D.; investigation, S.H.; resources, S.H. and Y.Z.; data curation, Q.Z. and S.H.; writing—original draft preparation, S.H.; writing—review and editing, Q.Z., J.Q. and Y.Z.; supervision, J.D.; project administration, Q.Z.; funding acquisition, S.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data will be made available on request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Qi, R.; Zhou, W.; Liu, J.; Zhang, W.; Xiao, L. Obstacle avoidance trajectory planning for gaussian motion of robot based on probability theory. *J. Mech. Eng.* **2017**, *53*, 93–100. [CrossRef]
2. Kong, L.; He, W.; Yang, C. Adaptive Fuzzy Control for Coordinated Multiple Robots With Constraint Using Impedance Learning. *IEEE Trans. Cybern.* **2018**, *49*, 3053–3063. [CrossRef] [PubMed]
3. LaValle, S. Rapidly-exploring random trees: A new tool for path planning. In *Research Report 9811*; Department of Computer Science, Iowa State University: Ames, IA, USA, 1998.
4. Shen, H.; Xie, W.F.; Tang, J.; Zhou, T. Adaptive manipulability-based path planning strategy for industrial robot manipulators. *IEEE/ASME Trans. Mechatron.* **2023**, *28*, 1742–1753. [CrossRef]
5. Zhang, Q.; Li, H.; Duan, J.; Qin, J.; Zhou, Y. Multi-Objective Point Motion Planning for Assembly Robotic Arm Based on IPQ-RRT* Connect Algorithm. *Actuators* **2023**, *12*, 459. [CrossRef]
6. Ferguson, D.; Kalra, N.; Stentz, A. Replanning with rrts. In Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, FL, USA, 15–19 May 2006; ICRA 2006. IEEE: Piscataway, NJ, USA, 2006; pp. 1243–1248.
7. Li, Y. An RRT-based path planning strategy in a dynamic environment. In Proceedings of the 2021 7th International Conference on Automation, Robotics and Applications (ICARA), Prague, Czech Republic, 4–6 February 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–5.
8. Han, D.; Nie, H.; Chen, J.; Chen, M. Dynamic obstacle avoidance for manipulators using distance calculation and discrete detection. *Robot. Comput. Integr. Manuf.* **2018**, *49*, 98–104. [CrossRef]
9. Chen, G.; Liu, D.; Wang, Y.; Jia, Q.; Zhang, X. Path planning method with obstacle avoidance for manipulators in dynamic environment. *Int. J. Adv. Robot. Syst.* **2018**, *15*, 1729881418820223. [CrossRef]
10. Mukadam, M.; Dong, J.; Yan, X.; Dellaert, F.; Boots, B. Continuous-time Gaussian process motion planning via probabilistic inference. *Int. J. Robot. Res.* **2018**, *37*, 1319–1340. [CrossRef]
11. Zhu, J.; Zhao, S.; Zhao, R. Path planning for autonomous underwater vehicle based on artificial potential field and modified RRT. In Proceedings of the 2021 International Conference on Computer, Control and Robotics (ICCCR), Shanghai, China, 8–10 January 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 21–25.
12. Wang, B.; Liu, Z.; Li, Q.; Prorok, A. Mobile robot path planning in dynamic environments through globally guided reinforcement learning. *IEEE Robot. Autom. Lett.* **2020**, *5*, 6932–6939. [CrossRef]
13. Cheng, Y.; Zhang, W. Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. *Neurocomputing* **2018**, *272*, 63–73. [CrossRef]
14. Bing, Z.; Brucker, M.; Morin, F.O.; Li, R.; Su, X.; Huang, K.; Knoll, A. Complex robotic manipulation via graph-based hindsight goal generation. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 7863–7876. [CrossRef]
15. Abdi, A.; Adhikari, D.; Park, J.H. A novel hybrid path planning method based on q-learning and neural network for robot arm. *Appl. Sci.* **2021**, *11*, 6770. [CrossRef]
16. Abdi, A.; Ranjbar, M.H.; Park, J.H. Computer vision-based path planning for robot arms in three-dimensional workspaces using Q-learning and neural networks. *Sensors* **2022**, *22*, 1697. [CrossRef] [PubMed]

17. Zhang, T.; Zhang, K.; Lin, J.; Louie, W.Y.; Huang, H. Sim2real learning of obstacle avoidance for robotic manipulators in uncertain environments. *IEEE Robot. Autom. Lett.* **2021**, *7*, 65–72. [CrossRef]
18. Zhang, S.; Xia, Q.; Chen, M.; Cheng, S. Multi-Objective Optimal Trajectory Planning for Robotic Arms Using Deep Reinforcement Learning. *Sensors* **2023**, *23*, 5974. [CrossRef] [PubMed]
19. Berenson, D.; Kuffner, J.; Choset, H. An optimization approach to planning for mobile manipulation. In Proceedings of the 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 1187–1192.
20. Xia, F.; Li, C.; Martín-Martín, R.; Litany, O.; Toshev, A.; Savarese, S. Relmogen: Integrating motion generation in reinforcement learning for mobile manipulation. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 4583–4590.
21. Yamada, J.; Lee, Y.; Salhotra, G.; Pertsch, K.; Pflueger, M.; Sukhatme, G.; Lim, J.; Englert, P. Motion planner augmented reinforcement learning for robot manipulation in obstructed environments. In Proceedings of the Conference on Robot Learning, London, UK, 8–11 November 2021; PMLR: New York, NY, USA, 2021; pp. 589–603.
22. Kontoudis, G.P.; Vamvoudakis, K.G. Kinodynamic motion planning with continuous-time Q-learning: An online, model-free, and safe navigation framework. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3803–3817. [CrossRef] [PubMed]
23. Chen, P.; Pei, J.; Lu, W.; Li, M. A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance. *Neurocomputing* **2022**, *497*, 64–75. [CrossRef]
24. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the International Conference on Machine Learning, Vienna, Austria, 25–31 July 2018; PMLR: New York, NY, USA, 2018; pp. 1861–1870.
25. Zhou, C.; Huang, B.; Fränti, P. Representation learning and reinforcement learning for dynamic complex motion planning system. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *35*, 11049–11063. [CrossRef] [PubMed]
26. Bai, C.; Zhang, J.; Guo, J.; Yue, C.P. Adaptive Hybrid Optimization Learning-Based Accurate Motion Planning of Multi-Joint Arm. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *34*, 5440–5451. [CrossRef] [PubMed]
27. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
28. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.