Expert Systems With Applications 193 (2022) 116379

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

MDA + RRT: A general approach for resolving the problem of angle constraint for hyper-redundant manipulator

Longfei Jia, Yuping Huang *, Ting Chen, Yaxing Guo, Yecheng Yin, Jing Chen

Beijing Institute of Precision Mechatronics and Controls, Laboratory of Aerospace Servo Actuation and Transmission, China

ARTICLE INFO

Keywords:
Hyper-redundant manipulator
RRT algorithm
Maximum deflection angle
Path planning

ABSTRACT

In this paper, a cable-driven hyper-redundant manipulator with 17 degrees of freedom is presented for narrow and complex environments firstly. Based on the mechanical structure, the MDA + RRT algorithms is proposed for path planning considering the maximum deflection angle of joint. First, the transformation relationship between the path parameters and manipulator parameters is obtained through the theorem of sine of triangles and two extreme points, which can convert limitation of the deflection angle of the joint to the limitation of the deflection angle of the path. Then, three corresponding improved algorithms are proposed: MDA-RRT, MDA-RRT*, MDA-QRRT* on the basis of the traditional RRT, RRT* and Q-RRT* algorithms. Finally, six algorithms are applied to plan the path of avoiding three different obstacles respectively by simulation. The results demonstrate that, three improved algorithms can guarantee that the planned path satisfies the constraint of deflection angle of joint without increasing the computational amount compared with the three traditional algorithms, only by limiting the selection range of the next random vertex. Among the six algorithms, the feasibility and optimization of the MDA-QRRT* algorithm are the best and the feasibility and optimization of the three improved algorithms are also better than the corresponding traditional algorithms.

1. Introduction

Path planning technology has been widely applied in many fields, such as: autonomous driving of unmanned vehicles (Jordi, & Alberto, 2019), autonomous planning of remotely controlled aircraft (Arun, & Craig, 2019), navigation of cruise missiles (Fang, Qiao, Zhang, Yang, & Peng, 2016), autonomous collision-free actions of robots (Singh, & Thongam, 2019), and so on. However, with the development of technology and the increase in demand, traditional algorithms for path planning cannot meet the planning conditions in complex environments. Probabilistic Road Map (PRM) algorithm (Wang, & Cai, 2018) has low search efficiency in complex environments. The computational complexity for A* algorithm (Koenig, Likhachev, & Furcy, 2004) depends on the completeness of grid resolution and the algorithm is not suitable for high-dimensional environmental path planning. Neural network algorithm (Singh, & Thongam, 2019; You, Lu, Dimitar, & Panagiotis, 2019) is highly dependent on the learning environment and has poor generalization. Particle Swarm Optimization (PSO) algorithm (Yasmine, & Abdelhamid, 2019) is easy to fall into a regional optimal solution and is not suitable for path planning in local and high-dimensional environments. And also, Heuristic search algorithms (Mohammad, 2019), gravitational search algorithm (Rashedi, Nezamabadi-Pour, & Saryazdi, 2009) and bat algorithm (Douglas, Luís, Rodrigo, Kelton, Yang, André, & João, 2014) are not suitable for path planning in high-dimensional complex environments. The Rapidly-Exploring Random Tree (RRT) algorithms based on vertex sampling has the advantages of not only being used for path planning of complete systems, but also for adding constraint conditions for autonomous path finding of non-holonomic systems, which provides a solution for path planning in high-dimensional and complex environments.

Research of RRT algorithm in the field of path planning has made remarkable progress, but the algorithm still has shortcomings such as large amount of calculation, unstable path, low vertex utilization, and difficulty in obtaining a path that meets the conditions. In response to these lacks, the relevant researches have been carried out, including optimized vertex sampling (Kuffner, & LaValle, 2000), sampling-heuristics (Urmson, & Simmons, 2003), optimization of step size (Liu, Feng, Li, Hu, & Zhang, 2020), multiple trees method (Qureshi, & Ayaz,

* Corresponding author at: Beijing Institute of Precision Mechatronics and Controls, Donggaodi, Fengtai District, Beijing 100076, China.
E-mail addresses: 2277393124@qq.com (L. Jia), huangyp@2008.sina.com (Y. Huang), 1539449013@qq.com (T. Chen), guoyaxing08@163.com (Y. Guo), yycheng_work@126.com (Y. Yin), chenjing@bjtu.edu.cn (J. Chen).

https://doi.org/10.1016/j.eswa.2021.116379
Received 30 January 2021; Received in revised form 30 November 2021; Accepted 4 December 2021
Available online 29 December 2021

2015; Zaid, Ahmed, Yasar, & Raheel, 2018), by combination with some intelligent algorithms (Cao, Zou, Jia, Chen, & Zeng, 2019) and other aspects. The DL-RRT* algorithm is proposed by combining RRT* algorithm and D* Lite algorithm (Chao, Liu, Xia, Peng, & Ayodeji, 2019) that can continuously optimize the existing path by sampling the search graph obtained during the grid searching process and improve the efficiency of path planning in the high frequently changing environment. (Wang, Li, & Meng, 2021) proposed a fast bi-directional path planning algorithm that can efficiently prune random trees: Kinematic constrained bi-directional rapidly-exploring random tree (KB-RRT*), which efficiently pry random trees by analyzing kinematic constraints. However, this algorithm is not general and does not analyze how to determine the sampling area in advance. The algorithm in this paper solves these two problems. The Q-RRT* algorithm that can converge to the optimal solution quickly is proposed, which enlarges the set of possible parent vertices by considering not only a set of vertices contained in a hypersphere, as in RRT*, but also their ancestry up to a user-defined parameter, thus, resulting in paths with less cost than those of RRT* (Jeong, Lee, & Kim, 2019). The PQ-RRT* (Li, Wei, Gao, Wang, & Fan, 2020) is proposed by adopting the advantages of P-RRT* (potential functions based RRT*) and Quick-RRT*, which not only produces a better initial solution, but also guarantees a rapid convergence to the optimal solution. Although the RRT algorithm has been improved in many directions, and many articles (Jeong, Lee, & Kim, 2019; Li, Wei, Gao, Wang, & Fan, 2020; Noé, Fernando, & Luis, 2018; etc.) have been mentioned in the future work on the issue of kinematic constraints, and they have not yet proposed corresponding measures to solve this issue.

However, some scholars have also carried out research on the kinematic limitation of robots or manipulators during the movement. A novel method based on a modified particle swarm optimization (MPSO) combined with the η^3-splines is proposed, which considers the planning problem of smooth global path for mobile robots with kinematic constraints (Song, Wang, Zou, Xu, & Fuad, 2019). And the maximum curvature or curvature derivative along the path should be bounded for the sake of the kinematic constraints of robots. However, this algorithm has a large amount of calculation and a long running time for a three-dimensional or complex environment, and it is difficult to plan a path with the deflection angle of a joint within a certain range. A fast online predictive approach to resolve the task-priority differential inverse kinematics of redundant manipulators under kinematic constraints is presented and the maximum deflection angle and maximum angular velocity of the joint are defined in the MATLAB (Marco, Manuel, Nicola, & Antonio, 2019). But as the number of moves increases, the calculation time required by the prediction algorithm increases significantly and it is sometimes difficult to obtain the optimal path. Whether the boundary of kinematics constraints is reached can be estimated by obtaining the state of the manipulator during its motion (Valerio, Naresh, Michael, Jeffrey, & Rustam, 2018). As the degree of freedom of the manipulator increases, some researchers propose a recursive algorithm to derive the kinematics and dynamics formulas. (Korayem, & Dehkordi, 2018) deduced the motion equations of a viscoelastic linkage and a mobile manipulator through recursive Gibbs-Appell formulation. (Shafei, & Shafei, 2018) developed an automatic method based on the combination of 3×3 rotation matrix and 4×4 transformation matrix. Based on recursive Gibbs-Appell formulation and Newton's collision law, the governing equations of the robot system in flight and collision stage are derived respectively. (Korayem, & Shafei, 2015) considering the interaction between the manipulator and the mobile platform, some coupling effects, and nonholonomic constraints, the Gibbs-Appell formulation is used to push the motion equation of the snake manipulator. The Gibbs-Appell formulation can be utilized to analyze the time-varying motion equation and the safe margins of motion. Although this method can realize real-time planning of the path, it requires real-time visual processing, which requires a large amount of calculation and a lengthy calculation time, and the finally obtained path also has considerable optimization space.

According to existing researches, algorithms that can plan a path by which a manipulator can pass through the narrow space under the condition of limited joint deflection angle is deficient. At present, some tools (such as Matlab, Robot Operating System) can be used to plan a path that meets the limitation of joint angle, but it is necessary to verify whether the constraints are met in real time during the planning process. Therefore, on the basis of the RRT algorithm, the improved algorithm that not only the randomness is retained, but also the maximum deflection angle of joint is considered, namely MDA + RRT algorithms, is proposed in this paper. The relationship between the deflection angle of the linear path and the joint deflection angle during the movement of the manipulator along the rectilinear path is analyzed. And the algorithm does not need to be checked and can directly plan a path that meets the limitation of the joint angle. If the manipulator moves along the path planned by the algorithm, it can pass through a narrow space to reach the target point, and it can also ensure that the joint angle of the manipulator is always within the limitation. In summary, the main contributions of this paper are described as following.

(1) A hyper-redundant manipulator with unequal length links is introduced, which greatly improves the flexibility of the backend of the manipulator.
(2) The relationship between the maximum deflection angle of the path under the different step lengths and the maximum deflection angle of the joint is deduced, which lays the groundwork for path planning in applying the MDA + RRT algorithms.
(3) The MDA + RRT algorithms are proposed that can ensure that the deflection angle of the joint is within the limitation under two-dimensional and three-dimensional environment. Compared with the RRT algorithm, the MDA + RRT algorithms does not increase the amount of calculation, and the feasibility and optimization of the planned path are preferable to the corresponding RRT algorithms.
(4) The method of this paper can be associated with any kind of RRT algorithm. Besides, the algorithm can plan a path that satisfies the angle constraint for various moving objects and can be invoked as a universal method.

The remaining of this paper are outlined as following. Section 2 introduces the features and models of the manipulator and analyzes the relationship between the maximum deflection angle of the path and the maximum deflection angle of the joint under different step lengths. Section 3 introduces the principles of the three RRT algorithms. The principle of the MDA-QRRT* algorithm in the MDA + RRT algorithms is shown in Section 4. Section 5 explains the obstacle models and the simulation results of six algorithms in three obstacles. Section 6 summarizes the contributions and discusses future prospects.

2. Modeling and theoretical analysis

2.1. Mechanical model

In this paper, a cable-driven hyper-redundant manipulator is proposed with 17 degrees of freedom that can perform detection, maintenance and rescue in narrow and complex environments, being composed of three parts: a manipulator, driving mechanism and propulsion platform. The manipulator is composed of eight rigid links of unequal length in series, where $L_1 = L_2 = L_3 = L_4 = L_5 = 486$ mm, $L_6 = L_7 = 206$ mm, $L_8 = 284$ mm, and the total length of the manipulator is 3126 mm. And two adjacent links are connected by a universal joint. Due to the limitation of the mechanical structure, the maximum deflection angle max (ψ_i) between two adjacent links cannot exceed 40°. The driving mechanism is composed of motors, couplings, lead screws, sliders, and driving cables. The lead screw is driven by the motor to control the front and back movement of the slider, thereby pulling the driving cables to control motion of the link. Each link contains two degrees of freedom,

namely the yaw angle α_i and the pitch angle θ_i relative to the previous link. By controlling the relative rotation of two adjacent links, the link can be adjusted to the specified spatial direction, thus the posture of the manipulator can be changed. Besides, the eighth link at the end is connected to the application tools such as a stereo camera and a gripper. And the length is adjustable that is fixed as $L_8 = 284$ mm in this paper.

The established D-H coordinate system is shown in Fig. 1, which includes the world coordinate system $\{O_{XYZ}\}$ fixed relative to the ground, the coordinate system $\{O_0\}$ of the base, and the coordinate system of the eight links $\{O_1\}$-$\{O_8\}$. The coordinate systems can be converted through the conversion matrix that is related to L_i, α_i, and θ_i. The driving cable passes through the hole that is fixed relative to the body coordinate system $\{O_i\}$ to drive the manipulator. Therefore, the deflection angle ψ_i of the link can be obtained according to the length of the driving cable.

The flexibility of the last three joints of the manipulator in this paper has been substantially improved, and the minimum turning diameter is only 602 mm. As shown in Fig. 2, when the manipulator enters a narrow space after perforation, due to the restriction of obstacles, there is a certain range that the end point of the manipulator cannot access. The unreachable range of the manipulator in this paper is significantly less than that of the traditional manipulator, which also confirms the flexibility of the end joint of the proposed manipulator.

2.2. Theory analysis

In some path planning algorithms, paths can be planned by connecting discrete points in a straight line. When the step length is l_w, the maximum angle between two adjacent straight lines is Ψ_{max}, it's necessary to ensure that when the link length L_i moves along this path, the deflection angle ψ_i between two adjacent links always does not exceed the limit range ψ_{max}.

In this part, on the premise that L_i and ψ_{max} are known, if the manipulator moves along the path as shown in Fig. 3, the relationship between the path step length l_w and the maximum angle Ψ_{max} between two adjacent straight lines should be analyzed, which provides theoretical support for subsequent algorithm design.

In one case, the relationship between these parameters should be analyzed when $l_w = L_i = L_{i\text{-}1}$ ($w = 1, 2, 3\ldots$). As shown in Fig. 3, the axes of two links and the three straight lines on the path form two triangles: ΔABC and ΔCDE. According to the theorem of sine of triangles and angle transformation, the mathematical model of ψ_i can be obtained, as shown in Eq. (1). And then we can derive the maximum value of ψ_i, as shown in Eq. (2).

$$\begin{cases} max\,\psi_i = 2\Psi_{max} - \beta_2 - \beta_1 \\ s.t. \dfrac{a}{sin(\beta_1)} = \dfrac{L_{i-1}}{sin(\Psi_{max})} \\ \dfrac{b}{sin(\beta_2)} = \dfrac{L_i}{sin(\Psi_{max})} \\ a + b = l_w \\ 0 \leqslant \overline{AB} \leqslant l_w \\ l_w = L_i = L_{i-1} \end{cases} \tag{1}$$

$$max(\psi_i) = 2\left[\Psi_{max} - arcsin\left(\frac{sin(\Psi_{max})}{2}\right)\right] \tag{2}$$

In other cases, when $l_w = k \cdot L_i = k \cdot L_{i\text{-}1}$ ($k \in (0,1]$, $w = 1, 2, 3\ldots$), the relationship between these parameters should be analyzed. Let the starting point of the $(i\text{-}1)^{th}$ link move along the w^{th} section of the linear path, changes in the movement process and the deflection angle at the i^{th} joint are shown in Fig. 4.

There are two positions where the maximal value of ψ_i appears. The first position, when the end point of the $(i\text{-}1)^{th}$ link, that is, the initial point of the i^{th} link moves to the start or end point of a straight path ($a = 0$ or $b = 0$), the value of ψ_{i_end} at this point is deduced as shown in Eq. (3). The second position, when the initial point of the i^{th} link moves to the midpoint of a straight path ($a = b$), the value of ψ_{i_mid} at this point is shown in Eq. (4). After calculating these two maximal values, the maximum deflection angle of the joint can be obtained by max (ψ_{i_end},ψ_{i_mid}).

$$\psi_{i\_end} = (2N_{end} + 1)\Psi_{max} - 2arcsin\left[\frac{l_w}{L_i} sin\left(\frac{N_{end} + 1}{2}\Psi_{max}\right) sin\left(\frac{N_{end}\Psi_{max}}{2}\right) \Big/ sin\left(\frac{\Psi_{max}}{2}\right)\right] \tag{3}$$

$$\psi_{i\_mid} = 2N_{mid}\Psi_{max} - 2arcsin\left[\frac{l_w}{L_i} sin^2\left(\frac{N_{mid}\Psi_{max}}{2}\right) \Big/ tan\left(\frac{\Psi_{max}}{2}\right)\right] \tag{4}$$

N_{end} denotes the amount of turning points between two end points of the link in the path when the end point of the link moves to the end of a straight path, as $N_{end} = 1$ in the first picture in Fig. 4. And N_{mid} shows the amount of turning points between two end points of the link in the path when the end point of the link moves to the midpoint of a straight path, as $N_{mid} = 2$ in the second picture of Fig. 4. When l_w is 300 mm, Ψ_{max} is 22.40°, L_i is 486 mm and $L_{i\text{-}1}$ is 486 mm, we can get that the value of

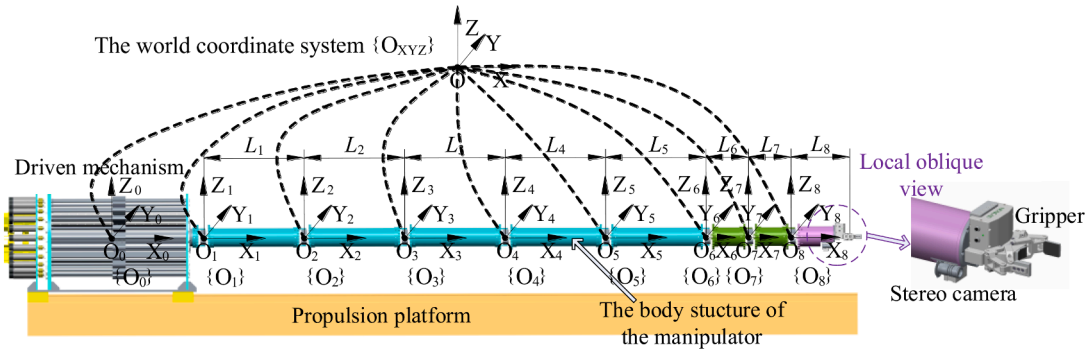
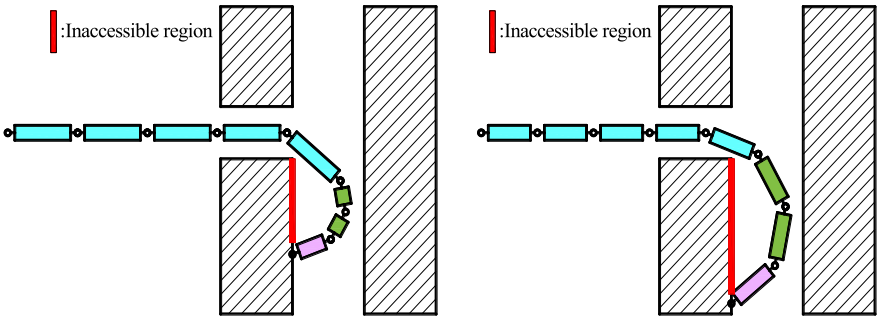
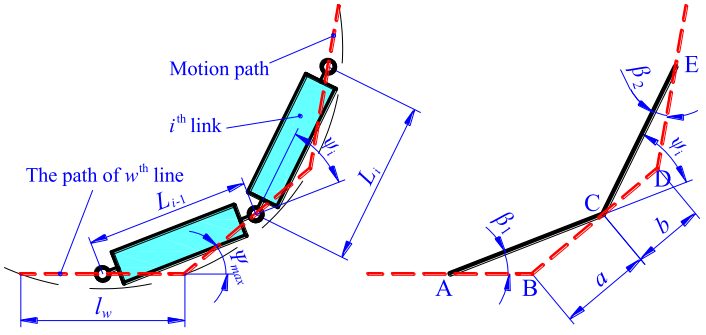


Fig. 1. Mechanical structure of the hyper-redundant manipulator.



(a) The proposed manipulator (b) The traditional manipulator

Fig. 2. The contrast of the inaccessible range.



(a) The manipulator is moving to a certain position (b) Trigonometric equivalent graph

Fig. 3. The position diagram and equivalent diagram of the manipulator.

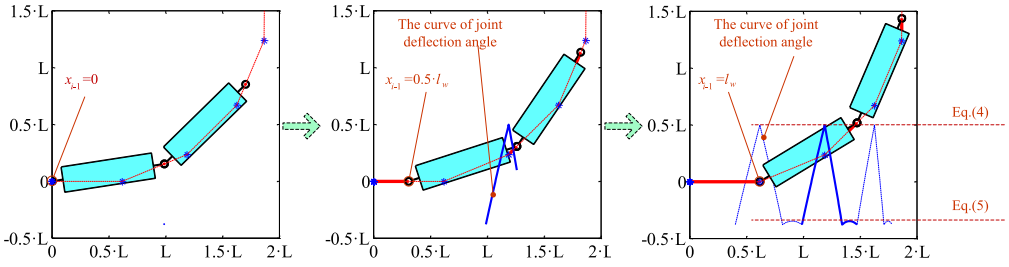


Fig. 4. Schematic diagram of the movement process of the manipulator in the straight path. ($l_w = 300$).

ψ_{i_end} is 39.99° and the value of ψ_{i_mid} is 35.77° through Eq. (3) and Eq. (4). The above description expresses that when the manipulator with the link length of 486 mm moves along the path with a step length of 300 mm where the maximum included angle between two adjacent straight lines is 22.40°, the deflection angle of the joint ψ_i does not exceed the limitation of 40° always.

According to the above formulas, we can determine the value of Ψ_{max} in the case of $l_w = k \cdot L_i = k \cdot L_{i-1}$ and $\psi_{max} = 40°$, as shown in Table 1. It can be observed in the table that the smaller the value of k, the smaller the value of Ψ_{max}, which means that the smaller the value of l_w, the smaller the value of Ψ_{max}, and two values keeps a non-linear relationship.

In this part, the relationship of four parameters including L_i, ψ_{max}, l_w and Ψ_{max} is analyzed considering the limitation of the deflection angle of the joint in the structure. Thus, the limitation of joint angle is transformed into the limitation of path angle, which provides the theoretical basis for the improved algorithms.

Table 1
Values of Ψ_{max} in different step lengths ($k = 0.1, 0.2 \ldots 1$).

k	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
N_{end}	10	5	3	2	2	1	1	1	1	0
N_{mid}	10	5	3	3	2	2	1	1	1	1
Ψ_{max}	3.91	7.82	11.53	15.12	19.43	21.99	24.65	28.08	32.77	37.88

3. Related work

3.1. Traditional RRT

After defining $X \subseteq \mathbb{R}^n$ as the bounded workspace in path planning, where n represents the dimensions of the workspace, path planning in two-dimensional ($n = 2$) and three-dimensional ($n = 3$) space is analyzed in this paper. Space X can be divided into obstacle areas (X_{obs}) and obstacle-free areas (X_{free}). Given the starting point $x_{init} \in X_{free}$ and the target point $x_{goal} \in X_{free}$ of the path, the planned path is represented by the continuous function σ:[0,1]. If $\sigma(0) = x_{init}$ and $\sigma(1) = x_{goal}$, and $\sigma(\tau) \in X_{free}$ for all $\tau \in [0,1]$, a feasible collision-free path is generated.

The traditional RRT algorithm adopts the starting point x_{init} of the path as the root vertices of the random tree T. The vertex x_i in the tree is stored in the vertices set V and the connection between the vertices is stored in the edges set E. All datas in V and E are located in X_{free}. In the traditional RRT algorithm, random vertices are selected from X_{free} to guide the growth direction of the tree. When the vertices in the tree spread to the target point, the search ends and the path connection from the target to the starting point is returned. The pseudocode for the Traditional RRT algorithm is shown in Algorithm 1.

Algorithm 1 Traditional RRT

```
1:V ← {x_init}; E ← ∅ ; T ← (V,E);
2:for i = 1 to Number do
3:   x_rand ← Random(i);
4:   x_near ← Nearest(V,x_rand);
5:   x_new ← NewVertex(x_near,x_rand);
6:   σ ← Steer(x_near,x_new);
7:   if CollisionFree(σ) then
8:      if Mindistance(x_new,V) > threshold then
9:         T=(V,E) ← Connect(x_new,x_near,T=(V,E));
10:     end if
11:  end if
12:end for
13:return T
```

The traditional RRT algorithm first initializes the random tree T (Line 1), and enters the loop when the number of operations has not reached the maximum number of iterations *Number* (Line 2). Then a random probability $p \in (0,1)$ is created in the Random function. If $p < pN$ ($pN = 90\%$ in this paper), a random value x_{rand} is selected within the map range, otherwise x_{rand} is equal to x_{goal} (Line 3). The vertex nearest to x_{rand} (x_{near}) is selected in the set V by using the Euclidean distance in the Nearest function (Line 4). The NewVertex function is illustrated in Eq. (5). The direction is determined by x_{near} and x_{rand}, and the step size δ determines the distance between x_{new} and x_{near} (Line 5). Given two configurations x_{near}, x_{new}, the Steer function returns the path σ, the path σ is the path between the two points x_{near} and x_{new} (Line 6). Then the CollisionFree function is adopted to analyze whether there are obstacles along the line (that is, path σ) between x_{new} and x_{near} is analyzed. If there is an obstacle, reelect x_{rand}, if not the algorithm continues. The CollisionFree function is applied to analyze whether there is a collision in the path σ, which can also be understood as judging whether various discrete points in the path are inside obstacles. Besides, the distance between adjacent discrete points used for the judgment is set to 0.1 mm (Line 7). The value of the threshold of all the algorithms in this paper is set to 20. If the closest distance between x_{new} and all vertices in V is greater than the set threshold (Line 8), the new vertices and edges generated are added to the random tree T (Line 9). And if the value is less than threshold, to reelect x_{rand}. When the Euclidean distance between x_{new} and x_{goal} is less than or equal to the step size δ, the loop terminates and the random tree T are output. If the maximum number of iterations *Number* is reached, the algorithm terminates and exits. After returning the random tree T, a feasible path from x_{init} to x_{goal} without collision from the random tree T is obtained.

$$x_{new} = x_{near} + \delta \frac{x_{rand} - x_{near}}{|x_{rand} - x_{near}|} \tag{5}$$

The input of traditional RRT is comprised of an initial point, target point and environment/map, and the result of the output is a graph containing a feasible path. Besides, the traditional RRT algorithm is to sample randomly in the entire space, therefore, it possesses good robustness in high-dimensional space. However, the algorithm still has disadvantages such as low vertex utilization, slow convergence speed, and an unstable path.

3.2. RRt

In order to surmount the shortcomings of traditional RRT, Karaman (Karaman, & Frazzoli, 2011) proposed the RRT* algorithm to ensure that the vertices of the random tree can converge to the optimal value. The principle of the RRT* algorithm is mostly the same as that of the traditional RRT algorithm. In the RRT* algorithm, after determining x_{new}, by querying the vertices within the radius r or k nearest neighbors, the neighbor vertex set $X_{neighbor}$ is obtained (Line 9). Then the new parent vertex x_{par} is found that minimizes the cost of x_{new} in $X_{neighbor}$, the connection between x_{new} and the original parent vertex x_{near} is deleted, and x_{new} and x_{par} are connected. If the weight of a vertex in the neighborhood is greater than the sum of the weight of the current x_{new} and the distance to the vertex, the current x_{new} vertex is defined as the parent of this vertex to ensure that the weight of the vertex in the neighborhood is always the best at present. After that, x_{new} is searched repeatedly and the joints between x_{new} and nearby vertices is trimmed repeatedly until reaching the target point (Line 10–12). The pseudocode of RRT* is presented in Algorithm 2.

Algorithm 2 RRT*

```
1:V ← {x_init}; E ← ∅ ; T ← (V,E);
2:for i = 1 to Number do
3:   x_rand ← Random(i);
4:   x_near ← Nearest(V,x_rand);
5:   x_new ← NewVertex(x_near,x_rand);
6:   σ ← Steer(x_near,x_new);
7:   if CollisionFree(σ) then
8:      if Mindistance(x_new,V) > threshold then
9:         X_neighbor ← Near(x_new,V,r or k);
10:           (x_par,σ_par) ← ChooseParent(x_new,x_near,X_neighbor,σ);
11:           T=(V,E) ← Reconnect(x_new,x_par,σ_par,T=(V,E));
12:           T ← Rewire(x_new,T,X_neighbor);
13:     end if
14:  end if
15: end for
16: return T
```

3.3. Q-RRt

In RRT* algorithm, the parameter r or k needs to be increased to improve the convergence speed, but it will also double the number of nearby vertices and the amount of calculation. The Q-RRT* algorithm not only considers the neighbor vertex set $X_{neighbor}$ of x_{new}, but also considers the ancestors of these vertices to share parent vertices (Line 9–10). The algorithm also checks whether the vertices near x_{new} and their parent vertices can be used to create a lower cost path (Line 11–13) to obtain an improved solution. In this algorithm, although the search range is expanded, it does not significantly aggrandize the calculation time, thereby augment the convergence speed. The pseudocode of Q-RRT* is submitted in Algorithm 3.

Algorithm 3 Q-RRT*

```
1:V ← {x_init}; E ← ∅ ; T ← (V,E);
2:for i = 1 to Number do
3:   x_rand ← Random(i);
4:   x_near ← Nearest(V,x_rand);
5:   x_new ← NewVertex(x_near,x_rand);
```

(continued on next page)

(*continued*)

Algorithm 3 Q-RRT*

6: $\sigma \leftarrow$ Steer(x_{near},x_{new});
7: **if** CollisionFree(σ) **then**
8: **if** Mindistance(x_{new},V) > threshold **then**
9: $X_{neighbor} \leftarrow$ Near(x_{new},V,r or k);
10: $X_{parent} \leftarrow$ Ancestry($X_{neighbor}$,V);
11: (x_{par},σ_{par}) $\leftarrow$ ChooseParent(x_{new},x_{near},$X_{neighbor} \cup X_{parent}$,$\sigma$);
12: T=(V,E) $\leftarrow$ Reconnect(x_{new},x_{par},σ_{par},T=(V,E));
13: $T \leftarrow$ Rewire(x_{new},T,$X_{neighbor} \cup X_{parent}$);
14: **end if**
15: **end if**
16:**end for**
17:**return** T

4. MDA-QRRt*

In Section 2, when a manipulator with a link of length L_i moves on straight lines path with a step length of l_w, how to set the maximum angle Ψ_{max} between two adjacent straight lines to ensure that the deflection angle ψ_i between the two links can never exceed ψ_{max} is introduced. In Section 3, three RRT algorithms are introduced: traditional RRT, RRT*, and Q-RRT*. On the basis of these two sections, this section takes the MDA-QRRT* algorithm improved by the Q-RRT* algorithm as an example to introduce how to plan a movement route that complies with the structural requirements of the manipulator. Pseudocode describing MDA-QRRT* is illustrated in Algorithm 4.

Algorithm 4 MDA-QRRT*

1:$V \leftarrow \{x_{init}\}$; $E \leftarrow \varnothing$; $T \leftarrow (V,E)$;
2:**for** $i = 1$ to *Number* **do**
3: $x_{rand} \leftarrow$ Random(i);
4: $x_{near} \leftarrow$ Nearest(V,x_{rand});
5: (Ψ_{max},Ψ_i) $\leftarrow$ Angle-MDA(x_{rand},x_{near},T);
6: $x_{new} \leftarrow$ NewVertex-MDA(x_{near},Ψ_i);
7: $\sigma \leftarrow$ Steer(x_{near},x_{new});
8: **if** CollisionFree(σ) **then**
9: **if** Mindistance(x_{new},V) > threshold **then**
10: $X_{neighbor} \leftarrow$ Near(x_{new},V,r or k);
11: $X_{parent} \leftarrow$ Ancestry($X_{neighbor}$,V);
12: $X_{in} = X_{neighbor} \cup X_{parent}$;
13: (x_{par},σ_{par}) $\leftarrow$ ChooseParent-MDA(x_{new},x_{near},X_{in},σ,Ψ_{max})
14: T=(V,E) $\leftarrow$ Reconnect(x_{new},x_{par},σ_{par},T=(V,E));
15: $T \leftarrow$ Rewire-MDA(x_{new},T,$X_{neighbor}$,Ψ_{max});
16: **end if**
17: **end if**
18:**end for**
17:**return** T

The MDA-QRRT* algorithm adopts the starting point of the path x_{init} as the root vertex of the random tree T. The vertex x_i in the tree is stored by the set V, which has n columns in the n-dimensional space. Each row in V corresponds to the position data of a vertex and the connection between vertices is stored in the edge set E where the first column indicates the vertex number of the parent of vertex x_i, the second column is the angle between the line segment formed by vertex x_i and its parent and the horizontal line, and the third column is the cumulative Euclidean distance from the vertex x_i to its parent, then to the parent of the parent until x_{init}. Always guarantee that all data in V and E are in X_{free} (Line 1). A x_{rand} is selected by the Random function and a vertex x_{near} closest to x_{rand} is selected. And then the selection range of x_{new} according to x_{rand}, x_{near} and T is analyzed in *Angle-MDA*, which is described in Algorithm 5 as shown in Section 4.1. After determining the direction of x_{new}, namely Ψ_i, the position of x_{new} can be calculated according to the NewVertex-MDA function (Eq. (6)) (Line 3–6). If there is no obstacle between x_{new} and x_{near}, and the distances between x_{new} and all vertices are greater than threshold, the vertices near x_{new} are stored in $X_{neighbor}$, and the ancestors of these nearby vertices are stored in X_{parent} to get a X_{in} set disposal by x_{new} (Line 7–12). Then the new parent vertex x_{par} that minimizes the cost of x_{new} and meets the deflection angle requirements are found in X_{in} by using the ChooseParent-MDA function as shown in Section 4.2. And the connection between x_{new} and the original parent vertex x_{near} being deleted. The connection with x_{new} and x_{par} is obtained and is trimmed in T (Line 13–14). If the weight of a vertex x_{from} in $X_{neighbor}$ is greater than the sum of the weight of the current x_{new} and the distance between x_{new} and x_{from}, and the deflection angle requirement is met simultaneously, the current vertex x_{new} is used as the parent of x_{from} as shown in Section 4.1(Line 15). After that, x_{new} is searched repeatedly and the connection relationship between x_{new} and nearby vertices is trimmed repeatedly until reaching the target point.

$$x_{new} = x_{near} + \delta \cdot \Psi_i \tag{6}$$

4.1. Angle-MDA

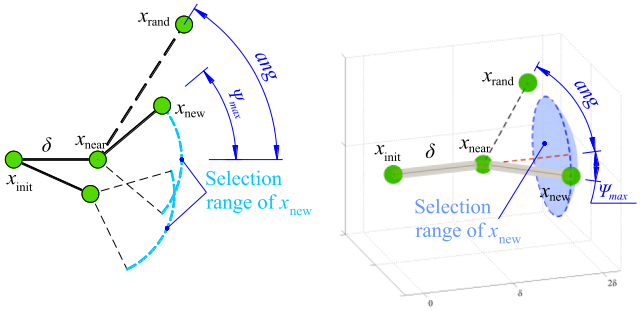
The selection range of x_{new} is analyzed by Angle-MDA function according to values of x_{rand}, x_{near} and T to determine the specific location of the next discrete point x_{new}. In this function, the angle ang_{rand} of the line segment connected by x_{rand} and x_{near} relative to the horizontal line is solved (Line 1), and the angle ang_{near} of the line segment connected by x_{near} and its parent relative to the horizontal line is found in T (Line 2). The purpose of this algorithm is to ensure that the deflection angle of the joint is always within a limited range. Nonetheless, the length of the manipulator is not constant, it is first necessary to analyze the value of Ψ_{max} according to the distance d_{goal} between the vertex x_{near} and the target point x_{goal}. Since the lengths of the links selected in this paper are unequal, the value of L_i is regarded as 486 mm in the foregoing path planning. When the distance between discrete points on the path and the target is less than the sum of L_7 and L_8, the value of L_i is regarded as 206 mm. According to the theoretical derivation in Section 2, if d_{goal} is less than the sum of L_7 and L_8, Ψ_{max} is equal to 40° and if d_{goal} is equal or greater than the sum of L_7 and L_8, then Ψ_{max} is equal to 22.40° (Line 4–8). After determining the maximum deflection angle Ψ_{max} between two adjacent straight lines in the path, whether the value of x_{new} exceeds the selectable range is analyzed. If x_{near} represents the starting point x_{init}, the direction of x_{new} is randomly selected between −60° and 60°. If the value of Abs(ang_{rand}-ang_{near}) that is the angle between two adjacent straight lines is lower than or equal to Ψ_{max}, the direction of x_{new} shows that Ψ_i is equal to ang_{rand}. If the value of Abs(ang_{rand}-ang_{near}) that is the angle between two adjacent straight lines is greater than Ψ_{max}, then Ψ_i expresses the boundary value in the selectable range. As shown in Fig. 5, in the two-dimensional situation, when the equation ($\Psi_i = ang_{near} + $ Sign (ang_{rand}-ang_{near})·Ψ_{max}) is set up, the selected point of x_{new} is unique. In the three-dimensional situation, when the angle between two adjacent lines is Ψ_{max}, the selection space of x_{new} is reduced from a space curved surface to a circle (Line 9–14).

Algorithm 5 Angle-MDA(x_{rand},x_{near},T)

1:ang_{rand} = NatureAngle(x_{rand},x_{near});
2:ang_{near} = T(near,n + 2);
3:d_{goal} = Distance(x_{near},x_{goal});
4:**if** $d_{goal} < L_7 + L_8$ **then**
5: Ψ_{max} = LineAngle(L_8);
6:**else**
7: Ψ_{max} = LineAngle(L_1);
8:**end if**
9$\Psi_i = ang_{rand}$;
10:**if** k_{near}==1 **then**
11: Ψ_i = Rand(-π/3,π/3);
12:**else if** Abs(ang_{rand}-ang_{near}) > Ψ_{max}
13: Ψ_i = MaxDirection(ang_{rand},ang_{near},Ψ_{max},n);
14:**end if**
15:**return** (Ψ_{max},Ψ_i);

4.2. ChooseParent-MDA

The function of ChooseParent-MDA is shown in Algorithm 6. After initialization the minimum parent vertex x_{min}, minimum path σ_{min}, and



(a) Two dimensions situation(n=2) (b) Three dimensions situation(n=3)

Fig. 5. Schematic diagram of the selection range of x_{new}.

minimum cost c_{min}, the vertex x_{in} in X_{in} is set as the new parent of x_{new} if the following three conditions are satisfied.

(1) If the vertex x_{in} is regarded as the new parent, the cost will be reduced.
(2) The path σ formed by x_{in} and x_{new} satisfies collision-free.
(3) The angle between the path from the parent of x_{in} to x_{in} and the path from x_{in} to x_{new} is not greater than Ψ_{max}.

The range of X_{in} not only includes the vertices near x_{new} but also the ancestors of these vertices, thus improving the optimization speed of the algorithm.

Algorithm 6 ChooseParent-MDA(x_{new},x_{near},X_{in},σ,Ψ_{max})

$x_{min} \leftarrow x_{near}$;
$\sigma_{min} \leftarrow \sigma$;
$c_{min} \leftarrow$ Cost(x_{min}) + Cost(σ_{min});
for all $x_{in} \in X_{in}$ **do**
  $\sigma \leftarrow$ Steer(x_{in},x_{new});
  $c \leftarrow$ Cost(x_{in}) + Cost(σ);
  ang_{new} = NatureAngle(x_{new},x_{in});
  $ang_{in} = T(in, n+2)$;
  ang = Abs(ang_{new}-ang_{in});
  if $c < c_{min}$ **and** $ang \le \Psi_{max}$ **then**
    if CollisionFree(σ) **then**
      $x_{min} \leftarrow x_{in}$;
      $\sigma_{min} \leftarrow \sigma$;
      $c_{min} \leftarrow c$;
    end if
  end if
end for
return (x_{min},σ_{min});

4.3. Rewire-MDA

In the MDA-QRRT* algorithm, it is not only necessary to trim the path related to x_{new}, but also to search for the better parent of all vertices in $X_{neighbor}$ in the Rewire-MDA function. In this function, whether the new paths formed by all vertices x_{nei} in $X_{neighbor}$ and x_{from} (x_{from} being the vertex belonged x_{new} or the ancestor of x_{new}) reducing the cost and meet the collision-free and restrictions of the angle is analyzed. And if it is satisfied, the parent of x_{nei} is updated to x_{from} to trim the path.

Algorithm 7 Rewire-MDA(x_{new},T,$X_{neighbor}$,Ψ_{max})

for all $x_{nei} \in X_{neighbor}$ **do**
for all $x_{from} \in \{x_{new}\} \cup$ ancestor(T,x_{new}) **do**
  $\sigma \leftarrow$ Steer(x_{nei},x_{from});
  $c_{ne} \leftarrow$ Cost(x_{nei});
  $c \leftarrow$ Cost(x_{from}) + Cost(σ);
  ang_{ne} = NatureAngle(x_{nei},x_{in});
  $ang_{from} = T(from, n+2)$;
  ang = Abs(ang_{nei}-ang_{from});
  if $c < c_{in}$ **and** $ang < \Psi_{max}$ **then**
    if CollisionFree(σ) **then**

(*continued on next column*)

(*continued*)

Algorithm 7 Rewire-MDA(x_{new},T,$X_{neighbor}$,Ψ_{max})

      T=(V,E) ← Reconnect(x_{nei},x_{from},σ,T=(V,E));
    end if
  end if
  end for
end for
return T;

5. Simulation

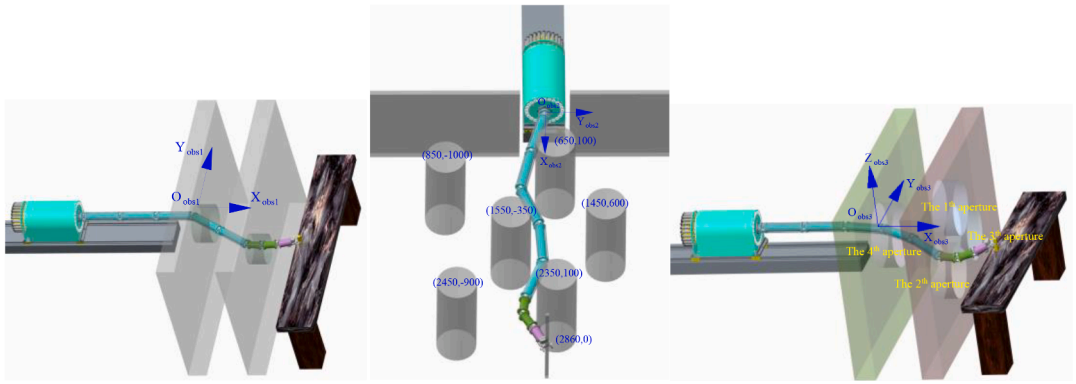
5.1. Three test environments

A hyper-redundant manipulator can be applied to detection operations in narrow space, enter into an environment that is difficult or inaccessible for human and bypass obstacles to accurately reach the desired position because of multiple degrees of freedom, which is suitable for work in confined and hazardous areas. In this section, six algorithms including traditional RRT, RRT*, Q-RRT*, MDA-RRT, MDA-RRT* and MDA-QRRT* in the three environments shown in Fig. 6 are compared. Whether these algorithms can plan the path where the eight-link manipulator can grab the target and retrieve it with not touching the obstacle and the deflection angle of the link is not exceeding the limit angle is analyzed. In addition to obstacles, the figure also contains the position and posture of the manipulator grasping the target along the planned path.

Since the axes of the two circular apertures in obs1 are on the same plane as the center of mass of the target, the three-dimensional path planning can be transformed into two-dimensional path planning. Similarly, the three-dimensional path planning in obs2 can also be converted to two-dimensional path planning. That is to say, in the obs1 and obs2, the passable space in the starting point, target point and obstacle can be analyzed at the same height, so only 2D path planning is required.

The sizes of three test environments are 2000 mm × 1665 mm, 3000 mm × 2500 mm, and 1700 mm × 2000 mm × 1400 mm, respectively. The size of each obstacle in obs1 can be interpreted from Fig. 7, and the size of each obstacle in obs2 can be read from Fig. 6(b). Besides, two walls and four circular apertures are contained in obs3. The axes of four circular apertures on the Y axis and Z axis are reflected as (-400 mm,400 mm), (-800 mm,-200 mm), (0 mm,-100 mm), (-250 mm,0mm) and the radius are 320 mm, 320 mm, 320 mm, 370 mm, respectively. And the positions and thickness of the two walls are identical to those of the two walls in obs1.

Taking obs1 as an example, how much the distance between the path and the obstacle should be maintained in order to ensure that the manipulator does not touch the obstacle is analyzed. The farthest distance between all points on the surface of each link and the axis of the link is 75 mm. Considering that only the two ends of the link move along the path when the link moves along a straight path, and control errors,



(a) Obs1 (Easy case) (b) Obs2 (Hard case) (c) Obs3 (3D case)

Fig. 6. Models of three obstacles.

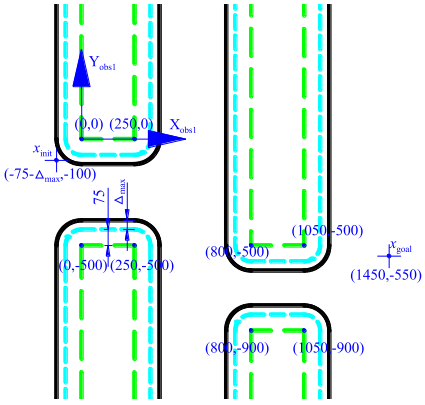
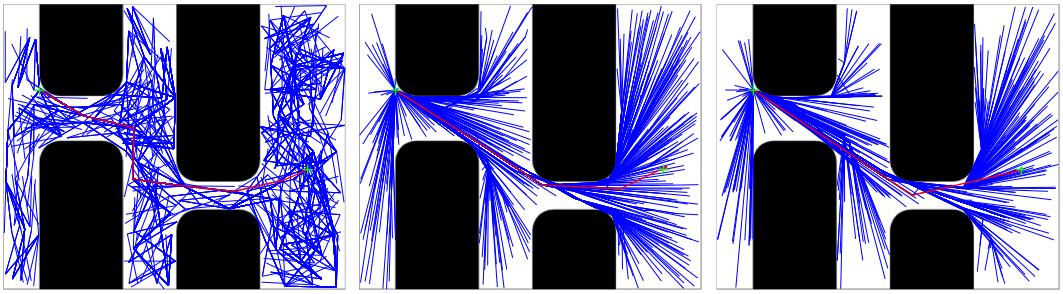


Fig. 7. Unreachable areas in the process of planning path.

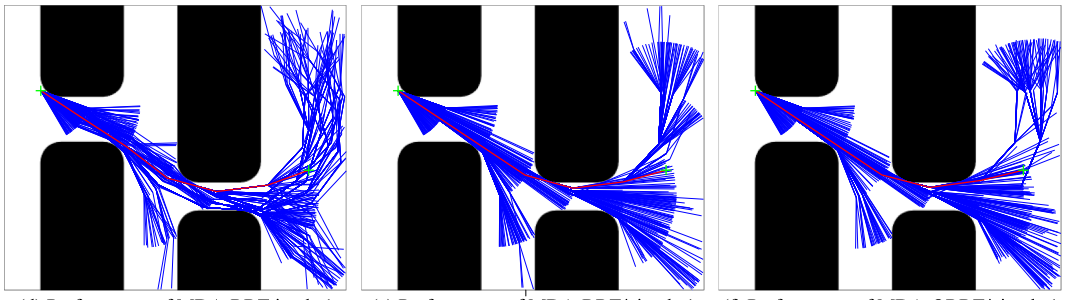
positioning errors and other errors are existing during the movement of the link, so there is a certain distance between the point on the axis of the link and the path (the maximum distance is considered Δ_{max}). Thus, the shortest distance between the planned path and the obstacle is expressed as $(75+\Delta_{max})$mm. The area where the distance to the obstacle is less than or equal to $(75+\Delta_{max})$mm is regarded as an unreachable area. That is, the area enclosed by the black frame shown in Fig. 7 is unreachable area. It can be judged whether the spatial point is reachable according to the closest distance between the spatial point and the obstacle. In this paper, for the convenience of observation, the obstacle model will be modified to the unreachable area model corresponding to the obstacle in the following path planning. The motion path of each joint center is the same/repeated as the motion path of the end-effector. Since the maximum distance $(75+\Delta_{max})$mm between each point on the surface of the manipulator and the path is taken into account, the body of the manipulator can be guaranteed to be collision free by constructing the unreachable region.



(a) Performance of RRT in obs1
t=3.148s, l=1949, max(ψ_i)>90°

(b) Performance of RRT* in obs1
t=12.110s, l=1810, max(ψ_i)=35°

(c) Performance of Q-RRT* in obs1
t=628.409s, l=1771, max(ψ_i)=34°

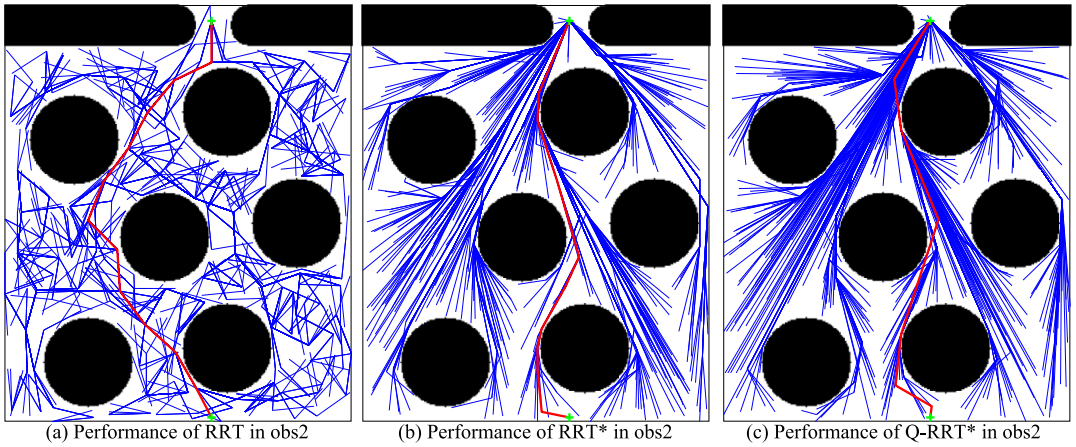


(d) Performance of MDA-RRT in obs1
t=11.460s, l=1756, max(ψ_i)=22°

(e) Performance of MDA-RRT* in obs1
t=16.812s, l=1739, max(ψ_i)=22°

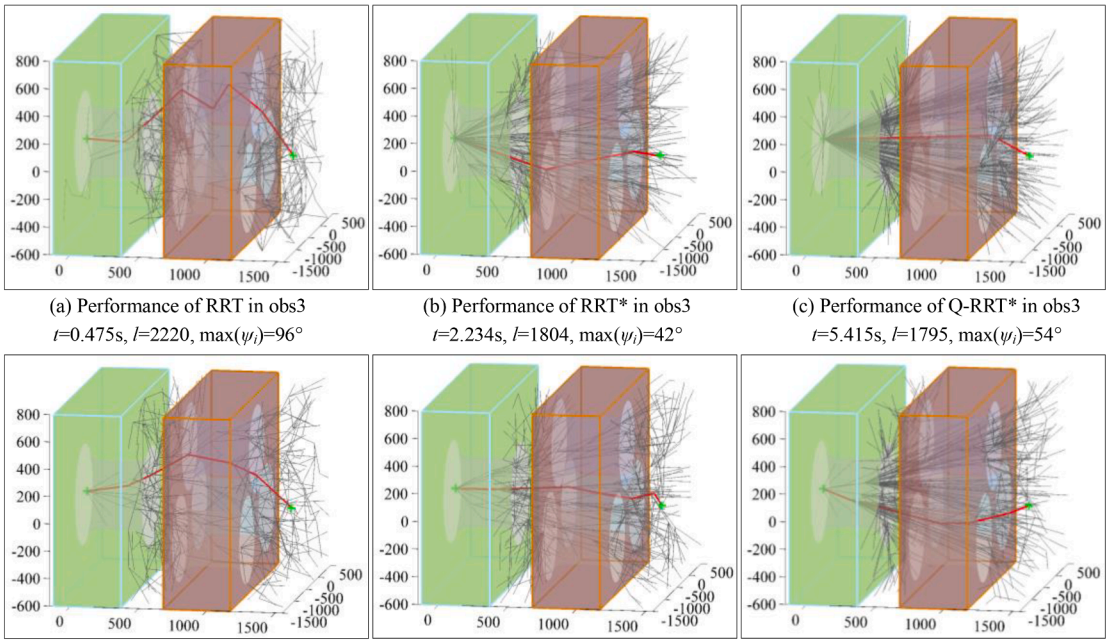
(f) Performance of MDA-QRRT* in obs1
t=569.689s, l=1736, max(ψ_i)=22°

Fig. 8. Generated path with 500 samples in obs1.



(a) Performance of RRT in obs2 t=0.454s, l=3526, max(ψ_i)>90°
(b) Performance of RRT* in obs2 t=2.283s, l=3185, max(ψ_i)=44°
(c) Performance of Q-RRT* in obs2 t=67.426s, l=3188, max(ψ_i)=66°
(d) Performance of MDA-RRT in obs2 t=0.961s, l=3450, max(ψ_i)=40°
(e) Performance of MDA-RRT* in obs2 t=2.403s, l=3219, max(ψ_i)=39°
(f) Performance of MDA-QRRT* in obs2 t=43.938s, l=3146, max(ψ_i)=40°

Fig. 9. Generated path with 500 samples in obs2.



(a) Performance of RRT in obs3 t=0.475s, l=2220, max(ψ_i)=96°
(b) Performance of RRT* in obs3 t=2.234s, l=1804, max(ψ_i)=42°
(c) Performance of Q-RRT* in obs3 t=5.415s, l=1795, max(ψ_i)=54°
(d) Performance of MDA-RRT in obs3 t=1.211s, l=1873, max(ψ_i)=40°
(e) Performance of MDA-RRT* in obs3 t=3.030s, l=1780, max(ψ_i)=39°
(f) Performance of MDA-QRRT* in obs3 t=6.130s, l=1778, max(ψ_i)=39°

Fig. 10. Generated path with 500 samples in obs3.

5.2. Simulation results under 500 samples

The simulation experiment is implemented in an Inter i5-6500 CPU with 4G of RAM. For the three obstacles, each algorithm is run once to generate 500 root vertices, and the random tree distribution diagrams shown in Fig. 8, Fig. 9, and Fig. 10 are obtained respectively. The step length δ in the algorithm is constant at 300 mm. The running time t of the algorithm, the total length l of the final selected path, and the maximum deflection angle $\max(\psi_i)$ of the joint when the manipulator moves along the path are all marked below the picture. The blue line in the Fig. 8 and Fig. 9 is a random tree that is expressed as the gray line in Fig. 10, the red line is the ultimate selected path, and the green cross axes express the initial point and the target point. As can be seen from the figures, although the time consumed by the traditional RRT algorithm is the shortest, the tree distribution map obtained by the traditional RRT algorithm is messy, and the final planned path length and the corresponding maximum deflection angle of the joint are larger than those planned by other algorithms. Because r is equal to 2.5δ in the algorithm, the difference between the random tree distribution diagrams obtained by the RRT* algorithm and the Q-RRT* algorithm is small. If these two algorithms are applied for planning the path in some easy obstacle models (such as obs1, obs3), after generating many root vertices, the path obtained may satisfy the requirements of the deflection angle of the joint. However, it is sometimes difficult to plan a path that meets the requirements of the deflection angle of the joint even if many root vertices are obtained, when the algorithm is applied to plan a path in a complex obstacle model (such as obs2). When there are more root vertices, the number of vertices in the neighbor vertex set X_{neighbor} will also increase correspondingly, which will increase the running time of the algorithm. Therefore, when the number of vertices is large, it is better to adopt the RRT* algorithm instead of the Q-RRT algorithm.

In the process of generating 500 root vertices, the running time of MDA-RRT* algorithm and MDA-QRRT* algorithm is close to that of RRT* algorithm and Q-RRT* algorithm respectively, which manifests that the improved algorithm does not increase the amount of calculation of the algorithm and only constrains the selection range of the next vertex. And the above constraint can ensure that the planned path meets the requirements of the deflection limit of the joint.

It can be seen from the simulation results that after 500 root vertices are generated. The three MDA + RRT algorithms all can plan a path that satisfies the requirements of the deflection angle limit of the joint in three obstacles. Due to randomization in the RRT algorithm, it is extremely important to quickly obtain a high-quality initial solution (initial vertex tree) in path planning related to the RRT algorithm. In this section, an optimal path based on 500 root vertices is selected, which cannot reflect the quality of the initial solution of these six algorithms, so the following will modify the termination condition from 500 samples to planning a path that can reach the target point. Then these six algorithms are all executed 50 times in each obstacle, and the time and quality required to plan the preliminary path using these six algorithms is analyzed.

5.3. Simulation results under initial solution

Based on the three obstacles, the six algorithms are implemented to plan an initial solution path from the initial point to the target point. The performance of the planned path is analyzed through three factors: the time t, the total length l of the path, and the maximum deflection angle $\max(\psi_i)$ of the joint when the manipulator moves along the path.

The statistical results of 50 simulation are shown in the boxplots. The three columns of Fig. 11 correspond to three obstacles. The first line is the time t/ms, and the second line is the total length l/mm of the planned path, and the third line is the maximum deflection angle $\max(\psi_i)/^\circ$ of the joint when the manipulator moves along the path. The six colors in the figure correspond to six algorithms. The boxplot consists of five numerical points: minimum (min), lower quartile (Q1), median (median), upper quartile (Q3), maximum (max). And the numerical points corresponding to the mean value are also added to the box chart. The color area in the figure reflects the degree of dispersion of the data in the middle 50%.

The corresponding running time t shows an increasing trend for traditional RRT, RRT* and Q-RRT*. Similarly, the corresponding running time t also shows an increasing trend for MDA-RRT, MDA-RRT* and MDA-QRRT*. Because three improved algorithms add constraint conditions, the running time t is longer than the corresponding RRT algorithm. And because there is randomness in random tree path planning, sometimes the planned path takes a very long time (for example, $t = 78517$ ms), so the mean of the running time of these six algorithms is greater than the median. Even in some cases, the mean is larger than the Q3. When planning a path in some easy obstacle models (such as obs1 and obs3), the running time t of the three RRT algorithms is significantly less than the running time t of the three MDA + RRT algorithms. But

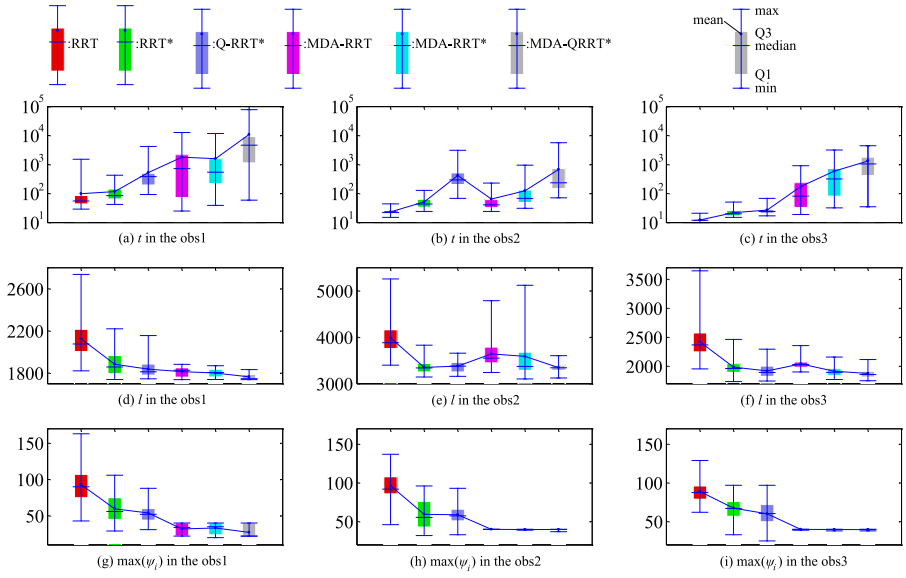


Fig. 11. Comparison of t, l and $\max(\psi_i)$ in every algorithm for planning 50 paths based on three obstacles.

when planning a path in a complex obstacle model (such as obs2), the running time t of the three MDA + RRT algorithms is approximately equal to the running time t of the corresponding RRT algorithms. If the number of vertex samples is the same, the simulation time is almost equal. Because the MDA + RRT algorithm restricts the selection range, the number of samples required to find a path that meets the constraints increases, which increases the running time.

The total path length l takes on a decreasing trend corresponding to the traditional RRT, RRT* and Q-RRT*. Similarly, the total path length l also shows a decreasing tendency corresponding to MDA-RRT, MDA-RRT* and MDA-QRRT*. Because the angle constraints are added to the three MDA + RRT algorithms, the total path length l is mostly smaller than the corresponding RRT algorithms. That is, three MDA + RRT algorithms greatly improve the optimization of the planned path compared to the three RRT algorithms. Moreover, the max, Q3 and mean of the total path length l planned by the MDA-QRRT* algorithm are the smallest among the six algorithms, which shows that MDA-QRRT* algorithm can not only plan a path that meets the angle constraints, but also make the path as short as possible.

The maximum deflection angle $\max(\psi_i)$ of the joint shows a decreasing trend for traditional RRT, RRT* and Q-RRT*, but the mean, median and Q1 of the $\max(\psi_i)$ are all greater than 40°. On the contrary, the $\max(\psi_i)$ corresponding to the paths planned by the three MDA + RRT algorithms are all less than or equal to 40°, which can meet the constraints of the angle limit of the joint. That is, three MDA + RRT algorithms greatly improve the feasibility of the planned path compared to the three RRT algorithms.

The feasibility and optimization of the six algorithms are analyzed according to data in Fig. 11 and Fig. 12. The path is regarded as a feasible path if meeting the following conditions.

(1) The value of $\max(\psi_i)$ is less than or equal to 40°.
(2) The value of l is less than or equal to 3126 mm.

The path is regarded as an optimal path if meeting the following conditions.

(1) The value of $\max(\psi_i)$ is less than or equal to 40°.
(2) The value of l is less than or equal to $1.05l_{\text{optimal}}$.

In this paper, obs$i$ represents the i^{th} obstacle among the three obstacles ($i = 1, 2, 3$). Besides, $n_{\text{fea\_obs}i}$ is the number to plan feasible paths in obs$i$, and $n_{\text{opt\_obs}i}$ is the number to plan optimal paths in obs$i$. $\eta_{\text{fea\_obs}i}$ is the feasible efficiency in obs$i$, which means the value of the number of feasible times divided by the total number of times, and $\eta_{\text{opt\_obs}i}$ is the optimization efficiency in obs$i$.

Table 2
The feasibility and optimization of six algorithms.

Algorithm	Traditional RRT	RRT*	Q-RRT*	MDA-RRT	MDA-RRT*	MDA-QRRT*
$n_{\text{fea\_obs1}}$	0	9	6	50	50	50
$n_{\text{opt\_obs1}}$	0	7	4	22	30	47
$\eta_{\text{fea\_obs1}}$	0%	18%	12%	100%	100%	100%
$\eta_{\text{opt\_obs1}}$	0%	14%	8%	44%	60%	94%
$n_{\text{fea\_obs2}}$	0	0	0	0	1	1
$n_{\text{opt\_obs2}}$	0	0	0	0	1	1
$\eta_{\text{fea\_obs2}}$	0%	0%	0%	0%	2%	2%
$\eta_{\text{opt\_obs2}}$	0%	0%	0%	0%	2%	2%
$n_{\text{fea\_obs3}}$	0	2	6	50	50	50
$n_{\text{opt\_obs3}}$	0	1	4	0	7	14
$\eta_{\text{fea\_obs3}}$	0%	4%	12%	100%	100%	100%
$\eta_{\text{opt\_obs3}}$	0%	2%	8%	0%	14%	28%

As can be seen from Table 2 and Fig. 12, the path planned by the traditional RRT is neither feasible nor optimized. When planning paths in some easy obstacle models (such as obs1, obs3), the paths planned by the three MDA + RRT algorithms all meet the feasibility. Among the six algorithms, the feasibility and optimization of the MDA-QRRT* algorithm are the best. The feasibility and optimization of the three MDA + RRT algorithms are also better than the corresponding RRT algorithms. Values of the feasible efficiency $\eta_{\text{fea\_obs}i}$ and the optimization efficiency $\eta_{\text{opt\_obs}i}$ of the six algorithms in the three obstacles can be seen from Fig. 12. And it can be seen that the $\eta_{\text{fea\_obs}i}$ and $\eta_{\text{opt\_obs}i}$ of the three MDA + RRT algorithms proposed in this paper are greater than the corresponding three existing RRT algorithms. Besides, in the three obstacles, the optimization of the MDA-QRRT* algorithm is greater than that of the MDA-RRT* algorithm, and the optimization of the MDA-RRT* algorithm is greater than that of the MDA-RRT algorithm.

5.4. Analysis of simulation time corresponding to the initial solution

As can be seen from the simulation results in Section 5.2, the improved algorithm does not increase the calculation amount of the algorithm. But only restricts the selection range of the next node. According to the simulation results of Section 5.3, it can be seen that while planning the initial path, the feasibility and optimization of the three MDA + RRT algorithms are also superior to the corresponding RRT algorithm. In this section, the average time $t_{\text{avg\_obs}i}$ corresponding to plan a path applying these six algorithms, and the average time $t_{\text{fea\_obs}i}$

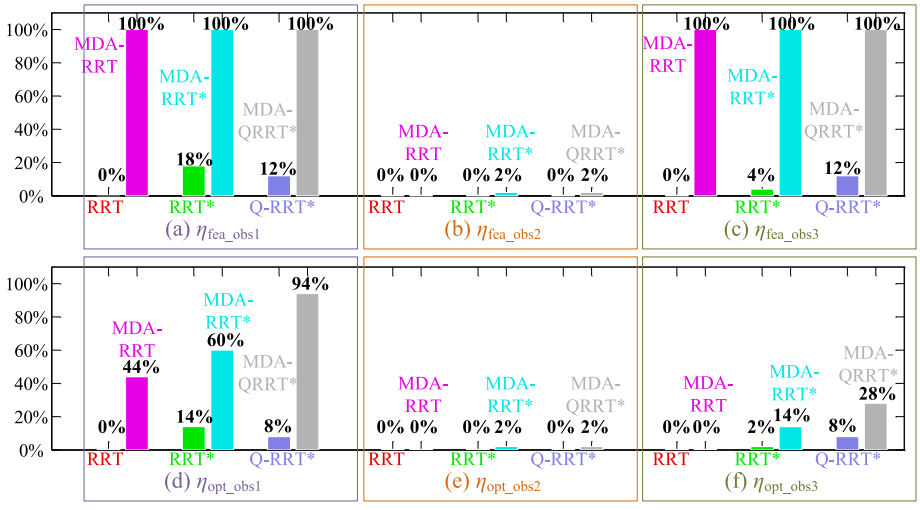


Fig. 12. Comparison diagram between the improved algorithms and traditional algorithms for $\eta_{\text{fea\_obs}i}$ and $\eta_{\text{opt\_obs}i}$.

corresponding to a feasible path applying these six algorithms, and the average time t_{opt_obsi} corresponding to an optimized path using these six algorithms are analyzed.

The path is directly planned by the traditional algorithm may not be feasible and it is also necessary to judge whether the planned path meets the conditions. Then, the geometric approach for follow-the-leader motion of manipulator is adopted to solve the inverse kinematics (Xie, Wang, Li, Hu, & Yang, 2019). It is not only necessary to calculate whether the angles of the corresponding eight joints exceed the limitation when the manipulator keeps in a static position, but also to calculate whether the angles of the eight joints meet the requirements at each position when the manipulator moves along the path. The above process will greatly increase the complexity of path planning, and increase the amount of calculation and calculation time. And it can only analyze whether the deflection angle of each joint exceeds the limitation when the manipulator moves to some discrete points on the path (rather than all points on the path).

In the improved algorithm proposed in this paper, the next random point within the range that meets the conditions is directly selected, so that the planned path must meet the requirements, and there is no need to judge whether the planned path meets the requirements of the joint deflection angle, which greatly improves the feasible efficiency and optimize efficiency. In other words, if the manipulator moves along the path planned by the improved algorithm, it can be satisfied that the deflection angle of each joint is within a limited range when the manipulator moves to any position.

The t_{fea_obsi} and t_{opt_obsi} corresponding to the path planned by the improved algorithm in this paper can be calculated according to the simulation results in Section 5.3, and the corresponding results are shown in Table 3. Because none of the paths planned by the traditional RRT algorithm meet the feasibility, there is no need to consider the t_{fea_obsi} and t_{opt_obsi} corresponding to the traditional RRT algorithm. The t_{fea_obsi} and t_{opt_obsi} corresponding to the path planned by the RRT* algorithm and the Q-RRT* algorithm need to be added to the time occupied by solving the inverse kinematics. The t_{avg_obsi} denotes the required time to plan a path applying six algorithms and determine whether the path meets the limitation of the joint deflection angle. Then the variables t_{fea_obsi}, t_{opt_obsi}, and t_{avg_obsi} satisfy the following formula

$$t_{fea\_obsi} = \frac{t_{avg\_obsi}}{\eta_{fea\_obsi}} \tag{7}$$

$$t_{opt\_obsi} = \frac{t_{avg\_obsi}}{\eta_{opt\_obsi}} \tag{8}$$

When the discrete point is selected at a distance of 0.1 mm in the path to solving the deflection angle of each joint, the value of t_{avg_obsi} in obs1 is 16.159 s and 16.362 s respectively and the value of t_{avg_obsi} in obs2 is 58.169 s and 58.794 s respectively with the value of t_{avg_obsi} in obs3 being 20.013 s and 20.036 s respectively by utilizing the RRT* algorithm and Q-RRT* algorithm. The above data are obtained by averaging 50 simulation samples and it can be seen that when a discrete point is selected every 0.1 mm, the calculation time for inverse kinematics will be very long. In order to reduce the calculation time of the traditional algorithm, a discrete point is selected every 1 mm in the path to solving the deflection angle of each joint. The corresponding values of t_{avg_obsi} adopting the RRT* algorithm and the Q-RRT* algorithm in the three obstacles are shown in Table 3. It can be seen from the table that when a discrete point is selected every 1 mm in the path for solving inverse kinematics, there is not much difference for values of t_{fea_obsi} and t_{opt_obsi} obtained by applying the RRT* algorithm and the Q-RRT* algorithm in the traditional algorithm and the corresponding improved algorithm in this paper respectively, but that the joint angles of the manipulator at all positions on the path are within the limit range cannot be guaranteed by using the traditional algorithm.

Table 3
The t_{avg_obs1}, t_{fea_obs1} and t_{opt_obs1} of six algorithms.

Algorithm	Traditional RRT	RRT*	Q-RRT*	MDA-RRT	MDA-RRT*	MDA-QRRT*
t_{avg_obs1}/ms	–	422	742	1781	1589	10,655
t_{fea_obs1}/ms	–	2344	6183	1781	1589	10,655
t_{opt_obs1}/ms	–	3014	9275	4048	2648	11,335
t_{avg_obs2}/ms	–	51	420	65	124	677
t_{fea_obs2}/ms	–	–	–	–	6200	33,850
t_{opt_obs2}/ms	–	–	–	–	6200	33,850
t_{avg_obs3}/ms	–	288	300	175	593	1327
t_{fea_obs3}/ms	–	7200	2500	175	593	1327
t_{opt_obs3}/ms	–	14,400	3750	–	4236	4739

The efficiency of six algorithms is analyzed based on the values of t_{fea_obsi} and t_{opt_obsi}, and the following conclusions can be drawn: (1) When the obstacle environment is not complicated and the joint deflection angle is not required strictly, traditional algorithms such as RRT* algorithm and Q-RRT* algorithm can be used for path planning, and the path is filtered by solving inverse kinematics for a few discrete points on the path. (2) When each point on the path is required to meet the limitation of deflection angle with time for the path planning as short as possible, the MDA-RRT* algorithm can be used for path planning. (3) When each point on the path is required to meet the limitation of deflection angle with the planned path length as short as possible, the MDA-QRRT* algorithm can be used for path planning.

5.5. Verification of feasibility of the improved algorithm

According to the improved algorithm in this paper, a feasible path and an optimized path can be planned. In the complex obstacle environment corresponding to obs2, it is difficult to plan a feasible path at one time (the corresponding maximum probability in Section 5.3 is 2%). A feasible path can be planned (as shown in Fig. 9 in Section 5.2) after the improved algorithm generates 500 root nodes.

The static states corresponding to the five positions of the manipulator during the movement and the static states when the manipulator grabs the target finally are shown in Fig. 13. The red line in the figure is a path with a length of 3085 mm planned by the MDA-QRRT* algorithm. Besides, obstacles, hyper-redundant manipulator, the size of the deflection angle ψ_i between the links and some notes are also included in the figure.

It can be observed in the figure that the manipulator never touches the obstacle during the movement and can grasp the target. And according to the previous simulation results, it can be obtained that the joint deflection angle is always less than 40°, which verifies the feasibility of applying the improved algorithm to plan the path. In this section, the repeated motion process of the hyper-redundant manipulator is shown by taking the process of the manipulator moving along a feasible path as an example that can prove the feasibility of planning the path.

In short, only whether some discrete points meet the requirements when the manipulator moves on the path are analyzed, and there are errors in the process of solving the inverse kinematics in the traditional algorithm. Besides, it is also difficult to plan feasible paths in difficult environments by traditional algorithms, but feasible and optimized paths can be effectively planned by applying the improved algorithms. After adopting the improved algorithm in this paper, it can be ensured that when the manipulator moves on a continuous path, the deflection limit is satisfied at any position, that is, in the process of path planning, there is no need to solve inverse kinematics to verify whether it meets the requirements.

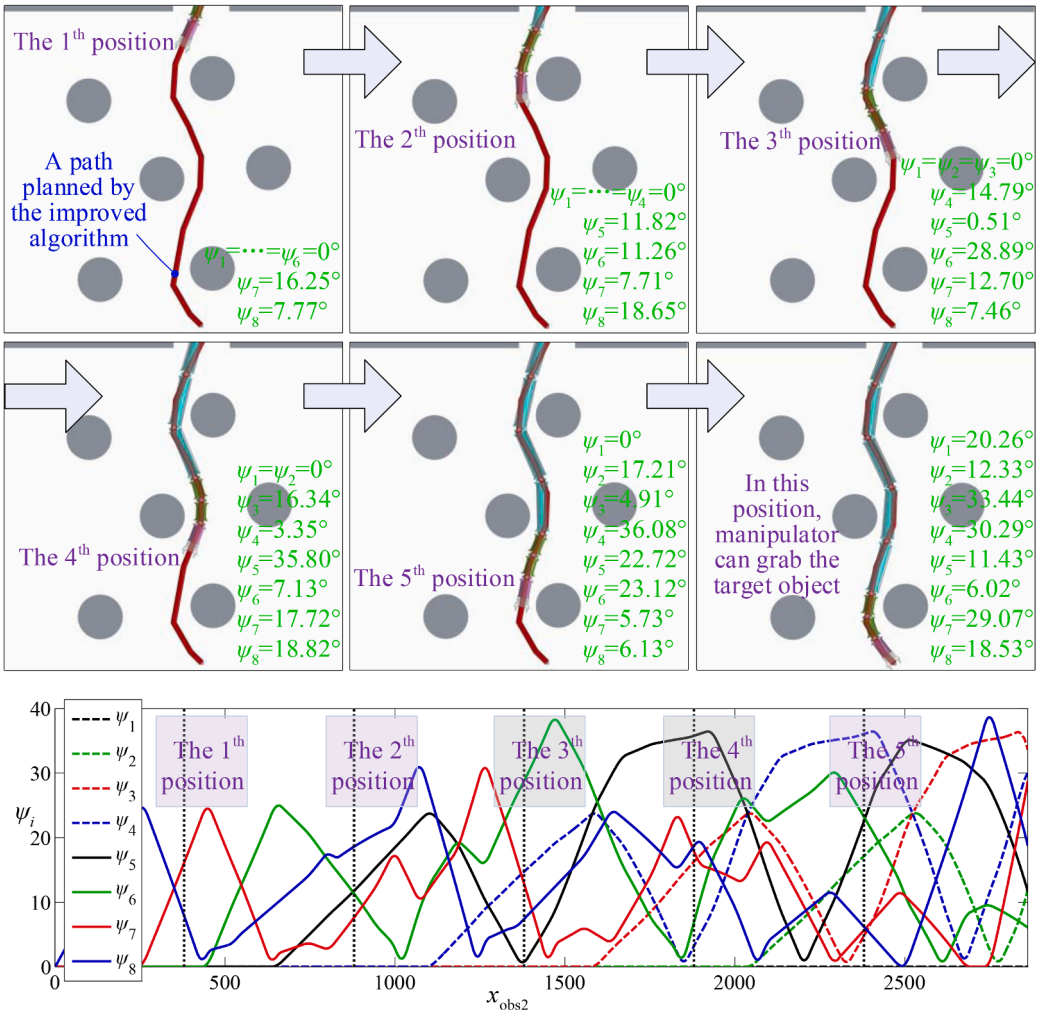


Fig. 13. The pose of hyper-redundant robot in few places along the path and the joint deflection angle ψ_i change diagram during the movement.

6. Conclusion

The MDA + RRT algorithm that can limit the joint deflection angle is proposed based on taking a hyper-redundant continuous manipulator as the moving object in this paper. The algorithm converts the limit of joint deflection angle to the limit of path deflection angle so as to analyze how to select the next random vertex under different circumstances, and can ensure that the collision-free and path angle restrictions are met when the parent is updated and the connection relationship is trimmed at the same time. By analyzing the data of the paths planned by the three RRT algorithms and the three MDA + RRT algorithms on the three obstacles, it is found that the MDA + RRT algorithm only restricts the selection range of the next vertex without increasing the computational amount of the algorithm. If the number of vertex samples is the same, the simulation time is almost the same. Besides, the feasibility and optimization of the three MDA + RRT algorithms are better than the corresponding RRT algorithm. Among the six algorithms, the feasibility and optimization of the MDA-QRRT* algorithm are the best.

In future, relevant researches will be taken into consideration, as following.

(1) In addition to adopting the MDA + RRT algorithm for path planning of hyper-redundant manipulators, the algorithm can also be applied to other moving objects, such as autonomous driving of unmanned vehicles, autonomous remote control aircraft planning, cruise missile evasion radar search, autonomous non-collision action of robots, to plan the path within the specified deflection angle.
(2) Based on the improved algorithm, variable-step RRT algorithm, bidirection RRT algorithm, or combination with other intelligent RRT algorithms can be implemented to plan a path that meets the constraints.
(3) The MDA + RRT algorithm can be combined with other path planning algorithms, such as PRM algorithm, A* algorithm, APF algorithm, to plan a path considering kinematics constraints.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.eswa.2021.116379.

References

Arun, D., & Craig, A. W. (2019). Workspace modeling and path planning for truss structure inspection by unmanned aircraft. *Journal of Aerospace Information Systems, 16*(1), 37–51.

Cao, X., Zou, X., Jia, C., Chen, M., & Zeng, Z. (2019). RRT-based path planning for an intelligent litchi-picking manipulator. *Computers and Electronics in Agriculture, 156*, 105–118.

Chao, N., Liu, Y., Xia, H., Peng, M., & Ayodeji, A. (2019). DL-RRT* algorithm for least dose path Re-planning in dynamic radioactive environments. *Nuclear Engineering and Technology, 51*, 825–836.

Douglas, R., Luís, A., Rodrigo, Y., Kelton, A., Yang, X., André, N., & João, P. (2014). A wrapper approach for feature selection based on Bat Algorithm and Optimum-Path Forest. *Expert Systems with Applications, 41*(5), 2250–2258.

Fang, Y., Qiao, D., Zhang, L., Yang, P., & Peng, W. (2016). A new cruise missile path tracking method based on second-ordersmoothing. *Optik, 127*(12), 4948–4953.

Jeong, I., Lee, S., & Kim, J. (2019). Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate. *Expert Systems with Applications, 123*, 82–90.

Jordi, P., & Alberto, S. (2019). Anticipatory kinodynamic motion planner for computing the best path and velocity trajectory in autonomous driving. *Robotics and Autonomous Systems, 114*, 93–105.

Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research, 30*(7), 846–894.

Koenig, S., Likhachev, M., & Furcy, D. (2004). Lifelong planning A*. *Artificial Intelligence, 155*(1–2), 93–146.

Korayem, M. H., & Shafei, A. M. (2015). Motion equation of nonholonomic wheeled mobile robotic manipulator with revolute-prismatic joints using recursive Gibbs-Appell formulation. *Applied Mathematical Modelling, 39*, 1701–1716.

Korayem, M. H., & Dehkordi, S. F. (2018). Derivation of motion equation for mobile manipulator with viscoelastic links and revolute–prismatic flexible joints via recursive Gibbs-Appell formulations. *Robotics and Autonomous Systems, 103*(7), 175–198.

Kuffner, J. J., & LaValle, S. M. (2000). RRT-connect: An efficient approach to single-query path planning. *Int. Conf. Robot. Automation. Symp. (ICRA)*, 995–1001.

Li, Y., Wei, W., Gao, Y., Wang, D., & Fan, Z. (2020). PQ-RRT*: An improved path planning algorithm for mobile robots. *Expert Systems with Applications, 152*, 1–11.

Liu, B., Feng, W., Li, T., Hu, C., & Zhang, J. (2020). A variable-step RRT* path planning algorithm for quadrotors in below-canopy. *IEEE Access, 8*, 62980–62989.

Marco, F., Manuel, B., Nicola, P., & Antonio, V. (2019). Predictive inverse kinematics for redundant manipulators with task scaling and kinematic constraints. *IEEE Transactions on Robotics, 35*(1), 278–285.

Mohammad, S. G. (2019). T*: A weighted double-heuristic search algorithm to find the shortest path. *International Journal of Computing Science and Mathematics, 10*(1), 58–70.

Noé, P., Fernando, C., & Luis, M. (2018). Teaching robot navigation behaviors to optimal RRT planners. *International Journal of Social Robotics, 10*(2), 235–249.

Qureshi, A. H., & Ayaz, Y. (2015). Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments. *Robotics and Autonomous Systems, 68*, 1–11.

Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Information sciences, 179*(13), 2232–2248.

Shafei, A. M., & Shafei, H. R. (2018). Dynamic modeling of tree-type robotic systems by combining 3×3 rotation and 4×4 transformation matrices. *Multibody System Dynamics, 44*(4), 367–395.

Singh, N. H., & Thongam, K. (2019). Neural network-based approaches for mobile robot navigation in static andmoving obstacles environments. *Intelligent Service Robotics, 12*(1), 55–67.

Song, B., Wang, Z., Zou, L., Xu, L., & Fuad, E. (2019). A new approach to smooth global path planning of mobile robots with kinematic constraints. *International Journal of Machine Learning and Cybernetics, 10*(1), 107–119.

Urmson, C., & Simmons, R. (2003). Approaches for heuristically biasing RRT growth. *International Conference on Intelligent Robots and Systems (IROS), 2*, 1178–1183.

Valerio, O., Naresh, M., Michael, M., Jeffrey, K., & Rustam, S. (2018). Vision-based framework to estimate robot configuration and kinematic constraints. *IEEE/ASME Transactions on Mechatronics, 23*(5), 2402–2412.

Wang, J., Li, B., & Meng, M. Q. (2021). Kinematic constrained bi-directional RRT with efficient branch pruning for robot path planning. *Expert Systems with Applications, 170*, 1–7.

Wang, Z., & Cai, J. (2018). Probabilistic roadmap method for path-planning in radioactive environment of nuclear facilities. *Progress in Nuclear Energy, 109*, 113–120.

Xie, H., Wang, C., Li, S., Hu, L., & Yang, H. (2019). A geometric approach for follow-the-leader motion of serpentine manipulator. *International Journal of Advanced Robotic Systems*, 1–18.

Yasmine, M., & Abdelhamid, D. (2019). Automatic generation of adaptive structuring elements for road identification in VHR images. *Expert Systems with Applications, 119*, 342–349.

You, C., Lu, J., Dimitar, F., & Panagiotis, T. (2019). Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning. *Robotics and Autonomous Systems, 114*, 1–18.

Zaid, T., Ahmed, H. Q., Yasar, A., & Raheel, N. (2018). Potentially guided bidirectionalized RRT* for fast optimal path planning in cluttered environments. *Robotics and Autonomous Systems, 108*, 13–27.