IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

E-RRT*: Path Planning for Hyper-Redundant Manipulators

Hongcheng Ji, Haibo Xie, Cheng Wang, Huayong Yang

***Abstract*—A hyper-redundant manipulator(HRM) can flexibly accomplish tasks in narrow spaces. However, its excessive degrees of freedom pose challenges for path planning. In this article, an ellipsoid-shape rapidly-exporing random tree (E-RRT*) method is proposed for path planning of HRMs in workspace, particularly those with angle limits. This method replaces line segments with ellipsoids to connect adjacent nodes. Firstly, an analysis of angle constraints of the HRM is conducted, providing restrictions on node selection during path planning. Secondly, a slow-speed informed guiding approach is introduced to optimize the sampling process. Finally, the obtained path is enhanced by adding control points and applying cubic polynomial interpolation to achieve path smoothing. Simulations demonstrate that the proposed E-RRT* method effectively solves the path planning problem for HRMs. Especially in narrow environments, appropriate informed guiding speeds enable E-RRT* to outperform other methods.**

***Index Terms*—Motion and path planning, collision avoidance, hyper-redundant manipulator.**

I. Intorduction

In RECENT years, hyper-redundant manipulators(HRMs) have been widely applied in many unique scenarios including spacecraft, nuclear power plants, medical treatment and tunnel boring machines [1]–[4]. Because an HRM has many degrees of freedom(DOFs), it is often used in exploration, observation, maintenance, inspection and other work [5], [6] in narrow environments [7]. Navigation technology is a basis to ensure that HRMs arrive at the goal safely and efficiently. Due to HRMs being used in confined spaces and having limitations such as angles and sizes, their path planning is complex and challenging [8], [9]. Therefore, planning a safe, reachable, and efficient path for HRMs is worth studying.

The two primary approaches for robot path planning are graph search algorithms, such as A* [10] and Dijkstra [11] , and sampling-based methods. Graph search algorithms create a graph representation of the environment to find the lowest-cost or shortest-distance path from the start to the goal. Despite their efficiency, these algorithms face challenges in computational complexity, memory usage, continuous spaces, and adaptability in dynamic environments [11].

Sampling-based methods, exemplified by Rapidly-exploring Random Trees (RRT) [12], randomly sample points and iteratively expand a tree by connecting these samples. They are well-suited for high-dimensional spaces and complex environments but may not guarantee optimal paths and can be sensitive to sampling strategies, leading to suboptimal solutions [13]. Moreover, they encounter difficulties in narrow or constrained environments [11]. As a result, many variants RRT-based improved methods have emerged, such as RRT-connect [14], EB-RRT [15], Fast RRT [16], RRT* [17], QRRT* [18] and informed RRT* [19].RRT* is a representative method known for asymptotically optimal path finding through node rewiring using cost functions in each iteration.

When traditional RRT-based algorithms are used for path planning of HRMs in configuration space, the algorithm dimensionality becomes exceedingly high due to the numerous DOFs [20]. The search space would be exceptionally large, resulting in low sampling efficiency. Additionally, collision detection would become highly complex and time-consuming [21]. Finding paths in ultra-high-dimensional spaces becomes challenging when optimize the cost of a path and obey feasibility constraints [20]. If the approach shifts to working in workspace, although the dimensionality is substantially reduced, it introduces other challenges. Due to the mechanical design of HRMs, the joint angles have certain limits. Therefore, the paths generated by RRT need to satisfy the angle constraints of HRMs to ensure successful traversal. Moreover, there are challenges related to collision avoidance and path smoothness [22], [23].

Consequently, when using the RRT method for HRMs path planning, optimization and improvement of the RRT method need to be tailored to the characteristics of HRMs. Some studies have already proposed modifications to the RRT method specifically for HRMs, resulting in the effective generation of paths. Wei et al. [22] present a specialized RRT (Sp-RRT) for follow-the-leader motion of HRMs, which can expand to multiple entrances and guarantee the final pose of the manipulator's end-effector. Jia et al. [23] introduce three corresponding improved algorithms: MDA-RRT, MDA-RRT*, MDA-QRRT* considering the maximum deflection angle of joint and compare their feasibility and optimization with traditional methods. Zhang et al. [24] combine RRT and shape control. They use a static and a dynamic curve to constrain the macroshape of HRM. However, the mentioned strategies above still have limitations in terms of angle constraints, path smoothing, and algorithm efficiency.

Manuscript received: July, 14, 2023; Revised September, 16, 2023; Accepted October, 8, 2023.

This paper was recommended for publication by Editor A.Bera upon evaluation of the Associate Editor and Reviewers' comments. This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant 226-2022-00016, in part by the National Key R&D Program of China under Grant 2022YFC3802302, and in part by the Major Program of National Natural Science Foundation of China under Grant 51890885. *(Corresponding author: Haibo Xie.)*

The authers are with the School of Mechanical Engineering, Zhejiang University, Hangzhou 310007, China(email:jhc@zju.edu.cn; hbxie@zju.edu.cn; chwang@zju.edu.cn; yhy@zju.edu.cn)

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

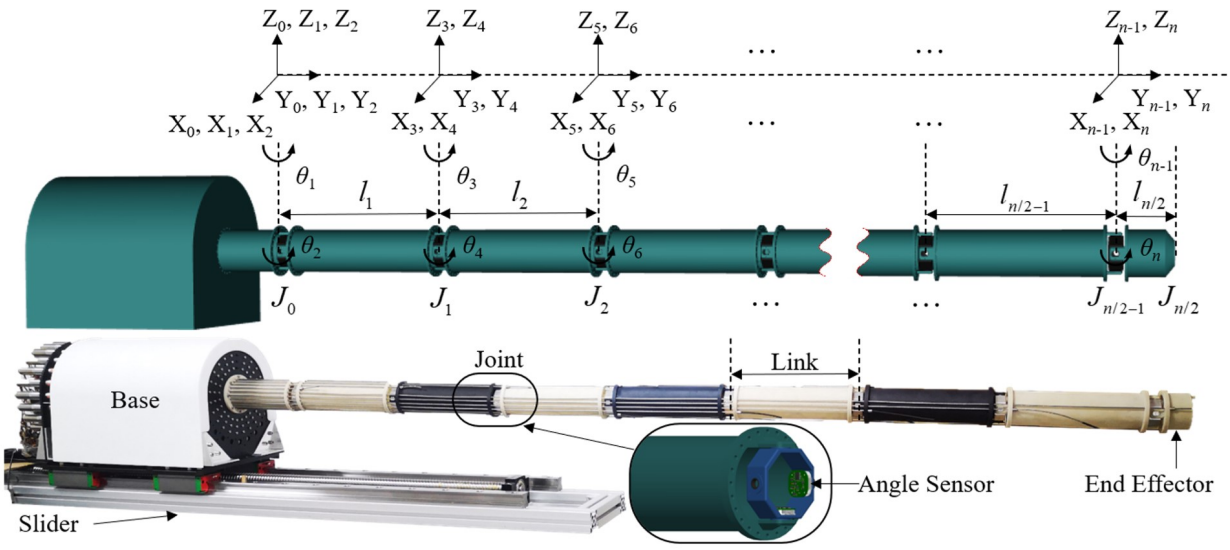

Fig. 1. D-H coordinate system of a n-DOF cable-driven HRM and a typical HRM consisting of a slider, base, arms, joints, etc.

In this letter, we propose an ellipsoid-shape RRT* (E-RRT*) algorithm for path planning of HRMs in workspace. We have noticed that some studies [25]–[27] incorporate ellipsoids or other objects into the path planning process to assist in obstacle avoidance, thereby improving the efficiency of obstacle avoidance with promising results. Our E-RRT* method also introduces ellipsoids to search for sufficient safe space. The path planning in this letter refers to tunnel-like route planning. The main contributions of this paper are as follows. 1) According to the characteristics of HRM, when using the RRT algorithm, an angle restriction is added, so that the traditional RRT algorithm can be used for HRM path planning; 2) An E-RRT* algorithm is proposed, which uses ellipsoids to connect nodes during the path planning process, so that the search result is a continuous space instead of several line segments; 3) Post-processing the nodes obtained by adding control points and polynomial interpolation which solves the problems of collision and path smoothing; 4) A slow-speed informed guiding approach is introduced, optimizing the sampling process.

The remainder of this letter is organized as follows. Section II presents the kinematic modeling of HRM and provides an exposition of the path planning problem. Section III introduces the traditional RRT algorithm and RRT* algorithm, followed by an analysis of their shortcomings when applied to HRMs. In Section IV, the E-RRT* algorithm is introduced, encompassing angle and size constraints, sampling optimization, and smoothing techniques. Section V presents simulation results and conducts a comparative analysis of the various algorithms. Finally, Section VI concludes the letter by summarizing the findings and providing future perspectives.

II. Modeling and Problem Formulation

A. Modeling of a HRM

A typical HRM with the universal joint layout comprises a base, multiple arms, joints, and an end effector. The robot's base is mounted on a slider, allowing it to move forwards and backward, with the driving device located in the base. Multiple arms and joints are arranged alternately between the base and end effector.

TABLE I
D-H Parameters of a Typical HRM

Link i	$\theta_i(^\circ)$	$\alpha_i(^\circ)$	a_i(m)	d_i(m)
1	θ_1	0	0	0
2	θ_2	0	l_2	0
...	...	...	...	...
n-1	θ_{n-1}	0	0	0
n	θ_n	0	l_n	0

We developed a kinematic model for an n-DOF cable-driven HRM using the classic Denavit-Hartenberg (D-H) coordinate system. Fig.1 displays the relevant parameters, some of which are listed in Table I. Link i whose $a_i = 0$ performs like a universal joint, so we call this structure a universal joint layout.

B. Problem Statement

The path planning problem for an HRM involves finding a collision-free path in workspace which starts from the initial point x_{start}, and reaches the target region $\mathcal{X}_{\text{goal}}$ as shown in Fig.2. While ensuring that the HRM can move without collisions and within its joint angle limits, the path should be made as short and as smooth as possible. The path here is a workspace tunnel-like pathway with length and width, defined by waypoints for HRMs to follow.

Let us consider an HRM with $n+1$ degrees of freedom. The configuration of the manipulator can be represented by a vector $\mathbf{q} = [q_0, q_1, q_2, ..., q_n]$, where q_0 is the position of the base and $q_i (i > 0)$ is equivalent to θ_i in Table I. This problem can be mathematically represented as finding $\boldsymbol{P}$ in workspace, such that

$$\boldsymbol{P}_0 = x_{\text{start}} \tag{1}$$

$$\boldsymbol{P}_N \in \mathcal{X}_{\text{goal}} \tag{2}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

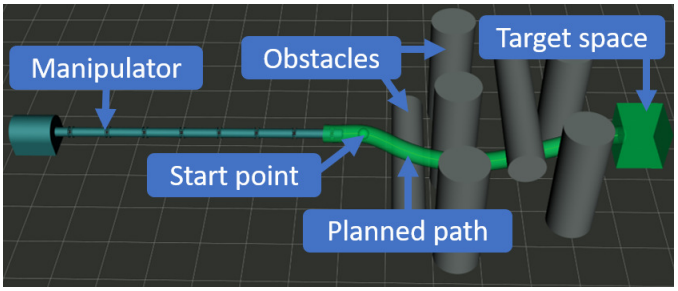

Fig. 2. Path planning description.

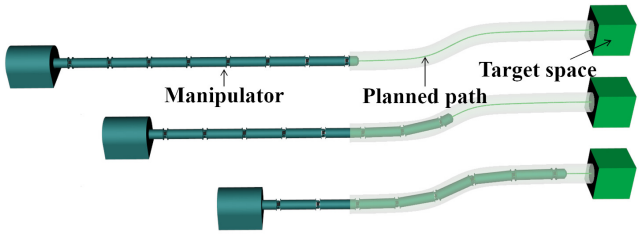

Fig. 3. Path following process.

$\boldsymbol{P}$ is an ordered set of points composed of several waypoints. By connecting the points in $\boldsymbol{P}$ sequentially in a specific manner, a path can be obtained. In traditional RRT algorithms, adjacent points are directly connected with line segments, resulting in a path composed of several line segments.

After finding such a path, the HRM proceeds along this path, as shown in Fig.3. This method of path following harnesses the flexibility of the HRM [28]. There are various approaches to path following. We adopt an intuitive approach: the HRM's joints strive to closely align with the expected path. While the base advances, joint angles from the root joint to the end-effector are computed sequentially to ensure the HRM follows the path. This process is iterated continuously until the end-effector reaches the target area. Throughout the entire path following process, the HRM needs to satisfy

$$q_{l,i} \leq q_i \leq q_{u,i}, \tag{3}$$

$$f_d(\mathbf{q}) \geq 0 \tag{4}$$

where $q_{l,i}$ and $q_{u,i}$ are the lower and upper joint limits, and $f_d(\mathbf{q})$ is a function that measures the distance between the manipulator and any obstacles in the environment.

III. Related Work and Shortcomings

A. Traditional RRT Algorithms

1) Terminology: According to the formulation as presented before [29], we have some definitions as follows. $\mathcal{X}_{\text{w}}$ denotes the work space. $\mathcal{X}_{\text{free}}$ is the free space within the boundary and without obstacles. $\mathcal{X}_{\text{forb}}$ denotes the forbidden space.

These are some simple functions used in the algorithms described later. $\texttt{Sample}()$ returns a sample point. $\texttt{NearestNeighbor}(x, V)$ returns the nearest neighbor of x within the set V. $\texttt{NearestNneighbors}(x, V, d)$ returns nearest neighbors of x within the set V whose distance is less than d. $\texttt{Steer}(x, y)$ returns a node z that is on the ray from x to y and the distance to x is the sampling step. $\texttt{CollisionFree}(\sigma)$ tests if σ is contained in $\mathcal{X}_{\text{free}}$. $\texttt{cost}(x)$ returns the path length from x_{start} to x.

Algorithm 1 RRT

$\mathcal{T}.V \leftarrow \{x_{\text{start}}\}$
while $\texttt{GeneratePath}()$ **do**
  $x_{\text{rand}} \leftarrow \texttt{Sample}()$
  $x_{\text{nearest}} \leftarrow \texttt{NearestNeighbor}(x_{\text{rand}}, \mathcal{T}.V)$
  $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})$
  if !$\texttt{CollisionFree}(\texttt{Line}(x_{\text{nearest}}, x_{\text{new}}))$ **then**
    CONTINUE
  end if
  $\mathcal{T}.V \leftarrow \mathcal{T}.V \cup \{x_{\text{new}}\}$
  $\mathcal{T}.\text{parent}(x_{\text{new}}) \leftarrow x_{\text{nearest}}$
end while

Algorithm 2 RRT*

$\mathcal{T}.V \leftarrow x_{\text{start}}$
while $\texttt{GeneratePath}()$ **do**
  $x_{\text{rand}} \leftarrow \texttt{Sample}()$
  $x_{\text{nearest}} \leftarrow \texttt{NearestNeighbor}(x_{\text{rand}}, \mathcal{T}.V)$
  $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})$
  if $\texttt{CollisionFree}(\texttt{Line}(x_{\text{nearest}}, x_{\text{new}}))$ **then**
    $\mathcal{T}.V \leftarrow \mathcal{T}.V \cup \{x_{\text{new}}\}$
    $\mathcal{T}.\text{parent}(x_{\text{new}}) \leftarrow x_{\text{nearest}}$
    $X_{\text{near}} \leftarrow \texttt{NearestNeighbors}(x_{\text{new}}, \mathcal{T}.V, d)$
    $\texttt{Rewire}(x_{\text{new}}, X_{\text{near}})$
  end if
end while

Algorithm 3 $\texttt{Rewire}(x_{\text{new}}, X_{\text{near}})$

for all $x \in X_{\text{near}}$ **do**
  if $\texttt{CollisionFree}(x_{\text{new}}, x)$ **then**
    $c \leftarrow \texttt{cost}(x_{\text{new}}) + \texttt{distance}(x_{\text{new}}, x)$
    if $c < \texttt{cost}(x)$ **then**
      $\mathcal{T}.parent(x) \leftarrow x_{\text{new}}$
    end if
  end if
end for

2) Algorithms: The RRT algorithm builds a tree of randomly generated waypoints that explores the work space of the robot. The pseudocode of the RRT algorithm is as shown in Algorithm 1.

RRT* is an extension of the RRT algorithm that guarantees asymptotic optimality by using a cost function to guide the growth of the tree. During tree expansion, RRT* tries to connect new nodes to existing nodes in a way that minimizes this cost function, which is the process of rewiring as demonstrated in Algorithm 3.

B. Shortcomings

1) Angle Issues: Traditional sampling-based methods overlook path angles. Directly using RRT and RRT* in work space can result in routes with random angles as shown in

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

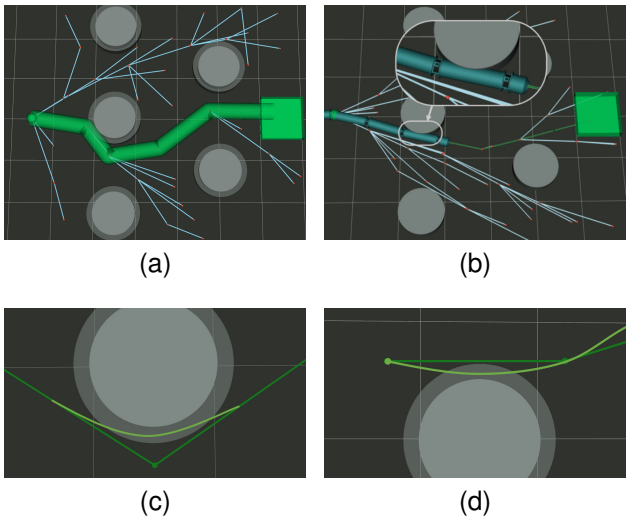

Fig. 4. Issues. In (a), the angle of the planned path exceeds the maximum joint angle HRM. In (b), HRM collides with the obstacle. In (c) and (d), the smoothed path collides with the obstacle.

Fig.4a, disregarding angle limits for HRMs. Therefore, angle restrictions must be considered when generating paths.

2) Collision Challenges: As mentioned earlier, obstacle avoidance can be expressed as $f_d(\mathbf{q}) \geq 0$, where $f_d(\mathbf{q})$ measures the distance between the manipulator and obstacles in the environment. Traditional RRT algorithms produce routes consisting of line segments, neglecting the actual volume of the robot. This oversight can lead to collisions when following the line-based route (refer to Fig. 4b). Some studies introduce $f_d(\mathbf{q}) \geq d$, with d controlled by the size of the moving object. While this solution works well in certain applications, it may result in inefficient or ineffective solutions in narrow environments.

3) Smoothness Concerns: As mentioned previously, the route found consists of line segments. Even if the angle falls within the restricted range, the route still contains numerous inflection points that negatively impact motion. Some studies tackle this problem by smoothing the route. However, in narrow conditions, this method may lead to collisions between the treated curve and obstacles (as shown in Fig. 4c and Fig. 4d). This reduces path planning efficiency and success rate. It is crucial to ensure that the smoothing function avoids collisions and adheres to joint limits.

IV. Approach

A. Size and Angle Restriction

Initially, we changed the way we connected two nodes from a straight line to an ellipsoid (ellipse in two-dimensional). The ellipsoid we use is a bit larger than a single arm of HRM. We expand this ellipsoid so that the node falls inside. In this way, we actually get a continuous area of ellipsoids, rather than a route consisting of several line segments.

The size of the ellipsoid needs to be further clarified. As talked in section II, the length of the HRM's link is l and the radius is R. We use an ellipsoid whose semi-major axis has length a and the other two semi-minor axes have length b.

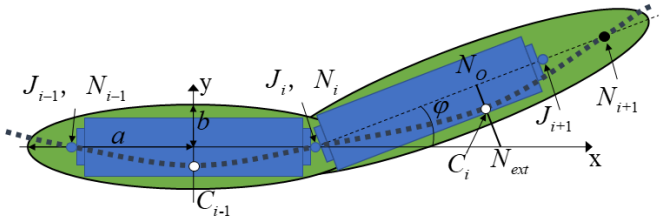

Fig. 5. Size of ellipsoid. N_i is the node. J_i represents the ith joint point. C_i is the control point. φ is the angle between adjacent path segments.

Algorithm 4 E-RRT*

$\mathcal{T}.V \leftarrow \{x_{\text{init}}\}$
$\mathcal{T}.\text{parent}(x_{\text{init}}) \leftarrow (x_{\text{init}} - (0, stepsize, 0))$
while `GeneratePath()` **do**
  $x_{\text{rand}} \leftarrow \texttt{SampleAR}(\mathcal{T}.V, \varphi_{max})$
  $x_{\text{nearest}} \leftarrow \texttt{NearestNeighbor}(x_{\text{rand}}, \mathcal{T}.V, \varphi_{max})$
  $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})$
  $\sigma \leftarrow \texttt{CreateEllipsoid}(x_{\text{nearest}}, x_{\text{new}})$
  if `CollisionFree`(σ) **then**
    $\mathcal{T}.V \leftarrow \mathcal{T}.V \cup \{x_{\text{new}}\}$
    $\mathcal{T}.\text{parent}(x_{\text{new}}) \leftarrow x_{\text{nearest}}$
    $X_{\text{near}} \leftarrow \texttt{NearestNeighbors}(x_{\text{new}}, \mathcal{T}.V, d, \varphi_{max})$
    `RewireAR`$(x_{\text{new}}, X_{\text{near}}, \varphi_{max})$
  end if
end while
`PerformPostProcessing()`

Since the link can be seen as circular, we equal the two minor axes of the ellipsoid. The ellipse should at least completely envelop the robot's single-section arm to find a collision-free area. When the size of an ellipse is at its minimum, it must satisfy $\frac{l^2}{4a^2} + \frac{R^2}{b^2} \leq 1$.

Therefore, when the parameters l and R of the robot are determined, the parameters a and b of the ellipse are in a negative correlation. The parameter a controls the length of the ellipse, while the parameter b controls its width. If a is too large, the distance between nodes will be too large, which can affect the efficiency of path planning. If b is too large, the ellipse will be too wide, which can affect the success rate of path planning in narrow environments. Therefore, it is necessary to determine the specific values of a and b based on the particular scenario, in order to ensure the smooth execution of path planning.

The angle between the parent and child ellipses, represented by φ, should be determined based on the mechanical arm's rotation angle limits. The angle φ between the ellipses must satisfy at least two conditions: the mechanical arm joints must always be inside the ellipse when passing through, and the joint angle must always satisfy $\theta_{l,i} \leq \theta_i \leq \theta_{u,i}$. The manipulator we use has the universal joint layout, which means that it has the same angles limits in both pitch and yaw directions. Therefore, the joint angle restriction can be described as

$$0 \leq \theta_i \leq \theta_{max}. \tag{5}$$

We consider the extreme case: there are multiple nodes,

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

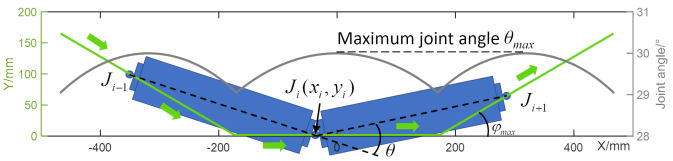

Fig. 6. Maximum joint angle. The green line is the trajectory of J_i and the gray curve reflects the change of the joint angle θ.

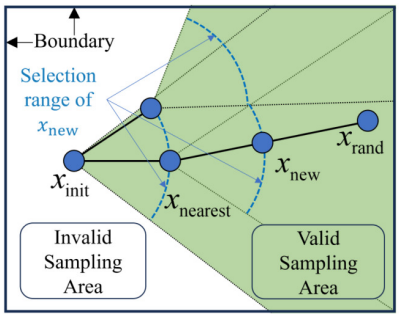

Fig. 7. An example of sampling process. After adding angle restriction, the sampling region changes accordingly.

Algorithm 5 `SampleAR`$(\mathcal{T}.V, \varphi_{max})$

while `SamplingFlag` **do**
  $x_{\text{sample}} \leftarrow (rand(\mathrm{x_l}, \mathrm{x_u}), rand(\mathrm{y_l}, \mathrm{y_u}), rand(\mathrm{z_l}, \mathrm{z_u}))$
  for all $x \in \mathcal{T}.V$ **do**
    $\varphi \leftarrow$ `CalculateAngle`$(x_{\text{sample}}, x, \mathcal{T}.\text{parent}(x))$
    if $\varphi < \varphi_{max}$ **then**
      `SamplingFlag` $\leftarrow$ FALSE
      BREAK
    end if
  end for
end while
return x_{sample}

and the distance between adjacent nodes is the minimum step length L, and the angle φ between the adjacent node connections is equal and in the same direction. In this case, we consider the maximum joint angle θ_{max} of the robotic arm. As shown in Fig.6, assuming that two adjacent arms pass through the path formed by the given nodes, the coordinate of the middle joint is $J_i(x, y)$, and the joint angle change is shown by the gray curve. When the joint J_i moves to the middle of a certain segment of the path, the joint angle is at its maximum. Assuming the maximum joint angle $\theta_{max} = 30.00°$, we can obtain the maximum angle between adjacent path segments $\varphi_{max} = 29.05°$ through Eq.(6). Therefore, when sampling and rewiring, the condition $\varphi_{max} = 29.05°$ needs to be taken into account which is shown in Algorithm 5 `SampleAR`() and Algorithm 6 `RewireAR`(). As shown in the Fig.7, the feasible sampling region in the two-dimensional space becomes a set of sectors.

$$\frac{L}{\sin(180° - \varphi_{max})} = \frac{L/2}{\sin(\varphi_{max} - \theta_{max}/2)} \tag{6}$$

Algorithm 6 `RewireAR`$(x_{\text{new}}, X_{\text{near}}, \varphi_{max})$

for all $x \in X_{\text{near}}$ **do**
  $\varphi \leftarrow$ `CalculateAngle`$(x_{\text{new}}, x, \mathcal{T}.\text{parent}(x))$
  if $\varphi < \varphi_{max}$ **then**
    $\sigma \leftarrow$ `CreateEllipsoid`$(x_{\text{nearest}}, x_{\text{new}})$
    if `CollisionFree`(σ) **then**
      $c \leftarrow$ `cost`(x_{new}) + `distance`(x_{new}, x)
      if $c <$ `cost`(x) **then**
        $\mathcal{T}.parent(x) \leftarrow x_{\text{new}}$
      end if
    end if
  end if
end for

B. Optimization of sampling

In traditional RRT* algorithm, the sampling is randomly done in $\mathcal{X}_{\text{C}}$, which means that each position has the same probability of being sampled. When we use E-RRT, since the ellipsoid has a certain volume, after generating several nodes, the nodes and ellipsoids can quickly occupy a large amount of space in $\mathcal{X}_{\text{free}}$. When new nodes are generated, the new ellipsoid often overlaps with the existing ellipsoids, resulting in few 'contributions' of new nodes and new ellipsoids to path planning.

To address the above problem, we adopt a slow-speed informed sampling method. In the early stage of path planning, we still use uniform sampling to let the algorithm find a suitable amount of nodes in $\mathcal{X}_{\text{free}}$ and initially scatter them in $\mathcal{X}_{\text{w}}$. After generating k nodes, we gradually change the sampling range, focusing on the target region $\mathcal{X}_{\text{goal}}$ from the initial entire $\mathcal{X}_{\text{w}}$, and finally just sample in $\mathcal{X}_{\text{goal}}$. Taking the x-coordinate of the sampled point x_{sample}(denoted as $\mathbf{x}_{\text{s}}$) as an example, when the number of successful samples t_s is less than k, $\mathbf{x}_{\text{s}}$ is uniformly sampled between the lower limit x_l and the upper limit x_u. When $k<t_s \leq 2k$, the sampling range gradually narrows down near $\mathcal{X}_{\text{goal}}$. In Eq.(7), $\Delta x_l = x_t - w - x_l$ and $\Delta x_u = x_u - x_t - w$. Once $t_s>2k$, the sampling range is fixed to $\mathcal{X}_{\text{goal}}$.

$$\mathbf{x}_{\text{s}} \sim \begin{cases} U[x_l, x_u], & t_s \leq k; \\ U[x_t - w - \xi\Delta x_l, x_t + w + \xi\Delta x_u], & k<t_s \leq 2k; \\ U[x_t - w, x_t + w], & t_s>2k. \end{cases} \tag{7}$$

$$\xi = 2 - \frac{t_s}{k} \tag{8}$$

This approach is beneficial for guiding the sampling process to approach the target position, and can improve the efficiency of sampling and path planning. However, when k is not chosen appropriately, if k is too small, the sampling process will be directed towards the target region too early, and the newly sampled nodes will have difficulty finding suitable parent nodes, leading to a local deadlock and the failure of path planning. Therefore, k needs to be determined according to the actual distribution of $\mathcal{X}_{\text{C}}$ and obstacles.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

C. Smoothness

We interpolate the obtained nodes with polynomial interpolation method. To ensure the obstacle avoidance performance of the robot, we need to add some control points so that the interpolation points and the manipulator can fall inside the ellipsoids.

1) Control Points: The role of control points includes two aspects: first, to increase the curvature radius of the curve and reduce the joint angle when the robot arm passes through, and second, to make the path closer to the center of the ellipse and ensure that the robot arm does not go beyond the ellipse when passing through.

After finding a path consisting of several ellipsoids, we can add control points between adjacent nodes before smoothing the path. For three adjacent path nodes, N_{i-1}, N_i, and N_{i+1}, the position of the control point C_i should lie between the extension of $N_{i-1}N_i$ and N_iN_{i+1} as illustrated in Fig5. Firstly, consider the midpoint N_O of N_iN_{i+1} as the foot of the perpendicular. A line is drawn, intersecting the extension of $N_{i-1}N_i$ at point N_{ext}. The control point C_i lies on the segment N0Ne, and the length of N_OC_i represents the maximum deviation from the center of the ellipsoid for that segment of the path. The specific length depends on the geometric size of the robotic arm, the size of the ellipsoid and the rotation angle. Here we let $N_OC_i = \frac{R\varphi}{2\varphi_{max}}$.

2) Interpolation: With E-RRT* and control points, we can get a set of waypoints as $W = [N_0, C_0, N_1, C_1, \cdots, N_{n-1}, C_{n-1}, N_n]$. There are $n+1$ nodes and n control points, so there are $2n+1$ path points and $2n$ intervals. For each interval, we interpolate in the xoy and yoz planes respectively. Assuming that the manipulator is moving along the y direction, y is regarded as an independent variable and x is regarded as a dependent variable. Suppose its equation satisfies $f_i(y) = a_iy^3 + b_iy^2 + c_iy + d_i$. To determine the coefficients a_i, b_i, c_i, and d_i within each interval, we can solve these unknowns using the endpoint function values, continuity of the first derivative, continuity of the second derivative, and boundary conditions (where the second derivative at the N_0 and N_n is zero). By applying this interpolation method, we can obtain a smooth path that enables the manipulator to move smoothly within the specified range.

V. Simulation and Modification

A. Case description

In this section, we conduct simulations on the proposed methods for different scenarios and cases. All methods are implemented in python and all experiments run on an Intel i5-12600K at 2.8GHz with 16 GB of RAM. We consider a simple 3D scenario(scenario 1) as shown in Fig.8a and Fig.8c and a challenging 3D scenario(scenario 2) as show in Fig.8b and Fig.8d. They have the same obstacle distribution, but the obstacle sizes are different. The obstacle sizes in scenario 2 are larger, making it narrower and more difficult for the manipulator to pass through. For each scenario, we consider three different cases. They share the same starting point $(0, 0, 0)$ and have different target points: $(0, 2400, 0)$, $(500, 2400, 0)$, and $(-500, 2400, -300)$(denoted as case 1, 2, and 3, respectively).

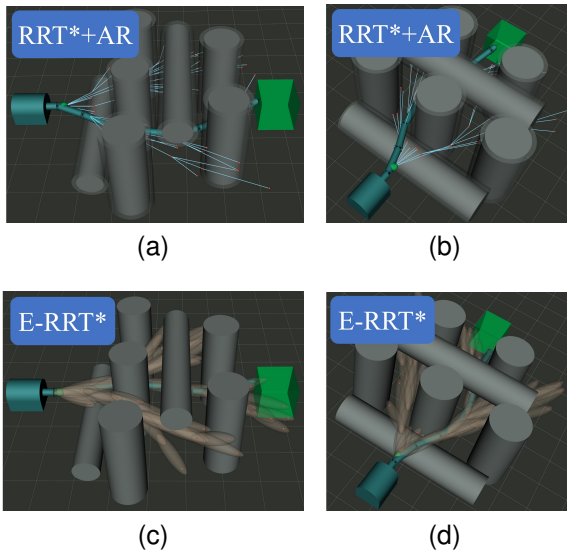

Fig. 8. Scenarios and performances. (a) RRT*+AR in easy scenario. (b) RRT*+AR in narrow scenario. (c) E-RRT* in easy scenario. (d) E-RRT* in narrow scenario.

For the above scenarios and cases, we apply different algorithms for simulation, including RRT with angle restriction(RRT+AR), RRT* with angle restriction(RRT*+AR), E-RRT*, and E-RRT* with different values of k.

B. Results and discussion

We execute 200 runs for each algorithm in each case. For a single run, we can get performances of different algorithms as demonstrated in Fig.8. We regard a single search with a total number of samples greater than 2000 and nodes more than 500 as path planning failure.

Fig.9 shows the success rate of each algorithm over time. In scenario 1 case 3, RRT+AR achieves a fast search completion with a 91% success rate. RRT*+AR is slower than RRT+AR as expected since RRT lacks a rewiring process. E-RRT* performs the worst with the slowest speed and lowest success rate. In both cases, E-RRT* and RRT*+AR show similar trends, with E-RRT* being slightly slower than RRT*+AR. This is because collision detection with ellipsoids takes more time compared to line segments. We use the Oriented Bounding Box collision detection method and find that the average time for a single collision detection with a line segment is 17.6ms, whereas for an ellipsoid, it is 27.0ms. These results align with the trend depicted in Fig.9.

When E-RRT* incorporates informed sampling, search speed and success rate get improved. In scenario 1 case 3, reducing the value of k to 20 markedly enhances the speed while decreases the success rate. This is due to premature focus on the target region, resulting in fewer nodes in $\mathcal{X}_{\text{w}}$ and difficulties in finding suitable parent nodes for newly sampled points, leading to local loop trapping and path planning failure. When $k = 40$, E-RRT* combines fast search speed with the highest success rate 97%. In scenario 2 case 3, the impact

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

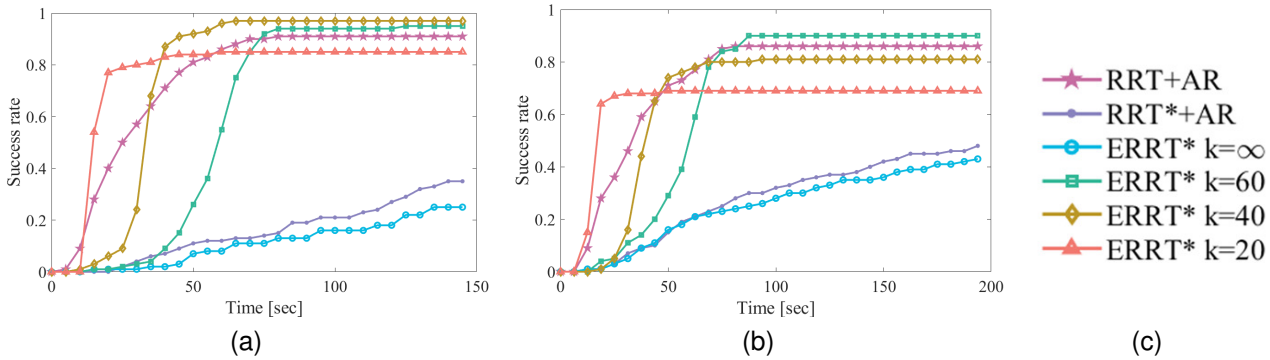

Fig. 9. Success rate over time. (a) Case 3 in scenario 1. (b) Case 3 in scenario 2. (c) Legend.

TABLE II
COMPARISON OF SUCCESS RATE

Case	RRT+AR	RRT*+AR	E-RRT* $k=\infty$	E-RRT* $k=60$	E-RRT* $k=40$	E-RRT* $k=20$
S1C1	**1.00**	**1.00**	0.99	0.99	**1.00**	0.98
S1C2	0.95	0.84	0.76	**0.97**	0.94	0.90
S1C3	0.91	0.72	0.75	0.95	**0.97**	0.85
S2C1	0.82	0.78	0.74	**0.94**	0.86	0.64
S2C2	0.90	0.88	0.80	**0.95**	0.80	0.60
S2C3	0.86	0.84	0.81	**0.90**	0.81	0.69

[1] SaCb represents case b in scenario a.

TABLE III
COMPARISON OF AVERAGE TIME CONSUMPTION

Case	RRT+AR	RRT*+AR	E-RRT* $k=60$	E-RRT* $k=40$
S1C1	6.17 ± 3.32	25.31 ± 16.47	46.71 ± 26.57	31.48 ± 14.19
S1C2	27.28 ± 13.80	165.57 ± 123.58	55.46 ± 11.22	33.20 ± 3.99
S1C3	26.08 ± 15.54	195.74 ± 152.71	56.72 ± 10.36	32.77 ± 4.64
S2C1	18.09 ± 10.51	145.43 ± 108.21	60.58 ± 18.47	48.54 ± 11.55
S2C2	38.05 ± 23.08	197.49 ± 166.15	59.58 ± 14.40	32.35 ± 7.10
S2C3	32.23 ± 18.30	179.39 ± 166.23	56.24 ± 14.22	36.50 ± 6.90

[1] SaCb represents case b in scenario a.
[2] Failed runs are not included.

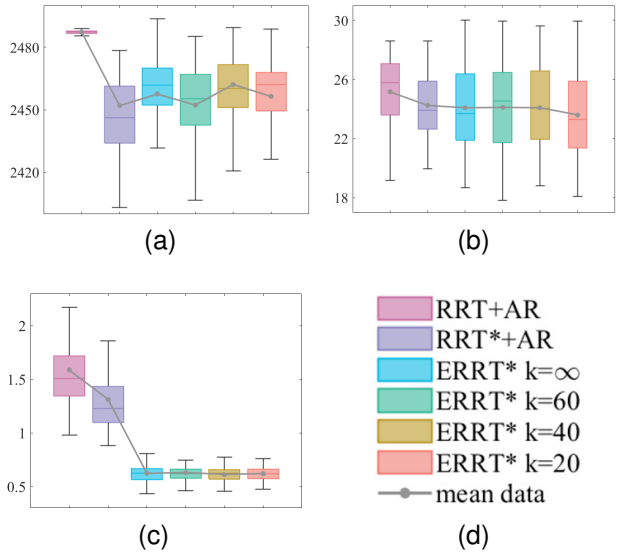

Fig. 10. Comparison of $l, \varphi, \theta, \omega$, and α in different algorithm for planning 100 paths in the narrow case. (a) road length l. (b) maximum joint angle θ. (c) maximum joint angular acceleration α. (d) legend.

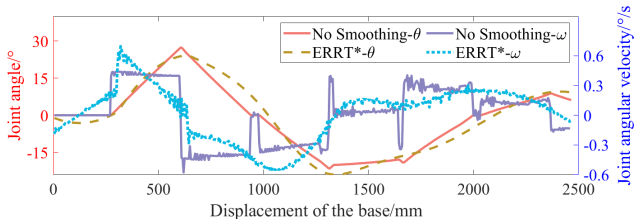

Fig. 11. Comparison of Smoothing vs. Non-Smoothing

of different values of k on the search efficiency of E-RRT* is more pronounced. Therefore, an appropriate value for k is crucial for E-RRT*, balancing search speed and success rate.

Table II presents the success rates of various algorithms in different cases. In the simplest case, S1C3, all algorithms exhibit high success rates. As the target area's position changes and the scenario becomes more confined, the success rates of all algorithms decrease to some extent. However, E-RRT* with an appropriate value of k can still maintain a high success rate. Table III compares the average completion time of various algorithms. RRT+AR consistently achieves faster search speeds, while RRT*+AR is relatively slower. With the introduction of k, E-RRT* can also achieve faster speeds.

Regarding path length, we measure it as the length of the whole path. As shown in Fig.10a, the paths obtained by RRT* and E-RRT* have significantly smaller lengths compared to the path generated by RRT. This is because the former algorithms incorporate the rewiring process resulting in reduced path lengths which is essential for manipulators with limited length.

The maximum joint angle should be less than 30°, which is validated in Fig.10b. In terms of joint angular acceleration, it is crucial to minimize the load on the HRM's driver and ensure smooth and stable movement. However, paths generated by RRT and RRT* consist of multiple line segments, resulting in numerous corners. As HRM traverses these paths, there are abrupt changes of joint angles. Fig.10c illustrates the maximum angular acceleration when the manipulator follows the path found. Fig.11 illustrates the variation of parameters θ

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

and ω of the penultimate joint with respect to the displacement of the base when HRM traverses two paths. These paths are composed of same nodes, with one path created by E-RRT* with adding control points and performing polynomial interpolation and the other without smoothing. When the HRM moves through these paths at a speed of 5mm/s, the unsmoothed path exhibits sudden changes in θ and ω, whereas the smoothed path performs much better.

VI. Conclusion

This letter proposes E-RRT*, an improved RRT* algorithm, to find a collision-free, smooth, angle-constrained, and asymptotically shortest path for HRMs in workspace. Unlike traditional RRT and RRT* algorithms applied in configuration space, E-RRT* operates in workspace to reduce the dimensionality of computation. E-RRT* incorporates angle restrictions during the sampling and rewiring processes and introduces slow-speed informed sampling. It connects adjacent nodes with ellipsoids. After obtaining a feasible path, control points are added, and polynomial interpolation is applied to achieve path smoothing. Compared to other methods, E-RRT demonstrates better performance in terms of path length and smoothness. By selecting an appropriate value for k, it can achieve a balance between path planning speed and success rate. For future work, we plan to introduce expanded ellipsoids and kinodynamic curves to further improve the efficiency and flexibility. We will also consider applying optimization-based motion planning algorithms, which do not require post-processing for smoothness.

References

[1] W. Wan, C. Sun, and J. Yuan, "Adaptive caging configuration design algorithm of hyper-redundant manipulator for dysfunctional satellite pre-capture," *IEEE Access*, vol. 8, pp. 22 546–22 559, 2020.

[2] J. Seetohul and M. Shafiee, "Snake robots for surgical applications: A review," *Robotics*, vol. 11, no. 3, p. 57, 2022.

[3] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, "Continuum robots for medical applications: A survey," *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1261–1280, 2015.

[4] R. Kang, A. Kazakidi, E. Guglielmino, D. T. Branson, D. P. Tsakiris, J. A. Ekaterinaris, and D. G. Caldwell, "Dynamic model of a hyper-redundant, octopus-like manipulator for underwater applications," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011.

[5] C. Canali, A. Pistone, D. Ludovico, P. Guardiani, R. Gagliardi, L. De Mari Casareto Dal Verme, G. Sofia, and D. G. Caldwell, "Design of a novel Long-Reach cable-driven hyper-redundant snake-like manipulator for inspection and maintenance," *Appl. Sci. (Basel)*, vol. 12, no. 7, p. 3348, 2022.

[6] A. Martín-Barrio, J. J. Roldán-Gómez, I. Rodríguez, J. Del Cerro, and A. Barrientos, "Design of a hyper-redundant robot and teleoperation using mixed reality for inspection tasks," *Sensors (Basel)*, vol. 20, no. 8, p. 2181, 2020.

[7] Z. Mu, L. Zhang, L. Yan, Z. Li, R. Dong, C. Wang, and N. Ding, "Hyper-redundant manipulators for operations in confined space: Typical applications, key technologies, and grand challenges," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 58, no. 6, pp. 4928–4937, 2022.

[8] C. Wang, H. Xie, and H. Yang, "An iterative path-following method for hyper-redundant snake-like manipulator with joint limits," *Ind. Rob.*, vol. 50, no. 3, pp. 505–519, 2023.

[9] S. Jainandunsing, J. L. Herder, Y. Takeda, and D. Matsuura, "Design of a compliant environmentally interactive snake-like manipulator," in *ROMANSY 21 - Robot Design, Dynamics and Control*. Cham: Springer International Publishing, 2016, pp. 233–240.

[10] R. Song, Y. Liu, and R. Bucknall, "Smoothed A* algorithm for practical unmanned surface vehicle path planning," *Appl. Ocean Res.*, vol. 83, pp. 9–20, 2019.

[11] A. R. Soltani, H. Tawfik, J. Y. Goulermas, and T. Fernando, "Path planning in construction sites: performance evaluation of the dijkstra, A*, and GA search algorithms," *Adv. Eng. Inform.*, vol. 16, no. 4, pp. 291–303, 2002.

[12] R. Kabutan and T. Nishida, "Motion planning by T-RRT with potential function for vertical articulated robots," *Electr. Eng. Japan*, vol. 204, no. 2, pp. 34–43, 2018.

[13] W. Zhang, L. Shan, L. Chang, and Y. Dai, "SVF-RRT*: A stream-based VF-RRT* for USVs path planning considering ocean currents," *IEEE Robot. Autom. Lett.*, vol. 8, no. 4, pp. 2413–2420, 2023.

[14] A. H. Qureshi and Y. Ayaz, "Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments," *Rob. Auton. Syst.*, vol. 68, pp. 1–11, 2015.

[15] J. Wang, M. Q.-H. Meng, and O. Khatib, "EB-RRT: Optimal motion planning for mobile robots," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 4, pp. 2063–2073, 2020.

[16] Z. Wu, Z. Meng, W. Zhao, and Z. Wu, "Fast-RRT: A RRT-based optimal path finding method," *Appl. Sci. (Basel)*, vol. 11, no. 24, p. 11777, 2021.

[17] I.-B. Jeong, S.-J. Lee, and J.-H. Kim, "Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate," *Expert Syst. Appl.*, vol. 123, pp. 82–90, 2019.

[18] C. Zhuge, Q. Wang, J. Liu, and L. Yao, "An improved Q-RRT* algorithm based on virtual light," *Comput. Syst. Sci. Eng.*, vol. 39, no. 1, pp. 107–119, 2021.

[19] M.-C. Kim and J.-B. Song, "Informed RRT* with improved converging rate by adopting wrapping procedure," *Intell. Serv. Robot.*, vol. 11, no. 1, pp. 53–60, 2018.

[20] D. Berenson, T. Simeon, and S. S. Srinivasa, "Addressing cost-space chasms in manipulation planning," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011.

[21] J. Huh and D. D. Lee, "Learning high-dimensional mixture models for fast collision detection in Rapidly-Exploring random trees," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016.

[22] H. Wei, Y. Zheng, and G. Gu, "RRT-based path planning for follow-the-leader motion of hyper-redundant manipulators," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021.

[23] L. Jia, Y. Huang, T. Chen, Y. Guo, Y. Yin, and J. Chen, "MDA + RRT: A general approach for resolving the problem of angle constraint for hyper-redundant manipulator," *Expert Syst. Appl.*, vol. 193, no. 116379, p. 116379, 2022.

[24] X. Zhang, J. Liu, and Y. Li, "An obstacle avoidance algorithm for space hyper-redundant manipulators using combination of RRT and shape control method," *Robotica*, vol. 40, no. 4, pp. 1036–1069, 2022.

[25] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Springer Tracts in Advanced Robotics*. Cham: Springer International Publishing, 2015, pp. 109–124.

[26] A. Amice, H. Dai, P. Werner, A. Zhang, and R. Tedrake, "Finding and optimizing certified, collision-free regions in configuration space for robot manipulators," in *Algorithmic Foundations of Robotics XV*. Cham: Springer International Publishing, 2023, pp. 328–348.

[27] S. Savin and A. Klimchik, "Morphing-enabled path planning for flying tensegrity robots as a semidefinite program," *Front. Robot. AI*, vol. 9, p. 812849, 2022.

[28] H. Ji, H. Xie, and H. Yang, "A spatial path following method for hyper-redundant manipulators by step-by-step search and calculating," in *2022 7th International Conference on Robotics and Automation Engineering (ICRAE)*. IEEE, 2022.

[29] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.*, vol. 30, no. 7, pp. 846–894, 2011.