2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-1714-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IROS51168.2021.9635876

RRT-Based Path Planning for Follow-the-Leader Motion of Hyper-Redundant Manipulators

Hanghang Wei, Yang Zheng and Guoying Gu, *Senior Member, IEEE*

***Abstract*— Hyper-redundant manipulators with slender body and high dexterity are widely applied for operations in confined spaces. Among the motion planning methods for these operations, the follow-the-leader motion controller is generally developed to avoid the obstacles, while the path trajectories are usually given. In this paper, we present an autonomous motion planner with a specialized rapidly exploring random tree (Sp-RRT) approach for follow-the-leader motion of hyper-redundant manipulators. Starting from the target pose in the workspace, the exploring tree can expand to multiple entrances while guaranteeing the final pose of the manipulator's end-effector. Meanwhile, the dexterity of hyper-redundant manipulators (even with different segments) can be utilized sufficiently with customized expanding parameters. Simulation results compared with existing methods are conducted to demonstrate the aforementioned characteristics and effectiveness. For further validation, we experimentally verify the development with our custom-built hyper-redundant manipulator to realize the generated path with follow-the-leader motion.**

I. Introduction

Hyper-redundant manipulators with slender body and redundant degrees of freedom (DOFs) can complete complicated operations in confined spaces, which leads to its wide applications in aerospace [1], nuclear facilities [2], medical surgery [3] and other fields [4] .

Despite continuum mechanisms are also utilized in some redundant manipulators [5, 6], articulated multi-link structures have become the main form of hyper-redundant manipulators. They are composed of several rigid links and corresponding revolute joints in a serial way, whose actuators are either integrated in joints [4, 7] or proximally arranged [1, 8].

Hyper-redundant manipulators possess high dexterity at the cost of extra complexity in motion planning [9]. Besides, end-effector's pose of the manipulators is crucial for most tasks such as inspections, cleaning and welding [10]. Therefore, motion planning in the presence of constraints on end-effector's pose as well as avoiding collisions in confined spaces for hyper-redundant manipulators is significant but challenging.

This work was supported in part by the China National Key R & D Program under Grant 2019YFB1311204, in part by the National Natural Science Foundation of China under Grant 52025057, in part by the Science and Technology Commission of Shanghai Municipality under Grant 20550712100. (Corresponding author: Guoying Gu.)

The authors are with Robotics Institute, School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China, and also with the State Key Laboratory of Mechanical System and Vibration, Shanghai Jiao Tong University, Shanghai 200240, China.

In order to deal with the motion planning of hyper-redundant manipulators, many works have been reported. The conventional approaches are generally based on the pseudo-inverse [11] or extended inverse [12] of the Jacobian matrix, which are difficult to meet specific configuration demands conveniently. The configuration space (C-space) approach maps obstacles to C-space of the manipulator [13], whose body is a point in C-space, making obstacle avoidance much easier to represent. However, the high dimensional C-space of hyper-redundant manipulator leads to low planning efficiency. The modal-based planning method [14] uses curves to represent the target configuration of the manipulator at each moment, and controls the manipulator to match the shape of the curve. Since the description parameters of the curve are generally less than the DOFs of the manipulator, the complexity of planning reduces significantly. However, the space swept by the manipulator is relatively large, so it is difficult to plan a collision-free trajectory in a confined space. With follow-the-leader strategies [15], the tip of the manipulator moves along a path, while the rest of the body follows the motion of the tip. This approach has sufficient obstacle avoidance ability and high planning efficiency, which enhances the ability of hyper-redundant manipulators to work in confined spaces.

A follow-the-leader strategy usually includes two steps: path planning for the tip in workspace and the corresponding motion realization. The realization of follow-the-leader motion have been well studied in the literature [16]. For the path planning step, researchers have developed abundant classic methods with collision-free constraints [17, 18] such as rapidly-exploring random tree (RRT), but most of them are designed to be used in C-space [19]. When these methods are directly applied to the workspace, less consideration is given to the smoothness of the path. The planning results in the workspace is also taken into account in several methods. An adaptive stepsize RRT planning algorithm [20] can automatically adjust the expanded stepsize in C-space, so that the stepsize in the workspace is approximately the same in spite of the nonlinearity of the mapping between C-space and workspace. However, it cannot restrict the curvature of the resulting path. The spline-RRT* method [21] is applied directly in the workspace, in which spline curves are used to connect the sample points to generate a smooth curve with end orientation constraint. Unfortunately, curvature constraints are not considered in the process of generation. Efforts have been made to limit the curvature of the path while planning [22]. However, the exact curvature constraint is difficult to describe for hyper-redundant manipulator with different segments.

Therefore, existing methods cannot meet all the requirements of paths for follow-the-leader motion of hyper-redundant manipulators: no collision, sufficiently

utilization of the dexterity, satisfying the end-effector's pose constraint and followed ability (i.e., the path can be followed). Currently, the path planning for follow-the-leader motion remains a challenging problem.

This paper presents a customized path planning approach for follow-the-leader motion of hyper-redundant manipulators. Compared with existing methods, the proposed approach possesses following characteristics: (1) by constructing an exploring tree starting from the target pose in the workspace, this algorithm can meet the pose constraints while realizing expandability to multiple entrances; (2) by expanding the tree, different parameters decided by layers of nodes and manipulator parameters are utilized for adaptation of the hyper-redundant manipulator with different segments; (3) the generated path is actually the final configuration of the manipulator, which ensures the followed ability of the path.

The rest of this paper is organized as follows: Section II briefly describes the structure and kinematics of the hyper-redundant manipulator, as well as its application scenarios; The specialized RRT algorithm (Sp-RRT) is presented in Section III; Section IV presents the experimental results. Section V concludes this paper.

II. Problem Descriptions and Design Objectives

A. Overview of the Hyper-Redundant Manipulator System

The hyper-redundant manipulator system [23, 24] mainly consists of two parts: a snake-like manipulator and a multi-DOF mobile base. The 24-DOF snake-like manipulator is composed of 12 serially arranged identical segments, as shown in Fig. 1. A single segment consists of a 150-mm link and a universal joint with structural angle limits of 45°, as shown in Fig. 1(a). The two DOFs of the universal joint are actuated by 3 driven cables distribute with equal interval of 120°. The multi-DOF mobile base provides desired starting poses with two electro-hydraulic linear actuators and the feeding motion with a feeding motor.

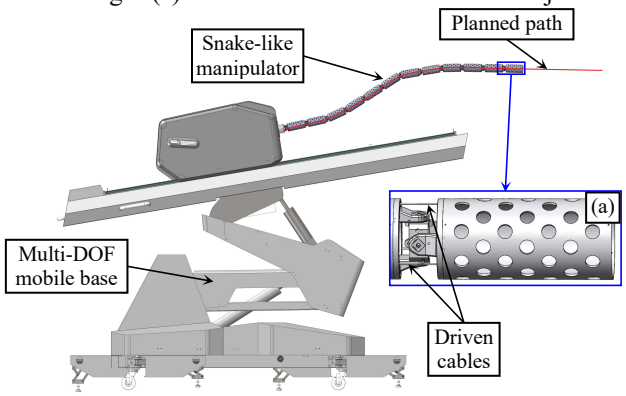

Fig. 1. The hyper-redundant manipulator system: (a) a single segment.

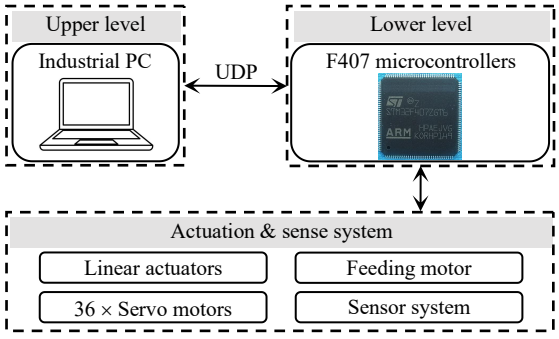

Fig. 2. The control diagram.

As illustrated in Fig.2, the control infrastructure can be divided into 3 parts: the upper level, the lower level and the actuation & sense system. The industrial PC in the upper level takes charge of path planning, motion control and coordination of the whole system. The upper level communicates with the lower level via user datagram protocol (UDP) over a local area network (LAN). Besides collecting feedback data from the sensor system as well as transferring the data to the upper level, the F407 microcontrollers in the lower level control all the actuators in the actuation & sense system via controller area network (CAN) protocol. In the actuation & sense system, the 36 MAXON servo motors pull the corresponding driving cables to control the 12 segments of the snake-like manipulator while the linear actuators and the feeding motor actuate the mobile base to the desired configuration. Based on the control infrastructure, the snake-like manipulator works in a closed-loop manner as proposed in [24].

Before the follow-the-leader motion, the multi-DOF mobile base moves to the appropriate pose and remains fixed. With the feeding motion, each joint of the snake-like manipulator is actuated to adjust the configuration, so that the tip of the manipulator moves along the planned path, and the rest of the manipulator follows the tip [16].

B. Kinematics of the Snake-like Manipulator

The kinematics of the snake-like manipulator can be expressed as mappings between 3 spaces, as shown in Fig. 3. Since the closed-loop control is aimed at the joint angles, the kinematic mappings from the configuration space to the actuation space and the workspace (f_1^{-1} and f_2) are detailed as follows.

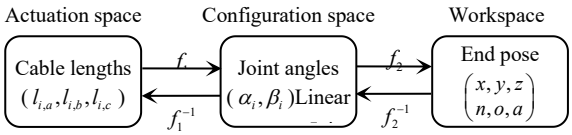

Fig. 3. Kinematics mappings.

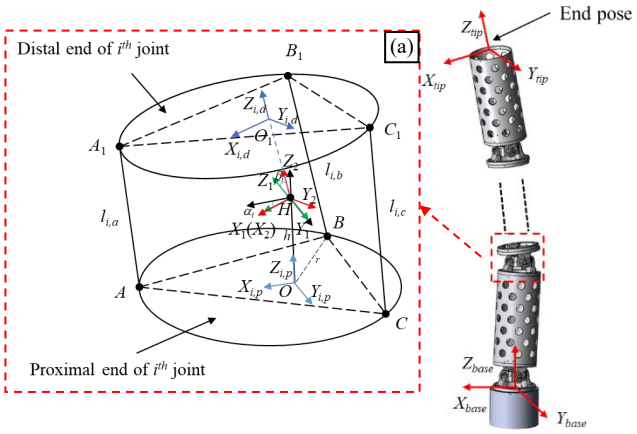

Fig. 4. The geometrical structure of the snake-like manipulator and (a) the i^{th} universal joint.

In the schematic diagram of the i^{th} universal joint (i is the index of the segments denoted from the proximal side) shown in Fig. 4(a): $l_{i,a}$, $l_{i,b}$ and $l_{i,c}$ refers to the portion lengths of the 3 driving cables passing through the i^{th} joint ($\|AA_1\|_2$, $\|BB_1\|_2$, $\|CC_1\|_2$); α_i and β_i are the yaw and pitch angles of the i^{th} joint respectively; $\mathbf{\Phi}_{i,p} = \{X_{i,p}, Y_{i,p}, Z_{i,p}\}$ and $\mathbf{\Phi}_{i,d} = \{X_{i,d}, Y_{i,d}, Z_{i,d}\}$ are the coordinates attached to the proximal and distal end of the i^{th} joint respectively.

As proposed in [23], the kinematic mapping from configuration space to actuation space of the i^{th} joint is denoted as

$$\begin{aligned} l_{i,a} &= \sqrt{2r^2 + 2h^2 + \overrightarrow{AO}\cdot\overrightarrow{HA_1} + \overrightarrow{OH}\cdot\overrightarrow{HA_1}} \\ l_{i,b} &= \sqrt{2r^2 + 2h^2 + \overrightarrow{BO}\cdot\overrightarrow{HB_1} + \overrightarrow{OH}\cdot\overrightarrow{HB_1}} \\ l_{i,c} &= \sqrt{2r^2 + 2h^2 + \overrightarrow{CO}\cdot\overrightarrow{HC_1} + \overrightarrow{OH}\cdot\overrightarrow{HC_1}} \end{aligned} . \quad (1)$$

And the kinematic mapping from configuration space to workspace of the i^{th} joint is denoted as

$$\begin{aligned} {}^{\mathbf{\Phi}_{i,d}}_{\mathbf{\Phi}_{i,p}}\mathbf{T} &= \mathrm{Trans}(0,0,h)\mathrm{Rot}(y,\ \alpha_i)\mathrm{Rot}(x,\beta_i)\mathrm{Trans}(0,0,h) \\ &= \begin{bmatrix} \cos\alpha_i & \sin\alpha_i\sin\beta_i & \sin\alpha_i\cos\beta_i & h\sin\alpha_i\cos\beta_i \\ 0 & \cos\beta_i & -\sin\beta_i & -h\sin\beta_i \\ -\sin\alpha_i & \cos\alpha_i\sin\beta_i & \cos\alpha_i\cos\beta_i & h\cos\alpha_i\cos\beta_i + h \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (2)$$

where Trans($\vec{\lambda}$) refers to the homogenous matrix of translation transformation along $\vec{\lambda}$, while Rot($\hat{\lambda}$,θ) refers to the homogenous matrix of rotation transformation about $\hat{\lambda}$ with θ.

For concise expression, $\mathbf{\Phi}_{1,p}$ and $\mathbf{\Phi}_{13,p}$ is regarded as the base coordinates $\mathbf{\Phi}_{base}$ and the tip coordinates $\mathbf{\Phi}_{tip}$ respectively, as shown in Fig. 4. Therefore, the end pose can be derived as

$${}^{\mathbf{\Phi}_{tip}}_{\mathbf{\Phi}_{base}}\mathbf{T} = \prod_{i=1}^{12} {}^{\mathbf{\Phi}_{i,d}}_{\mathbf{\Phi}_{i,p}}\mathbf{T}\,{}^{\mathbf{\Phi}_{i+1,p}}_{\mathbf{\Phi}_{i,d}}\mathbf{T} \quad (3)$$

where ${}^{o_{i+1,p}}_{o_{i,d}}T = \mathrm{Trans}(0,0,l)$ (l is the length of each link).

C. Planning Objectives

For the convenience of end-effectors' operations, there is an extra requirement for the end orientation of the path besides the end position. Conversely, the starting orientation of the path to entry the confined space is less required because it is convenient to reach with the mobile base. Besides, for confined spaces such as an airplane fuel tank model shown in Fig. 5, the multiple entrances can expand the operating scope.

As aforementioned, the objectives for this path planning algorithm are presented as follows:

- End-effector's pose constraint: When the follow-the-leader motion of the manipulator along the path is completed, the end-effector's pose should meet the operational requirement.
- Parametrical customization: To utilize the dexterity of hyper-redundant manipulators, algorithm parameters can be customized according to its structure.

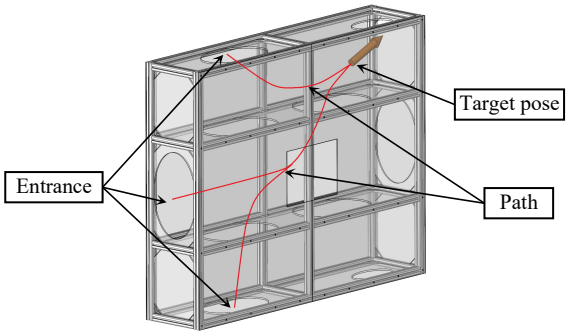

Fig. 5. An airplane fuel tank model.

- Followed ability: During the entire follow-the-leader motion, the rotation angle of each joint should be within its limitation.
- Planning efficiency in scenarios with multiple entrances.

III. Sp-RRT Algorithm

Based on the design objectives, the Sp-RRT algorithm for follow-the-leader motion of hyper-redundant manipulators is proposed in detail.

A. Path Planning

The Sp-RRT algorithm is designed to find a collision-free path between the target zone and several entrances in the workspace. Selecting the target point as the root node, a random tree is expanded with the limitation of manipulator's parameters to find feasible paths for different entrances.

The nomenclatures are as follows: Q, *Tree* and *Path* ($Path=\{Path(j)\}$, j = 1, 2, …, n.) represent a confined space with n entrances, the exploring tree and the path set generated from the j^{th} entrance to target zone, respectively; q_{rand}, q_{goal} and

Algorithm 1 : $Path \leftarrow$ Sp-RRT ($q_{\mathrm{goal}}, q_{\mathrm{start}}$)

q_{goal}.num = 1
q_{goal}.fatherNode = q_{goal} + $\mathbf{O}_F$
Tree.addNode(q_{goal})
for k = 1 : $k_{\max}$
  Randomly Sample $q_{\mathrm{rand}} \in Q$
  [q_{new}, flag] ← Expand(*Tree*, q_{rand})
  if flag
    for j = 1 : n
      if *Path*(j) already found
        continue
      end if
      if $\| q_{\mathrm{new}} - q_{\mathrm{start}(j)} \|_2 \le \varepsilon$
        $Path(j) = q_{\mathrm{new}} \rightarrow q_{\mathrm{goal}}$
      else if Connect($q_{\mathrm{new}}, q_{\mathrm{start}(j)}$) ==*true*
        $Path(j) = q_{\mathrm{start}(j)} \cup (q_{\mathrm{new}} \rightarrow q_{\mathrm{goal}})$
      end if
    end for
  end if
  if all path found
    break
  end if
end for
return *Path*

Algorithm 2 : $[q_{new}, \text{flag}] \leftarrow \text{Expand}(Tree, q_{rand})$

1: $Q_{abandoned} = \varnothing$
2: **while** $Tree - Q_{abandoned} \neq \varnothing$
3: $q_{near} = \text{Nearest}(Tree - Q_{abandoned}, q_{rand})$
4: $q_{new} = q_{near} + \frac{q_{rand} - q_{near}}{\| q_{rand} - q_{near} \|} \cdot L_{link}(q_{near}.\text{num})$
5: **if** $\text{Connect}(q_{near}, q_{new})$
6: $q_{new}.\text{num} = q_{near}.\text{num} + 1$
7: $q_{new}.\text{fatherNode} = q_{near}$
8: $Tree = Tree \cup q_{new}$
9: **return** $[q_{new}, true]$
10: **else**
11: $Q_{abandoned} = Q_{abandoned} \cup q_{near}$
12: **end if**
13: **return** $[q_{new}, false]$

Algorithm 3 : $\text{Connect}(q_1, q_2)$

1: $q_f = q_1.\text{fatherNode}$
2: $\theta = \text{Angle}(q_1 - q_f, q_2 - q_1)$
3: **if** $\|q_2 - q_1\|_2 > L_{link}(q_1.\text{num})$
4: **return** *false*
5: **else if** $\theta > \theta_{limit}(q_1.\text{num})$
6: **return** *false*
7: **else if** $\text{Collision}(q_1, q_2)$
8: **return** *false*
9: **else**
10: **return** *true*
11: **end if**

Algorithm 4 : $Path_{optimal} \leftarrow \text{Optimize}(Path)$

1: $Path = \{q_1, q_2, \ldots, q_{final}\}$
2: **Var** P_1, P_2
3: $P_1 = \text{Path}$, $P_2 = \emptyset$
4: $P_2.AddNode(q_1)$
5: $a=1$, $b=3$
6: **while** $b \leq \text{final}$
7: $\text{flag} \leftarrow \text{FeasibleJudge}(q_a, q_b)$
8: **if** flag == true
9: $b = b+1$
10: **else**
11: $P_2.AddNode(q_{b-1})$
12: $a = b-1$
13: $b = a+2$
14: **end if**
15: $P_2.AddNode(q_{final})$
16: **return** P_2

Algorithm 5 : $\text{flag} \leftarrow \text{FeasibleJudge}(q_a, q_b)$

1: $\theta_a = 180° - \angle q_{a-1}q_a q_{a+1}$, $\theta_b = 180° - \angle q_{b-1}q_b q_{b+1}$
2: $\theta_a' = 180° - \angle q_{a-1}q_a q_b$, $\theta_b' = 180° - \angle q_a q_b q_{b+1}$
3: **if** $\text{Collision}(q_a, q_b)$
4: flag = *false*
5: **else if** $\theta_a' > \theta_a$ or $\theta_b' > \theta_b$
6: flag = *false*
7: **else**
8: flag = *true*
9: **end if**
10: **return** flag

$q_{start(j)}$ represent the node selected randomly in Q, the target node and the j^{th} entrance nodes, respectively; The length of the i^{th} link is $L_{link}(i)$ and the joint angle limit of the i^{th} joint is $\theta_{limit}(i)$, where the index i is denoted from the tip of the hyper-redundant manipulator (differs from the definition in Section II B).

Algorithm 1 describes the main steps of Sp-RRT algorithm. For a tree node, q, q.num and q.fatherNode represent the Cartesian coordinates, the layer of this node and its father node, respectively. In Sp-RRT, each node is regarded as central point of a joint of the hyper-redundant manipulator, so q.num can be used to determine the stepsize (line 4 of Algorithm 2) and the joint angle limit (line 5 of Algorithm 3) during the expanding process. Firstly, the exploring tree is initialized with the target point q_{goal} as the root node, where $\mathbf{O}_F$ denotes the unit vector of the final orientation (Lines 1-3). Secondly, q_{rand} is randomly sampled in Q and determined if it can be used to expand *Tree* (Lines 5-7). Then whether *Path*(j) have been found is judged (Lines 8-11), if not, q_{new} will be used to determine if *Path*(j) can be completed (Lines 12-16). The algorithm will continue until *Path* is completely found or reaching the iteration limitation k_{max}.

Algorithm 2 describes the process of extending *Tree* towards q_{rand} by finding the closest node to q_{rand} in *Tree* that meets the expand requirements. $Q_{abandoned}$ is the set of nodes that do not meet the requirements. q_{near} is the nearest node to q_{rand} in *Tree* excluding $Q_{abandoned}$. While q_{near} is extended towards q_{rand} to reach the new node q_{new}, the step length is derived from the corresponding link length of the node q_{near} (Line 4).

Connect((q_1,q_2) (Algorithm 3) decides whether q_2 can be connected with q_1. There are three main requirements. The length limit (Lines 3-4) is only used in Algorithm 1 (Line 14) to determine whether or not an entrance has been reached; The rotation angle limit of q_1 is derived from the corresponding joint angle limit; The collision judgement is related to the representation model of obstacles. This model should be capable to judge whether a node in Q is collision-free. Octomap [25] is used in Section IV for collision detection.

Using Sp-RRT, the generated path is actually the final configuration of the manipulator when the follow-the-leader motion is completed.

B. Path Optimization

The randomness in the path planning may result in unnecessary windings as shown in Fig. 6 (highlighted with green dashed box). To solve this problem, A method of path optimization is presented. By trying to connect each node of the generated path, unnecessary windings are excluded from the path to decrease the length and turning angles of the generated path.

In Algorithm 4, P_1, P_2 are the original path and optimized path. As illustrated in Figure 6, this optimization algorithm tries to connect the non-adjacent nodes q_a, and q_b of P_1. If the connected path possesses feasibility, the two nodes are directly

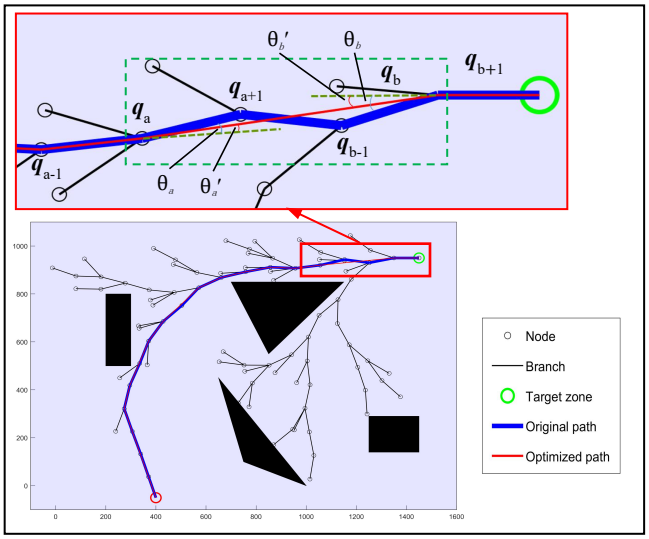

Fig. 6. Path optimization diagram.

connected in P_2 to eliminate unnecessary windings caused by randomness. The feasibility judgements are detailed in Algorithm 5.

In addition, the formulations in line 2 of Algorithm 5 can be replaced by $\theta_a' = \theta_{\text{limit}}(q_a.\text{num})$, $\theta_b' = \theta_{\text{limit}}(q_b.\text{num})$, which means the optimized rotation angle is within the rotation limit of the corresponding joint. In this condition, the feasibility requirements are loosened for shorter path length.

IV. Simulation and Experimental Validation

In this section, the proposed Sp-RRT algorithm is validated by simulation and experiments, as shown in the multimedia extension. A goal bias of probability 5% towards the target zone [17] is given in all experiments. The path following algorithm in [16] is used to test the followed ability of generated paths. All of the numerical experiments are performed on a platform with Intel Core i7-6700K 4.0 GHz.

*A. Path planning comparison with RRT and RRT**

In this part, a planar manipulator consisting of 20 100-mm links and 19 joints with rotation limits of 30° is used to demonstrate the characteristics of Sp-RRT compared with RRT and RRT* [26] in a confined space with several obstacles. The stepsize of all three algorithms is set to 100 mm, while the target final orientation is set to the positive direction of the horizontal axis. Besides, for stable results, 3000 nodes are used in RRT*.

Path planning tests are performed 100 times using each algorithm, and the corresponding statistics (average values) are shown in Table I, including path length, node quantity, computational time and orientation deviation. One of the results is shown in Fig. 7.

For each path, the follow-the-leader motion is carried out using the planar manipulator, and the maximum rotation angle of each joint is recorded. The average results (angles of the distal 14 joints) are shown in Fig. 8.

It can be observed that, RRT algorithm is the fastest but its poor performances in other aspects make its paths hard to follow. RRT* algorithm can find the shortest path, but the existing problems such as sharp corners, orientation deviation

TABLE I.
Comparison Statistics of RRT, RRT* and Sp-RRT.

	Path length	Nodes quantity	Computational time /s	Orientation deviation /°
RRT	2163	87.32	0.016	44.40
RRT*	1525.47	3000	35.48	59.35
Sp-RRT	1821.65	100.19	0.30	0.00

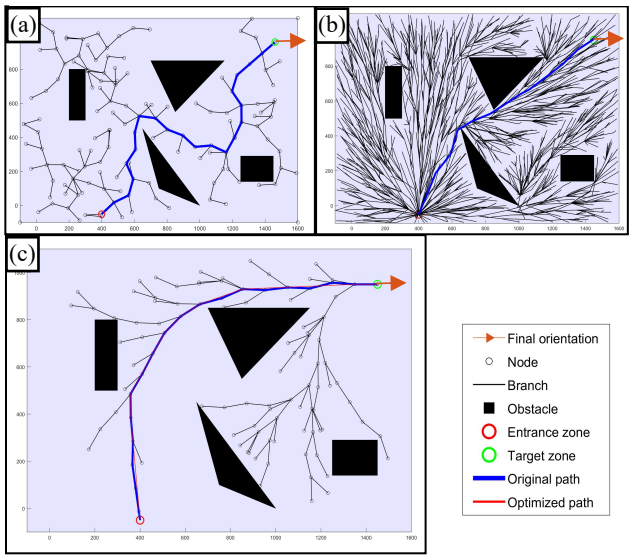

Fig. 7. The exploring tree and generated path of. (a) RRT; (b) RRT* (3000 nodes); (c) Sp-RRT.

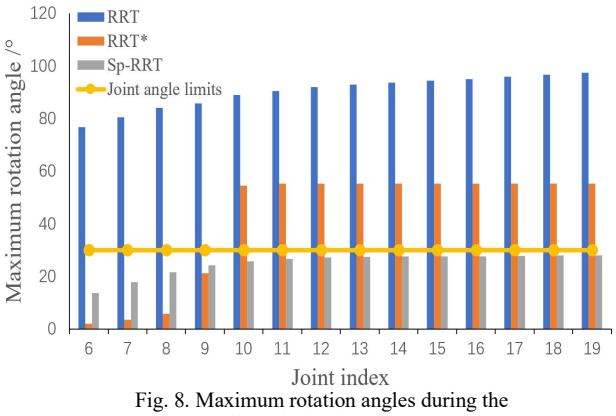

Fig. 8. Maximum rotation angles during the follow-the-leader motion.

and massive computational time remain to be solved. By contrast, Sp-RRT can find a path that meets all the requirements with an acceptable computational time.

B. Simultaneous Search Towards Multiple Entrances

By constructing the exploring tree from the target zone, Sp-RRT algorithm can expand to several entrances simultaneously. Paths from different entrances can share the exploring tree partly for higher efficiency.

Based on the scenario in Fig. 7, 3 extra entrance zones are added to construct a multi-entrance scenario. In this scenario, RRT and RRT* need to plan for each entrance independently, so the nodes and the computational time are the sum for each entrance. 100 contrast tests are performed for the multi-entrance scenario and the corresponding statistics are

TABLE II.
COMPARISON STATISTICS FOR MULTIPLE ENTRANCES.

	Nodes			Time /s		
	Single entrance n_1	Multiple entrances n_2	n_2/n_1	Singles entrance t_1	Multiple entrances t_2	t_2/t_1
RRT	87.32	296.55	340%	0.02	0.05	313%
RRT*	3000	12000	400%	35.48	140.89	397%
Sp-RRT	100.19	131.48	131%	0.30	0.37	125%

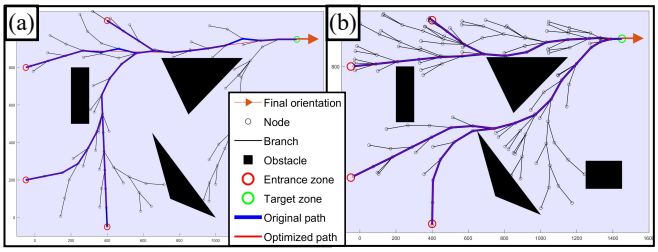

Fig. 9. The result of Sp-RRT in a multi-entrance scenario for a manipulator with. (a) identical segments; (b) more dexterous distal segments.

listed in Table II. One of the results of Sp-RRT algorithm is shown in Fig. 9 (a).

As shown in Table II, Sp-RRT algorithm consumes only 31.23% additional nodes and 24.55% additional time for the other 3 entrances compared with the results in Table I, while the additional consumptions of RRT and RRT* are quite more. Obviously, Sp-RRT has better adaptability for multi-entrance scenarios.

C. *Adaptability to Manipulator with Different Segments*

To enhance the capability to work in confined spaces, hyper-redundant manipulators may be designed with different segments, especially with more dexterous distal segments, which means shorter links and larger joint angle limits. Owing to the parametrical customization characteristic, path planning for those manipulators can also be performed by Sp-RRT algorithm.

For hyper-redundant manipulators with more dexterous distal segments, Sp-RRT can loosen expanding constraints at the beginning (neighbor area of the target zone) for better performance. Meanwhile, the followed ability of the whole path will remain because the less dexterous proximal part of the manipulator doesn't need to pass the partial path generated with loosen expanding constraints. It is worth mentioning that only when the path is searched reversely from the end, the constraints of links' length and joint angle limits can be realized conveniently.

In order to test the adaptability of Sp-RRT to hyper-redundant manipulators with more dexterous distal segments, the distal 3 links' length of the proposed planar manipulator is shortened to 50 mm, and the corresponding joint angle limit is increased to 45°.

The planned result in the multi-entrance scenario is shown in Fig. 9(b). Compared with Fig. 9(a), it can be concluded that Sp-RRT can take full advantage of the extra distal dexterity.

D. *Optimization Performance*

The spatial hyper-redundant manipulator proposed in Section II is used in this part. 100 tests of planning along with optimization are performed in the spatial multi-entrance scenario shown in Fig. 5. One of the test results is shown in Fig. 10.

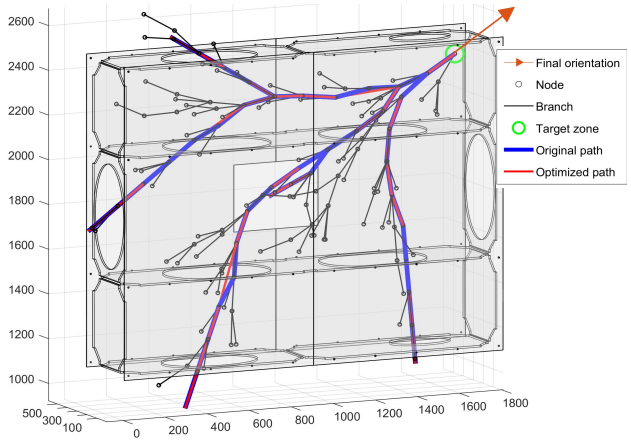

Fig. 10. Planning and optimization result of Sp-RRT in a 3D multi-entrance scenario.

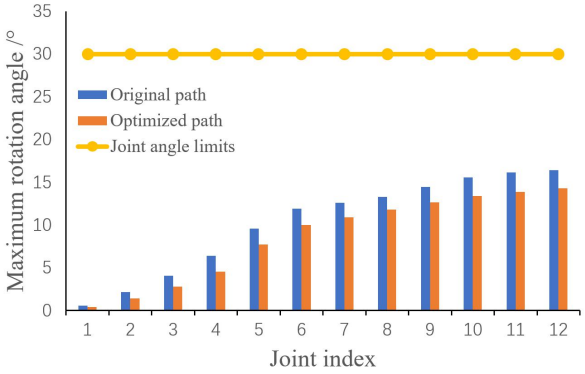

Fig. 11. Optimization performance.

To evaluate the performance of the optimization algorithm, the manipulator is used to follow the generated paths, and the recorded maximum rotation angles are shown in Fig. 11.

It can be observed that, the smoothness of the optimized paths is improved effectively benefiting from the decreased maximum rotation angles.

E. *Experimental Validation*

In this part, experimental validation is carried out on the proposed hyper-redundant manipulator system. In order to accomplish specific tasks in the airplane fuel tank model shown in Figure 5, Sp-RRT algorithm needs to plan several paths for multiple entrances firstly. After selecting the appropriate path from the results, the follow-the-leader motion is performed with the system, as shown in Fig. 12. The experimental results verify the followed ability of the path generated by Sp-RRT.

V. CONCLUSION

This paper presents the Sp-RRT approach for follow-the-leader motion of hyper-redundant manipulators. There are 4 characteristics: (1) The final pose of end-effector can be guaranteed;(2) Sp-RRT is adaptive to manipulators with different segments; (3) The exploring tree can expand towards multiple entrances simultaneously; (4) The generated paths can be followed easily.

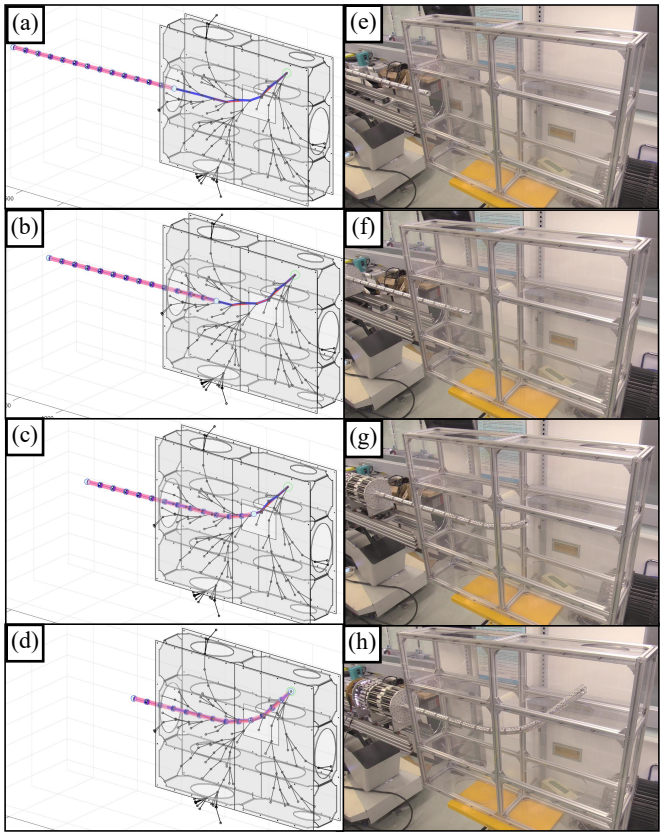

Fig. 12. Simulation and experimental validation of the follow-the-leader motion.

Experiments are performed with several manipulators in planar and spatial scenarios. Numerical experiments are conducted to demonstrate the aforementioned characteristics while the effectiveness is verified by the practical experiment.

All the links of manipulators concerned in this paper are rigid. For future works, extension of this method to continuum or soft manipulators should be considered.

REFERENCES

[1] R. Buckingham *et al.*, "Snake-arm robots: a new approach to aircraft assembly," 2008.

[2] S. Ma, S. Hirose, and H. Yoshinada, "Development of a hyper-redundant multijoint manipulator for maintenance of nuclear reactors," *Advanced Robotics,* vol. 9, no. 3, pp. 281-300, 1994.

[3] P. C. Giulianotti *et al.*, "Robotics in general surgery: personal experience in a large community hospital," *Arch Surg,* vol. 138, no. 7, pp. 777-784, 2003.

[4] P. Liljeback, O. Stavdahl, and A. Beitnes, "SnakeFighter - Development of a Water Hydraulic Fire Fighting Snake Robot," in *International Conference on Control*, 2006.

[5] V. Falkenhahn, F. A. Bender, A. Hildebrandt, R. Neumann, and O. Sawodny, "Online TCP trajectory planning for redundant continuum manipulators using quadratic programming," in *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2016.

[6] K. H. Yae, "Teleoperation of a Redundant Manipulator."

[7] V. C. Anderson and R. C. Horn, "Tensor arm manipulator," ed: Google Patents, 1970.

[8] L. Tang, J. Wang, Y. Zheng, G. Gu, L. Zhu, and X. Zhu, "Design of a cable-driven hyper-redundant robot with experimental validation," *International Journal of Advanced Robotic Systems,* vol. 14, no. 5, 2017, doi: 10.1177/1729881417734458.

[9] J. M. Ahuactzin, K. Gupta, and E. Mazer, "Manipulation Planning for Redundant Robots: A Practical Approach," *The International Journal of Robotics Research,* vol. 17, no. 7, pp. 731-747, 1998, doi: 10.1177/027836499801700704.

[10] D. Berenson, S. Srinivasa, and J. Kuffner, "Task Space Regions:A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research,* vol. 30, no. 12, pp. 1435-1460, 2011, doi: 10.1177/0278364910396389.

[11] Z. Zhang, G. Yang, and S. H. Yeo, "Inverse kinematics of modular Cable-driven Snake-like Robots with flexible backbones," in *IEEE 5th International Conference on Robotics, Automation and Mechatronics, RAM 2011, Qingdao, China, September 17-19, 2011*, 2011.

[12] T. F. Chan and R. V. Dubey, "A weighted least-norm solution based scheme for avoiding joint limits for redundant manipulators," in *IEEE International Conference on Robotics & Automation*, 2002.

[13] S. LaValle and J. Kuffner, "Randomized Kinodynamic Planning," *I. J. Robotic Res.,* vol. 20, pp. 378-400, 01/01 2001.

[14] M. Moll and L. E. Kavraki, "Path Planning for Deformable Linear Objects," *IEEE Transactions on Robotics,* vol. 22, no. 4, pp. p.625-636, 2006.

[15] H. Choset and W. Henning, "A Follow-The-Leader Approach To Serpentine Robot Motion Planning," *Journal of Aerospace Engineering,* vol. 12, no. 2, 1999.

[16] L. Tang, L. Zhu, X. Zhu, and G. Gu, "Confined spaces path following for cable-driven snake robots with prediction lookup and interpolation algorithms," *Science China Technological Sciences,* vol. 63, no. 2, pp. 255-264, 2020/02/01 2020, doi: 10.1007/s11431-019-1440-2.

[17] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[18] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research,* vol. 30, no. 7, pp. 846-894, 2011, doi: 10.1177/0278364911406761.

[19] A. C. Shkolnik and R. L. Tedrake, "Path planning in 1000+ dimensions using a task-space Voronoi bias," in *IEEE International Conference on Robotics & Automation*, 2009.

[20] B. An, J. Kim, and F. C. Park, "An Adaptive Stepsize RRT Planning Algorithm for Open-Chain Robots," *IEEE Robotics and Automation Letters,* vol. 3, no. 1, pp. 312-319, 2018, doi: 10.1109/LRA.2017.2745542.

[21] D. Lee and D. H. Shim, "Path planner based on bidirectional spline-RRT * for fixed-wing UAVs," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, 7-10 June 2016 2016, pp. 77-86, doi: 10.1109/ICUAS.2016.7502539.

[22] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Computer Communications,* vol. 149, pp. 270-299, 2020, doi: 10.1016/j.comcom.2019.10.014.

[23] L. Tang, J. Wang, Y. Zheng, G. Gu, L. Zhu, and X. Zhu, "Design of a cable-driven hyper-redundant robot with experimental validation," *International Journal of Advanced Robotic Systems,* vol. 14, no. 5, p. 172988141773445, 2017.

[24] Y. Zheng *et al.*, "Design and validation of cable-driven hyper-redundant manipulator with a closed-loop puller-follower controller," *Mechatronics,* vol. 78, p. 102605, 2021/10/01/ 2021, doi: https://doi.org/10.1016/j.mechatronics.2021.102605.

[25] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots,* vol. 34, no. 3, pp. 189-206, 2013.

[26] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. J. Teller, "Anytime Motion Planning using the RRT*," in *2011 IEEE International Conference on Robotics and Automation*, 2011.