RRT*-Based Path Planning for Continuum Arms

Brandon H. Meng, Isuru S. Godage, *Member, IEEE*, and Iyad Kanj

***Abstract*—Continuum arms are bio-inspired devices that exhibit continuous, smooth bending and generate motion through structural deformation. Rapidly-exploring random trees (RRT) is a traditional approach for performing efficient path planning. RRT approaches are usually based on exploring the configuration space (C-Space) of the robot to find a desirable work space (W-Space) path. Due to the complex kinematics and the highly non-linear mapping between the C-Space and W-Space of continuum arms, a high-quality path in the C-Space (e.g., a linear path) may not correspond to a desirable path/movement in the W-Space. Consequently, the C-Space RRT approaches that are based on C-Space cost functions do not lead to reliable and effective path planning when applied to continuum arms. In this letter, we propose a RRT* path planning approach for continuum arms that is based on exploring the W-Space of the robot as opposed to its C-Space. We show the successful applications of the proposed W-Space RRT* path planner in performing path planning with obstacle avoidance and in performing trajectory tracking. In all the aforementioned tasks, the quality of the paths generated by the proposed planner is superior to that of previous approaches and to its counterpart C-Space based RRT* approach, while the paths are generated in substantially less time.**

***Index Terms*—Motion and path planning, task and motion planning, flexible robotics.**

I. Introduction

CONTINUUM arms are bio-inspired manipulators [1] that exhibit continuous and smooth bending. They generate motion through structural deformation in a similar fashion to organic appendages such as elephant trunks and octopus arms. In this letter, we study the path planning problem for pneumatically-actuated human-scale continuum arms. Fig. 1 shows an example of the prototype of the multisection continuum arm under consideration in this letter. This arm is actuated by soft pneumatic muscle actuators (PMA) to facilitate smooth bending and to assume complex poses. Additionally, the inherent safety of soft material means that these continuum arms are well suited to serve as "co-robots"[2] in human-robot collaborative domains. Multisection continuum arms are constructed by serially attaching multiple continuum sections. Consequently, as each continuum section can bend in any plane, they can achieve a wide variety of spatial-shapes. Continuum arm-related research has seen rapid growth in recent years [3].

Manuscript received February 24, 2022; accepted April 20, 2022. Date of publication May 11, 2022; date of current version June 3, 2022. This letter was recommended for publication by Associate Editor O. Salzman and Editor H. Kurniawati upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation (NSF) and National Institute of Health (NIH). *(Corresponding author: Brandon H. Meng.)*

The authors are with the School of Computing, DePaul University, Chicago, IL 60604 USA (e-mail: bmeng1@depaul.edu; igodage@depaul.edu; ikanj@cs.depaul.edu).

Digital Object Identifier 10.1109/LRA.2022.3174257

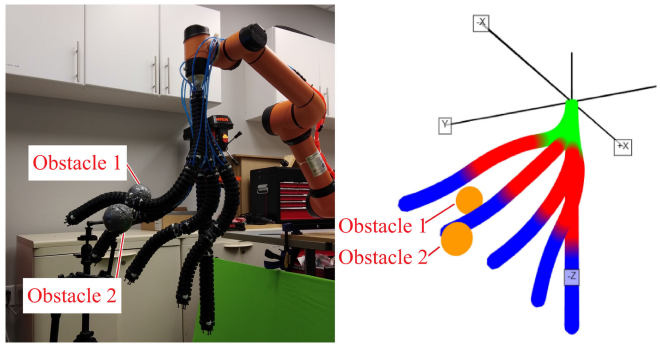

Fig. 1. Left: Five positions of a 3-section, pneumatically-actuated, continuum arm shown avoiding two obstacles. Right: The corresponding view in a simulated environment. The image on the left was generated by superimposing several static spatial-shapes.

Path planning is the problem of computing a trajectory for the robot to traverse while avoiding obstacles and obeying the physical constraints of the system. As reported in [4], the kinematics of continuum arms – the mapping between their configurations space (C-Space), or the corresponding joint-space, and work space (W-Space) – is highly nonlinear and complex. As a result, generated trajectories may include undesired or unnecessary movement that can quickly become unstable or infeasible to track when run on a prototype manipulator or a dynamic model thereof. Consequently, path planning for continuum arms is an ongoing research challenge.

A. Related Work

Inverse kinematics (IK) maps the W-Space to the C-Space and has been used extensively for path planning across a wide array of robotic models. A closed-form solution for the IK of the multisection continuum arms under consideration exists [5], but has major shortcomings. Notably, the approach discussed in [5] may not produce a solution, as the formulation does not take into account the geometrical constraints among the curve parameters. Additionally, this approach produces only one solution, though many solutions may be desirable for calculating a smooth trajectory. In [4], the authors discuss the shortcomings of the above approach and instead present an IK approach that relies on numerical techniques such as iterative optimization. However, this numerical approach, when used for path planning of continuum arms, leads to problems related to local minima, and the authors in [6] demonstrated its unreliability. Several graph-theoretic approaches were recently proposed to overcome the above-mentioned issues with the use of IK; these approaches relied on brute force (with an offline look-up table) [6] or refined

IK [7] to generate a very rich and dense set of configurations. While such approaches proved successful for performing path planning, they suffer significantly from extensive computation time. In all the aforementioned approaches [6], [7], finding a high-quality path may take several hours of computation time, and hence, cannot be useful for any real-time applications of path planning for multisection continuum arms. We mention that there exist planners that forgo the aforementioned IK approaches and instead rely on iterative, Jacobian-based methods to map the W-Space to the C-Space. For instance, the authors in [8] performed path planning with obstacle avoidance using repulsive forces in addition to the Jacobian-based method. Similarly, the authors in [9] used a Jacobian-based method to perform path planning with obstacle avoidance in very tight spaces.

Rapidly-exploring random trees (RRT) [10] is a traditional path planning approach that has been widely studied and applied to numerous robotic models. RRT is known for its simplicity, efficiency, and effectiveness, as random exploration allows for very efficient path planning with minimal computational requirements. In particular, the algorithm can be run entirely "online" with no offline storage or pre-computation needed. This algorithm is probabilistically complete, meaning that the probability of not finding a path decreases towards 0 as the number of samples heads towards infinity [11]. A notable variant of RRT, called RRT*, shares the benefits of RRT (notably probabilistic completeness) while proven to be asymptotically optimal [11], meaning that it converges to an optimal solution "almost surely" as the number of samples heads towards infinity. RRT has been extensively studied in a wide variety of domains since its introduction [12]–[15]. Moreover, it has been shown that RRT works well for redundant manipulators [16], [17].

There is a significant amount of work on RRT and Probabilistic Road Map (PRM) approaches for concentric tube robots, which is another notable type of continuum arms. For instance, the work reported in [18], [19] implemented a RRT approach for concentric tube robots and used precomputed roadmaps for rapid exploration. Additionally, the authors in [20] implemented a RRT* approach for a single section tendon-driven arm with 3 degrees of freedom (DoF) that rested on a mobile base. We note that there are major differences between concentric tube robots, tendon-driven continuum arms, and the continuum arm model under consideration. For example, concentric tube robots rely on tube translation (extension/contraction) operation, whereas tendon-driven continuum arms have "coupled" kinematics in the joint space due to the overlap of tendons actuating different sections. On the other hand, the continuum arm model under consideration has fixed-length, and individual continuum sections are kinematically independent in the joint-space. However, the C-Space, denoted by curve-parameters [4], for both tendon or PMA actuated fixed-length continuum arms are similar with decoupled kinematics. Thus, the path planning algorithm proposed herein can be generalized to tendon actuated continuum arms as well as any robotic manipulator with decoupled kinematics.

Many of the RRT (and PRM) approaches, including the above-mentioned ones for concentric tubes and tendon-driven arms are based on exploring the C-Space of the robot to find a desirable W-Space path (e.g., see [18], [19], [21]–[23]). Due to the complex kinematics and the highly non-linear mapping between the C-Space and W-Space of continuum arms, a high-quality path in the C-Space (e.g., a linear path) may not correspond to a desirable path/movement in the W-Space. Consequently, the RRT approaches that are based on C-Space cost functions do not lead to reliable and effective path planning when applied to continuum arms.

B. Contributions

We propose a W-Space based RRT* path planner for multisection continuum arms. The crux of our approach is a Jacobian-based method of configuration generation, allowing for W-Space search without the need for direct IK. This method may have further applications for other continuum manipulators as well. Additionally, goal-biasing and random variance (added through the use of the null space of the Jacobian) are employed to allow for rapid exploration while refining the quality of the generated paths. We demonstrate the success of our planner in path planning with obstacle avoidance and trajectory tracking and evaluate the planner efficiency and the quality of the generated paths. To assess the quality of a generated path, we use the same methodology from [7] that utilizes an accurate spatial dynamic model since real-time control for this prototype arm does not yet exist.

We show that the proposed planner generates superior-quality paths, while providing a substantial speedup over existing approaches with a more than 60% decrease in computation time over the approach in [6] while removing the need for offline computation. Furthermore, the proposed approach achieves a more than 90% decrease in computation time over the approach reported in [7]. Additionally, we demonstrate that our W-Space RRT* approach overcomes the shortcomings of a C-Space RRT* approach by implementing a C-Space RRT* path planner and comparing it to the W-Space RRT* planner. The W-Space RRT* path planner outperforms the C-Space RRT* one substantially, with improvements in both path quality and computation time.

II. System Model

A. Kinematic Model

The continuum arm to which we develop the proposed path planner is shown in Fig. 1. It has three sections – similar to prior works [6], [7] – and each section is actuated by 3 extension-mode, PMAs arranged around an inextensible rigid-linked chain (i.e., backbone) at $\frac{2\pi}{3}$ radians apart. Refer to [6] for more information about the manipulator design.

Without losing generality, consider any i^{th} section where $i \in \{1, 2, 3\}$. The differential length of PMAs – due to applied pressure – causes a section to bend in a circular arc shape. Unlike the continuum/soft section designs without backbones, such as [4], that require three curve parameters to describe the shape, only two parameters, namely $\phi_i \in [0, \pi]$, and $\theta_i \in [0, 2\pi)$, describe a section's bending angle subtended by the arc and bending plane angle, respectively (see Fig. 2). The curvature radius, λ_i, can be determined simply through the fixed-length L_i of a section and the curve parameter ϕ_i as $\lambda_i = \frac{L_i}{\phi_i}$. Thus,

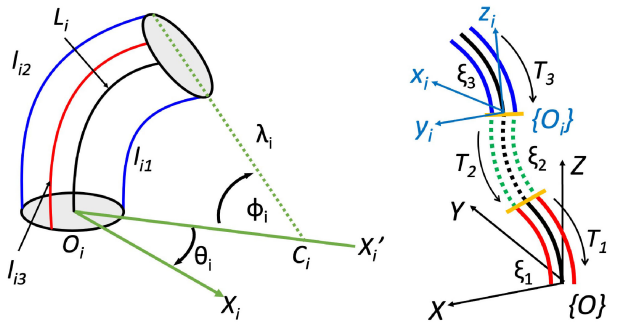

Fig. 2. Left: The curve parameters and their relationship to the arm's shape. Right: Schematic of a multisection continuum arm with 3 sections. Note that, as L_i is constant, $\lambda_i = \frac{L_i}{\phi_i}$.

in contrast to the designs reported in [4], only two length change variables are needed to describe the pose of a section. Additionally, the curve parameters have a direct mapping to PMA length changes [6]. In this model, the curve parameters make up the C-Space, and the length changes constitute the joint space. For length changes l_{ij} where $i \in \{1, 2, 3\}$ is the section number and $j \in \{1, 2\}$ is an actuator number, the curve parameters are derived as $\theta_i = \arctan(l_{i2}\sqrt{3}, 2l_{i1} - l_{i2})$, and $\phi_i = (2\sqrt{l_{i1}^2 - l_{i1}l_{i2} + l_{i2}^2})/(r_i\sqrt{3})$ where r_i is the radius of a section. Though there are 3 actuators in each section, the inclusion of the backbone introduces an over-constrained system such that the length of l_3 can be described by the lengths of l_1 and l_2. Due to this, l_3 can be disregarded [6]. Given this formulation, a configuration of the arm can be described using curve parameters as $c = (\phi_1, \theta_1, \phi_2, \theta_2, \phi_3, \theta_3)$.

As we have a one-to-one mapping between the length parameters and curve parameters, we will use curve parameters to formulate the planning algorithm in this work. Using curve parameters instead of length changes to describe the arm is desirable as they provide a more intuitive explanation of the arm's shape. Additionally, length changes are subject to constraints (as described in [6]), whereas curve parameters are only bounded. A homogeneous transformation matrix (HTM) can be constructed using a configuration for forward kinematics. Given the fixed length of a section L_i, the HTM $\mathbf{T}$ is constructed as $\mathbf{T} = \mathbf{T}_1\mathbf{T}_2\mathbf{T}_3$ where $\mathbf{T}_i = \mathbf{R}_Z(\theta_i)\mathbf{P}_X(L_i/\phi_i)\mathbf{R}_Y(\xi_i\phi_i)\mathbf{P}_X(-L_i/\phi_i)\mathbf{R}_Z(\theta_i)$. We denote by $\mathbf{P}_X$ the translation along the $+X$ axis and by $\mathbf{R}_Z$ and $\mathbf{R}_Y$ the rotations about the $+Z$ and $+Y$ axes, respectively. The parameter $\xi_i \in [0, 1]$ denotes the position/displacement of a point on the i-th section of the continuum arm (with respect to the base of the i-th section). This HTM can determine the coordinates of points along the arm, including the tip. Further information about the robotic model and its kinematics can be found in [6].

III. Preliminaries

A. RRT and RRT*

Rapidly-exploring random tree (RRT) [10] uses random sampling to explore the C-space of a robot. Starting from an initial node, corresponding to the starting point/configuration of the robot in this space, a "tree" $\mathcal{T}$ of nodes is grown in which nodes are inserted so that they are within some parameter distance ϵ from their parent nodes. For a given node n and target node t in $\mathcal{T}$, *steering* (at n) is the process of creating a new node n' that is on the line nt and within distance ϵ from n.

RRT*[11] is a variant of RRT that employs a cost measure to indicate the quality of the tree $\mathcal{T}$ and distinguish between a low-quality and high-quality path while *refining* the tree during exploration. The *cost* between two nodes is a metric specifying the cost of transitioning between these two nodes. The cost between two nodes $n, m \in \mathcal{T}$ is denoted $\zeta(n, m)$. For a node n in $\mathcal{T}$, let $P = (n_s = n_0, n_1, \ldots, n_k = n)$ be the unique path in $\mathcal{T}$ from the starting node n_s to n. We define the *path cost* of n as

$$\upsilon(n) = \sum_{i=0}^{k-1} \zeta(n_i, n_{i+1}). \tag{1}$$

To refine the tree, upon inserting a new node n', we check to see if n' "opens up" shorter paths to some existing nodes in $\mathcal{T}$. This process of examining the vicinity or *neighborhood* of a newly-inserted node and changing edges in the tree is called *rewiring*. The neighborhood is defined by a distance $\alpha \geq 0$. More formally, when a node n' is inserted in $\mathcal{T}$, if some node m within distance α of n' satisfies $\zeta(n,' m) + \upsilon(n') < \upsilon(m)$, then a shorter path exists to m (via n'). In this case, we replace the parent of m in $\mathcal{T}$ with n', and update the cost of m to reflect this change.

IV. W-Space RRT*

RRT* is a widely-used algorithm for path planning that is often employed to explore the C-Space of a robot. A C-Space based RRT* implementation for continuum arms faces several challenges, primarily due to their complex and highly non-linear kinematics. For instance, a linear trajectory in the C-Space may correspond to a highly-undesirable, non-linear path in the W-Space. Additionally, reorientation, when the arm makes large shifts in the pose without progressing along the path, can occur when two unrelated configurations occur sequentially in a path. This poses inherent hindrances for a C-Space-based implementation of RRT* that is capable of achieving high-quality path planning. Therefore, for continuum arms, an RRT* implementation should preferably be based on W-Space exploration, especially when obstacles are present in the environment.

Generating a mapping from the W-Space to the C-Space of continuum arms is a challenging task. For instance, a straightforward mapping of W-Space points to C-Space points by separate invocation of IK does not work, as separate invocations may produce uncoordinated spatial shapes; this issue can create an undesired movement for the continuum arm. There exist schemes for W-Space to C-Space mappings based on Jacobians that do not require the use of IK [4], [17]. Though this approach may also run into issues of convergence, it is possible to overcome these issues using randomness.

In this letter, we overcome the above challenges and present a RRT* algorithm for continuum arms that is based on W-Space exploration. This algorithm uses a Jacobian-based method to iteratively estimate the IK solutions, along with considerations

for hyper-redundancy, to make tree growth easier. Variance is added to the Jacobian-based method to avoid "getting stuck" in complex areas of the W-Space and C-Space.

To start, a tree is grown outward from a starting configuration. Candidate nodes are created by sampling the W-Space, steering and then using the Jacobian-based method to find suitable corresponding configurations. On some iterations, the target point is considered in the formulation, instead of a randomly sampled point, to bias the tree growth towards the target. Nodes that are found to collide with obstacles are discarded, and a queue of unsuccessful nodes is kept to avoid unnecessary exploration of unpromising tree branches. The procedure continues until the tree reaches the target point, and a corresponding configuration of the target point is computed. We now proceed to the details.

A. W-Space - C-Space Projection Via Jacobians

This method uses C-Space and W-Space displacements and velocities with their corresponding Jacobian to find a new configuration using a starting configuration and a desired change in tip position. Using forward kinematics, the continuum arm tip position, $\boldsymbol{p}$, can be extracted from $\mathbf{T}(\boldsymbol{q}) = [\mathbf{R}(\boldsymbol{q})\ \boldsymbol{p}(\boldsymbol{q}); 0\ 1]$ where $\boldsymbol{q}$ is a configuration and $\mathbf{T}$ is the HTM mapping the C-Space to the W-Space.The tip velocity can be described as $\dot{\boldsymbol{p}} = \mathbf{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$ where $\dot{\boldsymbol{q}}$ is configuration velocity.Given a change in configuration $\delta\boldsymbol{q}$, we can approximate a change in the position vector of the tip of the arm as $\delta\boldsymbol{p} \approx \mathbf{J}(\boldsymbol{q})\delta\boldsymbol{q}$ where $\mathbf{J}$ is the Jacobian of $\boldsymbol{p}$. Similarly, we can approximate a change in configuration $\delta\boldsymbol{q}$ as

$$\delta\boldsymbol{q} \approx \mathbf{J}_w(\boldsymbol{q})^{\dagger}\delta\boldsymbol{p}, \tag{2}$$

where $\mathbf{J}_w(\boldsymbol{q})$ is a weighted Jacobian [24] that ensures the validity of configurations generated using this method and $[\,]^{\dagger}$ denotes the Moore-Penrose pseudo-inverse operator. As $\delta\boldsymbol{p}$ increases, so does the error in this approximation. Using (2), we can use a small change in tip position to calculate a new configuration $\boldsymbol{q}' = \boldsymbol{q} + \delta\boldsymbol{q}$ as

$$\boldsymbol{q} + \delta\boldsymbol{q} \approx \boldsymbol{q} + \mathbf{J}_w(\boldsymbol{q})^{\dagger}\delta\boldsymbol{p}. \tag{3}$$

Thus, the new configuration vector will be adjacent to the current configuration $\boldsymbol{q}$. To use this formulation, an initial configuration q with corresponding tip $\boldsymbol{p}$ must first be selected. Then, a target point $\hat{\boldsymbol{p}}$ is also selected, and $\delta\boldsymbol{p}$ is calculated as $\delta\boldsymbol{p} = \hat{\boldsymbol{p}} - \boldsymbol{p}$. These values are plugged into (3), and we find a new configuration $\boldsymbol{q}' = \boldsymbol{q} + \delta\boldsymbol{q}$ that places the tip of the arm at $\hat{\boldsymbol{p}}$. We refer to this process as the Jacobian configuration-generation method (JCG).

B. Null Space

In the context of the continuum arm, the null space can correspond to the redundancy of the arm, or rather to the configurations that produce the same result at the tip. We can exploit the null space to modify the position/spatial shape of the intermediate sections of the arm while still producing the desired change in tip position.

We can describe the null space of the aforementioned Jacobian as $\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}$, where $\mathbf{I}$ is the rank 3 identity matrix. With the consideration of the null space, variance can be added to (3) using random configuration velocities. We start with the initial configuration $\boldsymbol{q}$. First, a random configuration $\boldsymbol{k}$ is picked from the C-Space. Then, $\boldsymbol{k}$ is used to create a "random" configuration velocity. A configuration $\boldsymbol{k}'$ is found that is within some small distance from $\boldsymbol{q}$ on the vector $\overrightarrow{qk}$. Next, a random C-Space velocity μ is found by calculating $\boldsymbol{\mu} = \boldsymbol{k}' - \boldsymbol{q}$. Lastly, the null space $\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}$ is multiplied by $\boldsymbol{\mu}$.

We apply $\boldsymbol{\mu}$ to (3) and obtain the following approximation

$$\boldsymbol{q} + \delta\boldsymbol{q} \approx \boldsymbol{q} + \mathbf{J}_w^{\dagger}\delta\boldsymbol{p} + (\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J})\boldsymbol{\mu}. \tag{4}$$

C. W-Space Exploration

Exploration of the W-Space space only occurs using viable C-Space points that correspond to the tip of the continuum arm. This ensures the tree contains only feasible nodes. The exploration is performed using a tree of nodes $\mathcal{T}$ where each node represents a valid C-Space and W-Space vector pairs.

A tree is grown outward from some initial node n_s with corresponding configuration $\boldsymbol{s}$. To perform exploration, we start by randomly selecting a point $\boldsymbol{r}$ in the W-Space. A node n is found in the tree whose tip point has the shortest distance to $\boldsymbol{r}$. We then use the steering process to find a point on the vector $\overrightarrow{nr}$ at distance δ from n; a new node n' is inserted into $\mathcal{T}$ that corresponds to that point. Using the difference between n' and n and the configuration $\boldsymbol{c}$ of n, we apply the JCG method to find a configuration $\boldsymbol{c}'$ corresponding to n'.

Given that the JCG method computes an approximate solution, it may be that the distance between n' and the W-Space point corresponding to $\boldsymbol{c}'$ is large. If this distance is larger than a certain threshold (specified based on the simulation results), we say that the invocation of the JCG method "failed" and deem the configuration $\boldsymbol{c}'$ "invalid." Given the aforementioned use of random null space vectors, it may be that the JCG method both succeeds and fails for the same input on different invocations. Assuming that the configuration is acceptable, we compute and update n' using $\boldsymbol{c}'$. Additionally, the distance δ is selected to be small such that the error produced by the JCG method is also small.

During the rewiring process, a node m may be rewired so that a new node n' becomes the parent of m. Therein, we calculate a new configuration for m (using the JCG method) between n' and m. If the JCG method fails and the distance between the current and new tip position of m is large, m is not updated. We explore the W-Space with the goal of finding a target point $\boldsymbol{t}$ and inserting it into $\mathcal{T}$. When a node n' is inserted into $\mathcal{T}$, the Euclidean distance between $\boldsymbol{t}$ and n' is calculated. If this distance is within some small acceptable margin, n' is marked as the target node n_t. There now exists a path from the initial node n_s to the target node n_t in $\mathcal{T}$.

Cost: To maximize the quality of a generated path, movement at the tip must proceed smoothly while minimizing unnecessary movement in the intermediate sections of the arm. The cost $\zeta(m, n)$ between two nodes $m, n \in \mathcal{T}$ is defined as the Euclidean distance between their corresponding arm tip points. Note that this does not take into account movement in the intermediate sections of the arm.

Goal-Biasing: The growth of the tree can be directed towards the target using *Goal-biasing*. In this scheme, random sampling occurs every $\omega - 1$ iterations of RRT*. However, in the ω^{th} iteration, the random node is replaced with the target node/point. In some applications where quality paths are prioritized over exploration, randomness can be reduced by decreasing ω. We refer to the process of selecting the target instead of a random node as *guiding*, and the ω^{th} iteration is called a *guiding iteration*. If a guiding iteration using the target point t and a node n that is closest to t fails to produce a configuration (due to obstacle collision, local minima, etc.), then no additional node is added. In a future iteration, it may be possible that the same node n is attempted again if no other node is closer to t, meaning that if the tree has not grown in the direction of the target, the algorithm could become stuck. Although repeated invocations of the JCG method can produce different results (due to the null space variance), it is possible that there is no feasible configuration, and n would be repeatedly attempted with no forward progress/growth. To avoid this scenario, a queue is created that has a size equaling some fixed percentage of $\mathcal{T}$'s size. Any node in this queue cannot be used during a guiding iteration. This encourages guided tree growth in other sections of the tree.

Starting Configurations: Since the JCG method relies on an initial configuration to produce a result, additional starting configurations or *auxiliary nodes* can be added to increase the diversity of the configurations included in the tree. This increased diversity can be useful in certain domains, like obstacle avoidance, where having a wide variety of spatial-shapes is important. These nodes are generally selected so that their corresponding tip points inhabit different areas of the W-Space.

Obstacles: Obstacles may or may not be present in the W-Space during path planning. If obstacles are present, collisions can be checked before the node is inserted into the tree. Using the configuration corresponding to a node n' and the HTM, points along the intermediate sections of the arm can be calculated. Each of these points is checked for collision with obstacles. If such a collision exists, n' is not inserted into the tree. As a consequence of choosing the step size δ to be small (to minimize the error produced by the JCG method), the tree grows slowly with small gaps between nodes. With small changes in tip position, collision detection is performed at very small intervals.

D. The RRT* Algorithm

The algorithm (see Alg. 1) starts with an initial configuration $\boldsymbol{s}$, a target tip position $\boldsymbol{t}$, and the minimum number of iterations i. A tree is created, and the starting node n_s and auxiliary starting nodes are inserted (lines 1–2). Then in line 3, the expansion of the tree begins with goal-biasing occurring every ω iterations. During guided iterations (lines 4–11), t is selected as the target point. The node n with the smallest Euclidean distance between n's tip and $\boldsymbol{t}$ is selected (line 5). The JCG method is used with n to create a new node n' assuming this node is valid and does not collide with obstacles (lines 7–8). The rewire routine is invoked (line 9). During non-guided iterations (lines 12–20), a random W-Space point r is sampled. Next, the node n with the smallest Euclidean distance between n's tip and $\boldsymbol{r}$ is found (line 14). The tip of n is steered towards $\boldsymbol{r}$ then the JCG method is invoked with n to create a new node n' assuming the node is valid and does not collide with obstacles (lines 16–17). After a node is added to the tree, the rewire subroutine is invoked (line 18).

Algorithm 1: WRRT*.

Input: Initial configuration s; target point t; minimum number of iterations i
Output: A sequence of configurations as a path P
1 Create tree $\mathcal{T}$ using s and t;
2 Create auxiliary starting-configurations and add to $\mathcal{T}$;
3 **while** t not in $\mathcal{T}$ and number of iterations $< i$ **do**
4 **if** guiding iteration **then**
5 Find a node n nearest to t;
6 Create a new node n' by steering from n to t;
7 **if** n' is valid and is obstacle-free **then**
8 Calculate a configuration for n' using the JCG method;
9 Insert n' into $\mathcal{T}$ and rewire;
10 **else**
11 Generate a random W-Space point r;
12 Find a node n nearest to r;
13 Create a new node n' by steering from n to r;
14 **if** n' is valid and is obstacle-free **then**
15 Calculate a configuration for n' using the JCG method;
16 Insert n' into $\mathcal{T}$ and rewire;
17 Check to see if t has been reached;
18 Generate a path P by following $\mathcal{T}$ from target;
19 **return** P;

V. C-Space RRT*

As another basis for comparison with other relevant approaches, a C-Space RRT* approach for the continuum arm model under consideration was implemented. The C-Space implementation follows to a large extent the description of the W-Space RRT* implementation (i.e., in terms of steering, goal-biasing, etc.), except that the tree growth occurs in the C-Space (and hence, there is no need for the invocation of the JCG method). Note that nodes in the tree $\mathcal{T}$ will still need to be mapped to the W-Space to perform tasks such as obstacle detection. Once a path to a target node in $\mathcal{T}$ is found, no additional mapping is required as the corresponding sequence of configurations can simply be outputted.

Although C-Space RRT/RRT* approaches are the standard for many robotic models, there are model-specific challenges that prevent a C-Space RRT* approach for the model under consideration from generating a high-quality path as alluded to in Section IV. Since path planning requires specifying a target end-effector position, IK must be employed to generate the corresponding configuration. While this behaves well in free path planning, it becomes a serious issue in trajectory tracking. In this task, the arm tip must stop a series of intermediate points or *stops* to have the effect of "tracking." To accomplish this task,

multiple invocations of IK are needed to generate configurations corresponding to each of the stops. These invocations are performed with random starting configurations as chaining together IK invocations leads to the knotting problem alluded to earlier. This leads to an uncoordinated movement along the trajectory, resulting in poor-quality paths, as shown by our simulation results in Section VI.

Some considerations are made to mitigate the above-mentioned issues in the implementation. First, The cost $\zeta(m, n)$ is updated so that, between two nodes $m, n \in \mathcal{T}$, it is defined as the Euclidean distance between their corresponding configurations. Second, to address the issue of uncoordinated movement along the trajectory, multiple (as opposed to one) configurations corresponding to each intermediate trajectory point were generated, and the best configuration (w.r.t. the Euclidean distance from the previous point) is chosen.

VI. Methodology and Results

A. Other Approaches

We compare our W-Space RRT* to the Look-up Table and the IK-based approaches introduced in [6] and [7], respectively, as well as the C-Space RRT* approach described in Section V. All these approaches were designed specifically for the continuum arm model under consideration. We summarize the approaches in [6], [7] briefly for the sake of comparison.

The approaches in [6], [7] are graph theoretic and rely on a decomposition of the W-Space. The Look-up Table approach works by first computing a discretization of the C-Space, calculating the corresponding tip points, and then storing the configurations and points in a look-up table. Next, the W-Space is discretized into a grid of cubes, and a path is found in the grid. Then, configurations are mapped to cubes if the corresponding tip position falls inside the cube. Lastly, a layered graph is constructed where each layer consists of all of the configurations in one of the cubes in the path of cubes. The letter in [7] introduced two IK-based approaches that had very similar results; we only discuss/compare one of them for the sake of brevity and refer to it as the Refined-IK approach. The Refined-IK approach is similar to the Look-up Table approach [6], but trades the look-up table for real-time invocation of IK. A cube is "populated" with configurations using repeated invocations of IK. Refined-IK specifically uses the configurations from one cube as a starting point for additional invocations of IK. Both approaches suffer from runtime issues. The Look-up Table approach requires the generation of an offline look-up table which, depending on the density, can take more than 30 minutes to compute. The Refined-IK approach, on the other hand, requires repeated calls to IK, which requires substantial computation time.

B. Test Framework

We extend the continuum arm dynamic model developed in [25] to model the inextensible continuum manipulator dynamics. The model was implemented on the Matlab Simulink environment with jointspace PI controller for each DoF (proportional gain, $P = 7$ and integral grain, $I = 420$). Here, the dynamic model incorporates the PMA dynamics [26], where input pressure controls the length variation of PMAs. Thus, in the controller design, we employed joint value saturation limits ($0 - 6$ bars to imitate the 6-bar input pressure limitation), joint slew rate (from experimental data showing ± 1 bar/sec), and 25 ms delay to the sensor feedback loop.

In order to verify the quality of paths generated using our algorithm, we use the dynamic simulations on the trajectories generated by path planners to assess their quality. This simulator tracks the arm under realistic dynamics. To judge the quality of a path, we expand or compress the execution time of the path until the dynamic simulator is able to track the arm with less than 0.01 cm root mean squared error at the tip. If two paths are otherwise identical, a shorter time would indicate higher quality. The dynamic simulator requires that a time be associated with every transition between configurations. We employ the method discussed in [7] to compute the path times. We refer to the time outputted by the dynamic simulator as *DynSim-time* and use it to measure the quality of a path.

C. Comparison of Approaches

Each of the algorithms is evaluated using two categories of tests. They will be evaluated by both their computation time (the amount of time required to compute a solution) and DynSim-time (the output of the dynamic simulator). In some of these tests, the arm will start in a "resting" position. The *resting configuration* or *resting position* corresponds to the configuration (0,0,0,0,0,0) where the arm has a neutral shape and is not bent or rotated. The corresponding arm tip position is $(0, 0, 0.45)$ m. Note that the feasible W-Space of our robotic model is in the range of $[-0.30, 0.30]$, $[-0.30, 0.30]$, $[-0.10, 0.45]$ m for the x, y, and z coordinates, respectively

The first category of tests evaluates obstacle avoidance. In these scenarios, the respective algorithms will be tasked with finding a path from the initial resting configuration to a point that is selected randomly for each scenario and is within the feasible W-Space. In addition, spherical obstacles will be placed randomly in the W-Space. We report the mean computation time needed by each algorithm to produce a solution to each of the trials. Additionally, the mean DynSim-time and its variance σ^2 are reported.

The second category of tests is for trajectory tracking. A collection of challenging shapes are selected and then represented as a trajectory in the W-Space. Each algorithm was used to produce a path that traces the prescribed trajectory. The computation time needed to produce a solution will be recorded alongside the DynSim-time. Additionally, given the randomness inherent to RRT*, the two RRT*-based approaches will be run 100 times on on each trajectory. The mean computation time and DynSim-time will be reported alongside the variance σ^2 of the DynSim-time. Throughout the tables of results, we refer to the Look-up Table approach, Refined-Ik approach, C-Space RRT* approach, and W-Space RRT* approach as **LT**, **REF-IK**, **C-RRT***, and **W-RRT***, respectively.

The algorithms discussed above were implemented and tested on a 64-bit Windows 10 PC using an Intel i7-9700 K 8 core 3.60 GHz CPU with 32 GB of RAM. The Look-up Table and

TABLE I
SUMMARY OF COMPUTATION TIME FOR OBSTACLE AVOIDANCE†

Approach	DynSim-Time [s]	σ^2	Computation-time [s]
C-RRT*	40.98	441.53	607.8
W-RRT*	13.11	43.62	50.61

†For each approach, the mean DynSim-time, the corresponding variance σ^2, and the mean computation time for 100 random trials in seconds.

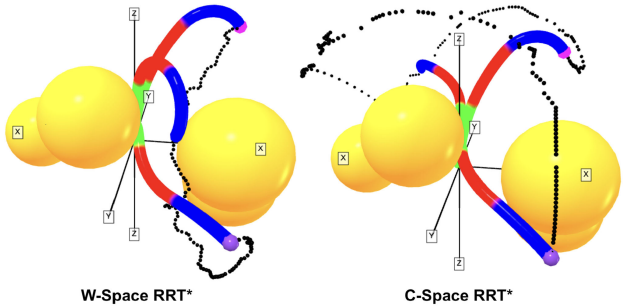

Fig. 3. A comparison of a trajectory produced by the W-Space RRT* (left) and C-Space RRT* (right) for the obstacle avoidance task, respectively. The target point is shown in purple.

Refined-IK approaches are implemented following the description in [7].

D. Category 1: Obstacle Avoidance

A group of 100 random trials was performed in which the algorithms were tasked to produce a path from a random starting configuration (chosen from within the feasible C-Space) to a random target point (from within the feasible W-Space). Obstacles were represented as spheres. Six spheres of random radius $[5, 12]$ were placed randomly in the W-Space. Note that for the RRT*-based approaches, the amount of random sampling dictated by ω is set at 10. Table I shows the results from these trials. Note that both approaches successfully produced a solution to each scenario. The mean DynSim-time, the corresponding variance, and the mean computation times are shown. Note that the W-Space RRT* approach has much lower DynSim-times, indicating that these paths are of higher quality. Additionally, the computation time is lower for these cases. Though the C-Space RRT* approach is capable of finding a solution, the undesired ancillary movement introduced by the C-Space exploration causes the quality to be much lower. Additionally, since this ancillary movement can cause the path to be longer, the C-Space RRT* trajectories take a longer amount of computation time to produce.

An example is given in Fig. 3 of the W-Space and C-Space RRT* approaches performing the obstacle avoidance task. The path produced by the C-Space approach in this example is nearly linear when plotted with respect to time. However, this path, when mapped back to the W-Space, corresponds to undesirable behavior that introduces a lot of unnecessary movement. In fact, many C-Space RRT* simulation results exhibited far worse behavior than the example reported above.

TABLE II
SUMMARY OF DYNSIM-TIME RESULTS FOR SMALL SHAPES (ABOVE LINE) AND LARGE SHAPES (BELOW LINE).†

Approach	LT	REF-IK	C-RRT*	σ^2	W-RRT*	σ^2
Circle	13.8	11.0	39.60	1607	9.9	1.1
Figure-8	20.8	17.0	97.40	276.6	16.1	7.1
Rot Cube	110.1	86.5	516.35	4354.97	71.5	106.5
Circle	Failure	57.71	191.70	1339	51.9	5.3
Figure 8	81.61	53.71	503.5	$7.1e^3$	51.0	2.6
Rot Cube	245.34	116.78	941.5	$1.8e^4$	118.3	295

†All numbers represent the DynSim-time in seconds. Variance is included for the RRT* approaches.

TABLE III
SUMMARY OF COMPUTATION TIME RESULTS FOR SMALL SHAPES (ABOVE LINE) AND LARGE SHAPES (BELOW LINE).†

Approach	LT	REF-IK	C-RRT*	W-RRT*
Circle	549.02	22450	27.31	14.26
Figure-8	255	22809	21.3	19.8
Rot Cube	2414	103914	126.6	98.3
Circle	N/A	79286	171.95	180.6
Figure 8	218.47	52966	66.3	180.8
Rot Cube	923.03	83927	316.6	191.75

†All numbers represent computation time in seconds.

E. Category 2: Trajectory Tracking

We select a collection of "demonstrative" shapes that are challenging for the arm to trace to demonstrate trajectory tracking. These are circles, figure-8 shapes (the shape resembling the infinity symbol), and cubes that have been rotated. Versions that are both large and small are generated. For example, the "small circle" is located at $z = 0.15\ m$ and has a radius of $0.1\ m$, whereas the "large circle" is located at $z = 0.25\ m$ and has a radius of $0.345\ m$. These shapes are passed as trajectories to each of the algorithms. Note that for the RRT*-based approaches, the amount of random sampling dictated by ω is set at 3.

Table II shows the DynSim-times for each of the approaches on all of the shapes. Table III shows the computation times for each of the methods for each of the shapes. The DynSim-times and computation times for the two RRT*-based approaches are averages after simulating each scenario 100 times. The average DynSim-time is reported alongside the variance σ^2. The "Failure" in the table indicates that the Look-up Table approach was not able to complete the task, as some of the cubes did not have any configurations, and therefore a path could not be produced. Fig. 4 shows a comparison between the approaches tracking a rotated cube.

The Look-up Table, Refined-IK, and W-Space RRT* approaches have similar quality, though our proposed W-Space RRT* approach has the best overall. However, the W-Space RRT* approach substantially improves computation time over the other two methods. The numbers highlighted in the table for the small circle show that W-Space RRT* is a 95% decrease in computation time over the next best result while also having a 26% reduction in DynSim-time relative to Look-up Table. Note also the inferior performance of the C-Space RRT* approach. As discussed, the reorientation required from this approach creates low-quality paths, which is reflected in the DynSim-times. Additionally, the computation time is surprisingly high for C-Space RRT*. This is due to the numerous invocation of IK discussed

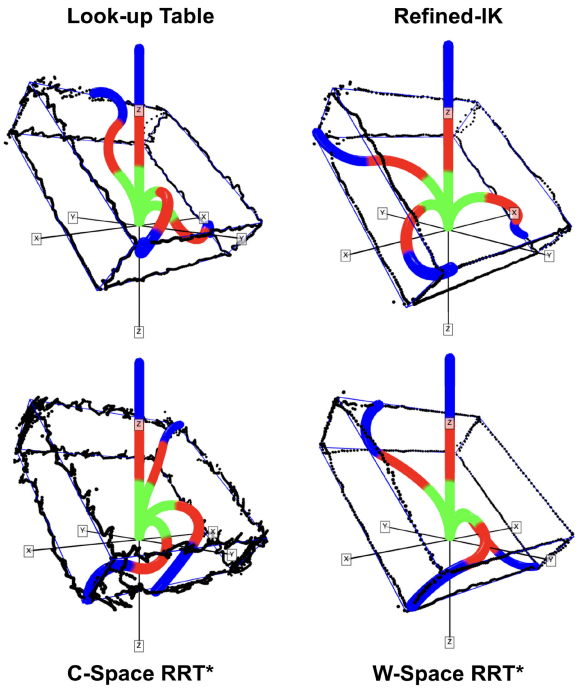

Fig. 4. A comparison of a trajectory traced by the Look-up Table (top left), the Refined-IK (top right), C-Space RRT* (bottom left), and W-Space RRT* (bottom right).

in Section V. Though the C-Space RRT* approach does have a better computation time than the W-Space RRT* approach for the large circle, the difference in DynSim-time between the two approaches is immense, with the W-Space RRT* approach producing a much higher quality path. Please see the associated video for examples of the four approaches performing trajectory tracking.

VII. Conclusion

We introduced a W-Space version of RRT* that was able to perform path planning for multisection continuum arms. Our approach is based on a Jacobian-based projection method of configuration generation to perform W-Space exploration. We demonstrated the effectiveness of the proposed planner in performing path planning tasks, including simple path planning, path planning with obstacle avoidance, and trajectory tracking. Additionally, we evaluated the paths generated via the proposed planner using a dynamic simulator and verified the quality of the trajectories generated in this manner. We demonstrated that our approach is able to produce high-quality paths with significantly less computation time than other approaches. Future work will focus on further improving the speed of RRT* through parallelization while extending it to additional applications, such as dynamic obstacle avoidance, anticipatory path planning, and/or multi-robot environments.

References

[1] B. Jones and I. Walker, "Kinematics for multisection continuum robots," *IEEE Trans. Robot.*, vol. 22, no. 1, pp. 43–55, Feb. 2006.

[2] E. S. Boy, E. Burdet, C. L. Teo, and J. E. Colgate, "The learning cobot," *ser. ASME Int. Mech. Engi. Congr. Expo., Dynamic Systems Control*, vol. 11, pp. 867–873, 2002.

[3] S. Kolachalama and S. Lakshmanan, "Continuum robots for manipulation applications: A survey," *J. Robot.*, vol. 2020, pp. 1–19, 2020.

[4] I. S. Godage, G. Medrano-Cerda, D. Branson, E. Guglielmino, and D. Caldwell, "Modal kinematics for multisection continuum arms," *Bioinspiration Biomimetics*, vol. 10, 2015, Art. no. 035002.

[5] S. Neppalli, M. A. Csencsits, B. A. Jones, and I. D. Walker, "Closed-form inverse kinematics for continuum manipulators," *Adv. Robot.*, vol. 23, no. 15, pp. 2077–2091, 2009. [Online]. Available: https://doi.org/10.1163/016918609X12529299964101

[6] J. Deng, B. H. Meng, I. Kanj, and I. S. Godage, "Near-optimal smooth path planning for multisection continuum arms," in *Proc. IEEE Int. Conf. Soft Robot.*, pp. 416–421, 2019.

[7] B. H. Meng, I. S. Godage, and I. Kanj, "Smooth path planning for continuum arms," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 7809–7814.

[8] I. S. Godage, D. T. Branson, E. Guglielmino, and D. G. Caldwell, "Path planning for multisection continuum arms," in *Proc. IEEE Int. Conf. Mechatronics Automat.*, 2012, pp. 1208–1213.

[9] J. Zhang, K. Lu, J. Yuan, J. di, and G. He, "Kinematics modeling and motion planning of tendon driven continuum manipulators," *J. Artif. Intell. Technol.*, vol. 1, pp. 28–36, 2021.

[10] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1999, pp. 473–479.

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

[12] X. Zhao, Z. Cao, W. Geng, Y. Yu, M. Tan, and X. Chen, "Path planning of manipulator based on RRT-connect and bezier curve," in *Proc. IEEE 9th Annu. Int. Conf. CYBER Technol. Automat. Control Intell. Syst.*, 2019, pp. 649–653.

[13] J. Xu, Z. Tian, W. He, and Y. Huang, "A fast path planning algorithm fusing PRM and P-BI-RRT," in *Proc. 11th Int. Conf. Prognostics Syst. Health Manage.*, 2020, pp. 503–508.

[14] M. Yu, J. Luo, M. Wang, and D. Gao, "Spline-RRT*: Coordinated motion planning of dual-arm space robot," *IFAC-PapersOnLine*, vol. 53, pp. 9820–9825, 2020.

[15] Z. Wang, J. Chang, B. Li, C. Wang, and C. Liu, "Application of improved rapidly-exploring random trees (RRT) algorithm for obstacle avoidance of snake-like manipulator," in *Proc. IEEE Int. Conf. Mechatronics Automat.*, 2020, pp. 490–495.

[16] Z. Chen, Y. Yang, X. Xu, and A. Rodic, "Path planning of redundant series manipulators based on improved RRT algorithms," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, 2019, pp. 553–557.

[17] M. Vande Weghe, D. Ferguson, and S.S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," in *Proc. 7th IEEE-RAS Int. Conf. Humanoid Robots*, 2007, pp. 477–482.

[18] C. Bergeles and P. E. Dupont, "Planning stable paths for concentric tube robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 3077–3082.

[19] K. Wu, L. Wu, and H. Ren, "Motion planning of continuum tubular robots based on centerlines extracted from statistical atlas," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 5512–5517.

[20] Z. Hawks, C. Frazelle, K. E. Green, and I. D. Walker, "Motion planning for a continuum robotic mobile lamp: Defining and navigating the configuration space," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 2559–2566.

[21] L. G. Torres, C. Baykal, and R. Alterovitz, "Interactive-rate motion planning for concentric tube robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 1915–1921.

[22] A. Kuntz *et al.*, "Motion planning for continuum reconfigurable incisionless surgical parallel robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 6463–6469.

[23] K. Leibrandt, C. Bergeles, and G.-Z. Yang, "Concentric tube robots: Rapid, stable path-planning and guidance for surgical use," *IEEE Robot. Automat. Mag.*, vol. 24, no. 2, pp. 42–53, Jun. 2017.

[24] J. Wan, J. Yao, L. Zhang, and H. Wu, "A weighted gradient projection method for inverse kinematics of redundant manipulators considering multiple performance criteria," *Strojniš ki Vestnik - J. Mech. Eng.*, vol. 64, no. 7/8, pp. 475–487, 2018.

[25] I. S. Godage, G. A. Medrano-Cerda, D. T. Branson, E. Guglielmino, and D. G. Caldwell, "Dynamics for variable length multisection continuum arms," *Int. J. Robot. Res.*, vol. 35, no. 6, pp. 695–722, 2016.

[26] I. S. Godage, Y. Chen, and I. D. Walker, "Dynamic control of pneumatic muscle actuators," 2018, *ArXiv:abs/1811.04991*. [Online]. Available: https://doi.org/10.48550/arXiv.1811.04991