# Path Planning for Continuum Arms in Dynamic Environments

Brandon H. Meng[1], Dimuthu D. K. Arachchige[1], Isuru S. Godage[2], and Iyad Kanj[1]

***Abstract*— Multisection continuum arms are bio-inspired manipulators that combine compliance, payload, dexterity, and safety to serve as co-robots in human-robot collaborative domains. Their hyper redundancy and complex kinematics, however, pose many challenges when performing path planning, especially in dynamic environments. In this paper, we present a W-Space based Rapidly Exploring Random Trees * path planner for multisection continuum arm robots in dynamic environments. The proposed planner improves the existing state-of-art planners in terms of computation time and the success rate, while removing the need for offline computation. On average, the computation time of our approach is below 2 seconds, and its average success rate is around 70%. The computation time of the proposed planner significantly improves that of the state-of-the-art planner by roughly a factor of 20, making the former suitable for real-time applications. Moreover, for application domains where the obstacle motion is not very predictable (e.g., human obstacles), the proposed planner significantly improves the success rate of state-of-the-art planners by nearly 50%. Lastly, we demonstrate the feasibility of several generated trajectories by replicating the motion on a physical prototype arm.**

## I. Introduction

Continuum arms are manipulators that use continuous bending as a means of actuation. Such devices employ a variety of models and construction materials, and are inspired by organic appendages (e.g., elephant trunks or octopus arms) [1]. In designing continuum arms, a combination of compliant and rigid materials can be employed to create an ideal mix of each component's strength. Rigid materials, on the one hand, excel in stability and potential payload, but may be quite dangerous when in contact with humans. Soft materials, on the other hand, allow for a great deal of novel shapes, and pose a far lower risk to humans, but are unable to manipulate larger payloads.

A multisection continuum arm consists of several continuum modules that are serially joined. These devices can assume a large number of poses, but at the cost of very complicated kinematics. This paper focuses on human-scale multisection continuum arms consisting of a mix of rigid and compliant materials. These devices exhibit an ideal mix of novel bending, inherent safety, and capable payload. It is these traits that make these devices perfectly suited to serve as co-robots [2] in domains like healthcare and manufacturing. Continuum arms have been a very active area of research in recent years [3].

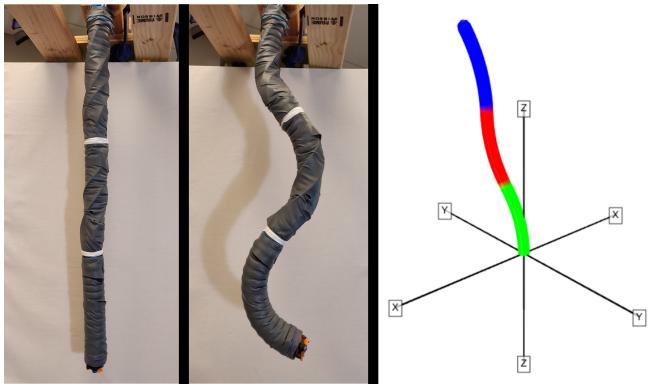


Fig. 1. Left: A multisection continuum arm at rest, Middle: A multisection continuum arm shown actuated, Right: A simulated view of a bent multisection continuum arm.

In this paper, we study the path planning of multisection continuum arms in dynamic environments, i.e., where obstacles are in motion. Continuum arms pose a unique challenge when performing path planning in dynamic environments, due to the huge number of complex spatial-shapes these devices can assume and the non-linear complex mapping between their configuration-space (C-Space) and work-space (W-Space). This complex mapping makes planning difficult, as a sequence of poses may have an excessive amount of movement if care is not taken to limit such behavior. This is problematic for inverse kinematics and other standard path planning methods. Path planning in dynamic environments (and path planning in general) for multisection continuum arms remains an open area of research.

### A. Related Work

Path planning in dynamic environments is a widely studied problem across a huge number of domains [4]–[6]. Notably, autonomous vehicles have seen an explosion of research [7]. There has been some work on path planning for continuum arms in dynamic environments, though this field is far less mature. Potential field approaches are standard solutions to this problem, and such a planner was presented in [8] for the robotic model in consideration. However, this approach was shown to be unreliable in [9]. An anticipatory approach that used a temporal-graph was presented in [9]. Anticipatory planners try to "anticipate" the motion of obstacles and plan around the predicted location of obstacles.

A substantial body of literature exists on static path planning (with stationary obstacles), and several planners

[1]School of Computing, DePaul University, Chicago, IL 60604, USA. {bmeng1,darachch,ikanj}@depaul.edu

[2]Department of Engineering Technology & Industrial Distribution, J. Mike Walker '66 Department of Mechanical Engineering (affiliated), and Department of Multidisciplinary Engineering (affiliated), Texas A&M University, College Station, TX 77843, USA. igodage@tamu.edu

This work was supported in part by the National Science Foundation (NSF) grants IIS-1718755, IIS-2008797, CMMI-2048142, CMMI-2132994 and National Institute of Health (NIH) R01 grant 5R01NS116148-04.

have been developed specifically for continuum arms. The approaches proposed in [10] and [11] are graph-theoretic, brute-force, and rely on a look-up table. These approaches were shown to be reliable and generate smooth paths, but used offline-computation and were slow to generate solutions. The authors in [10] also demonstrated the unreliability of inverse kinematics (IK) based path planners. IK is a ubiquitous tool used to map the W-Space of a robotic model to its C-Space. However, IK-based approaches can get "stuck" in local minima. The authors in [12] introduced a closed-form solution for the IK of multisection continuum arms. However this approach does not consider the geometrical constraints of the curve parameters, and may not arrive at a solution when applied to the model presented in this paper.

This paper focuses on a random sampling approach for path planning, and more specifically on the Rapidly Exploring Random Trees (RRT) approach [13]. RRT is notable for its simplicity and efficiency and has a very popular variant, RRT* [14], that achieves probabilistic completeness and asymptotic optimality. Probabilistic Road Maps (PRM) is another standard sampling approach, and has a similarly popular variant called Lazy PRM [15]. Though this field is broad, there have been recent contributions in RRT and PRM for dynamic environments. These include Dynamic RRT*-connect [16], LTR* [17], HIRO [18], and T-RPM [19]. RRT and PRM approaches are commonly applied in the C-Space of a robot, as they are easy to adapt to rigid-link robots, specifically with low degrees of freedom (DoF). In general, RRT and PRM search are known to slow down in higher dimensional spaces, due to the *curse of dimensionality*.

As was shown in [20], C-Space planning is ill-suited for multisection continuum arms. A W-Space based RRT* approach for multisection continuum arms was proposed in [20], but this approach is not suitable for dynamic environments. However, this approach did highlight some of the advantages of W-Space based RRT*. Notably, the W-Space RRT* approach reduced the dimensionality of the path planning problem by using W-Space exploration. Additionally, paths generated by this approach were shown to be smoother than other state-of-the-art planners.

### B. Contribution

The kinematics of multisection continuum arms are highly complex and present a major challenge when path planning. Standard C-Space RRT* approaches are not ideal for path planning in static environments, and these shortcomings become more prominent in dynamic environments. Previous work used W-Space exploration paired with Jacobian-based configuration generation to create a more suitable RRT* approach, but has not explored dynamic settings. To overcome these issues, we present a planner for dynamic environments. Our contributions are summarized as follows:

1. We propose a W-Space based RRT* path planner for multisection continuum arms in dynamic environments. This approach operates in two phases: planning and tracing. In the planning phase, a path is generated through the use of a RRT* random tree. In the tracing phase, the arm begins to follow the generated path, while updating the positions of the obstacles. At each step, the planner checks for imminent collision. When the arm is at risk of collision, the planner returns to the planning phase. In order to design the W-Space based RRT* planner for dynamic environments, four main methods were employed: Horizon planning, obstacle inflation, distance records, and a backup counter.

2. We demonstrate the superiority of the proposed planner by comparing it to the state-of-the-art planners. The comparisons target application domains with three types of motions: circular orbits, sinusoidal waves, and random walks.

i. We demonstrate a completely on-line planner.
ii. We show that our planner has a significant reduction in computation time. For most test environments, the computation time of our approach was below 1 second, a nearly 20 times improvement over the other planners.
iii. We illustrate the robustness of our approach, specifically in domains with less predictable obstacle motions. In these test environments, our approach produces a nearly 50% improvement over the state-of-the-art.

3. Lastly, we demonstrate the feasibility of a generated trajectory by replicating a restricted set of examples on the physical prototype arm.

We believe that the methods we develop are general, and that their applications extend beyond the continuum arms model, and can be adapted to develop W-Space based RRT* planners for other continuum manipulators.

## II. Preliminaries

### A. Kinematic Model

This paper is focused on the prototype continuum arm model proposed in [9]–[11], [20]. We will briefly summarize the model. This prototype continuum arm is composed of 3 continuum sections. Each continuum section consists of 3 pneumatic muscle actuators (PMA). These PMAs are joined around an inextensible backbone with $\frac{2\pi}{3}$ radians separation. Actuation occurs by adding or evacuating air or fluid from any of the PMAs. These continuum sections are joined serially to form the multisection continuum arm. The use of a backbone creates an over-constrained system, meaning that the third PMA in each section is kinematically redundant and therefore the bending can be described by just two curve parameters. These curve parameters are $\phi$ and $\theta$, and describe the bending angle subtended by the arc and the bending plane, respectively. A *configuration* for the pose of the complete multisection arm is a 6-tuple consisting of the $\phi$ and $\theta$ values for each of the 3 continuum sections. We refer the reader to [20] for more details.

### B. W-Space RRT*

The planner proposed in this paper utilizes some of the tools and methods from [20]. They are briefly summarized here. The W-Space RRT* planner uses W-Space exploration with W-Space-C-Space mapping to generate trajectories. In this approach, an RRT* tree $\mathcal{T}$ is grown in the W-Space, and nodes are inserted into $\mathcal{T}$ only if a corresponding configuration could be generated in the C-Space using the

*Jacobian-based configuration generation* method. The parent tip coordinates and parent configuration along with the desired child tip point are fed to the configuration generation method, and a configuration is outputted that places the arm's tip at the desired child tip point. The Jacobian-based method produces a configuration $c$ that places the tip at approximately the requested child tip point. If $c$ is deemed invalid, then it is discarded and not inserted into $\mathcal{T}$. This static approach utilizes *goal-biasing*, wherein $\mathcal{T}$ is grown in the direction of the destination for a fixed percentage of node insertions. An iteration of node insertion where goal-biasing occurs is called a *guiding iteration*.

## III. Dynamic W-Space RRT*

In this section, we present a W-Space RRT* planner for dynamic environments that we refer to as *WRRT*D*. The planner grows a sequence of trees in the W-Space, and traces a partial path in each of these trees, beginning from the original starting point and finally arriving at the destination point. During the tree growth, the radius of each obstacle is inflated, so that planning takes place with an implicit buffer around each obstacle. A tree is grown until its growth reaches a certain distance, at which point a partial path in this tree is generated. Once a path is generated, the arm begins to trace this path. While the arm is tracing a path, due to potential collision with obstacles, there may be a need to generate a replacement path. Specifically, if the arm is moving too close to an obstacle or away from the target, the path needs to be reconstructed. In such a case, a new tree is grown from the arm's current position. This sequence of growing trees, finding paths, and tracing paths is repeated until the destination is reached or the algorithm fails to find a collision-free path.

### A. The Planner

In dynamic path planning, generally, there is no information provided to the planner about the future positions of the obstacles. Therefore, dynamic planners must create a plan, and then update it as new information is acquired about the positions of the obstacles. Although path planners for static environments generally create a global path from the original starting point to the destination point, this is not suitable for dynamic planning.

Therefore, to address the challenges of dynamic path planning, our planner works in two phases. In the first phase, a partial path is created for the arm to follow. In the second phase, the arm attempts to trace the partial path. During this tracing, the current path may lead to a collision with an obstacle. In this case, a new partial path is created for the arm to follow starting from the current position. To avoid unnecessary computation, we use the *horizon planning* approach [21], [22]: Rather than attempting to grow a "global path" from the original starting point towards the target point, in horizon planning a random tree is grown outwards from the source until the depth of nodes reaches a certain threshold. Once this threshold is reached, a path is generated from the tree, whether or not the target is reached.

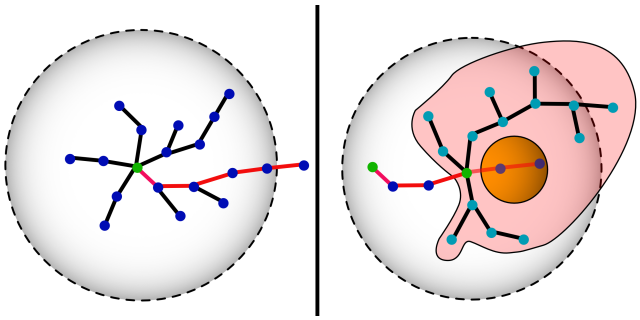

Fig. 2. Left: A tree is grown until a node crosses the horizon. The shortest path from the arm's current position (the green node) to the horizon is shown in red. Right: An obstacle obstructs the previously generated path while tracing it. A new tree is grown from the arm's new current position (the center green node).

Despite the use of horizon planning, partial paths are still costly to generate, so it is necessary to limit the frequency of replanning. To achieve this during the path tracing phase, we employ a number of strategies to detect the need for a replan and to limit the frequency of replans. Phase one is shown in Alg. 1 and phase two is shown in Alg. 2.

*1) Horizon Planning:* Let $h > 0$ be a fixed constant, referred to as the *horizon distance*. To grow a tree $\mathcal{T}$ in horizon planning starting from some node $u$, we proceed as described in Subsection II-B. At each iteration, the distance $\delta$ between $u$ and the newly-inserted node $n$ is calculated. There are two termination cases during horizon planning:

i. Node $n$ is the target node $t$. In this case, growth has been completed and a partial path from $u$ to $t$ is generated.
ii. The distance $\delta$ exceeds the horizon distance $h$. In this case, node $n$ marks a "crossing of the horizon" and the growth of the partial path is complete. We generate the partial path from $u$ to $n$.

Refer to the left side of Fig. 1 for an illustration. Note that, even though the global target may not be included within the horizon, the use of *goal-biasing* will direct the random exploration towards the destination. The pseudocode of this procedure is contained within Alg. 1.

*2) Tracing:* In a dynamic setting, planners only have knowledge of the current and previous positions of obstacles. In phase two of the planner, the planner traces the generated tree path. While the arm is tracing the path, the position of obstacles is updated to reflect the new state of the W-Space. The *critical distance* $\omega$ between an obstacle and the arm is a small distance wherein the obstacle and arm are very close and action must be taken to avoid collision. A *replan* must occur when the current partial path is not suitable for the arm given the current positions of the obstacles. If a replan is initiated, a new path is created using horizon planning and originating from the arm's current position. This means that a new tree $\mathcal{T}$ with the arm's current position as the source is grown. Refer to the right side of Fig. 1 for an illustration.

### B. Obstacle Inflation

To avoid collision with obstacles, the continuum arm needs to maintain a "buffer distance" from each obstacle present

in the environment. To maintain the buffer distance during planning, the radius of each obstacle is slightly increased during the horizon planning. Though arms may get arbitrarily close to an inflated obstacle during planning, the inflated region of the obstacle acts as a buffer, such that when the radius is reduced back to its normal value, the arm is implicitly maintaining a safe distance from each obstacle.

### C. Distance Record

The purpose of keeping a *distance record* is to keep track of the relation between the arm and nearby obstacles such that a replanning occurs only when the distance between the arm and an obstacle is decreasing and the obstacle is within the critical distance from the arm.

Let $A = [a_1, a_2 \ldots, a_i]$ be a sequence of points along the arm, where $a_1$ corresponds to the point at the base of the arm and $a_i$ corresponds to that at the tip of the arm. These points are uniformly distributed along each section of the arm. Let $O = \{o_1, o_2, \ldots, o_k\}$ be the set of obstacles, where $k$ is the number of obstacles in the workspace. Let $D = [d_1, d_2, \ldots, d_i]$ be the distance record of the arm such that for each $a_j$ for $j = 1, \ldots, i$, $d_j$ is the minimum distance between $a_j$ and the obstacles in $O$.

As the pose of the arm changes during tracing, the entries of $D$ are updated. If for a point $a_j$ along the arm, its new distance entry $d'_j$ is less than the minimum between its previously recorded distance entry $d_j$ and the critical distance $\omega$, then a replan occurs. This procedure is part of Alg. 2.

**Algorithm 1** HORIZON

**Require:** starting configuration $s$, destination point $t_g$
partial path $P' = \emptyset$ ▷ This is maintained as a queue
create a tree $\mathcal{T}$ using $s$ and $t_g$
inflate each obstacle $o \in O$
**while** no node has crossed the horizon and $t_g \notin \mathcal{T}$ **do**
  grow $\mathcal{T}$ by inserting a node $n$
  **if** $n == t_g$ **then**
    calculate the path $P$ from $s$ to $t_g$
  **else if** dist$(n, s) \geq h$ **then** ▷ $n$ crosses the horizon
    calculate the path $P$ from $s$ to $t_g$
  **end if**
**end while**
deflate the obstacles in $O$
**return** $P$

### D. Backup Counter

The backup counter is used to implement a simple strategy for ensuring that the arm makes sufficient progress toward the target without stalling in one area. It is possible that the arm might progress away from the target to avoid collision with an obstacle. However, if after several steps of obstacle movement, an opening exists towards the target, it would be ideal to reverse course and head towards the target.

The backup counter is an integer variable, $f$, that records the number of steps that the arm has taken away from the destination. The counter is initialized so that $f = 0$ at the start of a plan. Let $\beta$ be the number of permissible backup steps before a replan is necessary. As the arm traces the path, the distance between the arm's tip point and the goal target point is calculated. If this distance increases, then $f$ is incremented by 1. If at any moment $f > \beta$, then a replan is initiated. This procedure is part of Alg. 2.

**Algorithm 2** W-Space RRT* Dynamic

**Require:** global starting configuration $s_g$, global target point $t_g$, moving obstacle list $O$, horizon distance $h$
global path $P_G = \emptyset$
distance record $D = [\infty, \infty, \ldots]$
critical distance $\omega$
backup counter $f = 0$, maximum backups $\beta$
$P =$ HORIZON
current configuration $u = P.pop()$
add $u$ to $P_G$
**while** $t_g \notin P_G$ **do**
  update position of $o \in O$
  **for all** distance records $d \in D$ **do**
    calculate new distance entry $d'$
    **if** ($d' < d$ AND $d' < \omega$) OR $f > h$ **then**
      $f = 0$
      $P' =$ HORIZON
      BREAK
    **end if**
  **end for**
  $u = P.pop()$
  append $u$ to $P_G$
  **if** the arm moves away from $t_G$ **then**
    $f = f + 1$
  **end if**
**end while**
**return** $P_G$

## IV. Methodology and Results

### A. Test Environment

In order to evaluate the performance of the proposed dynamic path planner, we compare it to the state-of-the-art anticipatory approaches proposed in [9]. To summarize, these approaches use previous obstacle locations, a prediction scheme, and a temporal graph to generate paths that 'anticipate' the motion of obstacles. We subject each of the tested approaches to three test environments. Each environment uses spherical obstacles with a radius of either 2 or 3cm, with each obstacle assigned a radius randomly. When the pose of the arm is changed during tracing, the change in the obstacle position is roughly 75% of the change of the arm tip. The obstacle inflation radius was selected by experimentation. Let $r$ be the initial radius and $r'$ be the radius after inflation. This change in $r$ was selected such that $r' \leq 3r/2$. The parameters $h$, $\omega$, and $\beta$ were also selected experimentally as 15cm, 2cm, and 5, respectively. Each of the trials begins at the *resting pose* of the arm. This is the pose where none of the PMAs are actuated and corresponds to the configuration (and joint values) $[0, 0, 0, 0, 0, 0]$. The arm is tasked with navigating from the resting pose to a randomly-generated target point

TABLE I

ORBIT ENVIRONMENT

| Approach \ Obstacles | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| AN2 Succ. | 78% | 76% | 72% | 71% | 64% | 66% |
| AN2 Time (s) | 11.7 | 15.5 | 12.4 | 15.4 | 17.6 | 2.4 |
| AN3 Succ. | 78% | 76% | 71% | 73% | 64% | 66% |
| AN3 Time (s) | 11.7 | 9.6 | 19.4 | 23.2 | 25.3 | 29.8 |
| WRRT*D Succ. | 67% | 72% | 75% | 68% | 68% | 65% |
| WRRT*D Time (s) | 0.2 | 0.7 | 0.7 | 1.9 | 0.8 | 1.6 |

without colliding with any obstacles. Additionally, in all trials, a random target point is selected such that the point is within 25cm of the origin. A summary of each of the test environments follows.

**Environment 1 - Circular Orbit Motion:** In this environment, the obstacles are following a circular orbit around a fixed center. These orbit centers are selected along the straight-line between the initial tip position and the target position. Each orbit lies in a plane that makes an angle $-45° \leq \alpha \leq +45°$ with the $xy$-plane, where $\alpha$ is chosen at random with uniform probability. Additionally, the direction of the rotation (clockwise, counter-clockwise) is selected randomly with uniform probability.

**Environment 2 - Wave Motion:** This environment uses sinusoidal waves in order to generate obstacle movement in the W-Space. Obstacles are placed in a line near the target point which forms a $90°$ angle with the straight-line from the starting point to the target point. Obstacles are placed such that their respective centers are 4cm apart. The obstacles move with sinusoidal motion that lies on a plane and all obstacles intersect at the midpoint between the starting tip point and the target point. This is, in effect, a "wall" of obstacles advancing towards the arm that moves in a sinusoidal pattern.

**Environment 3 - Random Motion:** The random motion in this environment consists of random walks. Each of the obstacles is initially placed along the straight-line between the resting pose arm tip and the target point. For each obstacle, a random distance between 1 and 5cm is selected. Next, a random direction is selected. The obstacle then travels a random distance in a random direction. When the obstacle has traveled the specified distance, a new distance, and a new direction are selected, and the obstacle moves according to those new parameters. Each of these environments is tested on a range of obstacle counts, with $1, 2, \ldots, 6$ obstacles used. For each number of obstacles, the approaches are tested on the same 100 random trials. The success rate over the 100 trials is reported along with the average computation time that each approach took to generate a solution.

### B. Results

The results of each of the approaches for the test environments are reported in the tables below. We refer to the Temporal-2, Temporal-3, and WRRT*D approaches as *AN2*, *AN3*, *WRRT*D* respectively. The success rate reported in each table is the percentage of trials that were completed

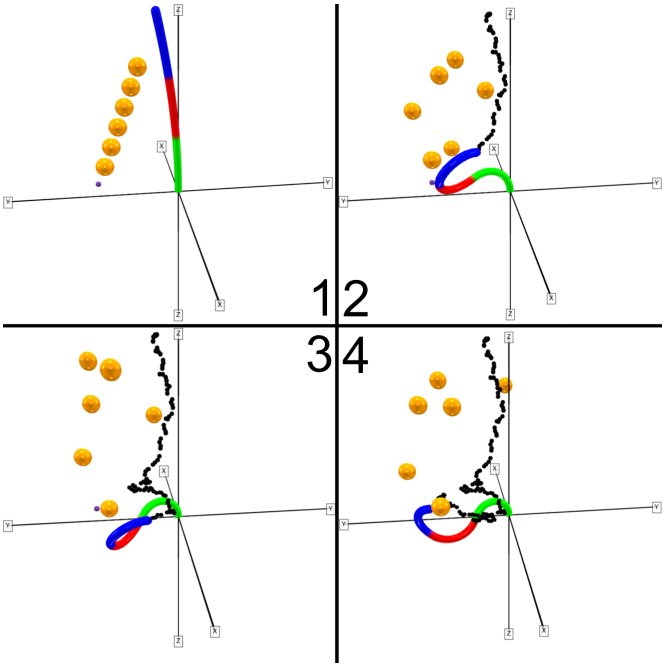


Fig. 3. A path generated by WRRT*D performed on Environment 3. Obstacles are shown as orange spheres and the target tip position is shown as a purple dot. Frames are shown of the initial position (1), 33% progress (2), 66% progress (2) (3), and the final position (4).

TABLE II

WAVE ENVIRONMENT

| Approach \ Obstacles | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| AN2 Succ. | 53% | 31% | 26% | 25% | 23% | 12% |
| AN2 Time (s) | 5.6 | 6.2 | 9.9 | 11.2 | 13.5 | 12.4 |
| AN3 Succ. | 65% | 42% | 45% | 28% | 26% | 19% |
| AN3 Time (s) | 11.2 | 13.7 | 17.0 | 17.7 | 20.5 | 22.6 |
| WRRT*D Succ. | 75% | 74% | 69% | 68% | 73% | 64% |
| WRRT*D Time (s) | 0.3 | 0.4 | 0.4 | 0.5 | 0.7 | 0.5 |

successfully. The columns correspond to the number of obstacles used in each set of trials. The rows contain the success rate and computation time of each of the approaches. Fig. 3 shows a path generated by WRRT*D for environment 3. For more examples, please see the associated video. Tabs. I, II, and III show the results from the Circular Orbit Motion environment, Wave Motion environment, and Random Motion environment, respectively.

In general, it seems that the AN2 and AN3 approaches are very sensitive to the quality of predictions, whereas the WRRT*D approach is more consistent across motions. Random motion, in particular, is challenging for the anticipatory approaches. The low success rate of the WRRT*D approach in certain settings is likely unrelated to the environment, and instead related to the method of calculating configurations.

### C. Continuum Arm Prototype Testing

The 3-section prototype arm shown in Fig. 1 is tested using the configurations generated in path planning simulations. Configurations are related to length changes of the arm using kinematic relationships [23]. Length changes are mapped to

TABLE III
RANDOM ENVIRONMENT

| Approach \ Obstacles | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| AN2 Succ. | 48% | 25% | 17% | 3% | 4% | 1% |
| AN2 Time (s) | 5.1 | 6.8 | 7.9 | 8.1 | 11.1 | 11.8 |
| AN3 Succ. | 39% | 18% | 10% | 4% | 2% | 2% |
| AN3 Time (s) | 9.5 | 10.8 | 12.9 | 14.3 | 16.3 | 18.8 |
| WRRT*D Succ. | 71% | 76% | 60% | 64% | 67% | 63% |
| WRRT*D Time (s) | 0.25 | 0.26 | 0.26 | 0.26 | 0.26 | 0.26 |

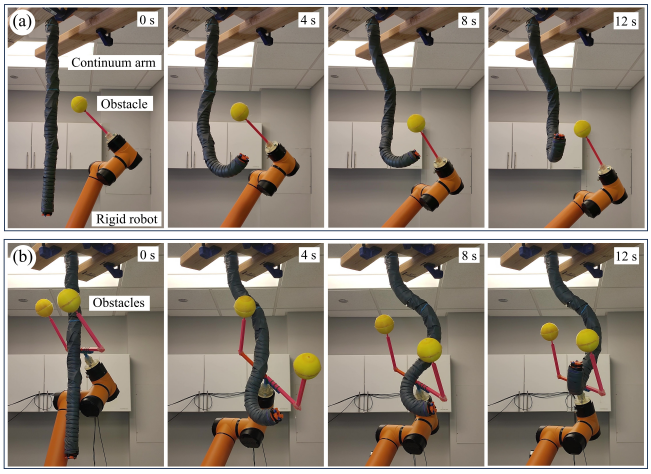


Fig. 4. Path planning progression of the continuum arm prototype under (a) single and (b) two obstacles.

actuation pressure trajectories using the joint space – pressure mapping approach proposed in [24] and fed to the arm via an experimental setup that consists of an 8 *bar* air compressor, 09 proportional pressure regulators (ITV3050-31F3N3, SMC USA) corresponding to 09 PMAs of the arm, and a MATLAB Simulink desktop real-time model interfaced with a data acquisition (DAQ) card (PCI-6703, NI USA). The DAQ card is coupled with pressure regulators and issues voltage signals corresponding to the pressure commands set in MATLAB Simulink to release air pressure.

To replicate the motion of dynamic obstacles, we attached plastic spherical objects to the end effector of a rigid robotic arm (AUBO-i5, AUBO-cobot USA). The obstacle trajectories that were generated relative to the base of the continuum arm are transformed relative to the base of the rigid arm. This helps generate obstacle trajectories by actuating the rigid arm. Fig. 4 shows a path generated by the WRRT*D for one and two obstacles following circular orbit motion. The arm prototype testing initially validates the proposed path-planning algorithm. Due to the operation limitations of the arm, every proposed path planning algorithm herein was not tested. Here we present proof of the concept through preliminary experimental results. Refer to the multimedia file submission to see the complete videos of the experiments.

## V. CONCLUSION

In this paper, we studied the problem of dynamic path planning for multisection continuum arms. We introduced a Dynamic W-Space RRT* path planner and demonstrated its strengths relative to other state-of-the-art path planning approaches for these models. Through the use of different test environments, we showed the speed of our proposed approach while also showing its ability to be adapted to a number of different domains. We showed that our approach outperformed the anticipatory approach, specifically where the predictions were not well-suited for the environment. Future work might center around real-time applications of this approach. Additionally, this model could be adapted for parallel robots, to show planning for multiple multisection continuum arms.

## REFERENCES


[1] D. Trivedi *et al.*, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied Bionics & Biomechanics*, 2008.
[2] E. Boy, E. Burdet, C. Teo, and J. Colgate, "The learning cobot," in *ASME Int. Mech. Eng. Cong. & Exposition*, 2002, pp. 867–873.
[3] S. Kolachalama and S. Lakshmanan, "Continuum robots for manipulation applications: A survey," *J. of Robotics*, pp. 1–19, 2020.
[4] B. Patle *et al.*, "A review: On path planning strategies for navigation of mobile robot," *Defence Technology*, pp. 582–606, 2019.
[5] C. Cheng, Q. Sha, B. He, and G. Li, "Path planning and obstacle avoidance for auv: A review," *Ocean Engineering*, vol. 235, 2021.
[6] R. Almadhoun *et al.*, "A survey on multi-robot coverage path planning for model reconstruction and mapping," *SN Applied Sciences*, 2019.
[7] K. Karur *et al.*, "A survey of path planning algorithms for mobile robots," *Vehicles*, vol. 3, no. 3, pp. 448–468, 2021.
[8] I. S. Godage *et al.*, "Path planning for multisection continuum arms," in *IEEE Int. Conf. on Mech. & Aut.*, 2012, pp. 1208–1213.
[9] B. H. Meng, D. D. Arachchige, J. Deng, I. S. Godage, and I. Kanj, "Anticipatory path planning for continuum arms in dynamic environments," in *IEEE Int. Conf. on Rob. & Aut. (ICRA)*, 2021.
[10] J. Deng, B. H. Meng, I. Kanj, and I. S. Godage, "Near-optimal smooth path planning for multisection continuum arms," in *IEEE Int. Conf. on Soft Robotics (RoboSoft)*, 2019, pp. 416–421.
[11] B. H. Meng, I. S. Godage, and I. Kanj, "Smooth path planning for continuum arms," in *IEEE Int. Conf. on Rob. & Aut. (ICRA)*, 2021.
[12] S. Neppalli *et al.*, "Closed-form inverse kinematics for continuum manipulators," *Ad. Robotics*, vol. 23, no. 15, pp. 2077–2091, 2009.
[13] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The Int. J. of Rob. Research*, vol. 20, no. 5, pp. 378–400, 2001.
[14] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *Rob. Science & Systems*, 2010.
[15] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *IEEE Int. Conf. on Rob. & Aut. (ICRA)*, 2000, pp. 521–528.
[16] Y. Chen and L. Wang, "Adaptively dynamic rrt*-connect: Path planning for uavs against dynamic obstacles," in *IEEE Int. Conf. on Auto., Control and Robotics Eng. (CACRE)*, 2022, pp. 1–7.
[17] T. Lai and F. Ramos, "Ltr*: Rapid replanning in executing consecutive tasks with lazy experience graph," in *IEEE/RSJ Int. Conf. on Intel. Rob. & Sys. (IROS)*, 2022, pp. 8784–8790.
[18] X. Huang *et al.*, "Hiro: Heuristics informed robot online path planning using pre-computed deterministic roadmaps," in *IEEE/RSJ Int. Conf. on Intel. Rob. & Sys. (IROS)*, 2022, pp. 8109–8116.
[19] M. Hüppi, L. Bartolomei, R. Mascaro, and M. Chli, "T-prm: Temporal probabilistic roadmap for path planning in dynamic environments," in *IEEE/RSJ Int. Conf. on Intel. Rob. & Sys. (IROS)*, 2022.
[20] B. H. Meng, I. S. Godage, and I. Kanj, "Rrt*-based path planning for continuum arms," *IEEE Rob. & Auto. Letters*, 2022.
[21] J. Wang *et al.*, "Learning to guide online multi-contact receding horizon planning," in *IEEE/RSJ Int. Conf. on Intel. Rob. & Sys. (IROS)*, 2022, pp. 12 942–12 949.
[22] J. Leu, M. Wang, and M. Tomizuka, "Long-horizon motion planning via sampling and segmented trajectory optimization," in *European Control Conference (ECC)*, 2022, pp. 538–545.
[23] D. D. Arachchige and I. S. Godage, "Hybrid soft robots incorporating soft and stiff elements," in *IEEE Int. Conf. on S. R. (RoboSoft)*, 2022.
[24] D. D. Arachchige *et al.*, "Soft steps: Exploring quadrupedal locomotion with modular soft robots," *IEEE Access*, 2023.