Kinematic-Model-Free Control for Space Operations with Continuum Manipulators

Chase Frazelle and Ian Walker
ECE Department
Clemson University
Clemson, SC USA 29630
{cfrazel,iwalker}@clemson.edu

Ahmad AlAttar and Petar Kormushev
Dyson School of Design Engineering
Imperial College London
London, UK SW7 2DB
{a.alattar19,p.kormushev}@imperial.ac.uk

***Abstract*—Continuum robots have strong potential for application in Space environments. However, their modeling is challenging in comparison with traditional rigid-link robots. The Kinematic-Model-Free (KMF) robot control method has been shown to be extremely effective in permitting a rigid-link robot to learn approximations of local kinematics and dynamics ("kinodynamics") at various points in the robot's task space. These approximations enable the robot to follow various trajectories and even adapt to changes in the robot's kinematic structure. In this paper, we present the adaptation of the KMF method to a three-section, nine degrees-of-freedom continuum manipulator for both planar and spatial task spaces. Using only an external 3D camera, we show that the KMF method allows the continuum robot to converge to various desired set points in the robot's task space, avoiding the complexities inherent in solving this problem using traditional inverse kinematics. The success of the method shows that a continuum robot can "learn" enough information from an external camera to reach and track desired points and trajectories, without needing knowledge of exact shape or position of the robot. We similarly apply the method in a simulated example of a continuum robot performing an inspection task on board the ISS.**

TABLE OF CONTENTS

1. INTRODUCTION

Rigid-link robots are, as the name suggests, traditionally comprised of rigid bodies connected through a finite series of joints. While this makes for tractable dynamics and permits reasonable controllability, this form of structure does not lend itself well to cluttered environments or scenarios in which collisions, even minor, can result in damage to the robot and environment. Conversely, continuum robots, which can bend at any point along their continuous backbones, are designed to be compliant, and in cluttered environments are able to experience contacts without causing damage. The compliant nature of continuum robots makes them ideal for inspection and sensing in restricted environments, such as cluttered cargo areas and hard-to-reach areas of interest. Unfortunately, this increase in compliance also carries an increase in the complexity of the robot's dynamics and inverse kinematics, especially in the case of a redundant continuum manipulator.

Continuum manipulators manifest a theoretically infinite number of Degrees-of-Freedom (DoF), distinctly different from even hyper-redundant rigid-link robots, such as snake robots [1]. The ability of continuum robots to adapt to their environments and maneuver in cluttered spaces has motivated their application in Space applications [2], [3]. There is by now a fairly extensive literature on continuum robot kinematics, many of which constrain these infinite DoF to ideal assumptions about the robot and its environment. Given these ideal conditions, the constant curvature based kinematic models [4], [5] have proven effective for approximating the shape of continuum systems and the location of end-effectors in open space. Further works have expanded into modeling non-constant curvature bending of continuum robots that are subject to internal and external loads [1], [6], [7], or collisions with the environment [8], [9]. These more realistic models, while better at approximating shape and predicting output under load, require significantly more information about the continuum system and do not lend themselves well to invertibility or the added complexity of implementation on multi-section continuum manipulators. Neither of these classes of kinematic models have proven trivial for the execution of task space path planning and following.

Likewise, dynamic modeling for continuum robot systems has received notable attention, especially of late [10], [11], [12], but the complexity of these systems, especially those composed of multiple sections, often proves to be unwieldy without making numerous simplifying assumptions. Indeed, our own recent approximate models [13], [14] are still computationally intensive even after removing the complexity of continuum mechanics and settling for approximate rigid-link models. Coupled with the effects of gravity, achieving motion along a desired path is dependent on reliable modeling and sufficient feedback. With respect to continuum robots in practical applications, the states of such solutions are often inherently unobservable, under-actuated, or both, without the implementation of computationally expensive sensor arrays or a suite of external cameras to extract full robot shape while avoiding occlusion.

In considering the complexity of kinematic and dynamic models for continuum robots, it is easy to see why the relatively few works that explore motion planning with continuum manipulators have relied on simplification of these models or reduction in the number of DoF. This simplification allows implementation of some of the popular motion planning methods used for rigid-link robots, such as Rapidly-exploring Random Trees (RRT) [15], [16] and reinforcement learning methods [17], [18].

In this paper, we explore the application of Kinematic-Model-

978-1-7281-7436-5/21/$31.00 ©2021 IEEE

Free (KMF) robot control as a potential solution for the many challenges facing task space path planning and automation of continuum robots, while simultaneously reducing the need for complex sensors or extensive knowledge about the continuum robot. In previous works by the authors [19], [20], [21], the KMF method has been shown to be extremely effective in permitting a rigid-link robot to learn approximations of local kinematics and dynamics (termed "kinodynamics") at various points in the robot's task space. These approximations then enabled a robot to follow various trajectories and even adapt to changes in the robot's kinematic structure. The approach learns the local kinodynamics through a series of exploratory actuation primitives and a k-Nearest Neighbor algorithm. The algorithm can predict what inputs to the robot's actuators will result in a motion towards a desired set point. A major advantage of this approach is the simplification of the feedback system: only a 3D camera is needed to track the location of the end-effector relative to the location of the desired set point.

The paper is organized in the following order: Section 2 reviews the architecture of KMF and the steps taken in this work to adapt KMF to continuum robots. Section 3 describes the experimental setup and results of using KMF with an extensible continuum manipulator. Discussion concerning the potential of KMF for continuum robots, especially considering a continuum manipulator deployed with KMF to enable ISS exploration is given in Section 4. Finally, conclusions are offered in Section 5.

2. METHODS

This section gives an overview of the established KMF method, detailing the overall structure of the algorithm and the specific details we utilize in our realization. Further, we present a series of mappings that convert actuation primitives from KMF into universal actions for continuum manipulators by exploiting ideal kinematics.

KMF Algorithm

As detailed in [19], [20], and [21], KMF operates on a learn-as-you-go premise by providing a robot with test motions, or actuation primitives, and then recording the resulting motion of the end-effector after the primitives are applied. A collection of these exploratory primitives across the robot's task space can then be used to approximate the local kinematics and dynamics of the system and provide a best-fit approximation of what actuation would provide desired motion. After the conclusion of each motion, the resulting movement is compared to the anticipated motion of the end-effector in order to evaluate the accuracy of the approximated "kinodynamic" model. If significant difference exists between expectation and reality, the algorithm triggers a new exploratory phase in order to better sample the local space.

In this work, we use a slightly modified implementation of that proposed in the original KMF works. First, we start with the premise of *actuation primitives*: a control signal $\tau(t)$, in this case either a voltage or pressure, that varies as a function of time:

$$\tau(t) = \begin{cases} \tau_p & \text{if } t \in [t_0, t_0 + d_p) \\ 0 & \text{if } t \in (-\infty, t_0) \cup (t_0 + d_p, \infty) \end{cases}, \quad (1)$$

where τ_p is defined to be the magnitude of the actuation primitive and d_p is the duration of the primitive. The value t_0 denotes the start time of the action. In this implementation, the value d_p is constant at 1 second throughout execution, and all individual actuation primitives share the same start value t_0 for each separate motion. Throughout the execution of KMF, the controller is recording the set p_i of all meaningful actuation primitives executed on the robot, including primitives from both exploration behavior and model predicted behavior.

Next, we describe the process of using the collected data set p_i to produce desirable actuation primitives that will drive the end-effector to a goal location. In this implementation, we will generalize the dimension of our actuator primitives, and subsequent results, to match our later implementation on hardware. To begin, let $\hat{p}$ be an actuation primitive whose parameters $\tau(\hat{p})$ will cause the end-effector to move towards a desired goal. We assume no knowledge about the robot's kinematics or dynamics, and given this, we must estimate the values $\tau(\hat{p})$ that will give us the desired motion. The desired primitive (b_1) consists of n elements – one for each DoF:

$$b_1 = \begin{bmatrix} \tau_p^1(\hat{p}) \\ \tau_p^2(\hat{p}) \\ \vdots \\ \tau_p^n(\hat{p}) \end{bmatrix}. \quad (2)$$

For traditional rigid-link robots, the number of elements in the primitive is also equivalent to the number of actuators. In this implementation, as with the initial implementation, the estimation of $\hat{p}$ is to be determined as a linear combination of the k-nearest neighbor (k-NN) primitives previously executed and saved in the controller's memory. These k-NN primitives are selected according to the distance between the current end-effector location and the starting position of each primitive executed in memory. The resulting k-NN primitives are labeled as $p_1 \dots p_k$. The linear combination of these k-NN primitives can be expressed in the matrix form:

$$A_1 x = b_1 \quad (3)$$

where $x = [x_0, x_1, \dots, x_k]^T$, is an as yet unknown weight vector. The matrix A_1 contains the parameters of the k-NN primitives:

$$A_1 = \begin{pmatrix} 1 & \tau_p^1(p_1) & \tau_p^1(p_2) & \dots & \tau_p^1(p_k) \\ 1 & \tau_p^2(p_1) & \tau_p^2(p_2) & \dots & \tau_p^2(p_k) \\ & \vdots & & & \\ 1 & \tau_p^n(p_1) & \tau_p^n(p_2) & \dots & \tau_p^n(p_k) \end{pmatrix}_{n \times (k+1)}, \quad (4)$$

where $\tau_p(p_i)$ is the magnitude of the i-th actuation primitive. We solve for the unknown coefficients x_i using readily available information concerning the results of our previously experienced k-NN actuation primitives. We thus describe the matrix:

$$A_2 = \begin{pmatrix} 1 & \Delta x(p_1) & \Delta x(p_2) & \dots & \Delta x(p_k) \\ 1 & \Delta y(p_1) & \Delta y(p_2) & \dots & \Delta y(p_k) \\ & \vdots & & & \\ 1 & \Delta z(p_1) & \Delta z(p_2) & \dots & \Delta z(p_k) \end{pmatrix}_{3 \times (k+1)}, \quad (5)$$

in which $[\Delta x(p_i) \Delta y(p_i) \Delta z(p_i)]^T$ is the relative displacement experienced by the end-effector upon execution of the

primitive p_i. Utilizing knowledge of both the manipulator's current end-effector location and the location of the goal destination, we can choose a simple desired displacement for the end-effector to move towards the goal. If the distance between the end-effector is sufficiently small, we can choose the next desired motion to move directly to the desired goal or take an incremental step towards our goal. Regardless, the desired motion (b_2) is summarized as a relative displacement of the end-effector in global coordinates:

$$b_2 = \begin{bmatrix} \Delta x(\hat{p}) \\ \Delta y(\hat{p}) \\ \Delta z(\hat{p}) \end{bmatrix} \tag{6}$$

After designating our desired motion, we can calculate the coefficients $\{x_i\}$ by solving for x in the equation:

$$A_2 x = b_2 \tag{7}$$

As discussed in [19], the rank of matrix A_2 is not guaranteed to be full, allowing variability in the solution for x. We once again solve this problem using least squares regression to find a best-fit approximation for x. Once calculated, we can use equation 3 to find the desired primitive parameters $\tau_p(\hat{p})$ in b_1. One final adjustment is the weighting of the k-NN primitives according to the distance between the current end-effector location and the starting location of each primitive. By adding this set of weights, $w_i \ldots w_k$, we obtain a weighted least squares solution when solving 7. Thus, equations 3 and 7 are adjusted as follows:

$$\begin{cases} A_1 W x = b_1 \\ A_2 W x = b_2, \end{cases} \tag{8}$$

where W = diag(1,$w_1, w_2, \ldots, w_k$).

Continuum OctArm Implementation

The OctArm, illustrated in Figure 1, is a 3-section, 9 DoF continuum robot, actuated by pneumatically driven McKibben actuators. Pressuring the McKibben muscle creates extension along the length of an individual muscle. By connecting three of these muscles (or sets of them at equal intervals) in parallel to form a continuum section, we can extend said section by pressurizing all muscles simultaneously by the same amount or create bending by differentially pressurizing the muscles. The OctArm is comprised of three of these serially connected sections, with each section actuated by three independently controlled pressures, and capable of independent extension and two DoF of spatial bending, providing the OctArm 9 DoF overall. The overall length of the OctArm can range from 1.07m to 1.38m.

In this work, we choose to model the continuum kinematics using the model described in [5]. This constant-curvature model describes a single extensible continuum section using 3 values: 2 bending magnitudes u and v, and arc-length s. The bending magnitudes convey bending along two orthogonal axes, u about the local X-axis, and v about the local Y-axis. The combined magnitude of u and v gives the overall bending magnitude of the section and their relative value gives the direction of bending as described by $\phi = \tan^{-1}(\frac{v}{u})$. We choose this kinematic description from among the other valid models due to the simplicity of mapping from kinematic values to actuator values, as provided in [22] and reproduced

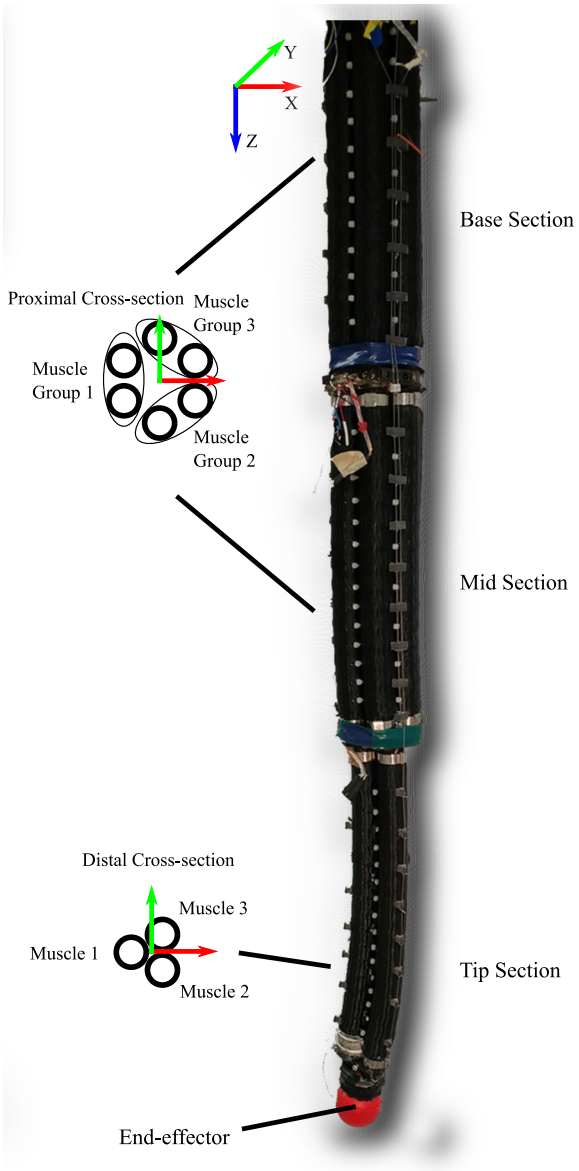


Figure 1. OctArm Continuum Manipulator

below:

$$l_1 = s(t) - d \cdot v(t) \tag{9}$$

$$l_2 = s(t) + \frac{d \cdot v(t)}{2} + \frac{\sqrt{3} d \cdot u(t)}{2} \tag{10}$$

$$l_3 = s(t) + \frac{d \cdot v(t)}{2} - \frac{\sqrt{3} d \cdot u(t)}{2} \tag{11}$$

KMF for continuum manipulator

As outlined in the initial development of KMF, the action primitives, $\tau(t)$, supplied by the method are not limited by action type or actuation method. When adapted to continuum robots, there are two main types of actuation to be considered: tendon driven devices actuated through electric motors and pneumatically driven artificial muscles. The cases of both extensible and non-extensible manipulators also need to be taken into account. In considering non-extensible, tendon driven robots, one cannot simply pull on a single tendon

and achieve desired motion without first or simultaneously letting all opposing tendons go slack or at least reduce tension in an amount proportional to the tendon being pulled on. This means that for fixed length robots, we need to address coupling for the robot by applying differential actuation.

Following from this idea of differential actuation, we make use of the kinematics mentioned in equations 9-11 to relate individual actuator values to KMF primitives. Here, for both extensible and non-extensible continuum robots, we can map one primitive to the kinematic value u to cause differential bending along the local Y-axis, as related by the sign change of the coefficient for u in actuators 2 and 3. Likewise, we can map another primitive to the value v to drive differential bending along the local X-axis. Exclusively for extensible continuum manipulators, a final primitive can be mapped to the arc-length value s that is present for all actuators. More explicitly for the OctArm, we can increase and decrease pressure to respective muscles using the following equations:

$$p_1 = \int \tau_s(t)dt - d\int \tau_v(t)dt \tag{12}$$

$$p_2 = \int \tau_s(t)dt + \frac{d}{2}\int \tau_v(t)dt + \frac{\sqrt{3}d}{2}\int \tau_u(t)dt \tag{13}$$

$$p_3 = \int \tau_s(t)dt + \frac{d}{2}\int \tau_v(t)dt - \frac{\sqrt{3}d}{2}\int \tau_u(t)dt \tag{14}$$

where p_i is the pressure value for muscle i in a section, and τ_u, τ_v, and τ_s are the primitives mapped to the kinematic values u, v, and s, respectively.

Mapping KMF to OctArm DoF

In order to implement KMF for the 9 DoF OctArm and observe the efficacy of the method, we began with a simplified mapping of KMF primitives to OctArm DoF. Here, we present 3 mappings of KMF primitives to OctArm DoF, gradually increasing the complexity and redundancy of the system in order to assess KMF.

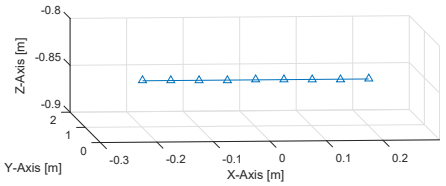
Mapping 1: 3 DoF— In this first mapping, we treat the OctArm as a single continuum section, providing identical inputs to muscle 1 of each section, and the same setup for muscles 2 and 3. Given that the OctArm is extensible by design, we can use the primitive mapping described for the basic extensible continuum section:

$$[\tau_1, \tau_2, \tau_3] = [\tau_u, \tau_v, \tau_s] \tag{15}$$

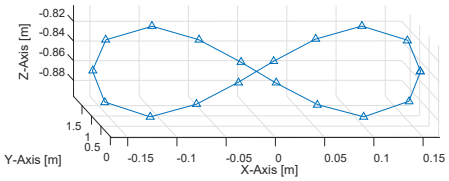
This mapping accentuates the under-actuated nature of continuum robots by displaying non-constant curvature. Non-constant curvature will be especially prevalent in the proximal (base) section, which must support the load of both the middle and distal sections. Even in this reduced number of DoF, traditional task space planning and control methods potentially suffer from modeling and sensing errors.

As noted when first reporting KMF, the order in which the primitives are arranged in this mapping, and subsequent mappings, is not important to the method. The order provided is simply for reporting purposes and for clarity of the mapping.

Mapping 2: 4 DoF—For the second mapping, we treat the OctArm as a non-extensible, 2-section continuum manipulator. We accomplish this by treating the base and mid-sections



(a) Line path in Kinect coordinate system



(b) Lemniscate path in Kinect coordinate system

Figure 2. Nominal Trajectories for End-effector tracing

of the OctArm as one section, receiving matching inputs to each of the muscle groups as in the first mapping. To simulate non-extensibility, we initialize each OctArm section to their respective mid-range extension value, essentially halfway between their minimum and maximum pressure. After initialization, only differential inputs utilizing u and v are given to the system. This provides the system with 4 DoF, with the primitive vector:

$$[\tau_1, \tau_2, \tau_3, \tau_4] = [\tau_{u_b}, \tau_{v_b}, \tau_{u_t}, \tau_{v_t}] \tag{16}$$

The second mapping serves to demonstrate using KMF in the non-extensible class of continuum robots. It also serves to introduce redundancy into the system.

Mapping 3: 6DoF—The final mapping makes use of each section of the OctArm independently of the others. As with the second mapping, we model the sections as constant length and provide only differential inputs to each section, giving the system a total of 6 DoF. Our primitive vector for this case is:

$$[\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6] = [\tau_{u_b}, \tau_{v_b}, \tau_{u_m}, \tau_{v_m}, \tau_{u_t}, \tau_{v_t}] \tag{17}$$

This mapping represents a 3-section non-extensible continuum robot, and implements the largest degree of redundancy explored in this work.

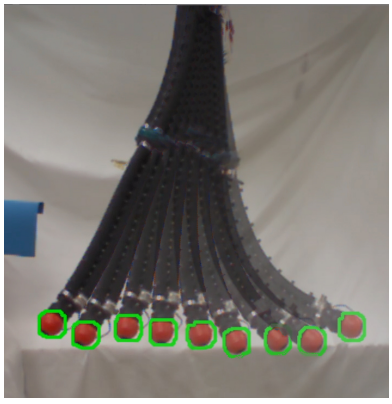
3. Experimental Validation

In evaluating the efficacy of KMF across the different mappings, we consider two paths in the OctArm's task space for the end-effector to follow. As part of a class of robots whose hardware naturally traces curves and bending motions, one of the most difficult motions for a continuum robot to perform for the end-effector is a straight line. Consequently, the first path we test is a straight line that runs through the taskspace, parallel to the x-axis. The path can be seen in Figure 2a. The exact path is a series of 9 points spaced evenly between [x,y,z] = [-0.2m,0.95m,-0.85m] and [0.2m,0.95m,-0.85m] in the camera's coordinate frame. In executing the path, the OctArm must travel to the start point (x=0.2m) then follow the full length of the path and back again to the start of the line.

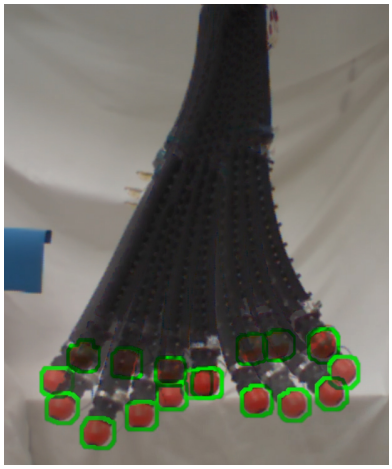
The second path considered is a planar lemniscate path sitting a plane that is parallel, but offset, from the OctArm's local

XZ-plane. The path consists of 19 discrete points (the start and end point are the same), shown in Figure 2b, and spaced approximately 5cm apart. For both paths, the end-effector must arrive within 2cm of error (as depicted in the plots by the green region) of the current goal point before moving to the next. An example of the OctArm executing both the line and lemniscate path can be seen in Figure 3.

Tracking of the OctArm end-effector along the desired trajectories is achieved through the use of a Microsoft Kinect [23] to track the center of the OctArm end-effector using hue filtering. We then use the Kinect API to map color pixel coordinates to depth values and consequently to real-world coordinates. Goal points along the desired paths are provided through a MATLAB script, with the coordinates themselves set with respect to the Kinect coordinate system. Kinect tracking is implemented in C++ using Visual Studio 2015, and the same program also sends the end-effector coordinates to a Simulink model via network socket. A second Simulink model is responsible for integrating the actuation primitives into pressure values that drive the pressure regulators controlling the OctArm. Finally, the MATLAB script that provides the goal locations as the end-effector moves through the task space also hosts the core KMF algorithm by receiving end-effector location, calculating and recording primitives and motions, and providing primitives to the Simulink model driving the OctArm.



(a) OctArm following line path through task space.



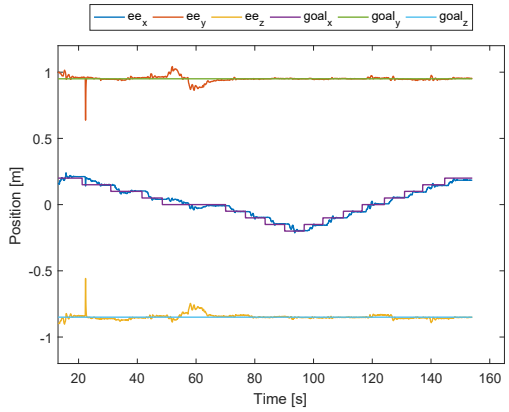
(b) OctArm following lemniscate path in task space.

Figure 3. OctArm Manipulator tracing paths in task space.

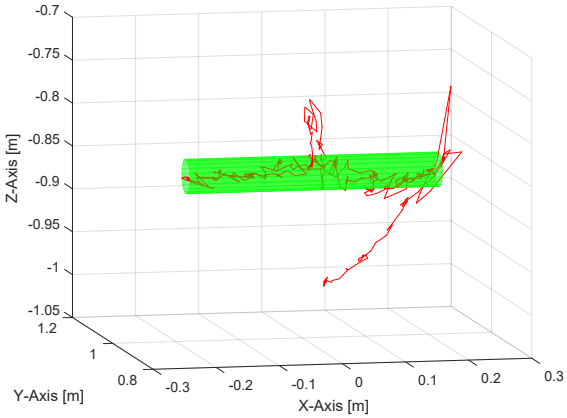
Results: 3 DoF Mapping

In implementing the first mapping, the KMF method was tasked with solving what would ideally be a relatively simple mapping of 3 DoF to 3 dimensions of movement for the end-effector. In reality, and as mentioned briefly when introducing the mapping, the compliance of the OctArm and the fact that the system is under the effects of gravity makes this a difficult problem to model, much less solve. In observing robot performance under KMF, seen in Figures 4 and 5, it can be seen that with sufficient exploration time the method is able to track both test trajectories from Figure 2 well and in reasonable time.

For the line trajectory, as seen in Figure 4a, starting from the time of arrival at the first point, KMF actuates the robot to step along the remaining 16 consecutive points with little to no error outside of the 2cm limit in under 3 minutes. This value is more impressive when considering that the system is paused (not a result of KMF) 3 seconds between reaching each goal point and starting to move to the next point on the path (1 second to ensure that system is settled, 2 seconds for clarity of the result).



(a) Measured end-effector and goal position while following line path (3DoF)

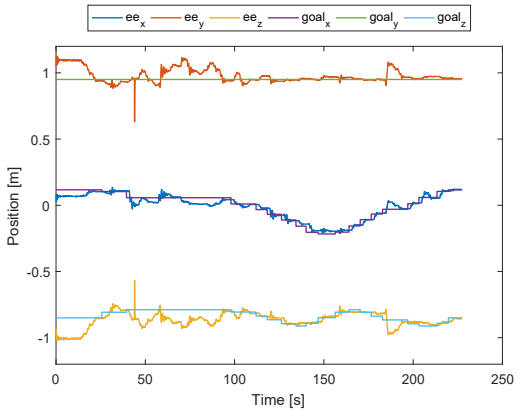


(b) Traversed path through task space with relevant margin for error (3DoF)

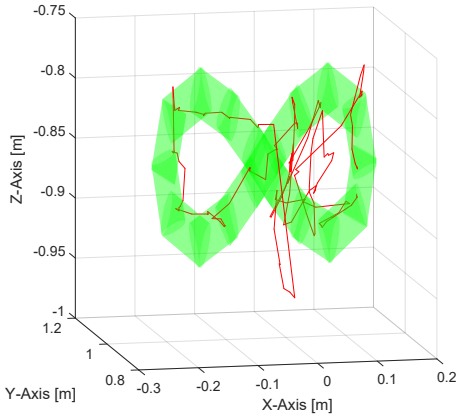
Figure 4. 3DoF OctArm line following

The 3 DoF mapping also proves capable of completing the lemniscate path, which requires additional motion compen-

sating against gravity. Here, we see more travel outside of the nominal path than we expect between consecutive points, but still have eventual recovery and convergence to each point. The travel outside the nominal path could potentially be improved either through smaller steps along the path or providing further exploration in this region of the task space.



(a) Measured end-effector and goal position while following lemniscate path (3DoF)



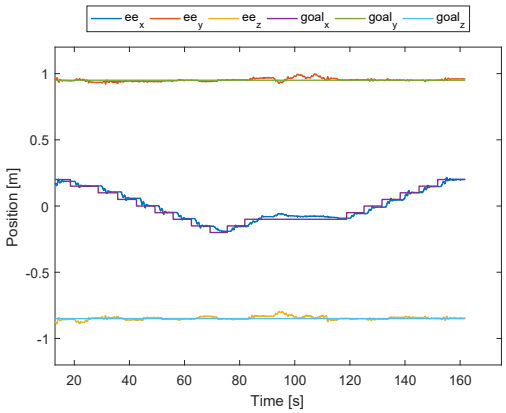
(b) Traversed path through task space with relevant margin for error (3DoF)

Figure 5. 3DoF lemniscate path following

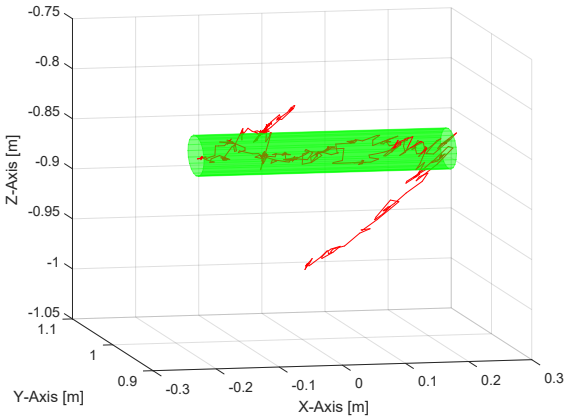
Results: 4 DoF Mapping

In implementing the second mapping, there was uncertainty about how introducing redundancy would impact the performance of the algorithm. In observing the results in Figures 6 and 7, we can clearly see that, at least for these two paths, the addition of another DoF did not greatly hinder the performance. While not shown in these snapshot results, it was noticeable during training that it took longer for the 4 DoF mapping to obtain enough experience to traverse the space intentionally. This is due in part to the fact that the 4-dimensional exploration primitives, even when orthogonal in primitive space, could not guarantee a diverse set of motion in the task space. The same can be said for the previous 3 DoF mapping given the non-linearity of the system, but the extra degree of redundancy does not appear to help exploration.

When comparing the results of the 3 and 4 DoF mappings at following the line path, there is little discernible difference between the two performances. In general, we observe that both mappings can still make inaccurate predictions at times, depending on the closest neighbor candidates at each given point, but both have similar completion time.



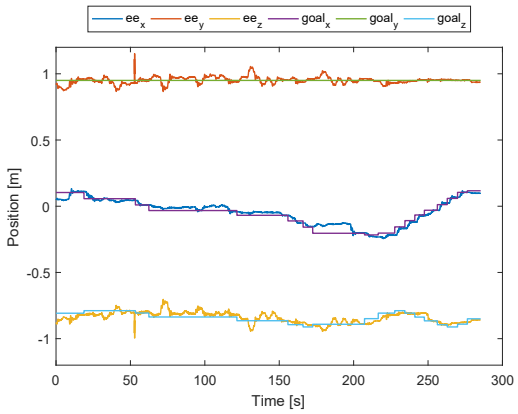
(a) Measured end-effector and goal position while following line path (4DoF)



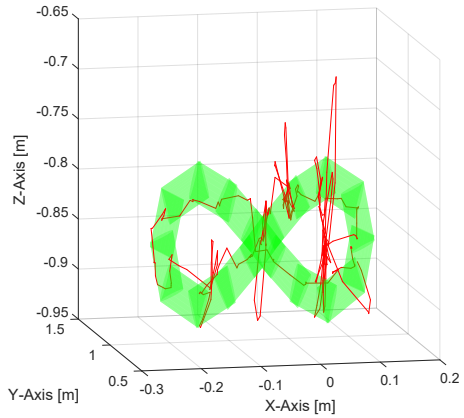
(b) Traversed line path through task space with relevant margin for error (4DoF)

Figure 6. 4DoF line following

Likewise for the lemniscate path, we see similar times of completion and accuracy of performance between the 3 and 4 DoF mappings. It is worth noting for later discussion that both mappings appear to deviate less from the acceptable region of error in the left loop of the path.



(a) Measured end-effector and goal position while following lemniscate path (4DoF)



(b) Traversed lemniscate path through task space with relevant margin for error (4DoF)

Figure 7. 4DoF lemniscate following

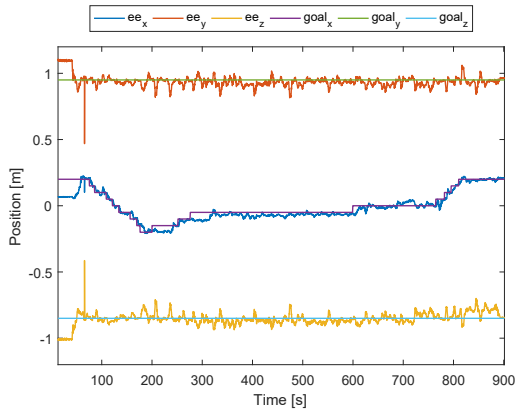
Results: 6 DoF Mapping

As with the 4 DoF mapping, it was unknown exactly what impact the extra redundancy would have on the performance, though it was anticipated that 6 DoF would be a greater challenge in generating a solution than the 4 DoF mapping, as is often the case with motion planning involving increased degrees of freedom. As can be seen in Figure 8a, the first notable impact is on completion time. The result seen here is one of faster examples we recorded using this mapping, but is still 4-5 times slower than the 3 and 4 DoF mappings. In observing Figure 8b, it becomes apparent that a large portion of that time difference is spent re-exploring and miscalculating primitives in an already explored region.

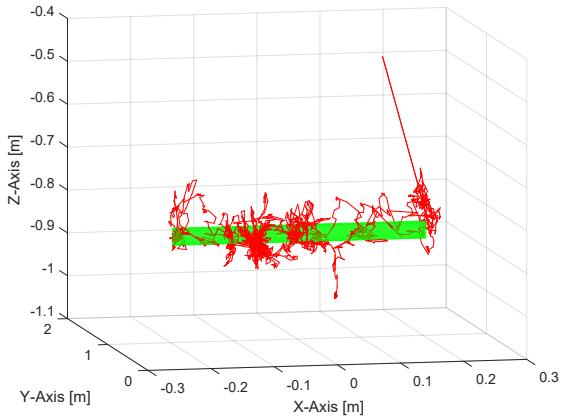
The 6 DoF mappings suffers from the same issue observed in the 4 DoF mapping in that there is no guarantee of sufficient task space exploration to use for predictions when given a diverse, or even orthogonal, set of actuation primitives. Predictably, where the 6 DoF mapping appears to suffer more than the 4 DoF mapping relates to the extra increase in redundancy. The extra redundancy increases the chance that two neighboring primitives could have been sampled from very different and relatively remote regions of the robot's configuration space.

Fortunately, KMF has already been shown to be able to adapt to changes in kinematics while actively tracking and learning. This is evidenced by the fact that the 6 DoF mapping is still able to converge to each point along the path, even if taking a different path between each point.

Given the difficulty of solving the line path after several learning trials in this example, we forgo attempting to trace the lemniscate path with the 6 DoF mapping for this reporting.



(a) Measured end-effector and goal position while following line path (6DoF)



(b) Traversed line path through task space with relevant margin for error (6DoF)

Figure 8. 6DoF line following

4. Discussion

In [20], the authors explored how KMF is capable of overcoming various unknown kinematic constraints and adjustments. In this work, we did not actively pursue adding unmodeled constraints to our assessment, but during our final testing, the results of which are presented in the previous section, the OctArm developed a considerable leak in muscle 1 of the distal section. The leak results in a considerable curve in the section when prior to the leak the section would be straight. Normally, such a leak would be catastrophic to control systems and models and prevent proper function, but in the case of KMF, this leak did not prove detrimental to the performance. As noted earlier, it is likely that this leak contributed to the difference in how well the OctArm follows the left-half loop of the lemniscate path versus the right-hand loop as the leak would reduce the predictability of motion in

the right-hand side of the task space.

In this initial reporting, the results demonstrate the capability of KMF to track end-effector locations along a path without placing limitations on memory. In practice, we allowed the method to keep all attempted primitives, regardless of whether they acted as predicted or not. Thus, the results presented here contain memory from a number of "good" and "bad" motions, collected over a series of attempts. The build-up of memory over multiple trials corresponded to a decrease in time of execution, meaning that with more memory and more diverse primitives, the OctArm traversed the paths faster, until the path could be traveled with little to no miss-steps. As such, the number of primitives and the density of the primitives available varied by both the path, number of completions, and the number of DoF. For example, the 3 DoF line path could be traversed with about two hundred primitives in memory surrounding the volume of the path while the lemniscate path takes several hundred more for execution. However, execution time for both paths greatly reduced when traversing the path multiple times. Future work will look to limit memory density across the entire task space by either replacing old primitives or by adding primitives that increase the diversity of motions in localized volumes of the task space.

With regards to increasing redundancy in our results, KMF proved to be able to enable the OctArm manipulator to arrive at points in the task space regardless of the number of DoF. Where redundancy does become a factor is in the transitioning between two consecutive points, which can vary depending on the configuration of the OctArm at the start of the motion. In the case of 3 DoF and 4 DoF, we observe desirable motion between discrete points in both paths, generally staying within the desired error. For 6 DoF, we observe greater variation in the motion between points and the configurations themselves at each point in time. While this trend is to be expected for more complex scenarios, we still observe the ability of KMF to improve performance as the system explores more of the task space. This is another example of KMF adapting to new changes in the local kinodynamics of the OctArm, as a new arrival at the same point in the task space no longer means the same kinematic structure of the OctArm. Future work will explore the use of time based memory to give more weight to more recent primitives as well as location in space, as these could be more relevant to the current robot configuration.

One challenge to this work, and to tracking end-effector's from a single camera in general, was the need to avoid occlusion of the end-effector from the Kinect camera. We accomplish this by implementing artificial bending limits at the actuator level, while still allowing KMF to be unaware of the system state. The limit itself is created by applying a dynamic saturation level for the maximum and minimum pressure for muscles 2 and 3, respectively. Since muscles 2 and 3 are responsible for bending towards and away from the camera, this dynamic saturation limit essentially prevents muscle 2 from being pressurized significantly greater than muscle 3 and likewise prevents muscle 3 from reducing pressure to be significantly below that of muscle 2. In future applications, this challenge of occlusion could be solved by using an array of cameras to track the end-effector.

As alluded to when introducing the respective mappings for the OctArm, part of the aim of this work is to display the applicability of KMF to continuum robotics beyond the OctArm. The results here show that KMF is capable of handling both the theoretical hyper-redundancy of continuum robots as well as designed redundancy of continuum manipulators with multiple sections. These results also lend credit to the idea that KMF primitives could be used as torque inputs to tendon driven continuum robots that only have the ability to bend and thus any inputs must be simultaneously pulling and releasing tendon.

Benchmarking KMF for Continuum Robots

In providing comparison for KMF performance to more traditional methods for closed-loop control of continuum robots, we discuss a collection of results from the literature that highlight both the success of KMF robot control in this paper and the difficulty of trying to directly compare the success with other methods on both the OctArm and similar hardware. To begin, we look at two works that explore the accuracy of forward kinematic models with respect to end-effector location explicitly performed on the OctArm manipulator. In [24], a series of forward kinematic models based upon constant curvature assumptions are directly compared for accuracy relative to real-world measurements of the OctArm's end-effector. The results here show that while some models perform better than others, the greatest accuracy seen across the samples of the robot workspace was in excess of 5% (as measured with respect to the OctArm's overall length), or over 5cm. These models also rely on the internal measurement of the OctArm's shape, giving no direct way to accurately relate assumption based measurements to real-world coordinates. In [25], the work compares the accuracy of constant curvature kinematics to statics-based models that are derived to be geometrically exact and provide a relationship between input pressure and end-effector location for the OctArm. In this work, the geometrically exact models prove to be better at predicting gravity related deflections for the OctArm than constant curvature models, but still provides approximately 5% error between the predicted and actual end-effector locations, and does not provide a means for mapping real-world coordinates to actuator inputs for control.

Next, we discuss a series of works that explore predicting and controlling end-effector locations of continuum robots, both in static and dynamic experiments. The works utilize various continuum robots with a variety of material and dynamic model related parameters. In [26], a combination of variable curvature Cosserat-rod based static and Lagrange dynamic models are presented to provide control for a two-section continuum manipulator that is similar in composition to the OctArm, containing a mixture of rigid and soft materials and using pneumatic actuation. While reporting much improved accuracy for their model in comparison to the previous state of the art, they report 6-8% error in static experiments and excess of 16% error in dynamic motions of the manipulator's end-effector. Another relevant work, [27], describes the statics and dynamics of a tendon driven robot with a flexible backbone through a combination of Cosserat-rod and Cosserat-string models. This work reports a 1.7% error between the predicted and measured end-effector location in static experiments for a single-section continuum robot, but does not provide a method for predicting actuator inputs for a desired end-effector location, which is likely not a unique solution for multi-section continuum robots. Finally, in [28], a non-dynamic model based approach based on forward and inverse kinematics coupled with an adaptive neural network control is tested. The combination of forward kinematics and the adaptive neural control provide for approximately 1% error in end-effector location when tracking multi-point paths. The paths are created using the robot's forward kinematics and are well suited for a continuum robot's inclination for

following curved paths.

In all, these results from the literature, among others, show various results below a 10% error margin with respect to robot length but also carry higher computational, sensing, and modeling costs. Thus, KMF robot control presented here establishes that reasonable tracking error for notably difficult motions and static end-effector location control ($\leq$2% error) can be obtained with this class of robot with a simple input-output framework and end-effector tracking through a 3D camera or network of standard cameras.

Expanding KMF to Space Deployment

One of the many potential advantages of KMF with respect to continuum robot automation, and robot automation in general, derives from the simplicity of the system required to implement it. Traditional robot control and planning relies on sensing, often on-board, to close the control loop and provide the system state to various models. In long-term deployment environments such as the International Space Station (ISS), the future Lunar Gateway, or even extra-planetary bodies, critical sensor malfunctions can be catastrophic to a mission.

In the case of our ongoing research, one specific continuum robot in our lab, the Tendril robot [29], is designed for applications concerning inspection and monitoring in Space operations. One of the target scenarios for this robot is deployment on the ISS for automated inspection of hard-to-reach locations and commensurately reducing astronaut workload. An example simulation of this scenario with Tendril deployed in and looking around the ISS can be seen in Figure 9. The simulated Tendril actuation has been driven by the KMF approach reported herein. Both aboard the ISS and the future Lunar Gateway, there are potential times when a deployed robot cannot be serviced, even for a faulty sensor, endangering mission success. KMF offers an alternative loop for maintaining these automated systems.

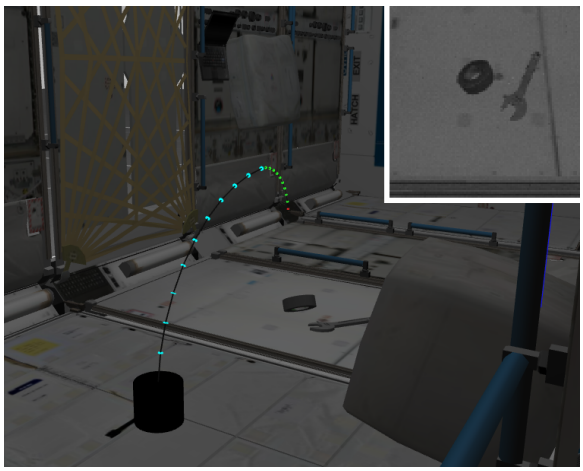


Figure 9. Tendril Continuum Manipulator in Simulated Inspection Task on ISS

5. CONCLUSION

Continuum robots, via their inherent compliance and ability to change shape throughout their structure, have the potential to perform tasks in congested and sensitive Space environments that are beyond the capabilities of traditional rigid-link robots. However, their dynamics and modeling are more complex than traditional robots, which complicates their modeling, sensing, planning, and control.

We report on experiments extending the Kinematic-Model-Free (KMF) robot control method to continuum robots, and highlight its ability to capture the kinodyamics of a continuum robot system in various parts of the task space. The success of the method shows that a continuum robot can "learn" enough information from an external camera to reach and track desired points and trajectories, without needing knowledge of exact shape or position of the robot. Experiments with a three section, nine Degree of Freedom (DoF) continuum robot are supported by simulations of continuum robot inspection on the International Space Station (ISS).

ACKNOWLEDGMENTS

This research was supported by NASA Space Technology Research Fellowship contract 80NSSC17K0173, and by the U.S. National Science Foundation under grants CMMI-1924721 and IIS-1718075.

REFERENCES

[1] G. Chirikjian and J. Burdick, "A modal approach to hyper-redundant manipulator kinematics," *IEEE Trans. Robot. Autom.*, vol. 10, no. 3, pp. 343–354, Jun. 1994.

[2] I. Walker, "Continuum robot appendages for traversal of uneven terrain in in-situ exploration," in *Proc. IEEE Aerospace Conf.*, Big Sky, MT, 2011, pp. 1–8.

[3] D. Nahar, P. Yanik, and I. Walker, "Robot tendrils: Long, thin continuum robots for inspection in space operations," in *Proc. IEEE Aerospace Conference*, Big Sky, MT, 2017, pp. 1–8.

[4] B. Jones and I. Walker, "Kinematics for multisection continuum robots," *IEEE Trans. Robot.*, vol. 22, no. 1, pp. 43–57, Feb. 2006.

[5] W. Felt, M. T. andT. Allen, G. Hein, J. Pompa, K. Albert, and D. Remy, "An inductance-based sensing system for bellows-driven continuum joints in soft robots," in *Robotics: Science and Systems*, The Hague, Netherlands, 07 2017.

[6] T. Mahl, A. Hildebrandt, and O. Sawodny, "A variable curvature continuum kinematics for kinematic control of the bionic handling assistant," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 935–949, 2014.

[7] I. Gravagne and I. D. Walker, "Manipulability, force, and compliance analysis for planar continuum manipulators," *IEEE Trans. Robots. Autom.*, vol. 18, no. 3, pp. 263–273, Jun. 2002.

[8] D. Rucker and R. Webster III, "Deflection-based force sensing for continuum robots: A probabilistic approach," in *Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst.*, San Francisco, CA, 2011, pp. 3764–3769.

[9] D. C. Rucker, B. A. Jones, and R. J. Webster III, "A geometrically exact model for externally loaded concentric-tube continuum robots," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 769–780, 2010.

[10] S. Sadati, Y. Noh, S. Naghibi, and A. Althoefer, "Stiffness control of soft robotic manipulators for minimally invasive surgery (mis) using scale jamming," in *Int. Conf. Rob. and Autom.*, Amsterdam, The Netherlands, 2015, pp. 141–151.

[11] W. S. Rone and P. Ben-Tzvi, "Continuum robot dynamics utilizing the principle of virtual power," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 275–287, 2013.

[12] E. Tatlicioglu, I. Walker, and D. Dawson, "Dynamic modeling for planar extensible continuum robot manipulators," *Int. Jour. Robot. Autom.*, vol. 24, no. 4, pp. 1087–1099, Apr. 2009.

[13] C. Wang, J. Wagner, C. G. Frazelle, and I. D. Walker, "Continuum robot control based on virtual discrete-jointed robot models," in *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2018, pp. 2508–2515.

[14] C. Wang, C. G. Frazelle, J. R. Wagner, and I. Walker, "Dynamic control of multi-section three-dimensional continuum manipulators based on virtual discrete-jointed robot models," *IEEE/ASME Transactions on Mechatronics*, 2020.

[15] A. Kuntz, A. W. Mahoney, N. E. Peckman, P. L. Anderson, F. Maldonado, R. J. Webster, and R. Alterovitz, "Motion planning for continuum reconfigurable incisionless surgical parallel robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6463–6469.

[16] A. Ataka, P. Qi, H. Liu, and K. Althoefer, "Real-time planner for multi-segment continuum manipulator in dynamic environments," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4080–4085.

[17] S. Satheeshbabu, N. K. Uppalapati, G. Chowdhary, and G. Krishnan, "Open loop position control of soft continuum arm using deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5133–5139.

[18] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 124–134, 2018.

[19] P. Kormushev, Y. Demiris, and D. G. Caldwell, "Encoderless position control of a two-link robot manipulator," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seattle, Washington, 2015, pp. 943–949.

[20] ——, "Kinematic-free position control of a 2-dof planar robot arm," in *Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst.*, Hamburg, Germany, 2015, pp. 5518–5525.

[21] A. AlAttar and P. Kormushev, "Kinematic-model-free orientation control for robot manipulation using locally weighted dual quaternions," *Robotics*, vol. 9, no. 4, p. 76, 2020.

[22] C. G. Frazelle, A. D. Kapadia, and I. D. Walker, "A haptic continuum interface for the teleoperation of extensible continuum manipulators," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1875–1882, 2020.

[23] Microsoft. (2020, October) Kinect for windows. Https://developer.microsoft.com/en-us/windows/kinect/.

[24] A. Chawla, C. Frazelle, and I. Walker, "A comparison of constant curvature forward kinematics for multisection continuum manipulators," in *2018 Second IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2018, pp. 217–223.

[25] D. Trivedi, A. Lofti, and C. Rahn, "Geometrically exact dynamic models for soft robotic manipulators," in *Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst.*, San Diego, CA, 2007, pp. 1497–1502.

[26] S. M. H. Sadati, S. E. Naghibi, I. D. Walker, K. Althoefer, and T. Nanayakkara, "Control space reduction and real-time accurate modeling of continuum manipulators using ritz and ritz–galerkin methods," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 328–335, 2018.

[27] D. C. Rucker and R. J. Webster III, "Statics and dynamics of continuum robots with general tendon routing and external loading," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1033–1044, 2011.

[28] A. Melingui, O. Lakhal, B. Daachi, J. B. Mbede, and R. Merzouki, "Adaptive neural network control of a compact bionic handling arm," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 6, pp. 2862–2875, 2015.

[29] M. Wooten, C. Frazelle, I. D. Walker, A. Kapadia, and J. H. Lee, "Exploration and inspection with vine-inspired continuum robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–5.

BIOGRAPHY

Chase Frazelle *received his B.Sc. and M.S. degree in electrical engineering from Clemson University, Clemson, SC, in 2015 and 2017, respectively, and is currently pursuing his PhD in electrical engineering at Clemson. His research is focused on the control and modeling of extensible continuum robots with the aim to improve the reliability and automation of such manipulators for space exploration and investigation. Other research interests include human-robot interfacing, embedded systems, and interaction with distributed systems.*

Ian D. Walker *received the B.Sc. Degree in Mathematics from the University of Hull, England, in 1983 and the M.S. and Ph.D. Degrees in Electrical and Computer Engineering from the University of Texas at Austin in 1985 and 1989, respectively. He is currently a Professor in the Department of Electrical and Computer Engineering at Clemson University. Professor Walker's research centers on robotics, particularly novel manipulators and manipulation. His group conducts basic research in the construction, modeling, and application of biologically inspired "trunk, tentacle, and worm" robots.*

Petar Kormushev *holds a PhD in Computational Intelligence from Tokyo Institute of Technology, an MSc in Artificial Intelligence, an M.Sc. in Bio- and Medical Informatics, and a B.Sc. in Computer Science. Dr Kormushev's research interests include machine learning and robot learning algorithms, especially reinforcement learning for intelligent robot behavior.*

Ahmad AlAttar *completed his Bachelor's degree with Honours in Mechatronic Systems Engineering in Simon Fraser University, Canada, in 2017 and his M.S. in Electronic Engineering with Management at King's College London in 2019. He is currently pursuing a PhD at Imperial College London and his research interests are robot learning, model-free control, and deep reinforcement learning.*