Smooth Path Planning for Continuum Arms

Brandon H. Meng, Isuru S. Godage, and Iyad Kanj

Abstract— **Continuum arms, with their mix of compliance, payload, safety, and manipulability, are perfectly suited to serve as co-robots, and their applications range from industry and manufacturing to human healthcare. Their hyper-redundancy serves as their most significant challenge for path planning and path planning approaches commonly used with rigid-link robots, such as inverse kinematics, that fail to provide reliable trajectories for continuum arms.**

We propose an Inverse Kinematics-based approach to address the limitations of previously-proposed Kinematics-based approaches. Using this new approach, we are able to efficiently generate very rich sets of configurations, which, in turn, lead to smooth path planning for such continuum manipulators.

To validate the smoothness of the paths generated by our approach, we apply dynamics constraints to the generated trajectories. We show that, when tracked by a controller, the paths that are generated using the proposed approach are much smoother than previously-proposed Kinematics-based approaches: The proposed approach allows the continuum arm to traverse the trajectories very accurately and in time less than half of that taken by previous (reliable) path planning approaches.

I. Introduction

Continuum arms are "bio-inspired" manipulators that seek to find a middle ground between rigid and soft robots. They are made of multiple sections, with each being a bundle of compliant actuators, such as a pneumatic muscle actuator. Figure 1 shows an example of a 3-section continuum arm at rest and with some of the pneumatic muscle actuators extended. Due to their mix of passive compliance, payload, inherent safety, and dexterous manipulability, these continuum systems are well suited to serve as co-robots [1]. Their applications range from industry and manufacturing to healthcare. However, the characteristics that make these devices promising also present significant challenges. As such, many of these devices have not yet reached maturity, and research on continuum arms has witnessed rapid growth in recent years [2]–[5].

In this paper, we study smooth path planning for continuum arms, where the goal is to compute a trajectory for the arm to traverse while avoiding any obstacles, and while obeying the physical constraints—including the actuators' limitations—of the system. Such properties of the path are paramount for continuum arms, as little perturbation in their trajectories, coupled with sensor feedback lag and actuator limitations, may result in uncontrolled oscillations with eventual instability. The above problem can take place in an environment with or without obstacles, where the latter naturally makes the the path planning problem much more challenging.

The authors are affiliated with the School of Computing, DePaul University, Chicago, IL 60604, USA. {bmeng1,igodage,ikanj}@depaul.edu.

This work is supported in part by the National Science Foundation Grants IIS-1718755, IIS-2008797, and IIS-2048142.

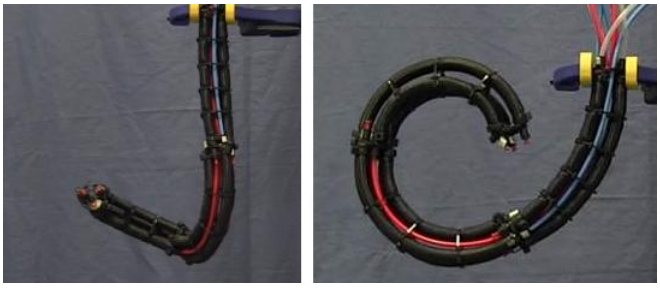

Fig. 1. A illustration of a 3-section continuum arm achieving different spatial shapes.

Path planning for continuum arms that produces both reliable and smooth paths remains a challenging task. Continuum arms are hyper-redundant and there is a complex, non-linear relationship between the C-space (the set of all possible configurations that the robot may assume) and W-space (the ambient 3D space of the physical continuum arm). Conventional path planning approaches based on Inverse Kinematics (IK) are unreliable for continuum arms—as was shown in [6], since they suffer from issues such as knotting and convergence to local minima. As the authors in [6] demonstrated, approaches that consider future and previous spatial shapes in path planning avoid these issues, and deliver high-quality paths. Since very small changes in the configurations of a continuum arm can lead to large changes in their spatial shape, the generation of dense sets of configurations allows for the selection of reliable and smooth paths.

The authors in [6] proposed a Kinematics-based approach that relies on generating a dense/rich set of configurations from which a smooth path is then selected. It is a brute-force approach that requires the generation of a look-up table prior to performing the path planning. As such, after the look-up table has been generated, this approach is limited to the set of configurations in the look-up table. The approach in [6] was shown to be superior to traditional IK in terms of its success rate. However, this approach becomes inefficient when smoothness is inadequate for a given discretization level. A larger look-up table can be generated, but this requires significant memory overhead and substantial pre-computation. Moreover, the approach in [6] was not validated to ensure the smoothness of the generated trajectories when tracked by a controller.

While IK remain one of the most widely used approaches to path planning [7]–[9], addressing the shortcomings of IK remains a challenge. An iterative approach to IK for

redundant manipulators was presented in [10] that relies on a ranking system to select a sequence of configurations. In this variant of IK, a set of configurations is generated from among which a "best" configuration is selected and is subsequently used as a seed for the generation of another set of configurations. While this approach seems to be a refinement on conventional IK implementations, it still relies on a greedy, local search, and hence suffers from the same issues of knotting and convergence to local minima that traditional IK suffer from.

Another use of IK was proposed in [11] for path planning of concentric tube robots. The authors in [11] are also critical of path planning approaches that rely on precomputed kinematic look-up tables. Instead, they propose a high-performance IK library that is used to randomly generate a set of points in the W-space that serve as a "roadmap" of the W-space; using IK, this set of points is then mapped back to generate a set of configurations in the C-space. A greedy search heuristic is then used to find a suitable path in the C-space among this set of configurations. This approach does not necessarily generate a dense set of configurations in areas of interest before employing global smoothness measures and instead relies on the generation of a set of configurations that serves as a roadmap of the whole C-space.

Lastly, an approach relying on break down of the W-space and configuration generation for rigid-link robots using IK was proposed in [12]. However, this approach focuses on estimation of IK within a region of the W-space and focuses on pose estimation of rigid-link robots rather than path planning.

In this paper, we propose an IK-based path planning approach for generating smooth trajectories for continuum arms. The proposed approach combines the best of both worlds: the flexibility and versatility of the local IK approach coupled with a global path planning approach that selects a path from a rich/dense set of configurations. The proposed approach can be viewed as an iterative use of IK to generate a rich set of configurations within each cell of a cell-path in a discretized W-space; the number and richness of the generated configurations in each cell of the path can be controlled on the fly as opposed to the brute-force approach in [6], which populates the look-up table prior to performing the path planning. A smooth path from among the generated configurations is then computed by casting the problem as that of computing a shortest path in a graph constructed based on the generated configurations and to which a smoothness metric—introduced in this paper—is used to differentiate candidate paths.

II. System Model

A. Kinematic Model

Our model is a continuum arm consisting of 3-sections; we will use index i, where $i \in \{1,2,3\}$, to refer to the i-th section of the continuum-arm, with index 1 indicating the section closest to the base. Each section consists of 3 pneumatic muscle actuators (PMA) bound together with radial symmetry. This model uses an inextensible backbone that introduces an over-constrained system making the third actuator kinematically redundant [6]. Thus, each section of the arm can be determined using only two length changes. Since the arm has 3 sections and each section can be determined by 2 actuators, a *configuration* defining the state of the full arm is a 6-tuple of length changes.

We follow the kinematic model introduced in [6]; see [6] for more specific details about the prototype. Note that this system model introduces curve parameters that are used to develop a homogeneous transformation matrix (HTM). Given a 6-tuple configuration space vector c, we can use the HTM to find the coordinates of the tip. See [6] for more information on the derivation of the kinematic model.

B. Dynamic Model

To validate the generated trajectories, we use the dynamic model of the continuum arm developed in [13]. The dynamic model was implemented on the Matlab Simulink environment. In order to assess the smoothness of trajectories we will apply this dynamic model with joint-space control. In the controller implementation, we employed joint value saturation limits (0-6 to imitate the 6-bar input pressure limitation), joint slew rate (from experimental data showing ± 1 bar/sec), and transportation delay (25 ms) to the sensor feedback. With these constraints in place, PI blocks were tuned utilizing Matlab PI control tuner that yielded P=7 and I=420 gains to ensure a 0.3 sec unit step response.

III. Preliminaries

A. Graphs

The configurations space (C-space) of the continuum-arm is the set of all configurations a robot may assume. The configurations graph has as its vertex-set the set of configurations w.r.t. a pre-specified discretization of the actuators' lengths, and in which there is an edge between any two configurations that are within one step variation w.r.t. the aforementioned discretization.

Each configuration corresponds to a 3-D point representing the tip and, by extension, pose of the continuum arm. The work space (W-space) of the continuum arm is the set of all such 3-D points corresponding to the configurations in the C-space.

If obstacles are present in the W-space, some of the configurations may become invalid as they lead to collision with the obstacles. In such cases, these configurations are removed from the C-space.

B. The Workspace Graph

We discretize the W-space using a uniform grid, in which each cell is a cube of dimension 1 unit. Each cube is represented by its center. The *cubes-graph* is a geometric graph whose vertex-set is the set of all grid cubes, and in which two vertices are adjacent if their corresponding cubes are adjacent in the grid; each edge is associated with a weight that is the Euclidean distance between the cube centers corresponding to the endpoints of the edge. A path in the cubes-graph is a called a *cube path*.

C. Brute-Force Approach

Brute-Force Path Planning (BF) was used in [6] and requires a mapping between the C-space and the W-space. This is created by generating a set of configurations in the C-space and calculating the corresponding points in the W-space. These configurations and points are then stored into a look-up table. Next, the *cube graph* is populated with configurations such that every configuration in the C-space is assigned a cube based on whether or not its corresponding W-space point is inside the cube. Cubes that do not contain any configurations are discarded.

BF then finds a shortest path, P, in the *cube graph* from c_s to c_t, where c_s is the cube containing the starting point and c_t that containing the destination point. Using the look-up table, all configurations whose corresponding points are contained within the cubes of P are enumerated. A layered graph is then formed, each of whose layers consists of all configurations whose corresponding points are in the same cube of P. Edges exist only between adjacent configurations in adjacent cubes and are associated with weights equal to the Euclidean distance between the configurations corresponding to the endpoints of the edge. A shortest path is then found in the layered graph from the starting configuration to a configuration in the cube c_t. The smoothness of a path is defined as the sum of the weights between all consecutive configurations on the path.

D. Inverse Kinematics

Inverse Kinematics (IK) map W-space points to corresponding C-space configurations using numerical methods. When supplied with a trajectory, IK can be a very effective tool for finding a sequence of configurations corresponding to W-space points that trace the trajectory. This is called Inverse Kinematics-based Path Planning and is a standard approach for path planning of rigid robots.

IV. Inverse Kinematics for Improved Configuration Generation

Whereas Inverse Kinematics are the conventional approach to path planning for rigid robots, the hyper-redundancy of the continuum arms make path planning for such robots much more complex. The standard IK-based planning approach—discussed above—may lead to non-convergence (trap in local minima) and self-knotting (loss of DoF available for planning). We propose an IK-based framework that addresses the above issues. Instead of relying on a single IK solution, we propose to use IK to generate a "rich" set of configurations whose corresponding W-space points lie within the vicinity of the W-space target point; by richness here, we mean sets of configurations that have a wide variety of valid spatial shapes. The proposed approach combines the configuration generation of IK with the layered graph approach to compute a suitable W-space path. We use an objective function that takes in a starting configuration and then tries to minimize the distance between the arm tip and the target point.

First, we generate a cube path as done by the BF approach discussed in Sec. III-C. After generating the cube path, we will use IK to populate the cubes along this path with a rich set of configurations on-the-fly. Afterwards, the layered graph is constructed and path planning is performed as discussed in [6] and described in Sec. III-C. The proposed framework allows for greater flexibility in terms of the number of generated configurations in a cube, the size of the cube, and the required pre-computation necessary for path planning. Additionally, it avoids the issues of knotting and convergence—as a failing sequence of configurations can be avoided altogether, and a diverse set of paths become available from which a smooth path may be selected.

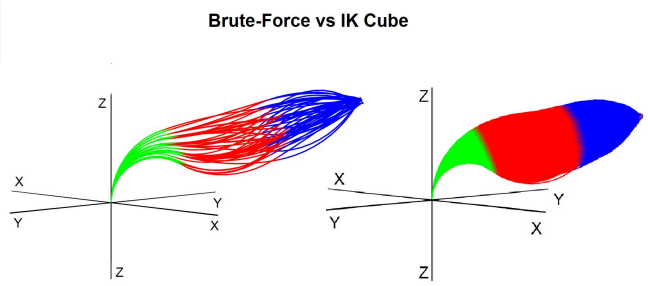

Fig. 2. A comparison for the cube (26,26,15) of the configurations stored in the look-up table vs. the Cube Configuration Generation Method

A. Cube Configuration Generation

In order to generate a dense set of configurations in a cube Q, we rely on multiple invocations of IK at the point located in the center of Q. We start by generating a large number of random starting configurations. Afterwards, we invoke IK with each of these starting configurations to find a target configuration that places the tip within some small distance from the center of Q. This allows us to generate a dense set of configurations within Q. When we use this method to populate a cube with configurations, we call it *Cube Configuration Generation*. Fig. 2 shows a comparison of the richness between a cube populated with the Cube Configuration Generation method and the look-up table.

B. Our Approach

Using the Cube Configuration Generation method, we propose a new approach for IK-based path planning that does not rely on a pre-computed look-up table of configurations. In [6], as discussed in Sec. III-C, to perform path planning, a cube path was generated and populated with configurations from a look-up table. In our approach, these cubes are populated using IK. We consider two variants of this approach that rely on two subroutines: The *Global-IK* subroutine and the *Sequential-IK* subroutine.

The Global-IK subroutine populates the first cube using the Cube Configuration Generation method. Each subsequent cube is also populated using the Cube Configuration method by using random starting points.

The Sequential-IK subroutine populates the first cube using the Cube Configuration Generation method. Each subsequent cube is populated with the configurations from the previous cube used as starting points. This means that only the first cube is generated using random starting points,

and each subsequent cube is populated using the output from previous invocations.

After populating the cubes, a path can be generated using the layered graph and weight metrics discussed in Sec. III. These methods, when paired with the two subroutines, form the *Global-IK Path Planning Approach* and *Sequential-IK Path Planning Approach* respectively. See Algorithms 1 and 2 for the pseudocode of the two subroutines.

Algorithm 1: Global-IK Subroutine

```
Input: A cube path P
Output: A list of configuration sets L
foreach cube in P do
    Create a set of configurations S = ∅;
    repeat n times
        Generate a random starting configuration
         start;
        Use the cube's center point as the target;
        Use the Cube Configuration Generation
         method as the objective function;
        Use IK with parameters start, target, and
         objective function to generate a (new)
         configuration and add it to S;
    end
    Add S to L;
end
return L
```

V. Trajectory Generation

Due to the compliant nature of continuum arms, their resonance frequency is much lower than that of rigid robots. As a result, path planning for continuum arms is much more complicated than that of their rigid counterparts, as tracking errors, coupled with feedback and actuation delays (low bandwidth actuators), can easily drive the system towards instability. Therefore, for path planning of continuum arms, it becomes critical to generate "smooth" trajectories in order to reduce the amount of instability.

In the kinematic sense, smooth paths are generally characterized by gradual and small changes over time in the corresponding arm's configurations. Usually, the amount of change in the configurations is inversely proportional to the smoothness of the corresponding path. Paths satisfying this property are easier and faster to track with feedback controllers without causing any oscillation issues while tracking. Thus, by maximizing trajectory smoothness, the speed of the continuum arm can be increased without causing oscillations as tracked by a controller.

We propose to incorporate the configurations space velocity of each actuator in path planning to compute a smooth trajectory and to accurately relate the generated paths with the actuated movement of the continuum arm. To that end, first, we define the notion of a velocity threshold, V_{th}, to be the maximum PMA velocity. We also define below a change limit, V_δ, that determines the maximum allowable change in velocity between two consecutive configurations on the path.

Algorithm 2: Sequential-IK Subroutine

```
Input: A cube path P
Output: A list of configuration sets L
for cube c in P do
    Create a set of configurations S = ∅;
    if cube is first then
        repeat n times
            Generate a random starting configuration
             start;
            Use the cube's center point as the target;
            Use the Cube Configuration Generation
             method as the objective function;
            Use IK with parameters start, target, and
             objective function to generate a (new)
             configuration and add it to S;
        end
    else
        foreach configuration c in the previous cube
         do
            start=c;
            Use the cube's center point as the target;
            Use the Cube Configuration Generation
             method as the objective function;
            Use IK with parameters start, target, and
             objective function to generate a (new)
             configuration and add it to S;
        end
    end
    Add S to L;
end
return L
```

In the path generation, we consider two consecutive configurations q and q^* and their corresponding lengths changes $l_{i,j}$ and $l^*_{i,j}$, where $i = 1,2,3$ denotes the arm's section and $j = 1,2$ denotes the actuator within a section. For section $i \in \{1,2,3\}$ and actuator $j \in \{1,2\}$, the shortest movement time, $t_{i,j}$, between $l_{i,j}$ and $l^*_{i,j}$, is calculated as

$$t_{i,j} = \frac{|l^*_{i,j} - l_{i,j}|}{V_{th}}. \tag{1}$$

Since no actuator can actuate in time less than $t_{i,j}$, the minimum time to actuate between q and q^*, $t_{shortest}$, is calculated as

$$t_{shortest} = \max\{t_{i,j}\},\ i = 1,2,3,\ j = 1,2. \tag{2}$$

After computing $t_{shortest}$, the robot is actuated from q to q^* in this shortest possible time. Note that, there are situations where (1) does not hold. For example, if the velocity of an actuator were to change from negative to positive, it is possible that the magnitude of the change could be quite large despite being within the velocity threshold. To mitigate this, we propose the use of a maximum change V_δ such that $V_\delta - V_{th} > 0$. If a sign change occurs, the new maximum actuator velocity becomes $V_\delta - V_{th}$ to ensure that this large change occurs slowly. Note that this velocity threshold is

used for normalizing the trajectory with a common actuator velocity. Once a time trajectory is generated, the duration of this trajectory is modified until a smooth trajectory is obtained under desired constraints during empirical tests.

A. Time as a Metric

As discussed in Sec. III-C, the Euclidean distance between configurations was used in [6] as edge-weight when constructing the layered graph. This metric prioritizes the smallest overall change in configurations in the path. Here, we propose to use time, as described above, as the edge-weight in the layered graph so that the new metric prioritizes the shortest time instead. That is, for consecutive configurations q_{k-1} and q_k, the weight between them is calculated as $weight(q_{k-1}, q_k) = t_{shortest}$. Intuitively, the path that can be traced by the continuum arm with a high velocity would have less overall changes with respect to its configurations.

VI. Results

A. Test Platform

Our algorithms were implemented and tested on a Windows-10 computer, with a 64-bit Intel i7-9700K CPU, 8 core 3.60 GHz processor, and 32 GB RAM. We use the Brute-Force approach as a basis for comparison with the proposed approaches. It is implemented as proposed in [6].

B. Comparison of Approaches

To test the Brute-Force algorithm from [6], we used the same method as discussed in [6] to generate $327,082,769$ configurations. Both of the IK-based approaches and the Brute-Force approach were implemented to be time weighted, as discussed in Sec. V-A.

All approaches used a cube size/dimension of 1 *cm* for a fair comparison, though the IK approaches are capable of using any cube size without re-enumeration. When Brute-Force was run with smaller cube sizes, there were many holes in the graph that lead to jagged, nonsmooth paths. These holes may be filled with a denser C-space, but that would necessitate a re-enumeration of all of the configurations, which was a stated shortcoming of this approach.

To test our approaches, we generated two sets of varied target trajectories that the IK-based and Brute-Force approaches had to trace. These included lines with obstacles and demonstrative geometric shapes. The first set of shapes were small and placed mostly in the center of the W-space. An additional, larger set of trajectories was created that focused on cubes towards the edge of the W-space. On both sets of trajectories, paths were generated that closely traced the desired trajectory with the arm tip. Next, these generated paths were validated using the dynamic model. We stretched or compressed the time for these paths until the Root Mean Square of the arm tip between the generated path and the dynamic path, what we use as an acceptable error, was less than 1 *cm*. Any error larger than that would indicate oscillation and instability. In some cases, this may not be possible, meaning that the path is unsuccessful.

TABLE I

Summary of Time Results for small shapes (above line) and large shapes (below line)

Approach	BF Time(s)	IK-Global Time(s)	IK-Seq Time(s)
Circle 1	24.46	16.08	23.67
Circle 2	13.81	11.01	12.90
Figure-8	20.82	17.03	18.02
Rotated Cube	110.14	86.45	83.53
Circle 1	**148.96**	**64.59**	**64.52**
Circle 2	DSF	68.37	67.19
Circle 3	Failure	57.71	57.01
Figure 8	81.61	53.71	56.47
Rotated Cube	245.34	116.78	131.64

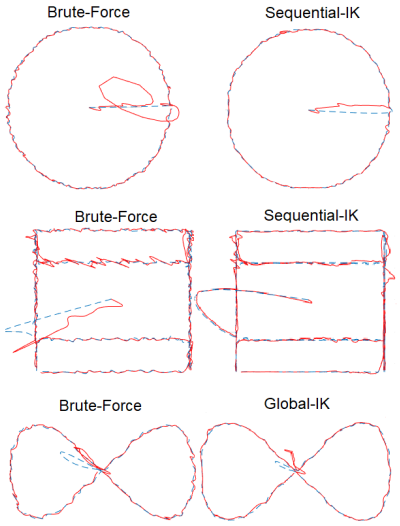

Fig. 3. A top down view of the results from the Dynamic Simulator for a circle, rotated cube, and figure-8 trajectory. The trajectory traced by the dynamic simulator (red) is displayed over the calculated trajectory (blue).

C. Trajectory Smoothness without Obstacles

Circles, figure-8 (the shape resembling the infinity symbol), and rotated cube trajectories were created for the arm to trace. Circles and figure-8 trajectories were two dimensional geometric shapes with a fixed Z coordinate. Rotated cubes were generated as fully three dimensional trajectories.

Table I shows the time needed to complete the tracing of the shapes. The small shapes are displayed above the double lines, and the large shapes are displayed below it. *BF* is the Brute-Force approach, while *Global-IK* and *Seq-IK* are the Global-IK and Sequential-IK approaches, respectively. "Failure" indicates that the trajectory could not be computed with a given approach. The term *DSF* indicates that the dynamic simulator could not find a tip error less than 1 *cm*. A shorter time indicates that the dynamic simulator was able to simulate the arm moving at a higher velocity. As discussed before, shorter times are desirable, so assuming no oscillation in our generated trajectories, a shorter time indicates a better performance for the algorithm.

In Table I, we see that there are not substantial differences in time between the Brute-Force, Globak-IK, and Sequential-IK approaches for the small shapes. However, it seems that IK-based approaches have a slight edge. Given that most of

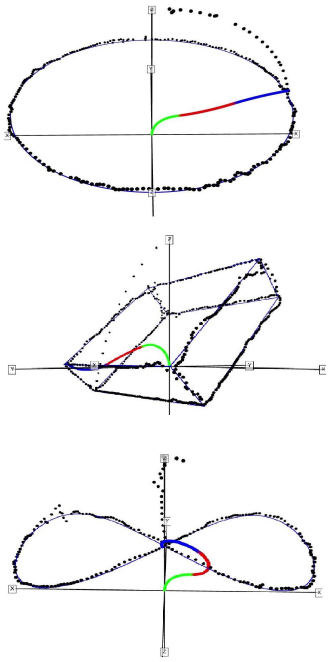

Fig. 4. A 3D view of the Sequential-IK approach simulated on the circle, rotated cube, and figure-8 trajectory. The traced path is shown in black and the target trajectory is shown in blue.

the redundancy in the W-space occurs in the center, the cubes towards the center of the W-space have quite rich and dense sets of configurations. In the case of the smaller trajectories, this means that the approaches are similar in the overall number of configurations that can be used, hence the similar times. For the large shapes in Table I, however, this is not the case. The larger trajectories focus on cubes towards the edge of the W-space where there is far less redundancy. Given how extended the arm must be to reach these points, there are fewer unique spatial shapes to assume towards the edge of the W-space. In the case of the Brute-Force algorithm, the restricted discretization of the C-space limits the number of configuration in each one of the edge cubes. However the IK-based approaches, with no offline configurations, can find an acceptable rich set of configurations. When applied to path planning, the IK-based approaches have far more configurations to choose from, meaning that IK-based approaches have a distinct advantage.

Fig. 3 compares the desired and achieved trajectories for a circle, figure-8, and rotated cube as traced by the Brute-Force and IK-based approaches. Both trajectories have a tip error of less than 1 *cm*. Despite similar tip errors, the IK-based results are smoother and had a lower time in each case. Fig. 4 shows the arm tracing large circle, figure-8, and rotated cube trajectories in 3D for the Sequential-IK approach.

D. Trajectory Smoothness with Obstacles

We also performed tests with straight-line trajectories in the presence of obstacles as a comparison to the results in [6]. A starting and ending point were given along with three obstacles. These were spherical obstacles placed randomly within the arm's reach and had a random radius in the range of $[4, 15]$. Table II shows the times for all three approaches. The results here very closely mimic those without obstacles.

TABLE II
SUMMARY OF TIME RESULTS FOR TRAJECTORIES WITH OBSTACLES

Approach	BF Time(s)	IK-Global Time(s)	IK-Seq Time(s)
Obstacle Trial 1	14.34	16.43	9.44
Obstacle Trial 2	12.89	14.22	7.11

VII. CONCLUSION

This paper proposed a novel IK-based path planning approach to address the limitations of previously-proposed kinematics-based approaches. Our IK-based approach can generate a rich set of configurations online and can find valid configurations in sparse regions of the C-space. Using this new approach, we implemented smooth path planning for continuum arms. We validated the smoothness of the generated paths by applying dynamics constraints in dynamic simulation and showed that, when tracked by a controller, the paths generated using the proposed approach are much smoother and can be traced by the arm much faster than previous approaches.

REFERENCES

[1] E. S. Boy, E. Burdet, C. L. Teo, and J. Colgate, "The learning cobot," 01 2002.

[2] D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied Bionics and Biomechanics*, vol. 5, no. 3, pp. 99–117, 1 2008.

[3] R. J. Webster and B. A. Jones, "Design and kinematic modeling of constant curvature continuum robots: A review," *Int. J. Rob. Res.*, vol. 29, no. 13, pp. 1661–1683, 2010.

[4] D. Palmer and D. Axinte, "Active uncoiling and feeding of a continuum arm robot," *Robotics and Computer-Integrated Manufacturing*, vol. 56, pp. 107–116, 2019.

[5] J. L. C. Santiago, I. S. Godage, P. Gonthina, and I. D. Walker, "Soft robots and kangaroo tails: modulating compliance in continuum structures through mechanical layer jamming," *Soft Robotics*, vol. 3, no. 2, pp. 54–63, 2016.

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

[7] C. López-Franco, J. Hernández-Barragán, A. Y. Alanis, N. Arana-Daniel, and M. López-Franco, "Inverse kinematics of mobile manipulators based on differential evolution," *International Journal of Advanced Robotic Systems*, vol. 15, no. 1, 2018.

[8] C. Lopez-Franco, J. Hernandez-Barragan, A. Y. Alanis, and N. Arana-Daniel, "A soft computing approach for inverse kinematics of robot manipulators," *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 104 – 120, 2018.

[9] S. Dereli and R. Köker, "A meta-heuristic proposal for inverse kinematics solution of 7-dof serial robotic manipulator: quantum behaved particle swarm algorithm," *Artificial Intelligence Review*, vol. 53, no. 2, pp. 949–964, 2020.

[10] L. Zhao, J. Zhao, H. Liu, and D. Manocha, "Efficient inverse kinematics for redundant manipulators with collision avoidance in dynamic scenes*," in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2018, pp. 2502–2507.

[11] K. Leibrandt, C. Bergeles, and G. Yang, "Concentric tube robots: Rapid, stable path-planning and guidance for surgical use," *IEEE Robotics Automation Magazine*, vol. 24, no. 2, pp. 42–53, 2017.

[12] M. Tarokh and M. Kim, "Inverse kinematics of 7-dof robots and limbs by decomposition and approximation," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 595–600, 2007.

[13] I. S. Godage, G. A. Medrano-Cerda, D. T. Branson, E. Guglielmino, and D. G. Caldwell, "Dynamics for variable length multisection continuum arms," *The International Journal of Robotics Research*, vol. 35, no. 6, pp. 695–722, 2016.