

An Optimization-Based Inverse Kinematics Solver for Continuum Manipulators in Intricate Environments

Yinan Sun, Sai Wang

Abstract—Continuum manipulators have gained significant attention as a promising alternative to rigid manipulators, offering notable advantages in terms of flexibility and adaptability within intricate workspace. However, the broader application of high degree-of-freedom (DoF) continuum manipulators in intricate environments with multiple obstacles necessitates the development of an efficient inverse kinematics (IK) solver specifically tailored for such scenarios. Existing IK methods face challenges in terms of computational cost and solution guarantees for high DoF continuum manipulators, particularly within intricate workspace that obstacle avoidance is needed. To address these challenges, we have developed a novel IK solver for continuum manipulators that incorporates obstacle avoidance and other constraints like length, orientation, etc., in intricate environments, drawing inspiration from optimization-based path planning methods. Through simulations, our proposed method showcases superior flexibility, efficiency with increasing DoF, and robust performance within highly unstructured workspace, achieved with acceptable latency.

I. INTRODUCTION

Continuum manipulators [1], characterized by their continuous bending along the entire length, pose unique challenges for inverse kinematics. Traditional methods, such as Jacobian-based approaches [2], incur significant computational costs when applied to continuum manipulators due to their highly redundant degrees of freedom and continuous structure, which lack closed-form solutions [3]. However, these manipulators are prized for their compliance, safety, and capacity to navigate intricate, constrained workspaces [4].

In addressing the IK problem for continuum manipulators, a common assumption is that each finite section of the continuum body follows a constant curvature arc [5], [6], allowing for a kinematic representation based on bending angle and arc length. The assumption is based on the fact that, most of the continuum manipulators are built by serially connect several bending actuators. These manipulators, with their high redundancy, often entail significant computational costs when applying conventional methods.

Existing approaches to IK for continuum manipulators often require substantial computational resources and offer no guarantee of a correct solution. For instance, the use of Geometric Jacobian [7], a prevalent choice, is encumbered by substantial computational demands, offering no assurance of a correct solution. Researchers have explored alternative approaches, such as Monte Carlo algorithms [8], which, while effective, suffer from prolonged computational times, particularly in systems with numerous DoF. Searching-based algorithms, while promising, still incur substantial computational costs as the number of sections increases. Also, it is

mainly used in manipulators with rigid links [9].

FABRIKc and its variants, including FABRIKx, are iterative algorithms designed to solve the IK problem for continuum manipulators [10], [11]. Inspired by the original FABRIK algorithm [12], these methods iteratively adjust poses, incorporating both forward and backward iterations to achieve real-time solutions with smooth postures. However, these algorithms have limitations, such as multiple constraints accommodation and a lack of consideration for intricate workspaces with multiple obstacles. To address these limitations, Wu et al. [13] developed a heuristic algorithm based on FABRIK for obstacle avoidance. This algorithm performs collision checks and updates key points away from obstacles. However, it only activates when a collision check fails during iteration, which can lead to passive avoidance and potential proximity to dangerous areas near obstacles.

Learning-based algorithms have also been developed for solving the IK of continuum robots [14], [15]. This type of algorithm can be used in the model-free control of the continuum robot with computational efficiency, while the quality of the output relies largely on the training sets, which might lead to reducing robustness in obstacle settings that are way different from the training sets. Also, the training set would be too large for a redundant continuum robot.

Sampling-based algorithms, commonly used for path planning in rigid robotic arms, are also employed for solving inverse kinematics in continuum robots. For instance, researchers have combined rapidly-exploring random trees (RRT) and follow-the-leader (FTL) algorithms to address obstacle avoidance in continuum robots [16], [17]. While these algorithms improve efficiency, they may not guarantee stable performance, especially in complex environments where obstacle settings require high accuracy for generating usable solutions considering obstacle avoidance.

Other algorithms also utilize path planning methods in IK solvers, such as SLINKI [18], a hybrid IK solver that has made significant advancements in solving the IK problem for continuum manipulators. SLINKI utilizes state lattice-based A* search algorithms, a path planning method, to generate configurations for the initial $(n - 2)$ segments with short latency. It then employs the FABRIKc algorithm for the accurate calculation of the remaining 2 segments, achieving a good balance between efficiency and accuracy. However, its time complexity may significantly increase with the number of segments considering the exponential computational complexity, posing challenges in computational efficiency as the DoF of the robot increases.

Despite the progress in IK for continuum robots, chal-

lenges remain. Existing methods may struggle with real-time performance as the number of DoFs increases, and they may not adequately consider multiple constraints or obstacle avoidance in complex environments.

In this paper, we present an optimization-based IK solver tailored for continuum manipulators, addressing IK challenges in intricate workspaces while accommodating constraints such as segment lengths and orientations. Our approach leverages the high degrees of freedom (DoFs) of continuum manipulators to navigate complex workspaces effectively. Our methodology employs an optimization-based path planning algorithm with an artificial potential field to navigate multi-obstacle environments. The path is defined by path points, which are the endpoints of continuous piecewise arcs. These arcs construct a continuous representation of the continuum robot. By calculating objective functions and employing optimization methods, we ensure that the continuum manipulator satisfies all constraints.

A key innovation of our approach is its ability to fully utilize the high DoF characteristic of continuum manipulators. We represent the manipulator as a holistic structure, making it easy to modify by adding more points on the path. Besides, our algorithm allows for obstacle avoidance even when the manipulator is not in direct contact with obstacles, leading to safer solutions.

Furthermore, our algorithm maintains computational efficiency, with a manageable time complexity of $O(n^2)$ as the number of segments increases. This efficiency ensures that our method remains practical and efficient for continuum manipulators of varying DoF.

The contributions of this research paper are outlined as follows:

- We have developed an optimization-based framework for solving inverse kinematics (IK) for continuum robots, addressing various requirements such as end orientation tracking, length control, and intricate obstacle avoidance. This framework includes the formulation of a cost function and update rules essential for optimization.
- Our research potentially marks a pioneering achievement by showcasing that our IK solver enables the end effector of a continuum manipulator to track paths collision-free within intricate workspace replete with multiple obstacles. This capability sets the stage for enhanced manipulator performance and safety in challenging real-world scenarios.
- Our approach demonstrates the potential of efficient continuum manipulator IK by achieving an $O(n^2)$ complexity algorithm, offering practicality for manipulators with varying numbers of segments and high Degrees of Freedom.

II. APPROACH AND METHODS

In addressing the challenges posed by the highly redundant structure of continuum manipulators, our methodology takes a holistic approach, treating the manipulator as a system of continuous paths. Central to our approach is the development of an innovative inverse kinematic solver designed to

determine optimal configurations for each segment within the manipulator's structure.

Our solver focuses on creating a smooth path consisting of points that connect the two designated endpoints. These points serve as the endpoints of a piece-wise arc, which we later characterize by G^1 continuity. Subsequently, we employ an optimization process to refine the initial configuration. This optimization considers length constraints associated with each segment, as well as obstacles, orientation, and other objective functions. This ensures the manipulator's operability within its environment.

This methodology forms the core of our research, enabling precise control and real-time operation of continuum manipulators, especially in intricate environments with complex obstacles.

A. Piece-wise Arc Representation and Gradient Calculation

To transition from path points to the continuum robot's configuration, we utilize a piece-wise arc representation that adheres to the constant curvature framework. Each segment of this piece-wise arc aligns with a corresponding segment in the continuum robot, ensuring G^1 continuity for coherence with adjacent robot segments.

Utilizing the piece-wise arc representation, we transform path points into a piece-wise arc connecting the path points and end points, aligning with the preset orientation at the start point. Subsequently, we optimize this parameterized piece-wise arc to align with constraints and avoid obstacles, employing an artificial potential field. This approach enables us to accurately represent the continuum manipulator's configuration while ensuring practical and functional configurations for real-world applications.

1) Shape Parameters and Control Points: To accurately represent the continuum manipulator's structure and achieve G^1 continuity, we employ a circular spline representation approach similar to Song et al.'s work [19]. This approach involves using rational Bézier curves to represent a single arc segment, with end points a and b , and a control point c lying on the bisector plane of the line segment ab . The arc equation using the rational Bézier representation is given by:

$$y(u) = \frac{(1-u)^2 a \pm 2u(1-u)\omega c + u^2 b}{(1-u)^2 \pm 2u(1-u)\omega + u^2}, u \in [0, 1],$$

where the \pm sign denotes the minor or major arc, and ω is determined by:

$$\omega = \frac{h}{\sqrt{h^2 + k^2}}$$

Here, h and k are components of the vector perpendicular to the line segment ab .

For our piece-wise arc representation with G^1 continuity, we compute the control point c of each arc segment based on the end points of the current arc segment and the control point c of the previous segment.

Figure 1 shows the rational Bézier representation of a single circular arc, and a piece-wise arc spline with G^1 continuity.

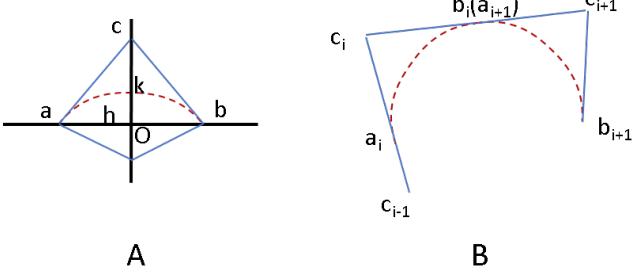


Fig. 1. A) Rational Bézier representation of a single circular arc. B) A 2 segments piece-wise arc spline with G^1 continuity.

For each arc segment, the control point c_{i+1} is determined by the end points a_{i+1} and b_{i+1} of the segment, as well as the control point c_i of the previous segment.

The control points are generated using the formula:

$$c_0 \vec{a}_0 = v_{initial}$$

Here, $v_{initial}$ represents the preset orientation at the start point (or base) of the continuum robot. For subsequent control points, we use:

$$c_{i+1} = f(a_{i+1}, b_{i+1}, c_i)$$

, where $f(a_{i+1}, b_{i+1}, c_i)$ calculates the control point based on the end points and the previous control point. This approach allows us to create a piece-wise arc with path points and preset initial orientation, ensuring G^1 continuity and proper control point positioning.

To establish an initial guess for our optimization process, we create an arc connecting the start point and the goal (with the direction at the start point aligned with the initial orientation). We then divide this arc into a piece-wise arc with segments of equal length, treating the points on the arc as path points for the optimization process. With these equidistant points, we create the control points (the c points) to ensure G^1 continuity within the piece-wise arc. This initial piece-wise arc serves as the starting configuration for our optimization process, providing a foundation for the subsequent optimization procedure.

2) Gradient Calculation: We consider the variation as the piece-wise arc itself, which is equivalent to optimizing the path points, as there is a bijection between the piece-wise arc and a set of path points with a fixed orientation at the start point. To effectively optimize the piece-wise arc, we calculate the gradient using numerical differentiation techniques with the following equation:

$$\frac{\partial F(p)}{p} = \frac{F(p + \delta p) - F(p - \delta p)}{2\delta p}$$

Here, p represents the vector of path points' positions, $F(p)$ represents the objective function, and δp represents a small disturbance on the path points. While this method is straightforward and easy to deploy, it is crucial for managing potential issues of gradient explosion and dimension scaling. The disturbance of a path point can significantly affect

TABLE I
PARAMETERS IN THE ARC FITTING AND OPTIMIZATION.

Parameters	Explanation
a	end point of an arc segment
b	another end point of an arc segment
c	control point of an arc segment, lies in the bisector plane of line segment ab
ω	weight parameter in the rational Bézier representation
m	central point of line segment ab
n	number of the continuum robot segments ab
i	index of arc segments
h	length of line segment am and mb
h	length of line segment cm
α	scale parameter preventing gradient exploding

the situation of an arc segment far away, similar to error accumulation. To optimize the piece-wise arc effectively and mitigate the potential for gradient explosion, we use the following gradient descent equation:

$$dq_i = \alpha^{j-i} \sum_{j=i}^n \frac{\partial F_j}{\partial p_i}$$

Here, dq_i represents the update value for the i^{th} path point, and $\frac{\partial F_j}{\partial p_i}$ calculates the partial derivative representing the effect of the i^{th} path point on the j^{th} arc segment during optimization. The parameter α serves as a scaling factor crucial for managing dimension scaling and stabilizing the gradient descent process. It determines the rate at which previous points' contributions diminish and helps control the magnitude of parameter updates. Typically, α is set to a small value, such as $1e-2$, to ensure a controlled and stable optimization process.

All the symbols and explanations are listed in Table I.

B. Cost Function Build

1) Obstacle Avoidance: The optimization-based algorithm utilizes a gradient descent mechanism that considers environmental obstacles and other constraints. This ensures that the path quickly converges to a locally optimal, collision-free, and continuous path, crucial for configuring a viable continuum manipulator. The approach integrates obstacle avoidance within the optimization framework, enhancing the manipulator's capability to navigate complex environments. The cost function (F_{path}) for obstacle avoidance is defined as follows:

$$F_{path} = \int_0^1 c(\xi(t)) \left\| \frac{d}{dt} \xi(t) \right\| dt$$

Here, $\xi(t)$ represents the path consisting of path points, and $c(\xi)$ represents the cost with respect to obstacles, derived from an artificial potential field. The calculation of $c(\xi)$ is described by the following formula:

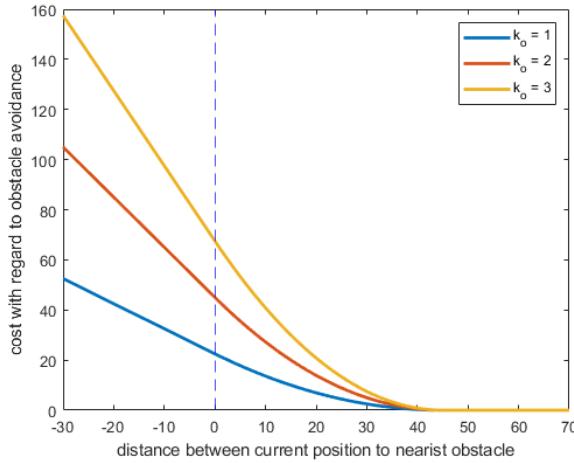


Fig. 2. Cost function for obstacles with relate to distance between current position and the nearest obstacle.

$$c(\xi(t)) = \begin{cases} k_o(-d(\xi(t)) + \frac{1}{2}\epsilon), & \text{if } d(\xi(t)) < 0 \\ \frac{k_o}{2\epsilon}(d(\xi(t)) - \epsilon)^2, & \text{if } 0 \leq d(\xi(t)) \leq \epsilon \\ 0, & \text{otherwise} \end{cases}$$

$d(\xi)$ denotes the distance between the path ξ and the nearest obstacle. $\|\frac{d}{dt}\xi(t)\|$ represents the 'velocity' of each path point. In this context, it refers to the mean value of the two arc segments adjacent to the current path point. This term is included in the cost function to discourage the behavior of enlarging the distance between path points to decrease the cost function value. ϵ signifies the maximum distance at which an obstacle still influences the path during the optimization procedure. The parameter k_o is introduced to adjust the slope of the artificial potential field. A higher k_o value leads to a steeper potential field, as illustrated in Figure 2. The function $c(\xi(t))$ is designed to be G^1 continuous with respect to $d(\xi(t))$ to ensure robust convergence and minimize the likelihood of oscillations or other convergence issues during optimization. Additionally, the function is linearized when $d(\xi(t)) < 0$ to prevent excessively large gradients that could impact convergence.

To facilitate efficient computation of the cost function for obstacle avoidance, we precompute a discrete artificial potential field offline (in the real world, it can be acquired by remote sensing, Lidar, RGBD camera, etc.). This involves recording the cost function value at grid nodes in the workspace with a reasonable density. During the optimization process, we can efficiently calculate the cost function by interpolating the values at the nearest grid nodes, thereby avoiding the need for costly real-time calculations.

As a continuum robot, ensuring collision-free operation is not guaranteed solely by the segments' endpoints being collision-free. To address this, we utilize the rational Bézier representation equation mentioned in Section II-A.1 to generate points along the continuum robot's body as a supplement, and for collision checking, as illustrated in Figure 3. Additionally, we employ the numerical gradient

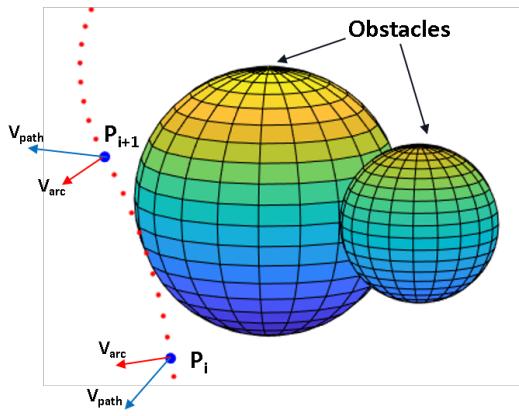


Fig. 3. A sketch for obstacle avoidance. Blue points represent the path points, which are also the end points of each arc segment. Blue vectors are generated by the optimization-based path planning algorithm, aiming to guide the overall path towards lower potential positions in the obstacle-related artificial potential field. The red points are samples taken from the piece-wise arcs, representing the continuum robot body. The red vectors depict the gradients calculated using a numerical method to move the piece-wise arcs away from obstacles during optimization.

calculation equation from Section II-A.2 to compute the gradient as part of the obstacle avoidance update vector. This allows the piece-wise arc to avoid obstacles, not just the path points. Consequently, the gradient related to obstacle avoidance is defined as:

$$\Delta F_{obs} = \Delta F_{path} + \beta \Delta F_{arcs}$$

Here, ΔF_{path} represents the update vector derived from F_{path} to move the path points away from obstacles. ΔF_{arcs} represents the gradient calculated using the points sampled from the piece-wise arc, aiding the arc in avoiding obstacles. The parameter β determines the weighting between these two gradients. ΔF_{obs} is the overall gradient for obstacle avoidance.

2) *Other Constraints* : To achieve precise control over the continuum manipulator's configuration, it is essential to manage the length of each segment within our inverse kinematics solver. Our methodology ensures that each segment closely aligns with the specified central length.

The optimization process begins by addressing the length of each segment within the piece-wise arc. Assuming the range of segment lengths is $[l_{min}, l_{max}]$, the objective function guiding this optimization is defined as:

$$F(p) = \sum_{i=1}^n (l_i - l_0)^2$$

Here, l_i represents the actual length of the segment, and l_0 denotes the mean value of the range l_{max} and l_{min} .

Subsequently, following the procedure outlined in Section II-A.1, we calculate the arc length of each segment. Then, we compute the gradient for optimization using the numerical gradient calculation equation from Section II-A.2.

To ensure that the end orientation closely matches the target orientation, we also impose constraints on the orientation of the end effector of the continuum robot. We

define the objective function guiding this constraint as the norm of the difference between the current orientation vector and the target orientation vector of the end effector. The calculation process for the gradient of each constraint is similar, allowing us to efficiently incorporate them into the optimization process.

C. Update Rules

According to Section II-B, the overall cost function value can be calculated as:

$$U(\xi) = \alpha_1 F_{obs}(\xi) + \alpha_2 F_{len}(\xi) + \alpha_3 F_{ori}(\xi)$$

Here, α_1 , α_2 , and α_3 are changeable weights of the terms during optimization. We utilize the simulated annealing algorithm to adjust the weights during optimization. Additionally, we employ other techniques, such as projecting the update vector for obstacle avoidance onto a surface perpendicular to the plane generated by the current path point and the adjacent path points, to aid in convergence.

To perform gradient descent, we first approximate the cost function with a first-order Taylor expansion:

$$U(\xi) \approx U(\xi_0) + \Delta U(\xi_0)^T (\xi - \xi_0)$$

To minimize changes in the average 'acceleration' of each path point, which is defined to be proportional to the difference between its distances towards adjacent path points, we add a regularization term $\frac{\lambda}{2}(\xi - \xi_k)^T A(\xi - \xi_k)$, where A is the acceleration matrix. Thus, the optimization problem becomes:

$$\begin{aligned} \xi &= \arg \min_{\xi} \left\{ U(\xi_0) + \Delta U(\xi_0)^T (\xi - \xi_0) + \frac{\lambda}{2} \|\xi - \xi_0\|_A \right\}; \\ \frac{\lambda}{2} \|\xi - \xi_0\|_A &= \frac{\lambda}{2} (\xi - \xi_0)^T A (\xi - \xi_0) \end{aligned}$$

Here, ξ_0 represents the current path points, and ξ represents the updated path points.

λ represents the regularization coefficient, serving as a trade-off between minimizing the cost function value and minimizing changes in the average acceleration of the existing trajectory, for better convergence. As a result, the gradient descent update rule can be written as:

$$\xi = \xi_0 - \frac{1}{\lambda} A^{-1} \Delta U(\xi_0)$$

The update will continue until all constraints are satisfied or the maximum iteration number is reached.

III. EXAMPLES AND RESULTS

A. Performance Comparison

To provide a clear comparison of the time consumption, we conducted experiments using a 3D test environment modeled after the settings in a previous research paper [18]. In this environment, our IK solver was tested against others with 5 segments, a segment length range of 45 mm to 70 mm, and an acceptable end effector orientation error of

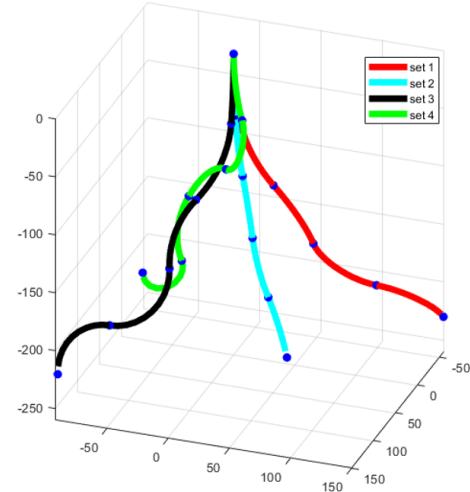


Fig. 4. The result of the four settings by the IK solver introduced in this paper. The piece-wise arcs represent the configuration of the continuum robot calculated by our IK solver, under four different settings of end position and orientation.

0.0175 rad, without obstacles. The start point's orientation was set to (0, 0, -1), and the four end orientations were (-0.7071, 0, -0.7071), (0, 0.2169, -0.9762), (0.5661, 0.2665, -0.7926), and (0, 0, 1) respectively.

The start point's position was set to (0, 0, 0), and the four end positions were (-60, 150, -230), (-10, 40, -260), (150, -90, -220), and (40, -60, -180) respectively, in mm. Each test was repeated 100 times for each setting to obtain the mean performance. Since our algorithm's nature guarantees the same solution, we aimed to achieve a more accurate estimate of time consumption.

The results are presented in Figure 4, and the data is summarized in Table II, focusing on the comparison with two specific sets as the previous paper did. Our algorithm is labeled as **Opti.path**.

Comparing the results, our algorithm demonstrates competitive time consumption, particularly excelling in 0 position error. However, being an optimization-based approach, its efficiency is influenced by the initial guess's proximity to the solution. For instance, in set 3, it took 15 iterations (15.9 ms) to find the solution, and even less if the initial guess was closer. Conversely, in set 4, where the solution was further from the initial guess, it took over 100 iterations, totaling 172 ms. This characteristic makes it suitable for IK during path tracking, as it can leverage previous results to quickly find nearby solutions, as we will demonstrate in the following section.

B. Obstacle Avoidance

In this section, we create complex environments with multiple spherical obstacles, each positioned randomly with random radius. The goal is to guide a continuum robot through these obstacles without collision. We segment the path into 180 points, and the robot is tasked with tracking a path on the opposite side of the workspace through the obstacles.

TABLE II
IK SOLUTIONS COMPARISON UNDER SAME ORIENTATION AND POSE SETTING

Orientation: (-0.7071, 0, -0.7071); Position: (-60, 150, -230)					
	time [ms]	position [mm]	pos-err [mm]	orientation	ori-err [rad]
exFABRIKc	183	(-60,149,-231)	0.863	(-0.71,0,-0.71)	0
Opti.DLS	112	(-62,152,-234)	4.52	(-0.4,0.77,-0.5)	0.85
SLInKi	12.2	(-59,150,-230)	0.766	(-0.71,0,-0.71)	0
Opti.path	16.9	(-60,150,-230)	0	(-0.70,-0.01,-0.71)	0.014
Orientation: (0, 0.2169, -0.9762); Position: (-10, 40, -260)					
	time [ms]	position [mm]	pos-err [mm]	orientation	ori-err [rad]
exFABRIKc	206	(-9.7,40,-261)	0.775	(0,0.22,-0.98)	0
Opti.DLS	124	(-10,40,-260)	0	(0,0.22,-0.98)	0
SLInKi	11.8	(-9.9,40,-259)	0.632	(0,0.22,-0.98)	0
Opti.path	2.8	(-10,40,-260)	0	(0.02,0.22,-0.98)	0.017

To begin, we solve the IK from the start point to the path's initial point, establishing an initial state. We then commence the optimization process by adjusting the end point and recalculating the IK using our optimization-based solver. We set a tolerance for segment length deviation, limiting it to one-tenth of the original length. The base orientation remains constant, but there are no constraints on the end effector orientation due to the complex environment.

Each experiment is repeated 30 times for each setup, and the mean time consumption is calculated. The results, presented in Figure 5, demonstrate an acceptable latency and the algorithm's ability to quickly reach local optimization during path tracking. Typically, solving IK for each path point takes between 6 ms and 20 ms. However, in narrow areas, this time can increase to over 100 ms, or even 200 ms.

Additionally, we designed other path configurations for the robot to track, showcasing different challenges. Due to page limitations, these configurations are not detailed here but can be viewed in the supplementary video.

C. Time Consumption With Relate to Number of Segments

In order to demonstrate the potential for our robot to solve IK for hyper redundant continuum robot, we run the continuum robot in simulation to track path through an obstacle rich intricate environment, while the number of segment varies from 5 to 10. Each set of experiment runs 20 times and gather the mean value as our result. We measure the time consumption per iteration for them and the result is shown in Figure 6. As shown in the figure, the time consumption doesn't increase too much as the number of segment increases, at least not exponentially increase.

In order to demonstrate the potential of our robot to solve IK for a hyper-redundant continuum robot, we conducted simulations where the robot tracked a path through a complex, obstacle-rich environment. We varied the number of segments from 5 to 10 and ran each experiment set 20 times each, averaging the results for analysis. The time consumption per iteration was measured, and the results are shown in Figure 6. Interestingly, the figure indicates that

the time consumption does not increase significantly as the number of segments increases, at least not exponentially.

IV. CONCLUSION AND FUTURE WORKS

In this research paper, we introduced a novel Inverse Kinematics solver designed specifically for continuum manipulators. Our approach integrates optimization-based path planning algorithms to establish an initial configuration and utilizes rational Bézier representation for arc length control. Through experiments, we demonstrate that the time consumption remains manageable even as the number of segments increases. Furthermore, we demonstrated the effectiveness of our IK solver in accurately tracking the end-effector path within complex environments featuring multiple obstacles.

For future work, this research paves the way for several avenues to enhance the efficiency and applicability of our algorithm. We intend to refine the updating mechanism by introducing adjustable waypoints and iteration numbers to optimize the IK solution further. Additionally, we will explore the integration of sampling-based path planning algorithms within the path generation process, with an array of more potential optimization objects. Extending our algorithm to three-dimensional scenarios represents another direction for future research.

Furthermore, we plan to implement our methodology on a physical manipulator equipped with touchless sensing techniques to enable real-time modeling of the surrounding environment. This will enable the development of an efficient motion planning framework tailored to continuum manipulators, capitalizing on the capabilities of our IK solver, to ensure rapid and practical applications in real-world scenarios.

REFERENCES

- [1] P. K. Singh and C. M. Krishna, "Continuum arm robotic manipulator: A review," *Universal Journal of Mechanical Engineering*, vol. 2, no. 6, pp. 193–198, 2014.
- [2] D. C. Rucker and R. J. Webster, "Computing jacobians and compliance matrices for externally loaded continuum robots," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 945–950.

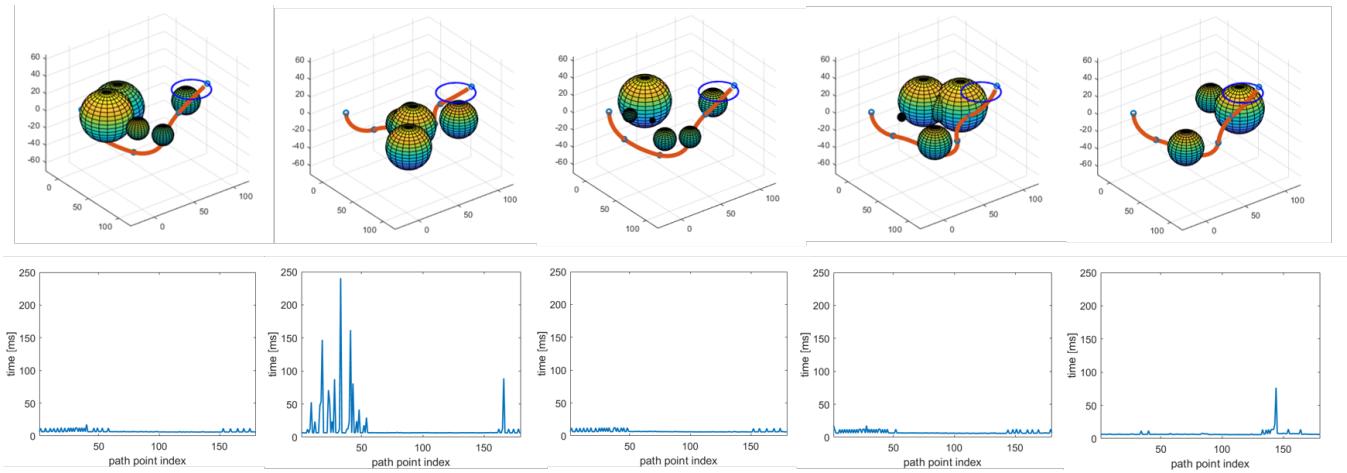


Fig. 5. Upper figures depict the continuum robot successfully navigating various intricate environments filled with sphere obstacles. The red piece-wise arcs with blue end points represent the continuum robot configuration calculated by our IK solver, and the blue lines represent the path to be tracked. The lower plots provide a visual representation of the corresponding time consumption. The x-axis represents the path point index, while the y-axis indicates the time consumption for each path point, measured in milliseconds.

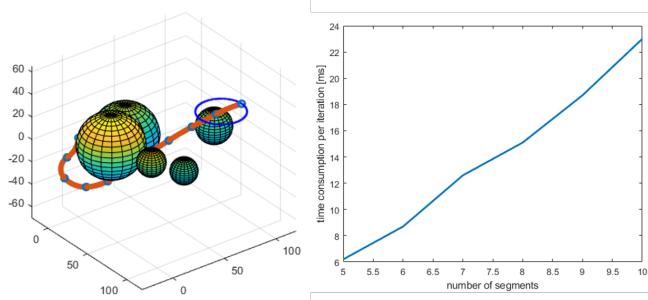


Fig. 6. Left: A 10-segment continuum robot tracking a path while avoiding obstacles. Right: The relationship between the number of segments and the time consumption per iteration.

- [3] J. Lai, K. Huang, and H. K. Chu, “A learning-based inverse kinematics solver for a multi-segment continuum robot in robot-independent mapping,” in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 576–582.
- [4] J. Santoso and C. D. Onal, “An origami continuum robot capable of precise motion through torsionally stiff body and smooth inverse kinematics,” *Soft Robotics*, vol. 8, no. 4, pp. 371–386, 2021.
- [5] A. Garriga-Casanovas and F. Rodriguez y Baena, “Kinematics of continuum robots with constant curvature bending and extension capabilities,” *Journal of Mechanisms and Robotics*, vol. 11, no. 1, p. 011010, 2019.
- [6] K. Wen and J. Burgner-Kahrs, “Modeling and analysis of tendon-driven parallel continuum robots under constant curvature and pseudorigid-body assumptions,” *Journal of Mechanisms and Robotics*, vol. 15, no. 4, p. 041003, 2023.
- [7] S. Lin, Y. Wang, A. Li, and W. Chen, “The piecewise constant curvature model for the forward and inverse kinematics of continuum robot based on lie theory,” in *2023 International Conference on Advanced Robotics and Mechatronics (ICARM)*. IEEE, 2023, pp. 386–391.
- [8] G. Qin, A. Ji, Y. Cheng, W. Zhao, H. Pan, S. Shi, and Y. Song, “A snake-inspired layer-driven continuum robot,” *Soft Robotics*, vol. 9, no. 4, pp. 788–797, 2022.
- [9] A. Sinha and N. Chakraborty, “Geometric search-based inverse kinematics of 7-dof redundant manipulator with multiple joint offsets,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5592–5598.
- [10] W. Zhang, Z. Yang, T. Dong, and K. Xu, “Fabrikc: An efficient iterative inverse kinematics solver for continuum robots,” in *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2018, pp. 346–352.

- [11] D. Kolpaschikov, O. Gerget, and V. Danilov, “Fabrikx: Tackling the inverse kinematics problem of continuum robots with variable curvature,” *Robotics*, vol. 11, no. 6, p. 128, 2022.
- [12] A. Aristidou and J. Lasenby, “Fabrik: A fast, iterative solver for the inverse kinematics problem,” *Graphical Models*, vol. 73, no. 5, pp. 243–260, 2011.
- [13] H. Wu, J. Yu, J. Pan, and X. Pei, “A novel obstacle avoidance heuristic algorithm of continuum robot based on fabrik,” *Science China Technological Sciences*, vol. 65, no. 12, pp. 2952–2966, 2022.
- [14] T. G. Thuruthel, B. Shih, C. Laschi, and M. T. Tolley, “Soft robot perception using embedded soft sensors and recurrent neural networks,” *Science Robotics*, vol. 4, no. 26, p. eaav1488, 2019.
- [15] J. Wu, B. Zhang, L. Wang, and G. Yu, “An iterative learning method for realizing accurate dynamic feedforward control of an industrial hybrid robot,” *Science China Technological Sciences*, vol. 64, no. 6, pp. 1177–1188, 2021.
- [16] N. Guochen, W. Li, G. Qingji, and H. Dandan, “Path-tracking algorithm for aircraft fuel tank inspection robots,” *International Journal of Advanced Robotic Systems*, vol. 11, no. 5, p. 82, 2014.
- [17] G. Niu, Y. Zhang, and W. Li, “Path planning of continuum robot based on path fitting,” *Journal of Control Science and Engineering*, vol. 2020, pp. 1–11, 2020.
- [18] S.-S. Chiang, H. Yang, E. Skorina, and C. D. Onal, “Slinski: State lattice based inverse kinematics-a fast, accurate, and flexible ik solver for soft continuum robot manipulators,” in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2021, pp. 1871–1877.
- [19] X. Song, M. Aigner, F. Chen, and B. Jüttler, “Circular spline fitting using an evolution process,” *Journal of computational and applied mathematics*, vol. 231, no. 1, pp. 423–433, 2009.