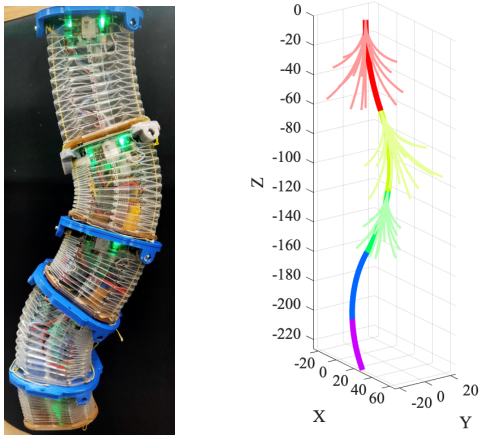
SLInKi: State Lattice based Inverse Kinematics – A Fast, Accurate, and Flexible IK Solver for Soft Continuum Robot Manipulators

*Shou-Shan Chiang, *Hao Yang, Erik Skorina and Cagdas D. Onal

***Abstract*— Soft continuum robots offer unique properties that cannot be achieved using rigid linkage based robot manipulators. Their dexterity and intrinsic compliance deliver the ability to navigate constrained environments and operate in unprecedented ways. Although Jacobian velocity matrix based method is a widely used approach to solve inverse kinematics (IK) problems for traditional rigid robots, the drawbacks of this method emerge obviously while solving IK problems of continuum robots, such as high computational cost with no solution guarantees. Attempts to provide alternative solutions suffer from limitations due to the computational complexity and vast functional workspace of continuum manipulator postures. Here, we propose a heuristic approach, State Lattice based Inverse Kinematics Solver (SLInKi), which is inspired by concepts originally developed for solving path-finding problems to solve the IK problem of a soft continuum robot. The algorithm implementation is intuitive, runs in real time, and combines the strengths of two algorithms in a unique package that surpasses existing methods in adjustability and efficiency. Several simulation case studies and real robot experiments demonstrate that the proposed approach is flexible, computationally efficient, and highly accurate as compared to the state of the art.**



(a) (b)

Fig. 1: (a) An origami-inspired continuum robot is modular and scalable. The current prototype consists of five modules, each being able to bend in 3-D and change in length. (b) An instance of robot pose simulated by SLInKi search algorithm, with alternate module configurations searched.

I. Introduction

Continuum robots bend continuously along their length instead of at discrete joints [1]. They have attracted attention during the past decade because of their compliance, safety and ability to negotiate constrained environments. These properties make inverse kinematics (IK) calculations difficult for continuum robots. Traditional numerical methods that work well for rigid robots, such as Jacobian based approaches, involve very high computational costs for continuum robots, with no guarantee to find a correct solution [2].

Continuum robot bodies trace smooth curves and, as a result, they possess theoretically infinite degrees of freedom (DoF). Making the common assumption that each finite section of the continuum body follows a constant curvature arc, bending angle and arc length are sufficient for their kinematic representation [3]. Using these parameters, end-effector poses can be calculated via forward kinematics [4]. In contrast, the inverse problem - using tip poses to calculate joint profiles - is much more difficult for continuum manipulators, since they do not always possess closed-form solutions [5]–[8].

In general, geometric Jacobian is still one of the most common approaches to solve IK problems [9], [10], but some researchers have presented other feasible methods due to its shortcomings. Courty et. al. implemented a Monte Carlo algorithm [11], but this scheme is very time consuming at each iteration, which makes the framework far less efficient for structures with larger DoFs. Shinha et. al. applied a searching-based algorithm to solve IK problems [12], but their approach is substantially developed for redundant manipulators composed of rigid bodies.

Continuum robots are highly redundant systems. It is possible to find inspiration for the continuum IK problem based on appropriate methods in other areas, such as motion planning. A path traced by a hypothetical mobile robot that follows bounded constant-curvature trajectories in 3-D space is similar to the shape of a continuum arm. This fact can be utilized to effectively solve the complicated IK problem of continuum robots [13]. State lattice-based searching is one of the motion planning algorithm that especially suits the scenarios where the orientation does matter, for instance, in mobile robot navigation process [14], [15]. Williams et. al. proposed an approach which involves A* to solve the inverse kinematics problem [16]. It exhibits the potential of a path-finding algorithm in IK problems. Chaichawananit et. al. also presented a solution for applying A* algorithm to solve the

This material is based upon work partially supported by the National Science Foundation (NSF) under Grant No. CMMI-1752195. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

The authors are with the Robotics Engineering and Mechanical Engineering Departments, Worcester Polytechnic Institute, MA 01609, USA. All correspondence should be addressed to Cagdas D. Onal cdonal@wpi.edu

*Shou-Shan Chiang and Hao Yang are co-first authors of this paper.

IK problem on 6-DoF rigid robot arms [17]. Xiao et. al. presented similar ideas working on continuum robots [18].

FABRIK [19], a newly developed algorithm for solving inverse kinematics, was presented in 2010. This method iterated forward and backward to adjust the poses, allowing for real-time solutions and smooth postures. Aristidou et. al. extended FABRIK from the original setting [20], in which the authors defined the convergence and let the link lengths be changeable. However, till then this framework was only applicable to rigid bodies. In 2018, Zhang et. al. proposed a method to expand FABRIK framework to continuum robots, denoted as FABRIKc [21], where the lowercase 'c' stands for 'continuum.' They introduced virtual links and joints to represent the continuum sections in a way that could be used by FABRIK. This algorithm inherits FABRIK's advantages and provides a smooth configuration with relatively low computational costs. Nevertheless, the fact that the arc length for each segment remains unchangeable poses a significant concern for this framework.

In this paper, we introduce a real-time inverse kinematic solver for continuum robots with high flexibility, shown in Fig. 1. This novel method combines the effectiveness of FABRIKc with the efficiency of state lattice-based search algorithms, denoted as State Lattice-based Inverse Kinematics (SLInKi). Specifically, state lattice-based searching is implemented to find a feasible configuration for the first $(n-2)$ segments, without involving orientation. Later on, we apply to the last 2 segments our modified FABRIKc framework, which is able to adjust arc length and bending angles to fit the desired end-effector position and orientation. To ensure the success of algorithmic integration, we also propose a method for redefining the searching goal. The detailed algorithm structure is discussed in Section II.

To validate the performance of SLInKi, we conduct several tests to assess multiple properties using simulation and experimental results with our custom origami-inspired continuum robot. This robot comprises 5 cable-driven origami-inspired modules with unique properties as presented in our prior work [3], [22]. First, we present three different cost functions for the state-lattice portion to comparatively evaluate their performances in simulation. The results indicate that minimized change in cable lengths provides the smoothest motion and highest efficiency, which leads to reduced dithering or shaking. We conduct both single pose and multi-poses experiments to validate this hypothesis. For the following tests, we implement cable-length-change minimization for SLInKi, and evaluate the overall performance of three different IK solvers (SLInKi, Optimization, FABRIKc) in simulation. From these tests, we conclude that SLInKi requires the shortest time to calculate a feasible solution, even for a 5-section continuum robot. The fast computation speed provides the real-time operation capability to continuum robots. To exhibit this, we perform trajectory following test on our experimental prototype, in which SLInKi showcases its real-time processing ability for the robot end-plate to trace a rectangular trajectory.

The contributions of this paper include:

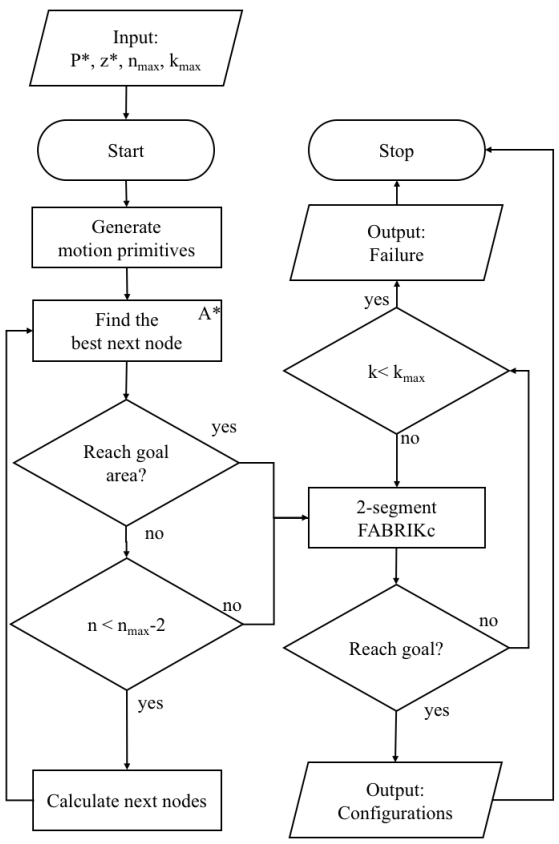


Fig. 2: Flow chart for the SLInKi algorithm. Notice that P^* & z^* denote goal position & orientation separately, while n_{max} and k_{max} represent the overall segment numbers and iteration limits of FABRIKc. The details of generating motion primitives and extended FABRIKc are presented in section II A & C. The pseudo-code for algorithms are placed in the Appendix.

- Applying motion planning concepts to robot kinematic problem.
- Investigating different cost functions for further research potentials in distinctive scenarios.
- Extending the capability of FABRIKc to solve the inverse kinematics of continuum manipulators, which not only bend continuously but also vary in segment lengths.
- Combining FABRIKc and a motion-planning algorithm, state lattice-based searching, to create a fast and feasible inverse kinematic solver for continuum manipulators.

II. Approach and Methods

Our approach treats the configuration profiles of continuum robot as a path. Due to its hyper-redundant nature, the end-effector is able to reach a desired pose with multiple configurations. The inverse kinematic solver is implemented to find proper configurations for each segment of the robot. From this aspect, the path-finding algorithms share a similar objective: to explore and connect adjacent nodes until arriving the destination node.

A. State Lattice-based Searching

In our application, each segment of the continuum robot has 6 DoFs in the task space. However, the position and orientation are correlated, which constrain the end-effector orientation when translating to the desired position. In this regard, we consider the IK of our robot to be a kinodynamic issue, which could be solved via a motion planning algorithm - state lattice-based searching [23].

Jones and Walker introduced arc length(s), curvature(κ) and the right-handed rotation angle(ϕ) as parameters of the continuum configuration [24]. Inspired from this, we can try to convert the continuous problem to a sequential decision process by utilizing discretization. We consider the tip position and orientation as states. The decisions, corresponding to parameters of next module (s, κ, ϕ), are made only at discrete states by following constant curvature assumptions. The tip poses of the module are the nodes in the lattice and the curves that connect the poses are the edges in the lattice.

We generate the next motion primitive with the parameters(s, κ, ϕ), and calculate the corresponding poses with forward kinematics. The detailed process is exhibited in Appendix II. We also choose a snap-width, which is inversely proportional to the size of the state-lattice. By choosing the snap-width, we can decide the density of motion primitives (nodes). Smaller snap-width would enlarge the coverage of searching path, unexpectedly increasing computational cost as well. Higher the coverage rate, better the path performance in a path-finding task. In our approach, higher coverage rate provides more options for every segment and generate a smoother, shorter configuration. However, this approach aims to create a fast, feasible inverse kinematics solver. A high density coverage is not essential for creating an acceptable configuration. On the other hand, optimization through length or smoothness may be considered as cost functions, which will be introduced in next paragraph.

To find configurations with candidate nodes, we apply the weighted A* search algorithm [25] [26] to our approach. This algorithm introduces two functions to assess and select the node for the next step during navigation. The $g(n)$ function is the path length from origin to node n, and the heuristics function $h(n)$ is the estimated distance from node n to the goal node:

$$h(n) = \sqrt{(x^* - x_n)^2 + (y^* - y_n)^2 + (z^* - z_n)^2} \quad (1)$$

where (x^*, y^*, z^*) is the goal node, the desired position, and (x_n, y_n, z_n) is the state of node n, which is the tip position of nth potential configuration.

A bias ε is applied on $h(n)$. The total cost function can be $f(n) = g(n) + \varepsilon h(n)$, which means the node n is rated with the difficulties not only from the start node to n but also from n to the goal node. The heuristics function with bias $(\varepsilon > 1)$ guiding search increases the performance obviously, especially in our application, which has rare obstacles in workspace of manipulator. We can expand the path toward the goal position straightforward without exploring too many unrelated nodes.

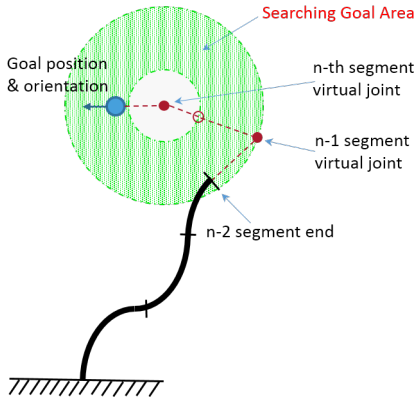


Fig. 3: The concept of the searching goal setup is shown in the figure: initially, (n-2) segments implement state lattice-based searching to access the searching goal area with any feasible orientations. The searching goal area is comprised of a ring, with the length of the final (n) segment being the inner radius and the distance to the virtual joint of the second-to-last (n-1) segment being the outer radius, padded green in the figure. Starting from this area, FABRIKc is guaranteed to find IK solutions in free space for the last 2 segments.

B. Cost Function Development

Although heuristics helps us exploit the shortest path from current tip to desired end, for continuum robot we specifically focus on the configuration profile, as introduced above, which is highly correlated to $g(n)$ function. According to our previous work [3], to achieve smooth trajectory control, the IK solver of a continuum robot can be treated as an optimization problem, in which we include inequality constraints (limits) on some variables into the numeric procedure. As mentioned in the paper, our robot module, designed as a tendon driven actuator, involves curvature, bending direction and arc length as intermediate variables to calculate desired cable length messages. These length messages will be transformed to desired encoder counts and then drive the DC motors, finally achieving goal poses. Therefore, a projection from curvature - arc length - bending direction (parameters for completely defining a module shape) domain to cable length domain is involved. This projection can be written as:

$$l_1 = 2\sin(\frac{\kappa s}{2})(\frac{1}{\kappa} - d\sin(\phi)) \quad (2)$$

$$l_2 = 2\sin(\frac{\kappa s}{2})(\frac{1}{\kappa} + d\sin(\frac{\pi}{3} + \phi)) \quad (3)$$

$$l_3 = 2\sin(\frac{\kappa s}{2})(\frac{1}{\kappa} - d\cos(\frac{\pi}{6} + \phi)) \quad (4)$$

where l_1, l_2 and l_3 are the lengths of cables 1,2 and 3, κ,s,ϕ represent curvature, arc length and bending direction respectively, d is the original distance from the top center of a module to the cable attachment point. Still, the deviation of equations above is stated in our pre-posted paper [3]. Notice that in cable length domain, straight line is the shortest distance from initial to target. Thus, we can derive the minimized cable length change cost function:

$$g_{cable}(n) = \sum_{i=1}^{n}[(l_{i1} - l_{i1p})^2 + (l_{i2} - l_{i2p})^2 + (l_{i3} - l_{i3p})^2] \quad (5)$$

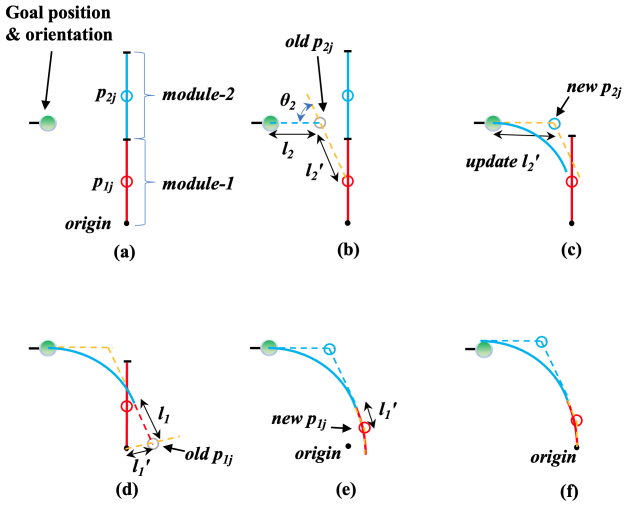


Fig. 4: An visualization of the extended FABRIKc on a 2-segment continuum robot. (a) The initial and the target poses, where p_{1j} and p_{2j} are the virtual joints for module 1 and 2. (b) Forward reaching phase: Update module-2 to satisfy the desired position and orientation. The upper virtual link 2 rotates and translate to reach the goal pose. The p_{2j} follows the link. The lower virtual link 2 rotates toward the virtual joint 1, and adjust the length to connect virtual joint 2 and 1, where is the extending step that does not happen in FABRIKc. (c) Update the length of upper visual link 2 with l_2'. Due to the length changing, translate the p_{2j} and lower virtual link 2. Module-2 is done with forward reaching phase. (d) Update upper virtual link 1 to connect module-2. The two virtual links should be in an identical orientation. Translate p_{1j} along with the virtual link. Define the distance between the p_{1j} and the origin as l_1'. (e) Update the module-1 with l_1' and have new p_{1j}. Because module-1 is the module that connects to the base, the orientation of the lower virtual link is always identical to the base orientation. Module-1 is done with forward reaching phase. (f) Backward reaching phase: Translate the whole robot to the origin. When it is done, the end-effect should be in a proper orientation with little position errors. Then, repeat from (b) to minimize the error.

where l_{i1}, l_{i2}, l_{i3} are cable lengths for current state of each module and $l_{i1p}, l_{i2p}, l_{i3p}$ are for previous state.

In addition, the arc length (Minimum Arc-length Cost) and curvature for each module (Minimum Curvature Cost) can also be effective cost functions, as presented in another paper we posted before [22]. The equations are shown below, and we will evaluate their performance with minimum cable change cost in the experiment section.

$$g_{arc}(n) = \sum_{i=1}^{n} s_i, \quad g_{cur}(n) = \sum_{i=1}^{n} \kappa_i \tag{6}$$

C. Extended FABRIKc

FABRIKc is used to refine the robot configuration in goal position & orientation. However, Zhang's approach [21] assumes a constant arc length, which does not agree with our robot design. We modify the forward reaching part and adjust the virtual link lengths to enable arc length variation, while two each length of the virtual link pair remains identical. When approximating the desired pose, we transfer the virtual links back into bending angles and arc lengths for each segment. An example of the detailed operation is exhibited in Fig. 4, and a supplemental pseudo code is posted in Appendix II.

D. Combined Inverse Kinematics Solver

The search strategy works well for reaching goal position, but not the same for orientation. On the other hand, although FABRIKc provides solutions for both position and orientation, the computational cost increase exponentially when adding segment numbers, due to its forward-backward iteration process. Our approach combines the high speed searching based strategy and the accurate end pose provider FABRIKc together: searching algorithm calculates configurations from base towards goal position; FABRIKc then finetunes the position and orientation using the last 2 segments. We name this combined inverse kinematic solver as State Lattice-based Inverse Kinematics (SLInKi). Its detailed algorithm framework is illustrated in a flow chart in Fig. 2.

E. Searching Goal

To decide where the path-finding inspired IK solver should hand over jobs to FABRIKc, we need to figure out workspace of last two segments, which implement FABRIKc to reach goal position & orientation. In extending FABRIK [20], the authors exhibited coverage equations. We quote their work and define the last 2-segments' FABRIKc reaching workspace within two concentric spheres, setting it as goal area for candidate tip positions of the (n-2) segments. Notice that this should be an orientation-free coverage area. No matter what orientation the end node is, FABRIKc will be able to connect it with final 2-segments, and the end-effector will be guaranteed to reach the desired goal position & orientation in free space. Based on this concept, we could now focus on approaching the goal position without considering orientation in searching part. In addition, in this setup the cost function only needs to be considered for distance and bending configuration. The detailed searching goal strategy is illustrated in Fig. 3.

III. Experiments

In this section, we conduct several experiments to verify the capability of our IK algorithm - SLInKi, as well as assessing multiple aspects of its properties. First, we adopt minimum cable length change as the cost function in the SLInKi program, as it is determined having the best performance in the previous work [22]. We compare the overall performance among three different IK solvers intuitively and evaluate throughout the results. Finally, a trajectory following test for real robot is conducted, in which SLInKi demonstrates it's real-time capability. Here we introduce a brand new origami-inspired modular continuum robot applied with high flexibility, shown in Fig. 1. We implement SLInKi to obtain IK solutions for a given rectangle trajectory and assess the real robot's following performance via observation.

A. Design, Manufacturing and Assembly

We developed a tendon-driven modular continuum robot, which is described in detail in our previous paper [3]. The robot module is capable of significant length change (about 1.25 times its original length) and highly resistant in torsion (73 times stronger than a similar size silicone module). These

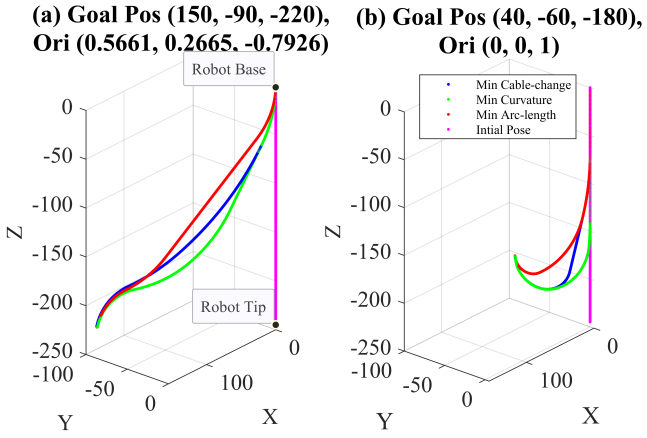


Fig. 5: Experimental results for comparing single pose performance of multiple cost functions. The magenta line is the continuum robot in the home position. Other curves are the final configurations from SLInKi with different cost-functions. The red, green, and blue curves are respectively optimized for minimum arc length, minimum curvature, and minimum cable length change. All three configurations lead the end-effector to the desired pose. The red curve tends to move toward the goal in shortest path. The green curve leans to stay straight and bends only when it is essential. The blue curve is similar to the green one because the configuration moves from the home position, where all segments are straight.

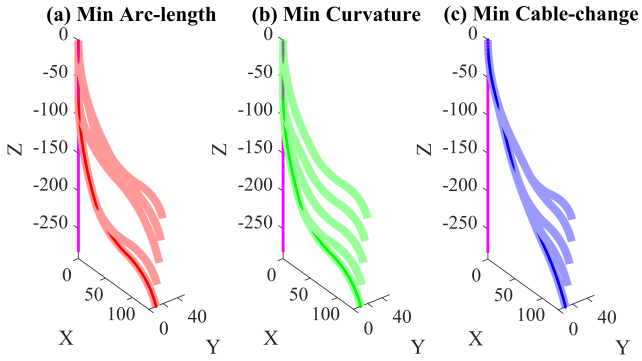


Fig. 6: Some results of the cost function experiments in sequence poses. The magenta line is the continuum robot in the home position. The task is to reach a sequence pose from (50,20,-230) at orientation (0.5547,0,-0.8321) to (90,-40,-222) at orientation (-0.2250,0,-0.9744) in 4 steps. (a) is optimization for arc length. (b) is optimization for minimum curvature. (c) is optimization for minimum cable length change. The configurations tends to stay in the previous status and only move when it needs to. This shows that (a) and (b) are unsuited to moving trajectories because some segments have to move back and forth frequently during the task.

two characteristics are attributed to the tubular accordion-like tessellated origami structure fabricated from a 0.178 mm-thin PET sheet. The origami structure was folded following the Yoshimura origami crease pattern. Each origami continuum module consists of a compliant plastic body, an acrylic end plate, and a custom-made PCB, using an 8-bit ATmega32U4 as microprocessor, for embedded control. Further, a total of four electrical wires run through the cavity of the origami structure in a helical shape, providing power and communication between continuum modules. The module control boards communicate to a desktop computer by using inter-integrated circuit(I2C) protocol.

B. Cost Function Comparison: Single Pose

By investigating the difference among the three cost functions, we shall determine the best performer and incorporate it to SLInKi. We aim to assign random target tip position and orientation within the robot workspace, then applying distinct cost functions to SLInKi and observing the features. Specifically, these functions are minimized arc length, where shorter arc length represents higher stiffness in our cable-driven origami structure; minimized curvature, where less curvature means more even tension on the cables and better endurance; minimized cable length change, where less changes lead to less jerking and power costs when transforming from one pose to another. In Fig. 5, we exhibit different configurations to achieve the same target, by involving separate cost functions. Meanwhile, to verify that minimum cable change cost is the most appropriate one towards our system, we should conduct another multi-pose test to let the robot follow a designated trajectory.

C. Cost Function Comparison: Multiple Poses

In a single pose experiment, we may still not be able to intuitively comprehend the superiority of minimized cable change cost function, so here we extend the desired target to a continuum path, particularly a straight line with multiple poses interpolated. As shown in Fig. 6, minimum cable change function involves the most fluent shape changing, or to say, the robot uses minimum power to switch its poses. Consider the case to change cable length from 20 to 40, is it better to jump from 20 to 60 and then back to 40, or to continuously move from 20, to 30, and then 40? Minimum cable change function provides solutions in the latter condition.

D. Algorithms Performance Comparison

Now that we have included minimum cable change as the cost function, we can proceed to validate our algorithm and prove its advantages. Specifically, we shall compare our algorithm with other IK solvers and evaluate their results. To achieve this, multiple single pose experiments are conducted. In each experiment, we randomly set a desired end-effector position and orientation, then implement three different IK algorithms: FABRIKc, numerical method based minimum cable optimized IK solver, and SLInKi to create joint configuration profiles. The overall robot shape generated from these profiles are compared, thus the algorithm performances can be assessed. Experiment results, including robot shapes and intermediate virtual joints produced from three separate algorithms are shown in Fig. 7, while the overall calculation time, actual end-effector position & orientation values and errors between desired ones are presented in Tables I-IV.

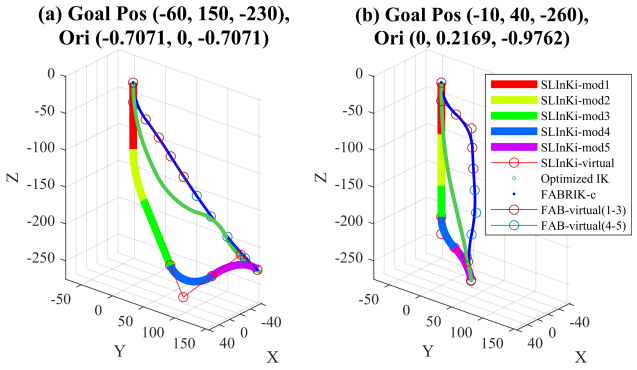


Fig. 7: Some results of the algorithms performance experiments. The blue curve is the IK solution from FABRIKc. The green curve is from the Jacobian method based IK solver optimized for minimum change. The multi-color curve is the result by SLInKi optimized for minimum change, with each color representing a manipulator segment. The skeletons on FABRIKc and SLInKi configurations display the virtual links and joints. These figures only show the configurations, while the details of time usage and errors are shown in the tables below. (a) corresponds to Table III. (b) corresponds to Table IV.

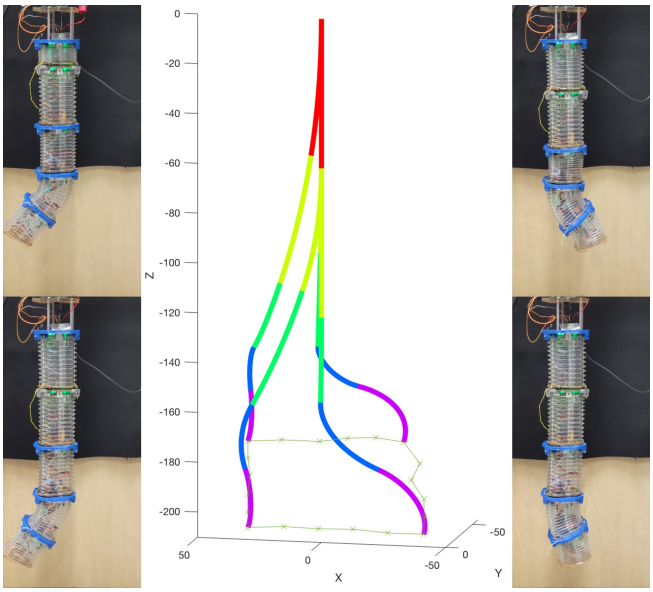


Fig. 8: This figure shows the rectangular trajectory following experimental results of both simulation and real robot test. The simulation, as shown in the middle, documents the 5-segment robot's trajectory following upper left - lower left - lower right - upper right sequence of the rectangle. Each side of the rectangle is interpolated by 5 waypoints, and the robot configuration profiles are calculated via SLInKi. The real robot shape shown in the figure corresponds to the configuration at four corners of a rectangle, respectively.

E. Real Robot Trajectory Following Test

The solutions of SLInKi for a series desired positions would not be similar due to its discretization property. Segments of the continuum robot have to bend, extend or shorten from this pose configuration to next pose configuration. It will lengthen response time and decrease energy efficiency. Adjusting the cost function in SLInKi can change the behavior of node selection and optimize the pose configuration for minimum change from previous pose configuration.

On our cable-driven origami manipulator, the bottleneck for the response time is the rotation speed of the motors. The minimum change of cable length between two configurations shorten the response time and increase the stability. From previous work, we know that the single segment can be mapping both general configuration space and cable-length space [22]. In cable-length space, a point means a configuration of a single segment. The distance between two points means the amount of change between the configurations. We apply sum of the cable length change amount into the cost function in SLInKi to count the cable-length change of all segments into configuration searching.

To validate the multi-poses performance of SLInKi, we conduct an experiment where simulation and real robot testing progress simultaneously. As shown in Fig. 8, the real robot configurations shown at the four corners correspond respectively to the simulation results of the four corners of the rectangular trajectory. The experiment shows that the robot is able to follow the trajectory in real time within an acceptable error performance, which certificates the real-time capability of SLInKi.

IV. Conclusions and Future Works

We propose the State Lattice-based Inverse Kinematics Solver (SLInKi) as a fast, accurate and flexible inverse kinematics solver for extensible continuum manipulators. SLInKi combines algorithms originally developed for solving path-planning problems with our modification of the existing continuum manipulator kinematic algorithm FABRIKc. This allows us to utilize the efficiency of path-planning algorithms while maintaining a high degree of accuracy. We have adopted a state lattice-based approach with weighted A* to find the feasible configurations for the manipulator to approach the objective quickly, and then modified FABRIKc to allow it to solve the kinematics of the extensible over the final stretch. We have verified this approach by applying it on the 5-segment origami robot [3]. The robot moves following the designed rectangular path smoothly.

One advantage of the weighted A* approach for most of the workspace is that it allows inverse kinematics to be performed free from a manipulator model. We can freely adjust the number of the section for the continuum robot, taking advantage of the modular nature of our manipulator. In addition, the use of the cost function allows us to easily swap out what the algorithm is minimizing, letting us fine-tune our computed states depending on the robot and the application.

Our future research on the SLInKi algorithm will cover several aspects, including i) obstacle avoidance (which is one of SLInKi's inherent features designed based on path-finding algorithms), ii) grow-to-shape implementation, and iii) feedback combination and application to a controller for continuum robots.

REFERENCES

[1] G. Robinson and J. Davies, "Continuum robots - a state of the art," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 4, pp. 2849–2854 vol.4.

[2] P. Sears and P. E. Dupont, "Inverse kinematics of concentric tube steerable needles," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 1887–1892, 2007.

[3] J. Santoso, E. H. Skorina, M. Luo, R. Yan, and C. D. Onal, "Design and analysis of an origami continuum manipulation module with torsional strength," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2098–2104, IEEE.

[4] A. Aristidou and J. Lasenby, "Inverse Kinematics: A review of existing techniques and introduction of a new fast iterative solver," p. 74.

[5] G. S. Chirikjian, "Inverse Kinematics of Binary Manipulators Using a Continuum Model," vol. 19, no. 1, pp. 5–22.

[6] 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)*, pp. 576–582.

[7] S. Neppalli, M. Csencsits, B. Jones, and I. Walker, "A geometrical approach to inverse kinematics for continuum manipulators," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3565–3570, IEEE.

[8] S. Neppalli, M. A. Csencsits, B. A. Jones, and I. D. Walker, "Closed-Form Inverse Kinematics for Continuum Manipulators," vol. 23, no. 15, pp. 2077–2091.

[9] L. Sciavicco and B. Siciliano, "A solution algorithm to the inverse kinematic problem for redundant manipulators," vol. 4, no. 4, pp. 403–410.

[10] W. A. Wolovich and H. Elliott, "A computational technique for inverse kinematics," in *The 23rd IEEE Conference on Decision and Control*, pp. 1359–1363.

[11] N. Courty and E. Arnaud, "Inverse Kinematics Using Sequential Monte Carlo Methods," in *Articulated Motion and Deformable Objects* (F. J. Perales and R. B. Fisher, eds.), Lecture Notes in Computer Science, pp. 1–10, Springer.

[12] 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)*, pp. 5592–5598.

[13] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles," vol. 1, no. 1, pp. 33–55.

[14] R. Knepper and A. Kelly, "High Performance State Lattice Planning Using Heuristic Look-Up Tables," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3375–3380, IEEE.

[15] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," vol. 26, no. 3, pp. 308–333.

[16] S. G. D. Williams, "Using the A-Star Path-Finding Algorithm for Solving General and Constrained Inverse Kinematics Problems,"

[17] J. Chaichawananit and S. Saiyod, "Solving inverse kinematics problem of robot arm based on a-star algorithm," in *2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pp. 1–6.

[18] J. Li and J. Xiao, "A general formulation and approach to constrained, continuum manipulation," vol. 29, no. 13, pp. 889–899.

[19] A. Aristidou and J. Lasenby, "FABRIK: A fast, iterative solver for the Inverse Kinematics problem," vol. 73, no. 5, pp. 243–260.

[20] A. Aristidou, Y. Chrysanthou, and J. Lasenby, "Extending FABRIK with model constraints: Extending FABRIK with model constraints," vol. 27, no. 1, pp. 35–57.

[21] 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)*, pp. 346–352.

[22] J. Santoso and C. D. Onal, "An Origami Continuum Robot Capable of Precise Motion Through Torsionally Stiff Body and Smooth Inverse Kinematics,"

[23] M. Pivtoraiko and A. Kelly, "Generating near minimal spanning control sets for constrained motion planning in discrete state spaces," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3231–3237.

[24] B. A. Jones and I. D. Walker, "Kinematics for multisection continuum robots," vol. 22, no. 1, pp. 43–55.

[25] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," vol. 4, no. 2, pp. 100–107.

[26] I. Pohl, "Heuristic search viewed as path finding in a graph," *Artificial intelligence*, vol. 1, no. 3-4, pp. 193–204, 1970.

APPENDIX I
COST FUNCTION PERFORMANCE COMPARISON TABLES

TABLE I: Performance Comparison for Cost Functions, Single Pose, Goal Position: (150;-90;-220) & Orientation: (0.5661;0.2265;-0.7926)

		min-arc len	min-curv	min-cable change
arc-length (mm)	1	56.00	60.67	60.67
	2	70.00	70.00	65.33
	3	70.00	60.67	65.33
	4	48.71	58.04	54.69
	5	45.57	52.98	49.51
curvature (10^{-3}/mm)	1	12.50	7.692	7.692
	2	0.000	0.000	3.571
	3	0.000	7.692	3.571
	4	9.470	8.750	6.141
	5	21.85	22.34	20.41

TABLE II: Performance Comparison for Cost Functions, Single Pose, Goal Position: (50;-100;-140) & Orientation: (0;0;1)

		min-arc len	min-curv	min-cable change
arc-length (mm)	1	60.67	65.33	65.33
	2	70.00	70.00	65.33
	3	46.67	51.33	56.00
	4	35.47	47.99	43.40
	5	30.30	52.04	48.92
curvature (10^{-3}/mm)	1	7.692	3.571	3.571
	2	0.000	0.000	3.571
	3	18.03	12.03	7.217
	4	25.69	27.25	34.41
	5	51.97	20.68	24.49

TABLE III: Single Pose IK Solutions for Desired Position: (-60,150,-230) & Desired Orientation: (-0.7071,0,-0.7071)

	FABRIKc	Opti. based	SLInKi
time(ms)	183	112	12.2(9.9+2.3)
position(mm)	(-60,149,-231)	(-62,152,-234)	(-59,150,-230)
pos-err(mm)	0.863	4.52	0.766
orientation	(-0.71,0,-0.71)	(-0.4,0.77,-0.5)	(-0.71,0,-0.71)
ori-err	0	0.85	0

TABLE IV: Single Pose IK Solutions for Desired Position: (-10,40,-260) & Desired Orientation: (0,0.2169,-0.9762)

	FABRIKc	Optimal IK	SLInKi
time(ms)	206	124	11.8(10.1+1.7)
position(mm)	(-9.7,40,-261)	(-10,40,-260)	(-9.9,40,-259)
pos-err(mm)	0.775	0	0.632
orientation	(0,0.22,-0.98)	(0,0.22,-0.98)	(0,0.22,-0.98)
ori-err	0	0	0

APPENDIX II
PSEUDO CODES FOR SLINKI

Algorithm 1: SLInKi

Result: The configurations of continuum robot
Generate Motion Primitives();
Set Goal Area;
A* search() approach goal area;
Extended FABRIKc find the final pose;
Export configurations ;

Algorithm 2: Generate Motion Primitives

Result: List of transformation matrix and the index of motion primitives
$\kappa \leftarrow 0, \phi \leftarrow 0, s \leftarrow 0, m \leftarrow 0$;
while $s < s_{max}$ **do**
  $T_m = ForwardKinematics(s, \kappa, \phi)$;
  $index_m = [s, \kappa, \phi]$;
  $s = s + \delta_s$;
  $m = m + 1$;
end
$\kappa \leftarrow \delta_\kappa$
while $\kappa < \kappa_{max}$ **do**
  $s \leftarrow \delta_s$
  while $s < s_{max}$ **do**
    $\phi \leftarrow 0$
    while $\phi < \phi_{max}$ **do**
      $T_m = ForwardKinematics(s, \kappa, \phi)$;
      $index_m = [s, \kappa, \phi]$;
      $\phi = \phi + \delta_\phi$;
      $m = m + 1$;
    end
    $s = s + \delta_s$;
  end
  $\kappa = \kappa + \delta_\kappa$;
end
T, index

TABLE V: Nomenclature used in kinematics modeling and algorithm

s, s_{max}	Arc length and maximum allowable value
κ, κ_{max}	Curvature and maximum allowable value
ϕ, ϕ_{max}	Bending orientation and maximum allowable value
m	Index of motion primitives
T_m	Transformation matrix of mth (s, κ, ϕ) setting
$\delta_s, \delta_\kappa, \delta_\phi$	snap-width of s, κ, ϕ
n	Number of segments
t	Index of the segments, t=1,2...n.$t = 1$ for the most proximal segment and $t = n$ for the most distal segment
s_t	Arc length of tth segment
κ_t	Curvature of tth segment
ϕ_t	Bending orientation of tth segment
e	Position error
ϵ	The threshold of the error
l_t	Length of the virtual links on tth segment
p^*	The target position in the world coordinate
$\hat{z}^*$	The target orientation in the world coordinate
p_{tj}	Virtual joint position of tth segment
p_{te}	End position of tth segment
p_{tb}	Base position of tth segment
$\hat{z}_{te}$	End orientation of tth segment
$\hat{z}_{tb}$	Base orientation of tth segment
k, k_{max}	The iteration index and the maximum allowable iterations

Algorithm 3: Extended FABRIKc

Result: List of p_e, p_j, p_b for each segment
while $e > \varepsilon$ *and* $k < k_{max}$ **do**
  // Forward reaching;
  $p_e \leftarrow p^*$;
  $\hat{z}_e \leftarrow \hat{z}^*$;
  $t \leftarrow 1$
  while $t \leq n$ **do**
    // translate and rotate the end of tth segment
    $p_{te} = p_e$;
    $\hat{z}_{te} = \hat{z}_e$;
    // get old virtual joint position and base orientation
    $p_{tj} = p_{te} - l_t \cdot \hat{z}_{te}$;
    $\hat{z}_{tb} = (p_{tj} - p_{(t+1)j})/||p_{tj} - p_{(t+1)j}||$;
    // adjust the length of virtual link
    $l_t = ||p_{tj} - p_{(t+1)j}||$;
    // update virtual joint and base of tth segment
    $p_{tj} = p_{te} - l_t \cdot \hat{z}_{te}$;
    $p_{tb} = p_{tj} - l_t \cdot \hat{z}_{tb}$;
    // set the pose for the next segment
    $p_e = p_{tb}$;
    $\hat{z}_e = \hat{z}_{tb}$;
    $t = t + 1$;
  end
  // Backward reaching;
  $p_b = p_{base}$;
  while $t \geq 1$ **do**
    $\nu = p_b - p_{tb}$;
    // translate tth segment
    $p_{tb} = p_{tb} + \nu$;
    $p_{tj} = p_{tj} + \nu$;
    $p_{te} = p_{te} + \nu$;
    // set the base position of the next segment
    $p_b = p_{te}$;
    $t = t - 1$;
  end
  $e = ||p^* - p_e||$;
  $k = k + 1$;
end