Towards Contact-Aided Motion Planning for Tendon-Driven Continuum Robots

Priyanka Rao¹, *Student Member, IEEE*, Oren Salzman², *Member, IEEE*, and Jessica Burgner-Kahrs¹, *Senior Member, IEEE*

Abstract-Tendon-driven continuum robots (TDCRs), with their flexible backbones, offer the advantage of being used for navigating complex, cluttered environments. However, to do so, they typically require multiple segments, often leading to complex actuation and control challenges. To this end, we propose a novel approach to navigate cluttered spaces effectively for a single-segment long TDCR which is the simplest topology from a mechanical point of view. Our key insight is that by leveraging contact with the environment we can achieve multiple curvatures without mechanical alterations to the robot. Specifically, we propose a search-based motion planner for a single-segment TDCR. This planner, guided by a specially designed heuristic, discretizes the configuration space and employs a best-first search. The heuristic, crucial for efficient navigation, provides an effective cost-to-go estimation while respecting the kinematic constraints of the TDCR and environmental interactions. We empirically demonstrate the efficiency of our planner-testing over 525 queries in environments with both convex and non-convex obstacles, our planner is demonstrated to have a success rate of about 80% while baselines were not able to obtain a success rate higher than 30%. The difference is attributed to our novel heuristic which is shown to significantly reduce the required search space.

I. INTRODUCTION

Tendon-driven continuum robots (TDCRs), bend their flexible backbones via tendon actuation, making them popular for medical and industrial applications for accessing areas with constricted entry-points. As navigating cluttered areas requires more intricate shapes, several segments are typically used, as each segment is capable of bending only in a C-shape. A multi-segment design comes at the price of complex actuation unit design [1] and tendon control. While methods like variable tendon routing [2], layer jamming [3], and locking mechanisms [4] vary curvatures, they demand mechanical modifications to the robot design. In this work we argue that in cluttered environments, we can use a simple single-segment long TDCR that exploits contacts with the environment (*contact-aided navigation (CAN)*) to achieve variations in curvature.

TDCRs can be used to deviate from the common obstacleavoidance paradigm, as they can interact with the environment due to their compliance. For example, they have been exploited for environmental sensing by actively tapping the surroundings [5]. They can be used for CAN as the environmental interactions deform the robot due to resulting



Fig. 1: (a) Problem statement: the end-effector of a TDCR must reach a given target pose in a cluttered environment, (b) Single-segment TCDR's unsuccessful solution (without CAN) which intersects an obstacle, (c) Three-segment TDCR's successful solution, (d) Single-segment's successful solution (with CAN) obtained by leveraging the obstacles to vary its curvature and reach the target pose.

external constraints on the robot shape, providing passive degrees of freedom (DOF) that can be leveraged to improve the workspace that the robot can access [6]. However, in scenarios cluttered with obstacles where the robot's endeffector must reach a target pose in order to carry out various tasks, a vital component is a motion planner capable of making combinatorial decisions about the necessary contacts to reach the desired pose. This motion planner, which is the focus of this paper, is responsible for identifying the sequence of actuation inputs required to navigate the robot.

Consider an environment with obstacles, where the robot needs to reach a target pose (see Fig. 1(a)). A singlesegment robot has limited workspace and cannot achieve the illustrated target pose, while avoiding contact with obstacles (Fig. 1(b)). To reach the target pose with obstacle avoidance, a multi segment robot is potentially required (Fig. 1(c)). A single segment long TDCR is essentially under-actuated for the task at hand as it only has two actuated DOFs (from bending and length insertion). However, CAN allows it to potentially leverage contacts with the obstacles to vary its curvature and reach the target pose, as shown in Fig. 1(d).

A. Related Work

Soft growing vine robots, despite being underactuated, leverage their follow-the-leader (FTL) deployment capability

©2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

¹Continuum Robotics Laboratory, Department of Mathematical & Computational Sciences, University of Toronto, ON, Canada

²Department of Computer Science, Technion, Haifa 3200003, Israel priyankaprakash.rao@mail.utoronto.ca.

for CAN. However, the robot path is modeled as a series of line segments [7], [8] to simplify trajectory generation between obstacles, and does not reflect the TDCR's continuous backbone. Selvaggio et al. [6] use inverse kinematics to actively steering vine robots, but only look at point-sized obstacles.

CAN has been exploited for hyperredundant robots where they have the advantage of additional DOFs that can be used to steer the end-effector towards the target pose.

Sampling-based approaches like Rapidly-exploring Random Trees [9] can perform poorly when required to traverse through narrow passages present between obstacles [10].

Search-based planning in the configuration space has been proposed to reach a target position [11], a target pose [10] and to find stable load configurations [12] for these multi-link robots. While search-based planning offers a more systematic approach to exploring the workspace, the proposed heuristics in the above works are tailored for multi-link robots, which offer increased maneuverability (due to higher DOF) for controlling the end-effector's pose.

Therefore, CAN for a single segment TDCR necessitates methods that respect the robot kinematics and the fact that it is underactuated. Existing controllers account for possible contacts [13], [14], but require user-defined trajectories. Given a trajectory, inverse-kinematics models respecting contact mechanics [15] could be used to follow the trajectory as well.

Naughton et al. [16] use reinforcement learning to develop the motion plan for a two-segment soft robot for position control amidst obstacles, although their work does not extend to achieving a target pose.

B. Contributions

In this letter, we propose a search-based motion planner for a planar single-segment TDCR to reach a target position and orientation. Our search-based planner discretizes the configuration space, and performs a best-first search while exploiting contacts with the environment. Key to the efficiency of our planner is a novel efficient-to-compute heuristic function. It captures the kinematics of the single segment TDCR and guides the planner to the goal, while exploring only a fraction of the entire search space.

II. PROBLEM DEFINITION

We consider a planar single-segment TDCR and denote its 2D workspace as $\mathcal{W} \in \mathbb{R}^2$, consisting of o known obstacles $\mathcal{W}_{obs} = \{\mathcal{O}_1, \ldots, \mathcal{O}_o\}$. The boundary of the j^{th} obstacle is defined by a closed curve $f_j(\mathbf{x}) = 0$ with points lying outside the obstacle satisfying $f_j(\mathbf{x}) > 0$.

Our robot is actuated via (i) inserting and retracting the base of the robot and (ii) pulling and releasing a single tendon.¹

To this end, a *joint-space value*, $\mathbf{q} = (\ell_{seg}, \ell_{ten})$ describes the total length of the robot that has been inserted into the workspace and the tendon length, respectively. We denote the joint-space, namely the space of all joint-space values, as Q. A *joint-space action* $\delta = (\delta_{seg}, \delta_{ten})$ describes the amount δ_{seg} that the base is inserted or retracted (corresponding to positive and negative values of δ_{seg} , respectively) and the amount δ_{ten} that the tendon is released or pulled (corresponding to positive and negative values of δ_{seg} , δ_{ten}) on a robot with the joint-space action $\delta = (\delta_{seg}, \delta_{ten})$ on a robot with the joint-space value $\mathbf{q} = (\ell_{seg}, \ell_{ten})$ is denoted by $\mathbf{q} + \delta$ and yields the new joint-space value ($\ell_{seg} + \delta_{seg}, \ell_{ten} + \delta_{ten}$). We will use σ to denote a *sequence* of joint-space actions and Σ to denote all such possible sequences.

Applying a joint-space action on a joint-space value is not always guaranteed to result in a feasible solution as the model might return a solution that does not satisfy the problem constraints. We refer to such actions as *invalid actions*, and all actions leading to feasible solutions as *valid actions*. Finally, we denote by $\mathbf{p}(\mathbf{q} + \sigma)$ and $\psi(\mathbf{q} + \sigma)$ the position and orientation of the robot's end-effector w.r.t. to the global frame of reference after applying σ to \mathbf{q} .

Now, the motion planning problem can be stated as follows.

Problem 1: Given a single-segment TDCR with known shape known for an initial joint-space value $\mathbf{q}_{\text{init}} \in Q$, moving amidst a set $\mathcal{O} = \{\mathcal{O}_1, \dots, \mathcal{O}_o\}$ of known obstacles, a tip goal pose, \mathbf{T}_{goal} containing the goal position and orientation $\mathbf{p}_{\text{goal}}, \psi_{\text{goal}}$, a position and orientation tolerance ε, ω , our motion-planning problem calls for computing a sequence of valid joint space actions $\sigma = \{\delta_1, \dots, \delta_k\}$ such that

$$|\mathbf{p}(\mathbf{q}_{\text{init}} + \sigma) - \mathbf{p}_{\text{goal}}||_2 \le \varepsilon, \tag{1}$$

$$|\psi(\mathbf{q}_{\text{init}} + \sigma) - \psi_{\text{goal}}| \le \omega.$$
 (2)

Namely, that the robot's tip reaches the goal pose within the goal tolerance as shown in Fig. 2(a). And that each joint-space action δ_i is a valid action when applied to $\mathbf{q}_{\text{init}} + \delta_1 + \dots + \delta_{i-1}$.

III. KINEMATIC MODEL FOR OBSTACLE INTERACTION

In this section we describe our kinematic model for a planar single-segment TDCR. Our approach adapts the model introduced by Ashwin et al. [17].² However, to reduce computational complexity, instead of employing the proposed four-bar mechanism, we use the so-called *piece-wise constant-curvature assumption* (PCCA). PCCA discretizes the robot into a series of m mutually-tangent constant-curvature arcs, each of curvature κ_i , $\forall i = 1, 2...m$ as shown in Fig. 2(b). These individual curvatures are collectively represented by the vector κ . The *configuration-space value* is represented by $\mathbf{c}(\mathbf{q}) = (\ell_{seg}, \kappa)$, and can be used to reconstruct the robot shape. As we will see, the kinematic

¹The bending in a TDCR is underactuated, where even though there are two antagonistic tendons, and thus two inputs, the robot only has one degree of freedom in bending from tendon actuation. For simplification, we consider the length of one tendon, $\ell_{\rm ten}^1$ as the actuation input. Once the corresponding robot configuration is calculated, the value of the second tendon, $\ell_{\rm ten}^{\prime 2}$ can be determined using geometry. We remove the superscript of the first tendon for convenience.

²This model assumes there is no friction in the system and that the tendons are partially constrained [18].



Fig. 2: (a) Robot shape at \mathbf{q}_{init} and at $\mathbf{q}_{\text{init}} + \sigma$ with respective endeffector positions. (b) Workspace \mathcal{W} with obstacles indicated in blue. The backbone is represented by a sequence of constant-curvature arcs subtending an angle θ_i with curvatures κ_i , where $i = 1, \ldots, m$. The tendons are indicated in violet.

model is treated as an energy-minimization problem, where the equilibrium configurations must (i) satisfy input tendon length constraints and (ii) remain outside the obstacles. In this section we first describe the optimization based kinematic model, and then use it to describe the required backbone parameters.

A. Optimization-based kinematic model

The objective of the kinematic model is to determine the values of κ for a given joint-space value **q**, while ensuring that the robot lies outside the obstacles. Since we consider only one tendon as the actuation input, the first nonlinear equality constraint enforces that the actuated tendon length, ℓ_{ten} should be equal to the calculated tendon length ℓ_{ten}^* . Calculating these tendon lengths for a configuration **c** is described in Sec. III-B.

$$g(\mathbf{c}(\mathbf{q})) = \ell_{\text{ten}}^* - \ell_{\text{ten}}.$$
 (3)

Recall that the boundary of an obstacle j is defined by the function $f_j(\mathbf{x})$. Let angle $\psi_{f_j}(\mathbf{x})$ be the angle made by the tangent to this obstacle at \mathbf{x} . Since we want the robot to lie outside these obstacles, we can formulate the nonlinear constraint as

$$f_j(\mathbf{X}(\mathbf{c}(\mathbf{q}))) \ge 0, \tag{4}$$

where $\mathbf{X}(\mathbf{c}(\mathbf{q}))$ represents a set of points approximating the robot's geometry. The selection of these points is detailed in Sec. III-B. Assuming the material is isotropic, minimizing the strain energy at equilibrium configurations is equivalent to minimizing the individual curvatures of the subsegments [17]. The resulting optimisation problem, using Eq. (3) and (4), is given by

$$\min_{\kappa} \quad \sum_{i=1}^{n} (\kappa_i)^2$$

s.t.
$$g(\mathbf{c}(\mathbf{q})) = 0$$

$$f_j(\mathbf{X}(\mathbf{c}(\mathbf{q}))) \ge 0, \forall j = 1, 2 \dots o.$$
(5)



Fig. 3: (a) Robot's initial configuration at \mathbf{q} . (b), (c) Overlayed robot shapes for joint sequences (b) σ (two length-extensions, and three bending actions) and (c) σ' (reverse order: three bending actions, and two length-extensions).

B. Robot kinematic representation

Recall that we discretize the backbone into a series of *m* circular arcs, also called *subsegments*³. For a robot of length ℓ_{seg} , each sub-segment has a length of ℓ_{seg}/m and curvature of κ_i . Thus, the subtending angle of the *i*'th sub-segment is $\theta_i = \kappa_i \cdot \ell_{seg}/m$. The vector κ contains the curvatures of all these subsegments.

The tip of each subsegment is denoted by \mathbf{p}_i . If we define a local frame of reference attached to p_i with its z-axis tangent to the backbone, then the transformation matrix $\mathbf{T}_{i-1}^i \in SE(2)$ is used to calculate the relative pose between two such consecutive frames. We consider the two tendons placed at a distance of r_d from the backbone and denote their position vectors at the *i*'th sub-segment as $\mathbf{p}_{i,\text{ten}}^1$ and $\mathbf{p}_{i,\text{ten}}^2$ respectively.⁴ Eq. (3) requires calculating the length of the first tendon which is given by

$$\ell_{\text{ten}}(\mathbf{c}(\mathbf{q})) = \sum_{i=1}^{m} ||\mathbf{p}_{i,\text{ten}}^{1} - \mathbf{p}_{i-1,\text{ten}}^{1}||_{2}.$$
 (6)

To define the nonlinear constraints in Eq. (4), we need to approximate the robot shape by a set of discrete points. In our setting, we will use a set of backbone and tendon positions. Specifically for a given c(q), we define the set $X(q, \kappa)$ of 3m points as

$$\mathbf{X}(\mathbf{c}(\mathbf{q})) = \left\{ \begin{bmatrix} \mathbf{p}_i & \mathbf{p}_{i,\text{ten}}^1 & \mathbf{p}_{i,\text{ten}}^2 \end{bmatrix}, \forall i = 1,\dots,m \right\}.$$
(7)

C. Applying a sequence of joint space actions

In the absence of obstacles, there is a one-to-one mapping between a sequence of joint-space actions applied to the robot starting at some initial joint-space value and the final shape of the robot. However, contacts between the robot and the obstacles affect its shape. This implies that given an initial joint-space value $\mathbf{q} = (\ell_{\text{seg}}, \ell_{\text{ten}})$ and two different joint-space sequences σ, σ' such that $\mathbf{q} + \sigma = \mathbf{q} + \sigma'$, may result in different robot shapes due to different contact histories. An example is illustrated in Fig. 3. Therefore, there can be multiple solutions for Eq. (5).

 $^{^3}$ Such discretization is natural as typically, a TCDR is built using a series of m disks used to route the tendons.

⁴See [18] for the exact formulae.

To solve Problem 1, we begin with the initial jointspace value \mathbf{q}_{init} , with known $(\ell_{\text{init}}, \kappa_{\text{init}})$. To compute the resulting shape after applying action δ_1 to \mathbf{q}_{init} , we use κ_{init} to warm-start the optimization in Eq. (5). The obtained curvature values can be used to calculate the endeffector position and orientation, and then used to solve for $(\mathbf{q}_{\text{init}} + \delta_1 + \delta_2)$. The model therefore, needs to be solved k times to obtain the κ for $(\mathbf{q}_{\text{init}} + \delta_1 + \delta_2 \dots + \delta_k)$. Subsequently, the end-effector's position and orientation are determined.

IV. METHOD

To address our motion-planning problem, we take a search-based planning approach [19], [20] where we run a best-first search on a discretized approximation of the continuous joint-space. This section details our implementation of the necessary primitive operations, data structures, and the proposed heuristic for this search.

A. Node representation

Each node *n* in our search algorithm corresponds to a sequence of joint-space actions $\sigma(n) = \{\delta_1, \ldots, \delta_k\}$ and each edge corresponds to one such joint-space action δ_i . Each node is uniquely represented by pair (\mathbf{q}^n, κ^n) where $\mathbf{q}^n = \mathbf{q}_{\text{init}} + \delta_1 + \ldots + \delta_k$ is a joint-space vector and κ^n is a vector corresponding the individual curvatures of the robot's subsegments. The curvatures κ^n resulting from applying $\sigma(n)$ to \mathbf{q}_{init} can be obtained by solving the forward kinematics model defined in Eq. (5). The curvatures along with the segment length from the joint space vector can be used to reconstruct the shape of the robot using Eq. (7).

B. Node extension

Our joint-space actions, $\mathcal{A} := (0, \delta \ell_{\text{ten}}), (\delta \ell_{\text{seg}}, \delta \ell_{\text{ten}}),$ involve either adjusting the robot's tendon by $\delta \ell_{\text{ten}}$ or inserting its base by $\delta \ell_{\text{seg}}$ while simultaneously adjusting the tendon by $\delta \ell_{\text{ten}}$. For a node $n = (\mathbf{q}^n, \kappa^n)$ and a jointspace action $\delta \in \mathcal{A}$, we define the operation of *extending* nby δ as creating a node $n' = (\mathbf{q}^{n'} = \mathbf{q}^n + \delta, \kappa^{n'})$, where $\kappa^{n'}$ is computed using the optimization in Eq. (5) with κ^n as the initial guess. We say that this extension is *valid* if δ is a valid joint-space action when applied to \mathbf{q}^n .

C. Duplicate detection

To avoid expanding similar, yet non-identical states, existing search-based approaches over continuous search spaces (as is our setting) group states into equivalence classes [21]. This is analogous to how standard search algorithms (over finite discrete graphs) reduce search effort by identifying previously-visited states, a process known as *duplicate detection* [22]. In our case, two nodes $n = (\mathbf{q}^n, \kappa^n)$ and n' = $(\mathbf{q}^{n'}, \kappa^{n'})$ are considered duplicates if (i) $\mathbf{q}^n = \mathbf{q}^{n'}$ and (ii) $||\mathbf{p}(\sigma(n))-\mathbf{p}(\sigma(n'))||_2 \leq d_{sim}$, where $d_{sim} > 0$ is some threshold parameter. That is, condition (ii) uses the distance between the two tip positions in order to detect duplicates.

D. Heuristic estimate of nodes

In heuristic-based search, heuristics are used to speed up the search by estimating the cost to reach the goal. In our case this means estimating the length of a path that will allow the end-effector of the TDCR to reach a target pose. In this letter, we propose a heuristic that considers the robot's limited steering properties around the known obstacles while avoiding the need to run the more computationally-complex forward kinematic model.

To achieve this, we suggest a simplified contact model for heuristic computation, which will employ constant-curvature (CC) arcs. Such arcs have been widely used to model loadfree configurations [23]. Specifically, here we model the robot's motion via a sequence of paths having a constant curvature. The robot can only change its curvature along the backbone when in contact with an obstacle. Specifically, we assume that the shape, and consequently the end-effector's trajectory to the goal is a series of mutually tangent CC arcs, with each arc beginning and ending on the surface of an obstacle to reflect the approximated contact mechanics. Therefore, in the absence of any contacts this trajectory would be a single CC arc. While CC curves do not accurately reflect the robot shape as there is curvature variation due to contact forces, our work posits that they could be used to provide a sufficient estimate for the cost-to-go as they reflect the TDCR's limited steering capabilities.

Therefore, our heuristic $h(n) : SE(2) \to \mathbb{R}$ takes as input the position and orientation of the robot's end-effector at a given node n and estimates the length of the path to reach the goal. To avoid calculating h(n) for every prospective node, we precompute h(n) by performing a backward search from the goal. We discretize SE(2) into cells, where each cell stores the length of the shortest path consisting of CC arcs from the goal to it. Using the precomputed values associated with each grid cell, the heuristic value for each node can be calculated during the search, as detailed in IV-D.4.

1) Overview: In order to precompute the cost of paths of CC arcs from each cell to the goal, we perform a backward search from the goal. We first find the set of all CC arcs whose end point orientation corresponds to the goal pose (Fig. 4(a)). From this set, we identify cells that could be potential contact points (Fig. 4(b)). From these contact cells, we perform the same search as before for CC arcs, but use the pose of these contact cells to populate consequent CC arcs (Fig. 4(c)). The process is repeated for all possible contact cells. Finally, we have a set of CC arcs from each cells that can reach the goal (Fig. 4(d)).

There are two main components to precomputing the heuristic: (1) PopulateCCArcs (pose): Determining the set of all cells that can reach a given target pose with a *valid* CC arc (to be defined shortly), (2) IdenitfyContact (cell): Identifies whether an input cell is a possible contact cell. This function is evaluated for each cell found in the above set. For the case where there are no obstacles, the heuristic computation only involves running step (1) with the goal pose as input.



Fig. 4: Diagrammatic representation of the heuristic calculation. (a) PopulateCCArcs ($\mathbf{p}_{goal}, \psi_{goal}$) evaluates cell poses that contain CC arcs (red) to reach a goal pose. Note that from each cell pose, only one arc reaches the goal. (b) Having obstacles (blue) invalidates some of the CC arcs. Running IdenitfyContact(cell) for each cell pose identified in the previous step identifies contact cells (green circles). (c) Using one of the contact cells, cell c, PopulateCCArcs(\mathbf{p}_c, ψ_c) is run, identifying new CC arcs that reach its pose (green) (d) After the above steps, we can see (i) positions from which arcs with several orientations were computed (e.g., cell b and cell a) and (ii) For the illustrated robot, the proposed heuristic informs the planner that based on its current end-effector orientation it can reach the goal via the CC arc connecting cell a and c, and consequently c to the goal.

2) Implementation: As mentioned, our heuristic will rely on a discretization of SE(2) into cells. Let this grid be denoted by C, with each cell $c \in C$ of size $\Delta \times \Delta \times \alpha$ where $\Delta \in \mathbb{R}$ and $\alpha = 2\pi/s$ for some $s \in \mathbb{N}^+$. Thus, we introduce mappings between cells and the pose they represent. Specifically, let CELL be a mapping taking a pose $(\mathbf{p}, \psi) \in SE(2)$ and returning the cell containing this pose. Similarly, let POSITION, ORIENTATION and POSE be mappings taking a cell $c \in C$ and returning the position, orientation and pose at the center of c, respectively. The heuristic value of every cell is initially set to ∞ .

(1)—PopulateCCArcs (pose): For any target pose $(\mathbf{p}, \psi) \in SE(2)$, there's a unique CC arc starting from $\mathbf{u} \in \mathbb{R}^2$ and reaching this target pose. Let $a = CC-ARC_{(\mathbf{p},\psi)}(\mathbf{u})$ be this arc. Its curvature, and length are denoted by $\kappa(a)$ and $\ell(a)$, respectively. Similarly, its orientation at \mathbf{u} w.r.t. the global x-axis is denoted by $\psi(a)$. We say that such an arc is *valid* for some curvature bound κ_{\max} if (i) $\kappa(a) < \kappa_{\max}$, (ii) $\theta(a) < \theta_{\max}$, and (iii) a does not intersect any obstacle.

Given a position $\mathbf{u} \in \mathbb{R}^2$, let $C_{\mathbf{u}}$ be the set of all cells that have the same position, but different orientations. Namely,

$$C_{\mathbf{u}} := \{ c \mid \exists \psi \in \mathbf{SO}(2) \text{ s.t. } \mathbf{CELL}(\mathbf{u}, \psi) = c \}.$$

Following the discussion above, given some target pose (\mathbf{p}, ψ) , if we compute a *valid* arc *a*, only one cell c_a in $C_{\mathbf{q}}$ contains this arc's orientation $\psi(a)$, i.e. $c_a =$ CELL $(\mathbf{u}, \psi(a))$. We set the heuristic value h_a of c_a to be the arc's length, i.e. $h_a := \ell(a)$. We use a breadth-first search (outlined in Alg. 1) to identify the set, $S_{(\mathbf{p},\psi)}$, of all cells with *valid* arcs reaching (\mathbf{p}, ψ) . This entire process is equivalent to identifying the workspace of a single CC segment as shown in Fig. 4(a).

(2)—IdenitfyContact(cell):

We say that a cell $c' \neq c$ is a *position neighbor* of c if $|\text{POSITION}(c)| = |\text{POSITION}(c')| \leq \Lambda$ and

$$|POSITION(c).x - POSITION(c).x| \le \Delta$$
 and $|POSITION(c).y - POSITION(c').y| \le \Delta$.

A cell c is said to be a *contact cell* if (i) there exists a position neighbor c' of c whose position POSITION(c') lies within some obstacle j, i.e., $\exists O_j \in W_{obs}$ s.t. $O_j \cap c \neq \emptyset$

Algorithm 1 Populating the CC arcs for a given pose

Input: Target pose (\mathbf{p}, ψ) Output: $S_{\mathbf{p},\psi}$ 1: **function** HEURISTICARC(*a*) 2: if $\kappa(a) > \kappa_{\max}$ or 3: $\theta(a) > \theta_{\max}$ or 4: InCollision(a) then return ∞ 5: 6: return $\ell(a)$ 7: function POPULATECCARCS(\mathbf{p}, ψ) OPEN \leftarrow (**p**) 8: 9: $c \leftarrow \texttt{CELL}(\mathbf{p}, \psi)$ $S_{(\mathbf{p},\psi)}$.insert(c)10: while not OPEN.empty() do 11: $\mathbf{u} \leftarrow OPEN.extract()$ 12: $a \leftarrow \text{CC-ARC}_{(\mathbf{p},\psi)}(\mathbf{u})$ 13: 14: $c_a \leftarrow \text{CELL}(\mathbf{u}, \psi(a))$ 15: $h_a \leftarrow \text{HEURISTICARC}(a)$ 16: if $h_a < \infty$ then $h(c_a) \leftarrow \min\{h(c_a), h_a + h(c)\}$ 17: $S_{(\mathbf{p},\psi)}$.insert (c_a) 18: for c' in PositionNeighbor(c_a) do 19: **OPEN.insert**(POSITION(c')) 20: 21: return $S_{(\mathbf{p},\psi)}$

and if (ii) the orientation of the cell differs from the local tangent of \mathcal{O}_j by at most ω_{contact} i.e., $|\text{ORIENTATION}(c) - \psi_{f_j}(\text{POSITION}(c'))| < \omega_{\text{contact}}$. Here, $\omega_{\text{contact}} > 0$ is some user-provided threshold.

3) Algorithm: To recall, we start by settling the initial heuristic value for all cells to ∞ . We start by calculating the set of all cells $S_{(\mathbf{p}_{goal},\psi_{goal})}$ that have valid CC arcs directed towards the desired goal pose $(\mathbf{p}_{goal},\psi_{goal})$. We do so by calling PopulateCCArcs $(\mathbf{p}_{goal},\psi_{goal})$ (detailed in Alg. 1) that updates the heuristic value for each cell to be the length of the calculated arc.

We then repeat the following process (summarized in

Algorithm 2 Algorithm for computing the heuristic

Inp	ut: Target pose $(\mathbf{p}_{\text{goal}}, \psi_{\text{goal}})$
Out	tput: $h(c) \ \forall c \in C$
1:	function COMPUTEHEURISTIC($\mathbf{p}_{\text{goal}}, \psi_{\text{goal}}$)
2:	$\textbf{OPEN} \leftarrow (\mathbf{p}_{\text{goal}}, \psi_{\text{goal}})$
3:	$h(c) \leftarrow \infty \ \forall c \in \mathbf{C}$
4:	$h(\text{CELL}((\mathbf{p}_{\text{goal}}, \psi_{goal})) \leftarrow 0$
5:	while not OPEN.empty() do
6:	$(\mathbf{p}, \psi) \leftarrow \text{OPEN.extract}()$
7:	$S_{(\mathbf{p},\psi)} \leftarrow \text{PopulateCCARCS}(\mathbf{p},\psi)$
8:	for c' in $S_{(\mathbf{p},\psi)}$ do
9:	if IdentifyContactCell(c') then
10:	OPEN.insert ($POSE(c')$)
11:	for c in C do
12:	$h(c) = \min\{h(c), h(c^+), h(c^-)\}$
13:	return h

Alg. 2): For each c in the priority queue OPEN (extracted in a first-in first-out manner) such that c is a contact cell and $h(c) < \infty$, we repeat the previous step by calling PopulateCCArcs(POSE(c)) to update the heuristic value of each cell that can reach c. If c_a is one such cell that can reach c with a valid arc a, we set its heuristic value as $h(c_a) = \min\{h(c_a), h_a + h(c)\}$

4) Returning the heuristic value for a node: Once the heuristic has been precomputed for grid cells using Alg. 2, it can be used to return the heuristic value for a given node. Given a node n, we compute the end effector (\mathbf{p}_n, ψ_n) of the robot at n. We then compute the cell c_n where this pose lies (i.e., $c_n = CELL((\mathbf{p}_n, \psi_n))$). As described earlier, the CC assumption is only an approximation for the contact mechanics and does not precisely reflect contact mechanics. Thus, we introduce a final postprocessing smoothing step to account for orientation errors. Specifically, to set the heuristic value for c_n we also consider the values of it's neighboring cells c_n^+ and c_n^- which share the same position as c but differ in the orientation by a value of $\pm \alpha$. Namely,

$$c_n^+ := \text{CELL}(\mathbf{p}_n, \psi_n + \alpha)$$
 and
 $c_n^- := \text{CELL}(\mathbf{p}_n, \psi_n - \alpha).$

Our final heuristic value for the node n will be

$$h(n) = \min\{h(c_n), h(c_n^+), h(c_n^-)\}.$$
(8)

E. Putting it all together—Search algorithm

In this work, we implement a greedy best-first search algorithm [24]. Specifically, we make use of a priority queue OPEN that orders search nodes (Sec. IV-A) according to the the node's heuristic value (Sec. IV-D.4).

The algorithm starts by initializing OPEN with a node corresponding to the initial configuration q_{init} and runs in iterations. Each iteration, the highest-priority node n is extracted from OPEN, and is *extended* (Sec. IV-B) by applying the set of actions A to its joint space values q^n . For each successor node n', we compute its corresponding joint

space values using our forward kinematic model (Sec. III). If n' is valid and not a duplicate (Sec. IV-C), it is added to OPEN. Finally, n is added to a so-called CLOSED set to avoid re-expanding previously expanded nodes.

This process continues until a node that satisfies the endeffector positional constraints specified in Problem 1, at which point the search is concluded successfully or until a maximal number of search iterations N_{max} is reached at which point the search is classified as unsuccessful.

V. EMPIRICAL EVALUATION

We evaluate our contact-based motion planner \mathcal{M}_{CAN} with our heuristic in simulation. Some sample motion plans and their implementation on a motorized prototype [25] are in the supplementary video.⁵

1) Kinematic model & planner parameters: The optimization problem in Eq. (5) is solved using an interior-point algorithm via MATLAB's *fmincon*. We consider a robot of radius 6 mm with a maximum length of 250 mm, represented by m = 30 arcs. The planner parameters are listed in Table I.

TABLE I: Parameters for the heuristic calculation and motion planner.

Heu	iristic	Motion Planner		
Parameter	Value	Parameter	Value	
Δ	$0.001\mathrm{m}$	ϵ	$0.01\mathrm{m}$	
α	45°	ω	15°	
$\kappa_{\rm max}$	$250 \ m^{-1}$	$\delta \ell_{ m seg}$	$0.001\mathrm{m}$	
$\theta_{\rm max}$	$270^{\circ} m^{-1}$	$\delta \ell_{ m ten}$	$0.001\mathrm{m}$	
$\omega_{\mathrm{contact}}$	2.815°	$N_{\rm max}$	7000	

2) Environments: We consider three workspaces, W_1, W_2 , and W_3 (see Fig. 5). W_1 has five equally sized convex circular obstacles while W_2 features six circular obstacles with random centers and radii that overlap, creating nonconvex shapes. Finally, W_3 resembles a 2D cross-section of the interior of a turbine engine. CAN's potential use in such cluttered environments guides our future work (refer to Sec. VI). Specifically, the non-convex turbine blade shapes are approximated by four circular obstacles each.

Recall that we compute a heuristic for the robot's end effector (Sec. IV-D) which can be modelled as a line segment translating and rotating in the plane. Thus, to ensure that we only consider collision-free end effector placements, heuristic computation is done after inflating obstacles by an amount equal to the robot's width [26].

3) Queries: We generated for each workspace a set of valid queries as follows: We start by setting the initial configuration to $\mathbf{q}_{\text{init}} = (1, 1)mm$. We then run a breadth-first search (BFS) by varying the tendon and segment length by 1 mm. This creates a set of valid configurations with corresponding end-effector poses for which a solution to Problem 1 is guaranteed to exist. For each workspace, we sample 175 such end-effector poses at random.

4) Baselines.: We compare our planner \mathcal{M}_{CAN} with two baselines. The first baseline, which we denote by $\mathcal{M}_{contactless}$, is a planner that avoids contacts with the obstacles. Specifically, in the kinematic model, contactless configurations

⁵Can also be found at https://youtu.be/da3eYGwzxts

have constant curvatures. Thus, $\mathcal{M}_{contactless}$ performs inverse kinematics [23] for points satisfying the position tolerance and checks if the obtained solution satisfies the angular tolerance as well and does not intersect with obstacles.

The second baseline, which we denote by $\mathcal{M}_{\text{CAN-simple}_h}$ is identical to our planner but uses a simple commonly-used heuristic function $h_{\text{simple}} := h_{xyz} + w \cdot h_{\theta}$ [27]. Here h_{xyz} represents the least-cost path from the tip to the goal, computed via Dijkstra's algorithm, h_{θ} is the absolute angular difference between the end-effector and the target orientation, and w is a constant used to balance the two, set to be 0.01.

5) Success rate for different planners: In Table II, we present each planner's success rate in different workspaces. Both baselines show significantly lower success rates, underscoring the need for CAN and the proposed heuristic. Failures of \mathcal{M}_{CAN} could be attributed to two factors: The first is that the heuristic's inexact approximation of contact mechanics may not always guide the robot successfully. The second is that, in many cases, if a particular path does not result for a given sequence of contacts, the planner tends to explore nearby nodes, without exploring paths with alternative contacts. A potential remedy is to discern similar configurations through soft duplicate detection [28], [29].

TABLE II: Success rate [%] of finding a solution for the different planning approaches on the three workspaces.

	\mathcal{W}_1	\mathcal{W}_2	\mathcal{W}_3
\mathcal{M}_{CAN} (proposed planner)	80.57	79.43	78.86
$\mathcal{M}_{\text{contactless}}$ (baseline 1)	16.57	25.14	26.86
$\mathcal{M}_{\text{CAN-simple}_h}$ (baseline 2)	29.71	19.42	28.57

6) Comparing the planner with BFS.: We compared our planner \mathcal{M}_{CAN} with the BFS planner used to generate queries, focusing on the number of contacts and the average nodes expanded (see Fig. 5). The heatmap analysis indicates that most solutions share a similar number of contacts, with a tendency for \mathcal{M}_{CAN} to identify solutions with fewer contacts if possible. Notably, for \mathcal{W}_2 with five contacts, \mathcal{M}_{CAN} finds a simpler solution, reflected in a lower value in the barplot. The barplot shows that the average number of nodes expanded by both planners increase with number of contacts, with BFS expanding more than 10 times the nodes expanded by \mathcal{M}_{CAN} .

VI. CONCLUSION & FUTURE WORK

In this letter, we present a motion planner coupled with a novel heuristic that computes plans to reach a desired pose. Achieving a success rate of roughly 80% over 525 target poses amidst convex and non-convex obstacles. This efficiency stems from the proposed heuristic that enables the planner to examine just a fraction of the entire search space.

We consider several directions for future work which we briefly elaborate on:

1) Providing guarantees on the quality of the solution.: The heuristic we propose in Section IV-D is not guaranteed to be admissible, meaning it does not provide a lower bound on the cost-to-go. Consequently, there is not guarantee on the quality of the obtained solution by our planner. A potential solution to this limitation is employing multi-heuristic A*



Fig. 5: Comparison of our motion planner \mathcal{M}_{CAN} with the BFS planner used to generate the queries across three workspaces. For each plot we report (Top right) a heatmap showing the number of successful solutions for both BFS (y-axis) and \mathcal{M}_{CAN} (x-axis), with numerical counts displayed within each cell. (Bottom right) N_{xp} , the average nodes expanded (×1e3 for \mathcal{M}_{CAN} (purple) and BFS (grey)) with increasing number of contacts. Please note that the y-axis scale differs between the two planners.

algorithms, which effectively incorporates both admissible and inadmissible heuristics in a structured manner [30], [31].

2) Extending the planner to 3D environments.: To extend \mathcal{M}_{CAN} for use in 3D environments, both the model and the heuristic require modifications. The design of the proposed motion planner is model-agnostic, allowing the potential use of any model in literature. The heuristic would require voxelizing the spaces \mathbb{R}^3 and SO(3). Additionally, a challenge would be to represent the potential twists in the robot's structure due to contact interactions. To address this, the heuristic could be computed using arcs of uniform curvature along the x and y-axes, combined with a constant rate of twist around the z-axis.

An additional challenge is the computational effort: The current 2D kinematic model accounts for 99.3% of the computational effort, and shifting to a 3D model could increase this due to the complexity of TDCR modeling. Using a simplified approach, such as Euler curves [32], to model the backbone under contact forces might offer

a feasible alternative. Another alternative may be to use multiple resolutions during the search [20], [33].

3) Using the planner for inspection of turbine engines.: While 21 DOF hyperredundant robots have been investigated for turbine blade inspections [10], TDCRs with CAN offer the advantage of performing the same task with fewer DOFs. TDCRs mitigate the risk of blade damage due to their compliance. With our proposed motion planner targeting specific poses, future work will evolve into search-based inspection planning [34], focusing on surface inspection by orienting the robot appropriately.

ACKNOWLEDGEMENTS

The authors would like to thank Quentin Peyron for discussions aiding in the conceptualisation of this problem statement, Puspita Triana Dewi and Chloe Pogue for helping with prototyping and experiments.

REFERENCES

- M. Russo, S. M. H. Sadati, X. Dong, A. Mohammad, I. D. Walker, C. Bergeles, K. Xu, and D. A. Axinte, "Continuum robots: An overview," *Advanced Intelligent Systems*, vol. 5, no. 5, p. 2200367, 2023.
- [2] A. Gao, H. Liu, Y. Zhou, Z. Yang, Z. Wang, and H. Li, "A cross-helical tendons actuated dexterous continuum manipulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 2012–2017.
- [3] Y.-J. Kim, S. Cheng, S. Kim, and K. Iagnemma, "Design of a tubular snake-like manipulator with stiffening capability by layer jamming," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 4251–4256.
- [4] C. Pogue, P. Rao, Q. Peyron, J. Kim, J. Burgner-Kahrs, and E. Diller, "Multiple curvatures in a tendon-driven continuum robot using a novel magnetic locking mechanism," in *IEEE/RSJ International Conference* on Intelligent Robots and Systems, 2022, pp. 472–479.
- [5] M. B. Wooten and I. D. Walker, "Environmental interaction with continuum robots exploiting impact," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10136–10143, 2022.
- [6] M. Selvaggio, L. A. Ramirez, N. D. Naclerio, B. Siciliano, and E. W. Hawkes, "An obstacle-interaction planning method for navigation of actuated vine robots," in *Proceedings - IEEE International Conference* on Robotics and Automation, 2020, pp. 3227–3233.
- [7] J. D. Greer, L. H. Blumenschein, R. Alterovitz, E. W. Hawkes, and A. M. Okamura, "Robust navigation of a soft growing robot by exploiting contact with the environment," *The International Journal* of Robotics Research, vol. 39, no. 14, pp. 1724–1738, 2020.
- [8] F. Fuentes and L. H. Blumenschein, "Mapping unknown environments through passive deformation of soft, growing robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2023, pp. 2522–2527.
- [9] L. Pfotzer, S. Klemm, A. Roennau, J. Zöllner, and R. Dillmann, "Autonomous navigation for reconfigurable snake-like robots in challenging, unknown environments," *Robotics and Autonomous Systems*, vol. 89, pp. 123–135, 2017.
- [10] M. S. Saleem, R. Sood, S. Onodera, R. Arora, H. Kanazawa, and M. Likhachev, "Search-based Path Planning for a High Dimensional Manipulator in Cluttered Environments Using Optimization-based Primitives," *IEEE International Conference on Intelligent Robots and Systems*, pp. 8301–8308, 2021.
- [11] A. Singh, Anshul, C. Gong, and H. Choset, "Modelling and path planning of snake robot in cluttered environment," in *International Conference on Reconfigurable Mechanisms and Robots (ReMAR)*, 2018, pp. 1–6.
- [12] X. Wang, M. Bilsky, and S. Bhattacharya, "Search-based configuration planning and motion control algorithms for a snake-like robot performing load-intensive operations," *Autonomous Robots*, vol. 45, no. 8, pp. 1047–1076, 2021.
- [13] M. C. Yip and D. B. Camarillo, "Model-less feedback control of continuum manipulators in constrained environments," *IEEE Transactions* on *Robotics*, vol. 30, no. 4, pp. 880–889, 2014.

- [14] Z. Zhang, J. Dequidt, J. Back, H. Liu, and C. Duriez, "Motion Control of Cable-Driven Continuum Catheter Robot Through Contacts," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1852–1859, 2019.
- [15] E. Coevoet, A. Escande, and C. Duriez, "Optimization-based inverse model of soft robots with contact handling," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1413–1419, 2017.
- [16] N. Naughton, J. Sun, A. Tekinalp, T. Parthasarathy, G. Chowdhary, and M. Gazzola, "Elastica: A compliant mechanics environment for soft robotic control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3389–3396, 2021.
- [17] K. P. Ashwin, S. Kanti, and A. Ghosal, "Profile and contact force estimation of cable-driven continuum robots in presence of obstacles," *Mechanism and Machine Theory*, vol. 164, p. 104404, 2021. [Online]. Available: https://doi.org/10.1016/j.mechmachtheory.2021.104404
- [18] P. Rao, Q. Peyron, S. Lilge, and J. Burgner-Kahrs, "How to Model Tendon-Driven Continuum Robots and Benchmark Modelling Performance," *Frontiers in Robotics and AI*, vol. 7, p. 223, 2021.
- [19] B. J. Cohen, M. Phillips, and M. Likhachev, "Planning single-arm manipulations with n-arm robots," in *Robotics: Science and Systems*, 2014.
- [20] M. Fu, O. Salzman, and R. Alterovitz, "Toward certifiable motion planning for medical steerable needles," in *Robotics: Science and Systems*, 2021.
- [21] J. Barraquand and J.-C. Latombe, "Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles," *Algorithmica*, pp. 122–155, 1993.
- [22] P. A. Dow and R. E. Korf, "Duplicate avoidance in depth-first search with applications to treewidth." in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2009, pp. 480–485.
- [23] R. J. Webster and B. A. Jones, "Design and kinematic modeling of constant curvature continuum robots: A review," *International Journal* of Robotics Research, vol. 29, no. 13, pp. 1661–1683, 2010.
- [24] S. J. Russell and P. Norvig, *Artificial intelligence a modern approach*. London, 2010.
- [25] R. M. Grassmann, C. Shentu, T. Hamoda, P. T. Dewi, and J. Burgner-Kahrs, "Open continuum robotics-one actuation module to create them all," arXiv preprint arXiv:2304.11850, 2023.
- [26] D. Halperin, O. Salzman, and M. Sharir, "Algorithmic motion planning," in *Handbook of Discrete and Computational Geometry*, 2017, pp. 1311–1342.
- [27] B. J. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for manipulation with motion primitives," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 2902–2908.
- [28] W. Du, S. Kim, O. Salzman, and M. Likhachev, "Escaping local minima in search-based planning using soft duplicate detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2019, pp. 2365–2371.
- [29] N. Maray, A. Vemula, and M. Likhachev, "Improved soft duplicate detection in search-based motion planning," in *International Conference* on Robotics and Automation, 2022, pp. 5792–5798.
- [30] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic A*," *Int. J. Robotics Res.*, vol. 35, no. 1-3, pp. 224– 243, 2016.
- [31] F. Islam, O. Salzman, and M. Likhachev, "Online, interactive user guidance for high-dimensional, constrained motion planning," in *International Joint Conference on Artificial Intelligence, IJCAI*, 2018, pp. 4921–4928.
- [32] P. Rao, Q. Peyron, and J. Burgner-Kahrs, "Shape representation and modeling of tendon-driven continuum robots using euler arc splines," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8114–8121, 2022.
- [33] M. Fu, K. Solovey, O. Salzman, and R. Alterovitz, "Resolution-optimal motion planning for steerable needles," in *International Conference on Robotics and Automation, ICRA*, 2022, pp. 9652–9659.
- [34] M. Fu, A. Kuntz, O. Salzman, and R. Alterovitz, "Asymptotically optimal inspection planning via efficient near-optimal search on sampled roadmaps," *Int. J. Robotics Res.*, vol. 42, no. 4-5, pp. 150–175, 2023.