Anticipatory Path Planning for Continuum Arms in Dynamic Environments

Brandon H. Meng, Dimuthu D.K. Arachchige, Jiahao Deng, Isuru S. Godage, and Iyad Kanj

***Abstract*— Continuum arms are more adaptable to their environments and inherently human-friendly compared to their rigid counterparts. Path planning of continuum arms is an active research area with many challenges. The hyper-redundancy of continuum arms, which renders them highly versatile, is their curse in path planning. This problem becomes even more challenging in dynamic environments in the presence of mobile obstacles. In this paper, we propose an anticipatory path planning approach for continuum arms in dynamic environments. Our approach is based on obstacle prediction coupled with temporal graphs to model the dynamic environment. We evaluate the proposed approach's performance and compare it to prevailing path planning approaches for continuum arms in dynamic environments.**

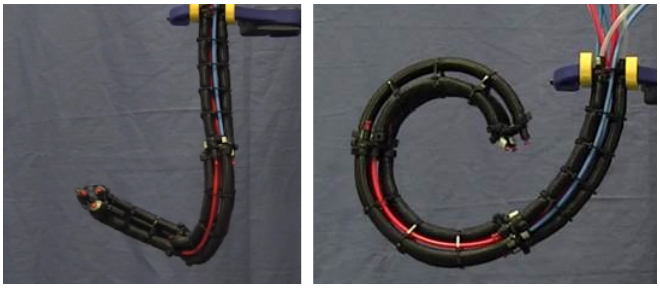

Fig. 1. Two different postures of a continuum-arm robot

I. Introduction

Continuum arms are inspired by the anatomies of animal appendages such as elephant trunks [1], tongues [2], and tentacles [3]. In contrast to rigid manipulators that use actuated joints, the continuum arm's movement is generated by structural deformation. As such, continuum arms can achieve postures that are impossible for rigid robots to achieve. Fig. 1 shows several postures of a continuum arm [4] while traversing a trajectory. Continuum arms can be used across various domains; larger macro-scale continuum arms can be applied in heavy-duty tasks such as car painting [5], nuclear reactor repair [6] and search and rescue [7]. Smaller continuum arms are used in areas that require more precision, such as minimally invasive surgeries [8]. Due to the inherent safety of continuum arms, they are ideal for manipulation tasks in dynamic environments with humans.

There has been considerable work on path planning for rigid mobile robots in dynamic environments. One such planning method is the potential field approach [9]–[11]. The work reported in [12], [13] uses the potential field approach for continuum arm path planning in dynamic environments. These approaches were implemented for restricted settings (for 1 or 2 obstacles in [13], and partially dynamic environments in [12]). As we demonstrate in this current work, given its greedy nature, the potential field approach fails in environments with many dynamic obstacles.

Other popular approaches are based on *anticipatory path planning*. Anticipatory path planning mainly involves two components: (1) predicting the state of the environment for the near future, and (2) performing path planning with respect to the predicted environment [14]. Based on this general approach, different combinations of prediction and planning algorithms have been proposed. The combination of a Gaussian process as the obstacle prediction algorithm with the use of sampling-based algorithms, such as Rapidly-exploring Random Tree (RRT) and Probabilistic Roadmap (PRM), has been widely used, in particular in human environments [15], [16]. Machine learning has also been used for human environment prediction [17], [18]. Other planning algorithms, such as A^* [19], genetic algorithms [20], and variants of the D^* algorithm [21]–[23], have also been used in anticipatory path planning. Even though sampling-based approaches (such as RRT and PRM) are used widely in robotics, few such approaches have been attempted for continuum arms. The relation between the Work Space ($\mathcal{W}$-Space) and the Configuration Space ($\mathcal{C}$-Space) for continuum arms is complex, and no closed-form inverse kinematic (IK) solutions exist. Consequently, sampling techniques do not translate well for continuum arms.

In general, reliable path planning for continuum arms must consider both previous and future positions of the arm to avoid making greedy choices that may drive the arm into knotting/contorted positions from which it can no longer make any progress. As was demonstrated in previous research on path planning of continuum arms in environments with stationary obstacles [24], path planning approaches based on Inverse Kinematics behave very poorly and have a low success rate.

In this paper, we propose an anticipatory path planning scheme for continuum arms in dynamic environments. The scheme combines a prediction process with a *temporal graph* approach and works as follows. We first predict the future positions of the obstacles during the next time window. This prediction is based on the history positions of the obstacles during the past time windows and is performed using a cubic

The authors are affiliated with the School of Computing, DePaul University, Chicago, IL 60604, USA. {bmeng1, darachch, jdeng5, igodage, ikanj}@depaul.edu.

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

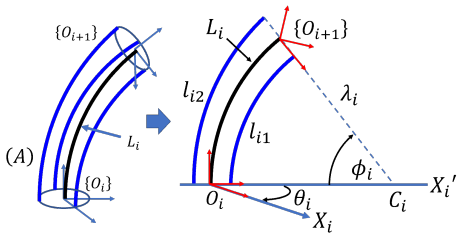

Fig. 2. (Left) diagram showing the layout of the actuator and the backbone; and (Right) the schematic of the i^{th} section when looking from an angle normal to the bending plane.

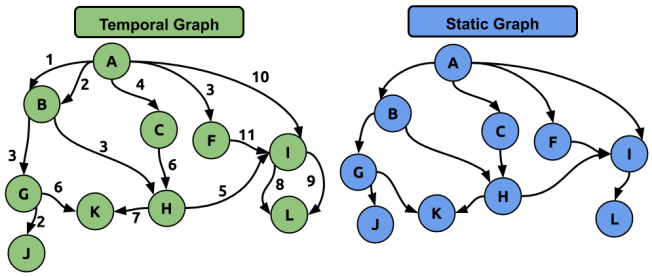

Fig. 3. An illustration of a temporal graph with $\mu = 1$ and $d = 0$ and its corresponding static graph [27].

interpolation [25]. We define a temporal graph whose edges indicate their availability during a short time window in the future, computed based on our prediction. We then compute a local shortest path in the temporal graph that the continuum arm can follow during the predicted time window. The above process is then repeated until the destination point is reached.

We implemented the proposed anticipatory path planning algorithm and compared it to conventional path planning approaches. We evaluated the performance of all algorithms by simulating them on a large number of instances and report our findings. For all considered scenarios, the proposed temporal-graph approach significantly outperforms the other approaches.

II. Preliminaries

A. System Model

The continuum arm prototype considered in this work is detailed in [24]. The arm consists of three inextensible continuum sections each with independent omnidirectional bending capability. Although the system can be extended to any number of sections, in our model, we use three sections after the prototype 3-section continuum arm available in our lab. The kinematic model used in this paper is the one proposed in [24]. Configurations are 6-tuples consisting of actuator length changes for each section. Given the inextensible backbone, each section can be defined by 2 length changes and deforms in a circular arc parameterized by the radius of the circular arc, λ_i, subtended angle, ϕ_i, and bending plane angle with respect to the +X axis, θ_i (Fig. 2). See [24], [26] for more details.

B. The Cubes-Graph

The cubes-graph is a critical piece of the approach described in [24] and will also be used extensively here. It allows us to discretize the $\mathcal{W}$-Space in a convenient way. To construct the cubes-graph, we discretize the $\mathcal{W}$-Space into cubes, each of dimension equal to 1cm, treat the center of each cube as a point/vertex, and connect each vertex corresponding to a cube q to the at most 26 vertices corresponding to the neighboring cubes of q. We assign the weight of an edge to be the Euclidean distance between its endpoints. We denote the cubes-graph as G_Q.

C. Temporal Graphs

To model a dynamic environment with moving obstacles, we use a temporal graph; this graph is capable of modeling which parts of the environment are accessible/available (i.e., devoid of obstacles) during certain times, which is crucial for path planning in such environments. As their name suggests, temporal graphs are graphs that have edges open only during certain time windows and closed otherwise [28]. More formally, in the model we use [27], a temporal graph is defined as a directed weighted multigraph $G=(V,E)$, where V represents the set of vertices of G and E represents the set of edges. Each edge $e \in E$ is a tuple (u,v,t,μ,d), where $u,v \in V$ are the start and end vertices of this edge, respectively; t is the start of the time window associated with the edge (in other words, this edge is open at time t); μ is the time taken to traverse the edge; and d the cost associated with traversing the edge.

Figure 3 shows an example of a temporal graph (left figure) with A as the source vertex, compared to its static counterpart graph (the right figure). In the figure, we assumed that $\mu = 1$ and that the cost of each edge is zero. The values on each edge stands for the start of the time window associated with that edge. Note that, for example, the path (A, I, L) is valid for the static graph but not for the corresponding temporal graph.

D. Smoothness Measure

Smoothness is a key measure of the quality of a path for continuum robots. In [24], the authors defined the shape curvature parameters θ, ϕ and λ. The orientation of the robot's arm can be represented as a vector whose coordinates/components are these three parameters. Since λ can be inferred from ϕ in our robot's model, we only need to use θ and ϕ to represent this orientation. For a given path, its smoothness can be measured as the cumulative change in orientations over the pairs of consecutive configurations on the path. This change is quantified using the Euclidean distance between the orientations of the pairs of consecutive configurations on the path. More specifically, consider a configuration C^m on the path with curvature parameters θ_i^m, ϕ_i^m, where $i \in \{1, 2, 3\}$ refers to the robot's arm section number. Consider the configuration C^{m+1} that follows C^m on the path and whose curvature parameters are θ^{m+1}, ϕ^{m+1}. The smoothness, Δ^m, between the two configurations C^m

and C^{m+1} is calculated as

$$\Delta^m = \sum_{i=1}^{3} \sqrt{(\theta_i^{m+1} - \theta_i^m)^2 + (\phi_i^{m+1} - \phi_i^m)^2}.$$

For a configurations path $P = (C^1, \ldots, C^k)$, the smoothness of P is calculated as:

$$S = \sum_{m=1}^{k-1} \Delta^m.$$

Our goal becomes to compute an obstacle-avoiding path P in the $\mathcal{C}$-Space that minimizes this cumulative value S. See [24] for more details about the curvature parameters.

E. Cubes Path and the Layered Graph

In a static environment, the cubes-graph was employed in [24] to create a *cube path* as follows. Following the construction of a cubes-graph, paths can be created between two desired cubes. For some starting point s and ending point t, two cubes q_s and q_t are found such that s and t, respectively, fall within these cubes. Using the cubes-graph G_Q, a path P is found from q_s to q_t using a shortest path algorithm, such as Dijkstra's algorithm. First, the $\mathcal{C}$-Space is enumerated (with respect to some level of discretization), and then each of the enumerated configurations is mapped to its corresponding $\mathcal{W}$-Space point. In order to find a smooth path of configurations that takes the arm tip from point s to t, a layered graph of configurations is constructed. The vertices in each layer consist of all the enumerated configurations whose $\mathcal{W}$-Space points fall within the same cube of P. Any configurations whose corresponding $\mathcal{W}$-Space points intersect with obstacles are purged. Then, each vertex/configuration in a layer is connected via a directed edge to all of the configurations in the next layer. The weight of an edge between two consecutive configurations C^m and C^{m+1} is defined using the smoothness metric Δ^m described in the previous section. Finally, the *Bellman-Ford Algorithm* [29] is used to compute a smooth path in the layered graph.

III. Anticipatory Path Planning Algorithm

In this section, we develop a temporal path planning scheme for dynamic environments. The scheme utilizes obstacle prediction and temporal graphs to accomplish path planning in a dynamic environment. We give a high-level description of this scheme before proceeding to the details.

Suppose that the arm tip is at some current cube q_c (initially the starting cube q_s). First, we predict the motion of the obstacles for h steps in the future based on the historical data of the obstacles' motion that we have collected. Second, we construct a (local) temporal cubes-graph, G_T, centered around q_c, that extends h hops in all directions by employing both the cubes-graph (described in Section II-B) and our prediction of the obstacles' motion. The temporal cubes-graph G_T allows us to model the availability/unavailability of certain areas in the $\mathcal{W}$-Space during certain time windows. Finally, we compute a subpath (of cubes) in this temporal cubes-graph from q_c to the cube q_b that is the closest cube in G_T to the target cube q_t. The above process is then repeated starting at q_b until q_t is reached.

A. Obstacle Prediction

Obstacle prediction is performed using cubic interpolation [30]. This is a widely used methodology for making predictions based on a small set of data [31]. For some given amount of input data, cubic interpolation constructs a polynomial of degree 3, with time as the variable, that includes all of the input data. Using this third degree polynomial, we can interpolate future times to obtain a data prediction.

As the obstacles move, their r most recent locations are stored as *historical obstacle data* in a fixed-length list/queue, denoted O. Though r can be as large as desired, in general r should at least be 4 as 4 data points are needed to determine a polynomial of degree 3. At each time step, the oldest location is removed and the current location is inserted. Given the historical obstacle data for each obstacle, we can employ cubic interpolation to predict the movement of the obstacle for a small number of hops, h, in the future, where $h \in \mathbb{N}$ is an input parameter to the algorithm. For each obstacle, we store h predictions in O, indicating our prediction of the locations of the obstacle for the next h time steps.

B. Temporal Cubes-Graph and Path

Suppose that the arm tip has reached some current cube q_a and that we have predicted the motion of the obstacles for the next h steps. To proceed, a *temporal cubes-graph* G_T is constructed, and a path in G_T is computed as follows.

Using Breadth-First Search [32], starting from q_a, we find all the cubes in the cubes-graph G_q within h hops from q_a. We then define a temporal subgraph G_T whose vertices are the cubes within distance h from q_a, and in which $(u, v, t, \mu = 1, d)$ is an edge if the arm can move from the cube q_u corresponding to u to the cube q_v corresponding to v at time t without colliding with obstacles, where d is the Euclidean distance between the centers of q_u and q_v. (We assume that the traversal time, μ, between any two adjacent cubes is 1.) Finally, a local target cube/vertex q_b in G_T is selected such that q_b is a closest cube in G_T to the final destination cube q_t. Using a shortest temporal paths algorithm [27], a sequence of cubes is found in the temporal cubes-subgraph from q_a to q_b. If q_t is more than h hops away from the starting cube, the above process of generating a temporal cubes path is repeated starting from q_b.

We note that, due to the temporal requirement, the shortest path between two vertices in a temporal graph cannot be computed using any of the folklore shortest path algorithms, such as Dijkstra's algorithm; however, a similar efficient algorithm to Dijkstra's was proposed in [27] and is used in this paper.

Algorithm 1 illustrates the construction of the temporal cubes-graph starting at a current cube q_a, and for a given number h of hops. The algorithm uses a similar idea to Breadth-First Search.

C. The Algorithm

The anticipatory path planning algorithm combines obstacle predictions and the temporal cubes-graph model to construct a sequence of configurations that navigates the arm tip from some starting point to a target point. We assume that we are given the configuration C^0 corresponding to the arm tip's initial position. This configuration is mapped to its corresponding $\mathcal{W}$-Space point s. First, we bin the starting point s into the starting cube q_s and the target point t into the target cube q_t. We then must wait 5 time steps so that the historical obstacle data can be collected. Next, the temporal cubes-subgraph at q_s is constructed using the process described in Subsection III-B. After a sequence of cubes from q_s to an intermediate target cube q_b is found, a layered configurations graph L is constructed as described in Section II-E using the configurations in each one of these cubes. The first layer includes the starting configuration (initially C^0) and the final layer includes all of the configurations whose corresponding $\mathcal{W}$-Space points fall in q_b. The Bellman-Ford algorithm is run on L, and a sequence of configurations P_L is computed. The first configuration C^0 in P_L places the arm tip at s (inside of q_s) and the last configuration C^b places the arm tip inside of q_b. These configurations in P_L are appended to a global path P and then the arm progresses to configuration C^b. If the target point has not been reached, the temporal cubes-graph is reconstructed centered around cube q_b. The first layer of the new layered graph consists of only configuration C^b. The process of constructing a temporal cubes-graph, finding a shortest cubes path, constructing a layered graph, and finding a sequence of configurations is repeated until the tip of the arm falls inside the final destination cube q_t. At this point, the global path P is returned. See **Algorithm 2** for the details.

Algorithm 1: *Temporal cubes-graph* construction

input : Current cube q_a, temporal prediction length h, end cube q_t, obstacle prediction data (for h steps) O
output : A temporal cubes-subgraph G_T and a cube q_b

$G_T := \emptyset$;
Use Breadth-First Search to find the set Q_a of cubes in G_Q within hop-distance h from q_a;
Define the vertices of G_T to be the vertices of G_Q corresponding to the cubes in Q_a;
Use the prediction data in O to construct the edges of Q_a; more specifically, there is a edge (u, v, t, μ, d), where $\mu = 1$ and d is the Euclidean distance between the centers of the cubes corresponding to u and v, if the arm can move from q_u to q_v without collision during the time interval $[t, t+1]$;
Using the coordinates of the centers of the cubes in G_T, find a cube q_b in G_T that is closest (w.r.t. its Euclidean distance) to q_t;
return G_T, q_b;

Algorithm 2: Shortest path for temporal approach

input : Starting configuration C^0, target point t, temporal prediction depth h
output : A list of configurations Ω

$\Omega := \emptyset$;
Let s be the $\mathcal{W}$-Space point corresponding to C^0;
Collect enough historical obstacle data to perform cubic interpolation and use it to initialize O;
Let $q_{cur} = q_s$, where q_s is the cube containing s, and let q_t be the cube containing t;
repeat until $q_{cur} = q_t$:
    Predict the future position of the obstacles for the next h steps and update O;
    Apply Algorithm 1 with $q_a = q_{cur}$, h, and O, to construct G_T and determine a local target cube q_b in G_T;
    Compute the shortest temporal cube path P in G_T from q_{cur} to q_b;
    Compute the layered graph L corresponding to P;
    Find a shortest path P_L of configurations in L by running Bellman-Ford Algorithm;
    Append P_L to Ω;
end
return Ω

IV. Results

In our implementation of the temporal approach, we used the built-in *interp1d* function provided by the *SciPy* library [33] to perform interpolation. This function allows us to input previous positions of the obstacles, and retrieve predictions as to their location for a limited number of time steps in the future. In our implementation, we interpolate on the positions of obstacles over the past 5 steps.

In order to evaluate our approach, we compared it against several approaches, including the folklore rapidly exploring random trees (RRT) [34] and potential field approaches, used for path planning of rigid robots. We also implemented two simple heuristics that rely on greedy methods. The empirical results obtained using the greedy heuristics demonstrate that the proposed anticipatory path planning approach for continuum arms is superior to baseline approaches, whereas the empirical results obtained using the potential field approach demonstrate that the anticipatory path planning approach for continuum arms is superior to some of the folklore methods used for path planning of rigid robots in dynamic environments.

A. First Heuristic Approach

One straightforward approach that may be used by the continuum arm to avoid mobile obstacles is to stop (or slow down) and wait for the obstacles to clear the way. This heuristic approach imitates the static obstacle planning approach in [24] but with modifications to account for the changing state of the graph. Since the obstacles are moving, our cube path must be constantly recomputed. At each

time step, we check to see if the next cube in the cube path is free of obstacles. If so, we compute the smoothest transition between the configurations in the current cube and the next cube using the method discussed in Subsection II-D. If the next cube contains an obstacle, that cube must be purged from the cubes-graph and then a new cube path is reconstructed. This proceeds until the arm collides with an obstacle, indicating a failure, or until the arm tip reaches the target point.

B. Second Heuristic Approach

The second heuristic approach is also based on the static path planning proposed in [24]. However, rather than recomputing a new path every time a cube is unavailable, the arm waits until the obstacle moves away. At each time step, we check to see if any obstacles are colliding with the next cube in the cube path. If not, the smoothest transition between the configurations in the current and next cube is calculated using the method discussed in Subsection II-D. If the obstacle is in the next cube, however, we simply wait until the obstacle is no longer colliding with the cube.

C. Potential Field Approach

A *Potential Field* [9] is a greedy method of path planning that uses attracting and repelling forces to move the arm tip towards a target point. A path planning implementation of a potential field approach for continuum arms was given in [13]. A similar approach was developed in [12]. Our implementation closely follows that in [13] and we compare our results to those obtained in [13].

The approach in [13] is a greedy approach that is unreliable in the presence of obstacles. When multiple obstacles are in the $\mathcal{W}$-Space, the magnitude and spread of repelling forces creates a repulsive "wall" that prevents the arm from moving towards the target. Additionally, in a dynamic environment, changes in the position of obstacles (and repelling forces) could reverse progress that the arm has already made towards the target point. These two issues, taken together, can cause the arm to move wildly back and forth.

V. Empirical Results

In order to test the performance of each approach fairly, we randomly generated test cases with moving obstacles that followed certain criteria. Each approach was tested on these trials, and their relative performance was recorded. Please refer to the supplemental multimedia material for a recording of some of these simulations.

The task for each algorithm is to output a sequence of configurations that brings the arm tip from some starting point to an ending point. If an obstacle collides with the arm at any point, that trial is considered a failure. Otherwise, the task is complete when the arm tip is within some small distance from the target point.For each number of obstacles $n \in \{1, 3, 5, 6\}$, we generate 100 test cases. For each n, we run each approach on each of the 100 test cases and keep track of the average running time (in seconds) and the success rate over all 100 test cases. Obstacles are modeled as spheres with their radius randomly chosen as either 2cm or 3cm. These obstacles orbit around random points along the straight line joining the starting to the ending point. The radius of these orbits is selected randomly in the range of 6cm to 15cm. The total range of x-axis, y-axis, and z-axis of the entire $\mathcal{W}$-Space are around $[-40.20cm, 40.20cm]$, $[-40.20cm, 40.20cm]$, $[-29.95cm, 45cm]$, respectively. For purposes of simplicity, we treat time as a discrete unit. Time is discretized into steps of 1 second each. In all of our simulations, we assume the obstacles move at a relatively low speed. Speed is a randomly generated number that is no greater than $1.5\ cm/s$. This is a reasonable assumption as today's high speed camera can capture moving objects at the frequency of multiple frames per second, which means the object moves at a low speed between frames [35].

TABLE I

Simulations of obstacles between start and end points

Algo. \ # Obs	1		3		5		6	
T_2 Time(s)/Succ	31.86	92%	39.37	69%	35.01	55%	32.58	46%
T_3 Time(s)/Succ	42.56	93%	50.61	72%	42.98	56%	40.19	46%
H Time(s)/Succ	10.41	58%	21.30	20%	30.31	14%	33.36	4%
H_s Time(s)/Succ	5.31	58%	12.23	21%	17.32	10%	18.57	4%
PF Time(s)/Succ	39.07	17%	41.22	17%	37.93	19%	70.27	19%

Table I shows the simulation results. We compare five different algorithms. T_2 and T_3 in the table refer to the temporal approaches that run on the next 2 and 3 predicted positions of obstacles while the H and H_s refer to the first and second heuristic approach, respectively. PF refers to the potential field approach.

As indicated from the simulation results, both temporal approaches perform significantly better than the heuristic approaches and potential field approach in all cases. Understandably, as the number of obstacles increased, the performance of all approaches decreased. However, the temporal approaches were consistently the best performing algorithm. As we discussed before, the limitations of potential fields prevented the arm tip from reaching the target point in most cases. Though this was expected with high numbers of obstacles, the potential field approach also under performed with few obstacles. The heuristic approaches were also understandably worse than the temporal approach as they do not consider the predicted motion of the obstacles. Although the larger prediction window should give T_3 an advantage over T_2, in some of our simulation cases, we observed that T_2 performed better than T_3. This is mainly caused by the fact our prediction algorithm (cubic interpolation) does not predict the obstacle motions at the third time step as accurately as the first two.

VI. Conclusion

In this paper, we studied path planning for continuum arms in dynamic environments. We proposed an anticipatory path planning algorithm based on temporal graphs, and compared its performance to folklore path planning approaches used for rigid robots in dynamic environments. Our findings

demonstrate the superiority of the proposed approach over other standard approaches.

Our work strongly advocates for the use of temporal graphs for performing anticipatory path planning in dynamic environments, and opens up many avenues for future research. In the current work, we used the simple cubic interpolation—as a proof of concept—as our prediction method. One obvious research avenue is to explore the use of more sophisticated prediction methods, or even the use of a set of different methods that each is suitable for a specific environment, such as prediction methods that rely on machine learning techniques [15], [16].

Finally, we mention that the construction of the temporal graph in each stage of our algorithm would be time consuming if implemented for a large prediction depth. To keep our algorithm efficient, we limited the depth to 2 and 3. While this depth still achieves a reasonable success rate compared to other approaches, the success rate gets lower as the number of obstacles increases. To achieve a high success rate in the presence of a large number of obstacles, one would need to explore paths in larger temporal graphs, which would require a faster construction of the temporal graph. We leave this direction as another avenue for future research.

REFERENCES

[1] M. W. Hannan and I. D. Walker, "Kinematics and the implementation of an elephant's trunk manipulator and other continuum style robots," *Journal of robotic systems*, vol. 20, no. 2, pp. 45–63, 2003.

[2] B. A. Jones, R. L. Gray, and K. Turlapati, "Three dimensional statics for continuum robotics," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 2659–2664.

[3] B. A. Jones and I. D. Walker, "Kinematics for multisection continuum robots," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 43–55, 2006.

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

[5] S. Hirose and M. Mori, "Biologically inspired snake-like robots," in *2004 IEEE International Conference on Robotics and Biomimetics*. IEEE, 2004, pp. 1–7.

[6] R. Buckingham, "Snake arm robots," *Industrial Robot: An International Journal*, vol. 29, no. 3, pp. 242–245, 2002.

[7] H. Ohno and S. Hirose, "Study on slime robot (proposal of slime robot and design of slim slime robot)," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, vol. 3. IEEE, 2000, pp. 2218–2223.

[8] R. J. Webster III, J. S. Kim, N. J. Cowan, G. S. Chirikjian, and A. M. Okamura, "Nonholonomic modeling of needle steering," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 509–525, 2006.

[9] Y. K. Hwang, N. Ahuja *et al.*, "A potential field approach to path planning." *IEEE Transactions on Robotics and Automation*, vol. 8, no. 1, pp. 23–32, 1992.

[10] K. P. Valavanis, T. Hebert, R. Kolluru, and N. Tsourveloudis, "Mobile robot navigation in 2-d dynamic environments using an electrostatic potential field," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 30, no. 2, pp. 187–196, 2000.

[11] S. Ge and Y. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous Robots*, vol. 13, pp. 207–222, 11 2002.

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

[13] I. S. Godage, D. T. Branson, E. Guglielmino, and D. G. Caldwell, "Path planning for multisection continuum arms," in *2012 IEEE International Conference on Mechatronics and Automation*, 2012, pp. 1208–1213.

[14] S. J. Guy and I. Karamouzas, "Guide to anticipatory collision avoidance," *Game AI Pro*, vol. 2, 2019.

[15] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, "Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 1056–1062.

[16] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, vol. 35, no. 1, pp. 51–76, 2013.

[17] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 981–986.

[18] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun, "Learning motion patterns of people for compliant robot motion," *The International Journal of Robotics Research*, vol. 24, no. 1, pp. 31–48, 2005.

[19] A. Cosgun, E. A. Sisbot, and H. I. Christensen, "Anticipatory robot path planning in human environments," in *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2016, pp. 562–569.

[20] A. Ismail, A. Sheta, and M. Al-Weshah, "A mobile robot path planning using genetic algorithm in static environment," *Journal of Computer Science*, vol. 4, no. 4, pp. 341–344, 2008.

[21] A. Stentz, "The focussed d* algorithm for real-time replanning," ser. IJCAI'95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 1652–1659.

[22] S. Koenig and M. Likhachev, "D*lite," in *Eighteenth National Conference on Artificial Intelligence*. USA: American Association for Artificial Intelligence, 2002, pp. 476–483.

[23] S. C. Yun, V. Ganapathy, and T. W. Chien, "Enhanced d* lite algorithm for mobile robot navigation," in *2010 IEEE Symposium on Industrial Electronics and Applications (ISIEA)*. IEEE, 2010, pp. 545–550.

[24] J. Deng, B. H. Meng, I. Kanj, and I. S. Godage, "Near-optimal smooth path planning for multisection continuum arms," in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2019, pp. 416–421.

[25] J. Stoer and R. Bulirsch, *Introduction to numerical analysis*. Springer Science & Business Media, 2013, vol. 12.

[26] I. S. Godage, G. A. Medrano-Cerda, D. T. Branson, E. Guglielmino, and D. G. Caldwell, "Modal kinematics for multisection continuum arms," *Bioinspiration & biomimetics*, vol. 10, no. 3, p. 035002, 2015.

[27] H. Wu, J. Cheng, S. Huang, Y. Ke, Y. Lu, and Y. Xu, "Path problems in temporal graphs," *Proceedings of the VLDB Endowment*, vol. 7, no. 9, pp. 721–732, 2014.

[28] V. Kostakos, "Temporal graphs," *Physica A: Statistical Mechanics and its Applications*, vol. 388, no. 6, pp. 1007–1023, 2009.

[29] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms,*, 3rd ed. The MIT Press, 2009.

[30] F. Lekien and J. Marsden, "Tricubic interpolation in three dimensions," *Journal of Numerical Methods and Engineering*, vol. 63, pp. 455–471, 2005.

[31] E. Maeland, "On the comparison of interpolation methods," *IEEE Transactions on Medical Imaging*, vol. 7, no. 3, pp. 213–217, 1988.

[32] G. B. Dantzig, "On the shortest route through a network," *Management Science*, vol. 6, no. 2, pp. 187–190, 1960.

[33] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online]. Available: http://www.scipy.org/

[34] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[35] I. C. Jeong and J. Finkelstein, "Introducing contactless blood pressure assessment using a high speed video camera," *Journal of medical systems*, vol. 40, no. 4, p. 77, 2016.