Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 54-20 (2021) 586–591

Efficient Path Planning of Soft Robotic Arms in the Presence of Obstacles

Preston R. Fairchild*, Vaibhav Srivastava*, Xiaobo Tan*

**Department of Electrical and Computer Engineering*
Michigan State University
East Lansing, MI 48824 USA
(e-mail: fairch42@msu.edu, vaibhav@egr.msu.edu, xbtan@egr.msu.edu)

Abstract: Soft robotic manipulators have seen growing interest in recent years and have many applications in the medical and industrial fields. Path planning algorithms for these soft continuum arms often have features to include obstacle avoidance. The redundant nature of soft robots allows for these manipulators to avoid obstacles while moving towards a goal. In this paper, a novel, efficient path planning algorithm is proposed for a soft robotic arm to navigate multiple obstacles in its workspace. The method aims to reduce the computational complexity and increase the precision of modeling curve-like obstacles by representing them with parametric equations instead of a set of points. The closest point between a function modeling the robotic arm and a function approximating the obstacle is updated in real-time and used to accommodate obstacle avoidance in path planning. The method is tested in simulation with a soft continuum arm represented by the piecewise-constant curvature model and the performance is compared to the traditional approach where an obstacle is defined by a set of points. The efficacy of the proposed approach is supported by simulation results from multiple obstacle settings, including one emulating multiple tree branches.

Keywords: path planning, obstacle avoidance, soft robotic arms, redundant manipulators.

1. INTRODUCTION

Soft continuum robotic manipulators are often based on biological designs such as octopus tentacles (Neppalli et al., 2007) and snakes (Luo et al., 2015). The continuum nature of these manipulators enables redundancy of control, allowing for sophisticated path planning algorithms that take into account obstacles that need to be avoided. Soft robots that work on complex tasks such as picking berries from bushes (Uppalapati et al., 2020) or performing non-invasive surgery (Diodato et al., 2018) require obstacle avoidance. Path planning for continuum arms has been examined in a number of settings (Xiao and Vatcha, 2010; Mbakop et al., 2020). Planning algorithms for redundant manipulators (Maciejewski et al., 1985) have been adapted to work for multi-segment continuum arms (Godage et al., 2012). Additionally, potential-field-based path planning (Khatib, 1985) has also been utilized with multi-segment continuum arms with additional considerations such as mechanical constraints of the manipulator (Ataka et al., 2016).

A common factor in these path planning algorithms is the necessity of accommodating obstacles. While the main objective of the algorithm is to reach an end goal, a secondary objective that must be achieved is avoiding collision with the obstacles. While methods for modeling the robotic arm and checking for collisions with the obstacles have been developed (Li and Xiao, 2012) and methods of detecting the obstacles as a set of points are available (Rakprayoon et al., 2011), the general method for modeling these obstacles has remained the same. The obstacles are typically represented through a point or set of points in 3D space. This method has many advantages: an adequate set of points could represent any object, it is simple to calculate the distance from the manipulator to these points, and the vectors between these points and the manipulator can be easily obtained. The disadvantage of this method lies in the increasing computational cost of a large number of points. Complex obstacles cannot be easily modeled without a large number of points unless features of the obstacle are simplified. Additionally, non-3D obstacles such as lines and planes require a high number of points to ensure there are no gaps in-between points mistaken for open space.

In this paper, a novel path planning method for soft continuum arms is proposed that efficiently accommodates multiple curve-like obstacles. The method applies to soft robots with workspaces containing curve-like obstacles such as tree branches, piping, wiring, etc. Geometric models for obstacles have been considered in literature in contexts such as motion planning for mobile robots and rigid robots [REFs]. In this paper, the motion planning framework for continuum robots developed in Ataka et al. (2016), in which obstacles were represented by discrete points, is built upon. For curve-like obstacles, it is shown that using geometric models of the obstacles enables efficient motion planning. Specifically, using these geometric models and optimization techniques, we propose a systematic approach to represent a set of obstacles and a soft robot through parametric equations and use these within the artificial potential-based planning framework. While such ideas have been used for artificial potential-based planning for rigid manipulators [REFs], to the best of our knowledge, such ideas have not been explored for soft robots. To facilitate the discussion of the main idea, the piecewise-constant curvature model is adopted for the soft robotic manipulator. The method for modeling an obstacle using a curve equation in 3D space is presented, as well as how it can be utilized in path planning algorithms in place of, or in

10.1016/j.ifacol.2021.11.235

tandem with, a set of points. The method is then tested in simulation with multiple settings for the obstacles, with the robotic arm described by a 3-segment constant curvature model. The proposed approach is compared to an alternative method where curve-like obstacles are modeled with a set of points, in terms of both computation time and ability to avoid collision with the obstacle.

2. METHOD

2.1 Kinematic Model

This section covers the kinematic model of the continuum arm used throughout this paper: a piecewise-constant curvature arm as described in Webster and Jones (2010). Fig. 1 illustrates the model with a 3-segment example.

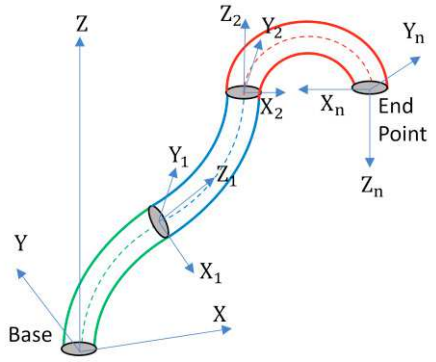
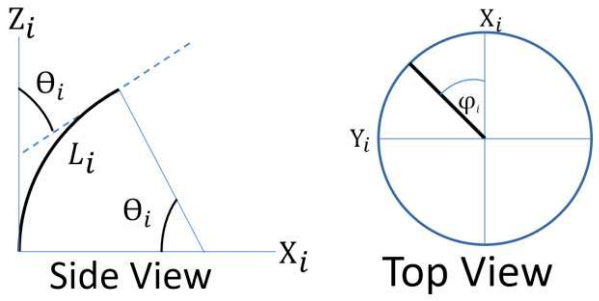


Fig. 1. Illustration of a continuum arm model with piecewise-constant curvature.

Each segment i, with $i = 1, 2, \dots, n$ for any number n of segments, of the arm is composed of an actuator of length $L_i = L_{0i} + l_i$, where L_{0i} is the base length of the segment and l_i is the length variation for the segment. Each segment has a base, starting at either the global base of the arm or the tip of the previous segment, with its own local coordinate system X_i-Y_i-Z_i. Z_i coincides with the tangential direction of the segment at its base while X_i-Y_i follow the right-hand rule and are determined by the bending of the previous segment. The bending on each segment can be modeled by two parameters, φ_i, which defines the direction of bending in the X_i-Y_i plane, and Θ_i, the magnitude of the angle the segment curves; see Fig. 2 for illustration. This bending results in a change in orientation from the base to the tip of a segment by a rotation about the Z_i axis by φ_i, followed by a rotation about the successive Y_i axis by Θ_i. A single segment can be described as an arc of a circle of length L_i and angle Θ_i, as shown in Fig. 2(a), with the direction of bending defined by φ_i, illustrated in Fig. 2(b). The radius of the circle the arc is formed from, r_i, is equal to L_i/Θ_i, with the curvature of the arc, $k_i = 1/r_i$.



(a) (b)

Fig. 2. (a) A side view of a segment with length $\boldsymbol{L_i}$ and bending of Θ_i. (b) A top view of a segment with bending direction $\boldsymbol{\varphi_i}$.

For a given segment i with configuration variables $q_i = [L_i \quad \Theta_i \quad \varphi_i]$, the displacement matrix relating its tip to its base is given by

$$H_i^{i-1} = \begin{bmatrix} R_i & P_i \\ 0_{1x3} & 1 \end{bmatrix}, \tag{1}$$

where R_i is the rotation matrix representing the orientation at the end of the segment relative to its base and P_i is the set of coordinates of the endpoint of the segment relative to its base. R_i and P_i can be calculated as follows (Webster & Jones, 2010):

$$R_i = \begin{bmatrix} \cos(\varphi_i)\cos(\Theta_i) & -\sin(\varphi_i) & \cos(\varphi_i)\sin(\Theta_i) \\ \sin(\varphi_i)\cos(\Theta_i) & \cos(\varphi_i) & \sin(\varphi_i)\sin(\Theta_i) \\ -\sin(\Theta_i) & 0 & \cos(\Theta_i) \end{bmatrix} \tag{2}$$

$$P_i = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} [1-\cos(\Theta_i)]\frac{L_i}{\Theta_i}\cos(\varphi_i) \\ [1-\cos(\Theta_i)]\frac{L_i}{\Theta_i}\sin(\varphi_i) \\ \sin(\Theta_i)\frac{L_i}{\Theta_i} \end{bmatrix}. \tag{3}$$

A value of 0 for Θ_i would lead to an arc of infinite radius and an undefined value for P_i. Instead, if $\Theta_i = 0$, P_i can be expressed as $[0 \quad 0 \quad L_i]^T$. The displacement matrix for each segment can be extended to include any number of n segments relating the base of the arm to the tip of the nth segment by

$$H_n^0 = \prod_{i=1}^{n} H_i^{i-1}. \tag{4}$$

2.2 General Path Planning Algorithm

This section covers a general path planning algorithm that utilizes the piecewise constant curvature model. For this manipulator model, any given point s relative to the base on a segment i of the arm can be calculated by

$$P_{si} = R_{i-1}^0 \begin{bmatrix} [1-\cos(\Theta_i s)]\frac{L_i}{\Theta_i}\cos(\varphi_i) \\ [1-\cos(\Theta_i s)]\frac{L_i}{\Theta_i}\sin(\varphi_i) \\ \sin(\Theta_i s)\frac{L_i}{\Theta_i} \end{bmatrix} + P_{i-1}^0, \tag{5}$$

where R_{i-1}^0 is the matrix representing the orientation of the ith segment base relative to the base of the arm, P_{i-1}^0 is the set of coordinates for the base of the ith segment with respect to the

base of the arm, and *s* is a parametric variable with a range [0 1] that defines the proportional distance along the segment. The Jacobian of the 3D position for any point *s* on the segment *i* can be derived with respect to the set of all configuration variables $q = [q_1, q_2, \dots, q_n]$ as $J_{si}(q) = \partial P_{si}/\partial q$. The Jacobian expresses the change in position of a point on the arm relative to any change in the configuration variables. Using this, the linear velocity for any point *s* on the arm segment *i*, $\dot{P}_{si}$, can be calculated as follows using the rate of change in the configuration variables $\dot{q}$.

$$\dot{P}_{si} = J_{si}(q)\dot{q}. \tag{6}$$

Conversely, the rate of change in the configuration variables needed to achieve a desired linear velocity can be expressed as

$$\dot{q} = J_{si}(q)^{+}\dot{P}_{si}\ . \tag{7}$$

Here $J_{si}(q)^{+}$ is the pseudo-inverse of the Jacobian and can be calculated as

$$J_{si}(q)^{+} = J_{si}(q)^{\mathrm{T}}[J_{si}(q)J_{si}{}^{\mathrm{T}}(q)]^{-1}, \tag{8}$$

with $J_{si}{}^{\mathrm{T}}(q)$ being the transpose of the Jacobian. In a scenario without obstacles, a path planning algorithm would assign $\dot{P}_{si}$ at the endpoint of the arm equal to a velocity that drives the endpoint of the arm directly to its desired end position. Given the redundant nature of soft robotic arms, a secondary objective can be assigned to a path planning algorithm such as avoiding obstacles. A general path planning algorithm that incorporates obstacle avoidance for a soft continuum arm takes the following form to specify the rate of change for the configuration variables $\dot{q}$ similar to the one presented in Ataka et al. (2016):

$$\dot{q} = J_e(q)^{+}F_e(V_e) - \textstyle\sum_{j=1}^{m} J_{sj}(q)^{+}F_{sj}(V_{sj}), \tag{9}$$

where $J_e(q)^{+}$ is the pseudo inverse of Jacobian at the endpoint of the manipulator and $F_e(V_e)$ is some gain function for the vector between the endpoint on the arm and end goal of the manipulator, V_e. For the *j*th obstacle of any number, *m*, obstacles, $J_{sj}(q)^{+}$ is the pseudo-inverse of the Jacobian of the point on the arm closest to the obstacle and $F_{sj}(V_{sj})$ is some gain function for the vector between the closest point on the arm and the obstacle, V_{sj}. This algorithm works by attracting the end of the manipulator to the end goal while repelling it away from any obstacle. The point on the arm closest to an obstacle can be calculated by discretizing the arm with a set number of points and comparing their distances to the obstacle. These obstacles are typically defined as a set of points, making the distance calculation trivial. However, as the number of points on the manipulator increases and the number of obstacles also increases, the computational cost for this calculation rises, as not only the number of distance calculations increases, the inverse Jacobian must be calculated for each point that falls within a critical range. This can present a problem, as the path planning algorithm should compute $\dot{q}$ repeatedly, as it allows for a dynamically changing end goal position and obstacles. This leads to a compromise needing to be made for the number of points on the arm and the obstacles in regards to computation time and accuracy of modeling. This compromise can be avoided if the obstacles resemble a curve-like object. In this case, the obstacle can be modeled as a function in space along with the manipulator, and the closest point between these two functions can be found instead and utilized in the path planning algorithm, reducing the computational complexity without sacrificing the accuracy in representing the obstacles. This is the key concept exploited in our proposed approach.

2.3 Curve-Like Obstacle Model

Using the method proposed in this paper, a curve-like obstacle can be avoided by evading the single point on the obstacle closest to the manipulator. This allows for a curve-like obstacle to be represented as a single point, which greatly simplifies the computation. In path planning algorithms, obstacles can be modeled by sets of points in 3D space that when combined can represent a variety of shapes. However, a small number of points may fail to properly represent elongated objects such as lines and curves. If the number of points is reduced, the manipulator may not properly avoid the obstacle. Instead, a curve-like obstacle can simply be represented as parameterized functions in 3D space. This allows for the accuracy of modeling the object with an infinite number of points, without the computational intensity of a large number of points. These curve-like objects in 3D space can be represented by

$$\mathrm{F}_0(u) = [X = \mathrm{f}_x(u), Y = \mathrm{f}_y(u), Z = \mathrm{f}_z(u)]^T, \tag{10}$$

where u is a parametric variable and f_x, f_y, and f_z are some functions of u. These equations could be used to represent objects such as tree branches, pipelines, wires, etc. The equations themselves could be derived from a set of points from a point cloud using techniques such as linear or quadratic regression. The curve itself could be of any length defined by the limits of u, allowing for longer objects to be modeled. In order to use the obstacle avoidance algorithm, the minimum distance between the obstacle curve and the arm must be found. Given a soft robotic arm of *n* segments, points on each individual segment *i* could be represented by parametric equations as follows

$$\mathrm{F}_{Ri}(s) = R_{i-1}^{0}\begin{bmatrix}(1-\cos(\Theta_i s))\frac{L_i}{\Theta_i}\cos(\varphi_i)\\(1-\cos(\Theta_i s))\frac{L_i}{\Theta_i}\sin(\varphi_i)\\\sin(\Theta_i s)\frac{L_i}{\Theta_i}\end{bmatrix} + P_{i-1}^{0}. \tag{11}$$

Using these curves, the distance between points on the obstacle and a segment for any set of parametric variables u and *s* can be calculated by

$$D_i(\mathrm{s},u) = \|F_{Ri}(s) - F_o(u)\|. \tag{12}$$

To calculate the minimum distance between the obstacle and segment *i*, three cases must be tested. The minimum distance between the segment and the obstacle will be the minimum of these three cases and the global minimum will be the minimum among the distances between the obstacle and all segments. The first case is when the partial derivatives of D_i with respect to both *s* and u are equal to 0:

$$\frac{\partial D_i}{\partial s} = 0 \Rightarrow \mathrm{F}_{Ri}(s)^T \dot{\mathrm{F}}_{Ri}(\mathrm{s}) - \mathrm{F}_o(u)^T \dot{\mathrm{F}}_{Ri}(\mathrm{s}) = 0 \quad (13)$$

$$\frac{\partial D_i}{\partial u} = 0 \Rightarrow \mathrm{F}_o(u)^T \dot{\mathrm{F}}_o(u) - \mathrm{F}_{Ri}(s)^T \dot{\mathrm{F}}_o(u) = 0. \quad (14)$$

For these equations $\dot{\mathrm{F}}_{Ri}(s)$ and $\dot{\mathrm{F}}_o(u)$ are the derivative of the curves with respect to *s* and *u* respectively. A solution (*s*, *u*) to both equations implies either a local minimum or maximum for the distance between the curves. If the point is a local maximum, the distance values obtained in the following cases checking the edge cases will be smaller, and thus this type of local extremum does not need to be calculated. The extremum may not necessarily be unique, so all solutions within the bounds of *s* and *u* should be calculated for the minimum of all solutions.

The second case is at the limits of either *s* and *u* where the derivative of the other parametric variable is zero. Values of *s* and *u* that solve these conditions can be found from any the equations below.

$$\frac{dD_i}{ds} = 0 \Rightarrow \mathrm{F}_{Ri}(s)^T \dot{\mathrm{F}}_{Ri}(s) - \mathrm{F}_o(u_{min})^T \dot{\mathrm{F}}_{Ri}(s) = 0 \quad (15)$$

$$\frac{dD_i}{ds} = 0 \Rightarrow \mathrm{F}_{Ri}(s)^T \dot{\mathrm{F}}_{Ri}(s) - \mathrm{F}_o(u_{max})^T \dot{\mathrm{F}}_{Ri}(s) = 0 \quad (16)$$

$$\frac{dD_i}{du} = 0 \Rightarrow \mathrm{F}_o(u)^T \dot{\mathrm{F}}_O(u) - \mathrm{F}_{Ri}(0)^T \dot{\mathrm{F}}_O(u) = 0 \quad (17)$$

$$\frac{dD_i}{\partial u} = 0 \Rightarrow \mathrm{F}_o(u)^T \dot{\mathrm{F}}_O(u) - \mathrm{F}_{Ri}(1)^T \dot{\mathrm{F}}_O(u) = 0. \quad (18)$$

The values of *s* and *u* for any solution to the above equations should be only included if they exist within the limits of the parametric variables.

The third and final case is where the minimum distance corresponds to the endpoints of both the obstacle and manipulator. All four combinations of endpoints need to be checked for their distances.

Once the individual point on the manipulator that is closest to the obstacle curve is found, the minimum distance can be used in the original obstacle avoidance algorithm (9) with the obstacle being treated as a single point. The algorithm can be extended to include any number of curved obstacles without inhibiting the ability for other obstacles to be modeled by a set of points in the same path planning algorithm.

3. SIMULATION RESULTS

The proposed path planning algorithm is evaluated in MATLAB simulation, with a soft continuum arm of three segments. The simulations are done on a Windows 10 PC with an Intel Core i5 @3.80GHz and 16.0 GB RAM. The method is compared with an alternative where the obstacles are approximated with a discrete set of points evenly distributed with respect to *u* along the curve, and another set of points representing the discretization of the robotic arm. The inverse Jacobian is *not* calculated for each point of the obstacle, only once per obstacle. The majority of the computational time for the method using sets of points arises from the distance calculations and comparisons. To find the shortest distance for the method utilizing a set of points, the MATLAB function *dsearchn* is used. The MATLAB function *lsqnonlin* is used to solve the nonlinear equations from (13-18). Table 1 presents the parameters of the simulated arm.

Table 1. Simulation parameters for the soft arm.

Number of segments	3
Static length of segments (L_i)	1
# Of points per segment on the arm for distance calculations	10

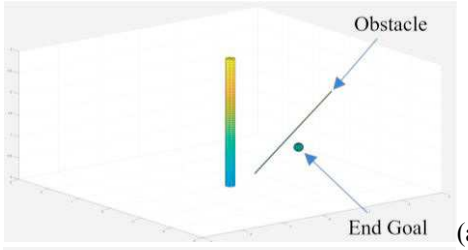
The computational time and the number of iterations (i.e., the number of times $\dot{q}$ needs to be calculated as in (9) to reach the end goal) for the different methods are calculated in differing scenarios and given in Tables 2-4 with visual representations shown in Figs. 3-5.

3.1 Simulation 1

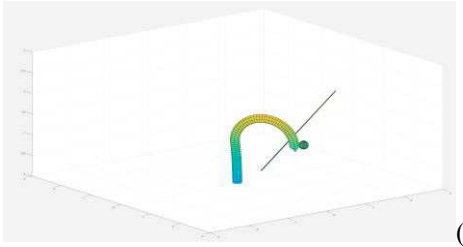
Simulation 1 has the manipulator navigate to a goal position while avoiding a linear obstacle. Results of different methods are shown in Table 2, with snapshots of the arm under the proposed method presented in Fig. 3.

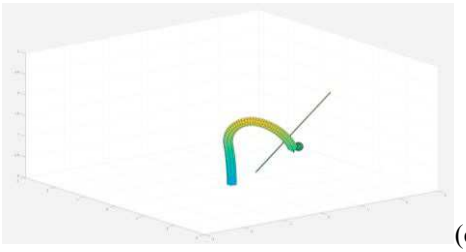
Table 2. Simulation 1: Line segment

End goal	[1.1, -0.8, 0.9]	
Obstacle equation	$X = 1, Y = u, Z = 0.5 - u$ -2 < *u* < 2	
Obstacle model used in simulation	Iterations to reach goal	Average computation time (s)
Proposed method	60	0.2914
1 Point	24	0.0053
10 Points	24	0.0690
50 Points	24	0.4254
100 Points	66	1.6679



(a)


(b)



(c)

Fig. 3. Snapshots of arm movement under the proposed algorithm for simulation 1 at (a) 0 iterations, (b) 40 iterations, (c) final iteration.

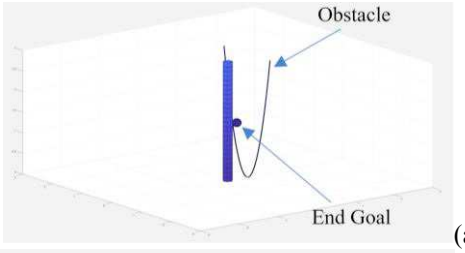
In simulation 1, the proposed method has a similar computation time to that of modeling the obstacle with 50 points. However, the 50-point method did not properly model the obstacle and the manipulator collided with the obstacle. Only the proposed and the 100-points methods led to manipulator avoiding the obstacle, although the 100-points simulation had a higher computation time relative to the proposed method.

3.2 Simulation 2

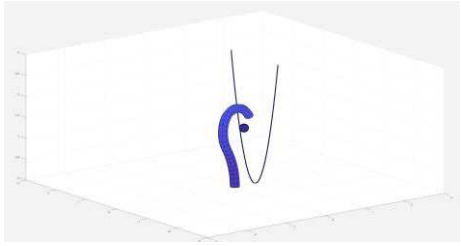
Simulation 2 has the manipulator navigate to a goal position while avoiding a curved obstacle. Results of differing methods are shown in Table 3, with snapshots of the arm under the proposed method presented in Fig. 4.

Table 3. Simulation 2: Curved line segment

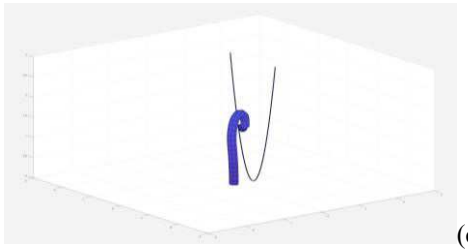
End goal	[1, 1, 1]	
Obstacle equation	$X = 0.5, Y = u, Z = 5u^2$ $-2 < u < 2$	
Obstacle model used in simulation	Iterations to reach goal	Average computation time (s)
Proposed method	30	0.2971
1 Point	23	0.0044
10 Points	23	0.0599
50 Points	24	0.2720
100 Points	23	0.5794



(a)



(b)



(c)

Fig. 4. Snapshots of arm movement under the proposed algorithm for simulation 2 at (a) 0 iterations, (b) 20 iterations, (c) final iteration.

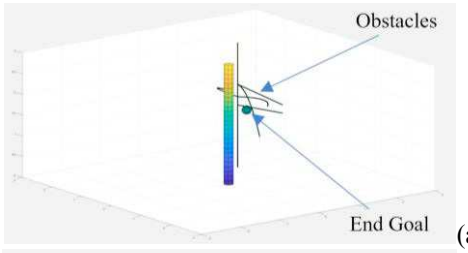
In simulation 2, the proposed method again has a similar computation time to that of the 50-point method. Of all tested methods including the more computationally expensive 100-point method, only the proposed method avoided collisions.

3.3 Simulation 3

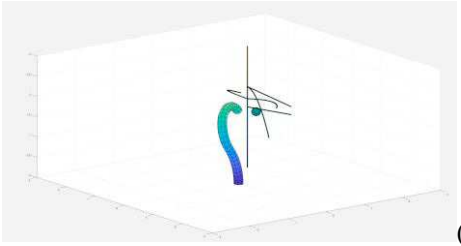
Simulation 3 has the manipulator navigate to a goal position while avoiding six curved obstacles. The obstacles were chosen to represent a tree-like environment. Results of differing methods are shown in Table 4, with snapshots of the arm under the proposed method presented in Fig. 5.

Table 4. Simulation 3: Multiple curves

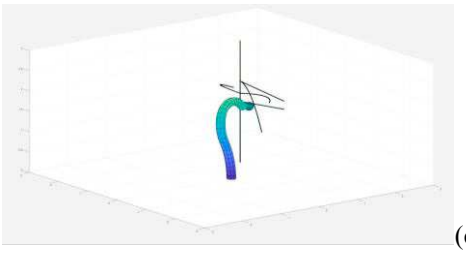
End goal	[1, 0.7, 1.45]			
Obstacle equations	X	Y	Z	u
	$1 - 0.2u$	$1 - u$	$2 - u^2$	0-1
	$1 - 0.5u$	$1 + u - u^2$	$1.7 + 0.3u$	0-1.5
	1	$1 - u$	$2 - 0.1u$	0-1.5
	$1 + u - u^2$	$1 - u$	1.7	0-1
	1	$1 - u$	$1.5 + 0.1u$	0-1.5
	1	1	$u - u^2$	0-3
Obstacle model used in simulation	Iterations to reach goal	Average computation time (s)		
Proposed method	32	0.7414		
1 Point	19	0.1090		
10 Points	19	0.7594		
50 Points	32	5.7699		



(a)



(b)



(c)

Fig. 5. Snapshots of arm movement under the proposed algorithm for simulation 3 at (a) 0 iterations, (b) 20 iterations, (c) final iteration.

In Simulation 3, the proposed method performs significantly faster than the 50-point representation, comparable to that of the 10-point method. The arm avoids the obstacles in both the proposed and 50-point methods. In all 3 simulations, the proposed method avoided collisions and had a lower computation time than other methods that avoided collisions.

4. CONCLUSION

In this paper, an efficient path planning approach was presented for soft robotic arms. Most existing obstacle avoidance algorithms using sets of points to represent obstacles will need a minimal alteration in order to implement the proposed method. The proposed method showed a reduction of computational complexity compared to approaches using many points representing curve-shaped obstacles and was demonstrated to be effective in handling complex obstacle scenarios including one emulating a target behind multiple tree branches.

Future work will extend the approach to soft robotic arms without the piecewise-constant curvature assumption, to accommodate realistic shapes under effects of gravity and other loadings. The proposed algorithm will also be tested experimentally on soft robotic arms.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation (ECCS 2024649) and the Michigan State University Strategic Partnership Grants program (16-SPG-Full-3236).

REFERENCES

Ataka, A., Qi, P., Liu, H., and Althoefer K. (2016). Real-time planner for multi-segment continuum manipulator in dynamic environments. In 2016 IEEE International Conference on Robotics and Automation, Stockholm, Sweden, pp. 4080-4085.

Diodato, A., Brancadoro, M., Rossi, G.D., Abidi, H., Dall'Alba, D., Muradore, R., Ciuti, G., Fiorini, P., Menciassi, A., and Cianchetti, M. (2018) Soft robotic manipulator for improving dexterity in minimally invasive surgery. *Surgical Innovation*, 25 (1), pp. 69-76.

Godage, I.S., Branson, D.T., Guglielmino, E., and Caldweel D.G. (2012). Path planning for multisection continuum arms. In 2012 IEEE International Conference on Mechatronics and Automation, Chengdu, China, pp. 1208-1213.

Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In 1985 IEEE International Conference on Robotics and Automation, St. Louis, MO, USA, pp. 500-505.

Li, J., and Xiao, J. (2012). Exact and efficient collision detection for a multi-section continuum manipulator. In 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, pp. 4340-4346.

Luo, M., Pan, Y., Skorina, E.H., Tao, W., Chen, F., Ozel, S., and Onal, C.D. (2015). Slithering towards autonomy: A self-contained soft robotic snake platform with integrated curvature sensing. *Bioinspiration & Biomimetics*, 10 (5), 055001.

Maciejewski, A.A., and Klein, C.A. (1985). Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *The International Journal of Robotics Research*, 4(3), pp. 109-116.

Mbakop S., Tange G., Lakhal O., Merzouki R., and Drakunov S.V. (2020) Path planning and control of mobile soft manipulators with obstacle avoidance. In 2020 3rd IEEE International Conference on Soft Robotics, New Haven, CT, USA, pp. 64-69.

Neppalli, S., Jones, B., McMahan, W., Chitrakaran, V., Walker, I., Pritts, M., Csencsits, M., Rahn, C., Grissom, M. (2007). OctArm - a soft robotic manipulator. In 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, pp. 2569-2569.

Rakprayoon, P., Runchanurucks, M., and Coundoul, A. (2011) Kinect-based obstacle detection for manipulator. In 2011 IEEE/SICE International Symposium on System Integration, Kyoto, Japan, pp. 68-73.

Uppalapati, N.K., Walt, B.T., Havens, A.J., Armeen, M., Chowdhary, G., and Krishnan, G. (2020) A berry picking robot with a hybrid soft-rigid arm: design and task space control. In Robotics: Science and Systems 2020, Corvalis, Oregon, USA.

Webster, R.J., Jones, B.A. (2010) Design and kinematic modeling of constant curvature continuum robots: A review. *The International Journal of Robotics Research, 29* (13), pp. 1661-1683.

Xiao, J., and Vatcha, R. (2010). Real-time adaptive motion planning for a continuum manipulator. In 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, pp. 5919-5926.