2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Task-constrained Continuum Manipulation in Cluttered Space

Jinglin Li and Jing Xiao
Department of Computer Science
University of North Carolina at Charlotte
jli41@uncc.edu, xiao@uncc.edu

***Abstract*— Continuum manipulators do not contain rigid links and can deform continuously to perform a whole arm manipulation. Hence, they are much more flexible than articulated manipulators to perform tasks in cluttered space. However, autonomous manipulation constrained by tasks other than grasping has not been studied for continuum manipulators. In this paper, we introduce a general and efficient approach for autonomous continuum manipulation under task constraints. We consider a spatial continuum manipulator with multiple uniform-curvature sections if not deformed. We further apply the approach to an example of inspection task in a cluttered environment to verify its effectiveness. The high-efficiency of our approach makes it suitable to run on-line for guiding task-constrained manipulation in real-time.**

I. Introduction

Unlike conventional articulated manipulators, continuum manipulators [1], [2] are inspired by invertebrate structures found in nature, such as octopus arms [3], [4] and elephant trunks [5]. A representative continuum manipulator is the OctArm (Fig. 1). Smaller scale continuum manipulators are also developed for medical surgery applications [6], [7].

The shape of a multi-section continuum manipulator can continuously deform via changing the controllable degrees of freedom, such as bend, extend/contract, and torsional turn the arm sections; the manipulator is also passively compliant due to their infinite number of passive degrees of freedom. Hence, continuum manipulators are very flexible and particularly suitable for performing tasks in cluttered environments. It can be infeasible for an articulated manipulator to reach a target blocked by obstacles in a cluttered space. A continuum manipulator, however, can bend and curl its arm to reach the target behind an obstacle without collision. Fig. 2 shows one example of a cluttered pipe environment, courtesy of the Electric Power Research Institute (EPRI): certain pipes in this environment can be reached by a continuum manipulator for inspection but are difficult to reach by an articulated manipulator.

Fig. 1. An OctArm manipulator (courtesy of Ian Walker)

Fig. 2. A cluttered pipe environment (courtesy of EPRI)

Continuum manipulation was mainly performed via teleoperation in the past [8], [9]; only recently, motion planning algorithms of surgery insertion using pre-curved concentric tube robots have been proposed in [10], [11]. Autonomous algorithms of grasping a target object using continuum robots have been introduced in [12], [13], while avoiding obstacles [14], [15], and in cluttered space [16]. However, certain tasks, such as inspection of power structure, require that the end-effector follow a specified continuous path in the task space, but autonomous manipulation under task constraints is an open problem for continuum manipulators.

Existing research on manipulation under task constraints was focused on conventional (mobile) articulated manipulators [17]–[19], where inverse (instantaneous) kinematics was incorporated in motion planning to satisfy task constraints.

Inverse kinematics for a continuum manipulator was studied as the problem of finding the arm shape and pose given specific positions of the tip and the base. [20] proposed a numerical algorithm to compute the inverse-kinematics solutions for the arm. However, how to handle the arm redundancy considering the interactions between the arm and the objects in the physical space was not considered, and the cost of solving a numerical solution increases significantly as the number of arm sections increase. [21] required specifying the end positions of each arm section as input and proposed a geometric method to compute the inverse-kinematic solutions for each section. However, how to decide the end positions of each section under task constraints was not considered.

Moreover, how to enable a continuum manipulator to satisfy both kinematic constraints and task constraints while avoiding obstacles, especially in a cluttered environment, has not been studied thus far.

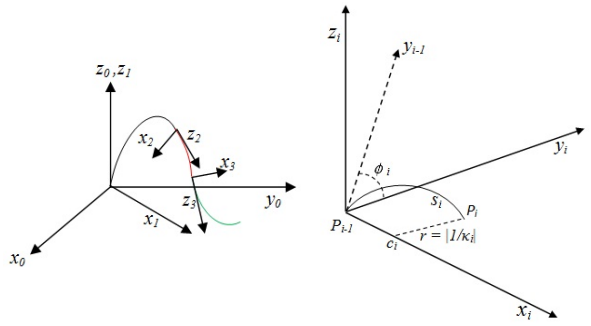
In this paper, we introduce a general and efficient approach for autonomous continuum manipulation under task constraints. The rest of the paper is organized as follows. Section

II introduces the manipulator model considered. Section III defines task constraints. Section IV describes our approach to solve the manipulation problem under task constraints. Section V provides an example application of our framework in pipe inspection. Section VI presents the experimental results. Section VII concludes our paper.

II. Manipulator Model

We first briefly review an n-section continuum manipulator model [16], inspired by the representative OctArm manipulator (see Fig. 1). The base frame of the robot is set at p_0 with z_0 axis tangent to the central axis of the first section sec_1. The frame of the i-th section sec_i is formed at p_{i-1} with the z_i axis tangent to the central axis of sec_i at p_{i-1}. The base of sec_i is the tip of sec_{i-1}. Adjacent sections are connected *tangentially* at the connection point p_{i-1} as shown in Fig. 3(a).

Each section i has three degrees of freedom, i.e., controllable variables: curvature κ_i, length s_i, and rotation angle ϕ_i from y_{i-1} axis to y_i axis about z_i axis. Fig. 3(b) shows one example seg_i, its frame, and controllable variables.



(a) Central axes of multiple sections and their frames (b) Section i and its controllable variables

Fig. 3. Section frames and controllable variables per section

We denote a *super-section* $Ssec_{1,i}$ formed by sections from sec_1 to sec_i when the central axes of these connected sections share the same curvature and on the same plane (i.e., $\phi_j = 0$, $1 < j \leq i$), and therefore on the same circle, see Fig. 4 for illustration. The super-section $Ssec_{1,i}$ now behaves like a large, single section with three degrees of freedom: curvature κ_1, rotation angle ϕ_1, and the sum $s_{1,i}$ of section lengths from sec_1 to sec_i. In the rest of the paper, we use *arm segment* i to denote either the arm section sec_i or the super-section $Ssec_{1,i}$.

In general, the *arm configuration* of an n-section continuum manipulator is determined by the control variables of each section, denoted as $C = \{(s_1, \kappa_1, \phi_1), ..., (s_n, \kappa_n, \phi_n)\}$. We further denote the *partial arm configuration* for sections from sec_k to sec_i $(1 \leq k \leq i)$ as $C_{k,i}$. Note that $C_{i,i}$ indicates the configuration of a single section sec_i; whereas, $C_{1,n}$ also indicates an (entire) arm configuration.

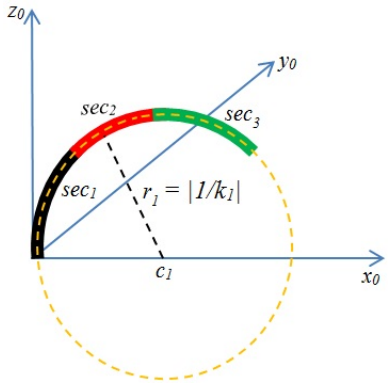


Fig. 4. A super-section $Ssec_{1,3}$ consisting of sec_1 (black), sec_2 (red) and sec_3 (green) with their central axes on the x_0-z_0 plane

III. Task Constraints

In a robot working environment, an object is a target object if it is to be handled by the robot as required by a task. Other objects are obstacles to avoid in performing the task. Each object can be represented by polygonal meshes. In general, in order to perform a task, a continuum manipulator has to decide how to move, not only to reach a goal while avoiding obstacles but also to satisfy certain task constraints.

We define task constraints in the Cartesian space in terms of restrictions on the position of the tip of a continuum manipulator required by a task. Specifically, we focus on the case that the tip of the manipulator has to follow a curve in the Cartesian space, i.e., the task constraint is defined by a *task curve* in this paper. A task curve could be viewed as either describing the goal of a task or ending at the goal configuration of a task. Here we assume the task curve is either given (for simple tasks) or obtained through motion planning of the tip point [19], [22].

We call an arm (or partial arm) configuration a *feasible* configuration if it does not make the manipulator collide with obstacles and also satisfies the task constraint for the task.

IV. Task-constrained Manipulation

Given a task curve, which can be described in terms of a discrete sequence of points $G = \{g_1, g_2, ..., g_m\}$ in the Cartesian space that the tip of a continuum manipulator must follow, the problem of planning for task-constrained manipulation is to find a feasible path of arm configurations for the tip to follow the task curve while the arm avoiding obstacles efficiently. This problem can be divided into two intertwined subproblems: search of arm shapes in the configuration space of the manipulator and search of base positions for all arm sections in the Cartesian space, so that a corresponding arm configuration can be computed by [21] given the base positions for arm sections. In the following, we address these subproblems and present our approach for planning.

A. Search of arm shapes

Even though the configuration space of a continuum arm is constrained by the task curve that the tip of the arm has to

follow, it is still huge because of the high dimensionality, e.g., 9-D for the OctArm, which consists of three arm sections. Direct search of such a space, even by random sampling, would be very inefficient. Therefore, our strategy is to search from a subspace and gradually relax the subspace as needed to find feasible arm configurations. This core idea is presented in a recursive function **Algorithm 2**, which is called by the main algorithm **Algorithm 1** for our approach.

Algorithm 1 starts by considering the shape of the arm in the simplest form as one super-section $Ssec_{1,n}$, whose tip needs to reach the first point g_1 on the task curve, and calls **Algorithm 2** to get a feasible arm configuration. **Algorithm 2** first computes the corresponding configuration of the super-section by inverse kinematics [21]. If no feasible arm configuration exists (i.e., either no inverse solution for the tip at the target point or the solution is not free of obstacles), our algorithm relaxes the shape of the arm by breaking it into a shorter super-section $Ssec_{1,n-1}$ and the last arm section sec_n, whose tip must reach the point g_1 on the task curve. This requires deciding the target position q for the base of sec_n, i.e., also the tip of $Ssec_{1,n-1}$, as described in section IV.B, to find a feasible configuration for sec_n, and the corresponding partial arm configuration $C_{1,n-1}$ when the tip of $Ssec_{1,n-1}$ reaches q.

If there is no feasible configuration for super section $Ssec_{1,n-1}$, **Algorithm 2** further breaks down $Ssec_{1,n-1}$ into a shorter super-section $Ssec_{1,n-2}$ and arm section sec_{n-1}, and so on. So in general, the considered arm shape can be in terms of a super section $Ssec_{1,j}$ and arm sections sec_{j+1}, ..., sec_n, $1 \leq j < n$. In the most relaxed case, where $j = 1$, the arm shape consists of no super section and just n arm sections.

For the arm tip to reach the next point g_t, $t = 2, ..., m$, on the task curve, **Algorithm 1** calls **Algorithm 2** to decide the next arm configuration C_t based on the current feasible arm configuration C_{t-1} where the arm tip is at g_{t-1}. **Algorithm 2** first tries to decide the configuration of the last arm segment, which can be either section sec_n or super-section $Ssec_{1,n}$, to reach the new tip position g_t without changing the segment's base position via inverse kinematics [21]. If the solution for the segment exists and is feasible, then the resulting arm configuration C_t is feasible, because the rest of the arm below the last segment is intact as part of the feasible configuration C_{t-1}. Otherwise, the base position of the last segment is changed as described in section IV.B in order to find a feasible configuration.

Our strategy is very efficient for several reasons. For collision checking between the arm and the obstacles (which may also include the target object), we use the collision detection algorithm [23] that treats a super-section just as a regular arm section as a primitive unit for efficient check. The fewer sections (including a super-section) in an arm, the less time for collision checking of an arm configuration. Also, the fewer sections, the fewer target points connecting sections need to be searched. Moreover, a super-section can be considered just as a regular section for inverse kinematics [21]. Thus, the fewer sections in an arm, the less time for solving inverse kinematics. Finally, for the arm tip to move from point g_{t-1} to point g_t on the task curve, the new arm configuration C_t can often re-use the previous arm configuration C_{t-1} for some arm sections without repeatedly solving for inverse kinematics and checking for collisions.

Algorithm 1: $GenPath$

input : *Task curve* $G = \{g_1, g_2, ..., g_m\}$, arm model and an initial config. C_0
output: A $path$ of arm configuration for the tip to follow G or report failure

```
1  begin
2      path ← null;
3      C_current ← C_0, flag ← "super-section";
4      for t = 1 to m do
5          p_n ← g_t;
6          if NewArmConfig(C_current, p_n, n, flag)
           returns C_{1,n} then
7              C_current ← C_{1,n};
8              path ← path ∪ {C_current};
9          else
10             return "No path is found";
11         end
12     end
13     return path;
14 end
```

B. Search of base position for an arm segment

As described earlier, if a considered arm shape is in terms of a super-section $Ssec_{1,j}$ and arm sections sec_{j+1}, ..., sec_n, $1 \leq j < n$, the base positions of sections sec_{j+1}, ..., sec_n need to be determined given that the tip of sec_n must reach a target point g_t on the task curve, so that a feasible arm configuration can be computed via inverse kinematics [21]. If the base of the manipulator is not fixed, then the base position for $Ssec_{1,j}$ also needs to be determined.

Algorithm 2 generates a new base position q_{new} for an arm segment according to its current base position q and its required tip position p_i by searching in a neighborhood Q of q in the Cartesian space. The scope of the neighborhood can be determined by the range of length of the segment such that from any point in Q, p can be reached. Q can also be determined based on task-specific information, as shown in section V. Depending on the shape and size of Q, either random sampling or systematic sampling of Q can be used to find q_{new}.

The found q_{new} not only ensures that sec_i's configuration $C_{i,i}$ is *feasible*, but also that a corresponding partial arm configuration $C_{1,i-1}$ with its tip at q_{new} is *feasible*. This means that, in the worst case, base positions of sec_1 to sec_{i-1} will also be changed by the recursive **Algorithm 2**[1], and the number of base positions searched per segment is limited by the threshold max. Even though the worst-case time

[1]Note that sec_1's base is the manipulator base, and it can be either fixed or changeable.

Algorithm 2: $NewArmConfig(C, p_i, i, flag)$

input : Arm configuration C, new tip position p_i for segment i, $flag$ to indicate if segment i is a *"super-section"*

output: a feasible new partial arm configuration $C_{1,i}$ or "no feasible solution"

begin
  if *segment i can reach p_i from its current base in a feasible configuration $C_{i,i}$* **then**
    return the new $C_{1,i}$ from sec_1 to sec_i, where configurations of sections with indices lower than segment i remain the same as in C;
  end
  if $flag =$ *"super-section"* **then**
    break segment i into $Ssec_{1,i-1}$ and sec_i;
    $flag \leftarrow$ *"section"*;
    return $NewArmConfig(C, p_i, i, flag)$;
  end
  if $i = 1$ *with fixed base* **then**
    return *"no feasible solution"*;
  end
  $\#tries \leftarrow 0$;
  while $\#tries < max$ **do**
    generate q_{new} as a new base point for sec_i;
    if *segment i-1 is a super-section* **then**
      $flag \leftarrow$ *"super-section"*;
    else
      $flag \leftarrow$ *"section"*;
    end
    if $NewArmConfig(C, q_{new}, i-1, flag)$ *returns* $C_{1,i-1}$ **then**
      if *sec_i can reach p_i from q_{new} in a feasible configuration $C_{i,i}$* **then**
        return new arm configuration $C_{1,i} \leftarrow C_{1,i-1} \cup C_{i,i}$;
      end
    end
    $\#tries = \#tries + 1$;
  end
  return *"no feasible solution"*
end

complexity for base point search is max^i, the average time in practice is much, much less because (a) a tip position of a segment (with 3-DOFs) can usually be reached from many base positions in a sizable continuous region (or conversely, from a base position, many tip positions can be reached, forming the "dexterous workspace" of the arm segment), and (b) either the current q is already feasible or a feasible q_{new} can often be obtained by a small change of the current q. The experimental results in Section VI for an example of inspection task confirms the efficiency of base point search.

V. Task of Inspection

In this section, we describe how to apply our strategy for task-constrained manipulation to the task of autonomously inspecting the surface of an object, such as a pipe, for possible defects in a cluttered environment (e.g., see Fig. 2).

A. Task description

Assuming that an inspection device (e.g., a camera) is mounted on the tip of a continuum manipulator. The tip of the manipulator will have to cover the entire surface of an object following certain specific path, which defines task constraints. Without losing generality, we consider the tip scan the object surface line by line (or curve by curve if the surface is not flat), and each line/curve scan defines a task curve. For example, Fig. 5 shows a cylindrical object and N task curves for inspecting it; each task curve is a circle surrounding the object, and all task curves are parallel.

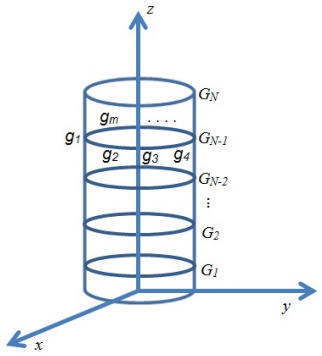


Fig. 5. A cylindrical object and the task curves for inspection

In general, we can describe a task curve with respect to the object frame o-xyz. Let z be the axis that task curves stack along (see, for example, Fig. 5). Each point on a task curve can be further described by cylindrical coordinates (d, θ, z). By using such a cylindrical coordinate system in the Cartesian space, the possible neighborhood Q for searching the base point of a segment can usually be considered as 2-D (instead of 3-D) with a more or less fixed d value (i.e., fixed distance to the z axis).

Consider sec_i. If its current base point q does not result in a feasible configuration $C_{i,i}$, in searching a better q_{new}, the θ and z values of q are adjusted in the direction of approaching its target tip point; if $C_{1,i-1}$ is not feasible, the θ and z values of q are adjusted in the direction of approaching the robot base point.

We define the direction of a scan (to complete a task curve) based on if the continuum manipulator extends or contracts itself. If the manipulator arm extends itself in a scan, the scan is called in *forward direction* and is a *forward scan*; Otherwise, if the manipulator arm contracts in a scan, the scan is called in *backward direction* and a *backward scan*.

B. Inspection algorithm

Algorithm 3 outlines the main algorithm for the inspection task. The manipulator alternates a *forward scan* with a

backward scan to visit all *task curves* around the object. In this way, after the manipulator finishes a forward scan, with the arm largely extended, a small update of its tip along the z axis of the object frame is often sufficient to set it on course for the next scan in reverse direction (i.e., a backward scan).

In the first scan, our strategy described in Section IV is applied to lead the arm follow its first *task curve*, and all base points of arm sections are recorded along the way (for each arm shape consists of one or more arm sections in addition to the super-section). In a subsequent scan, the recorded base points of arm sections from the previous scan are re-used with small adjustments, as the algorithm calls **Algorithm 2**, taking advantage of the fact that the current task curve is more or less a simple shift of the previous task curve along the z axis.

Clearly, the time taken for generating a path of arm configurations for a subsequent scan is usually shorter than that for the first scan.

Algorithm 3: *Inspection*

input : A set of *task curves* $\{G_1, G_2, ...G_N\}$ for inspection, the initial arm configuration
output: The feasible path of arm configurations for inspection or report failure

begin
  $Path \leftarrow null$;
  $scan \leftarrow 1$; // *"1" indicates forward scan*
  call **Algorithm 1** with *task curve* G_1;
  if *no feasible path of arm configurations is returned* **then**
    return "No solution";
  end
  record the feasible path $path_1$ and section base positions for every configuration on $path_1$;
  $Path \leftarrow Path \cup path_1$;
  for $i = 2$ **to** N **do**
    $scan \leftarrow 1 - scan$; // *change scan direction*
    $path_i \leftarrow path_{i-1}$; // *copy previous path*
    foreach *point* g_t *on* G_i *along scan direction,* $1 \leq t \leq m$ **do**
      $p_n \leftarrow g_t$;
      call **Algorithm 2** to update configuration C_t on $path_i$ with the new tip position p_n;
      if *no feasible configuration is returned* **then**
        return "No solution";
      end
    end
    record the updated $path_i$;
    $Path \leftarrow Path \cup path_i$;
  end
  return $Path$
end

VI. Implementation and Results

We have implemented all the algorithms on a 2.40GHz Intel(R) Xeon(R) CPU with 4.00 GB RAM and tested them on a three-section continuum manipulator with a fixed base for the task of inspecting a tilted cylindrical object in a cluttered space as shown in Fig. 6. Ten scans are defined by 10 *task curves* $\{G_1, G_2, ...G_{10}\}$, and each *task curve* is discretized into 500 target points for the tip of the arm to visit. We set $max = 6$ for $\#tries$ in **Algorithm 2**. Our algorithms return a feasible path of arm configurations for every task curve. Snapshots of one forward and backward scan, as the results of applying our inspection algorithm, are shown in Fig. 7.

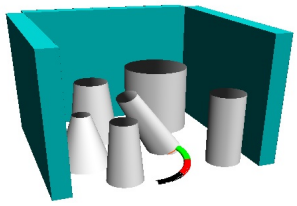
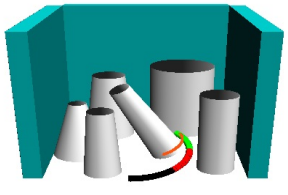
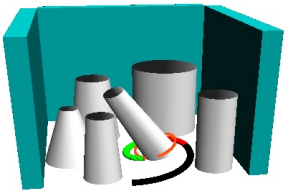


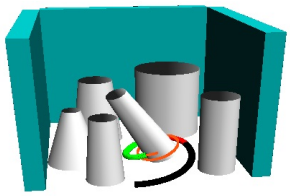
Fig. 6. A target object (the middle oblique object) and obstacles in a cluttered environment



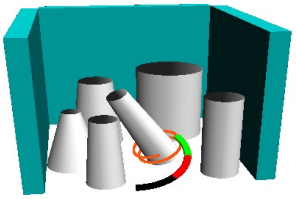
(a) Forward scan for G_1 with the arm shape of $Ssec_{1,3}$



(b) Forward scan for G_1 with the arm shape of sec_1, sec_2 and sec_3



(c) Backward scan for G_2 with the arm shape of sec_1, sec_2 and sec_3



(d) Backward scan for G_2 with the arm shape of $Ssec_{1,3}$

Fig. 7. Snapshots of a forward scan of *task curve* G_1 and a subsequent backward scan of *task curve* G_2, with the results shown as the red curves

In Table I, we present the related costs of the two scans, with the forward scan as the first scan and the backward scan obtained by updating the results of the forward scan. As expected, the backward scan takes much less time than the forward scan (about half of the time of the forward scan) to find feasible configuration. The table also shows different time costs to find a feasible arm configuration under different arm shapes. For the most constrained and simplest shape $Ssec_{1,3}$, as expected, our algorithm used the least amount of time on average (2 ms in both forward and backward scans) to find a feasible configuration. Whereas, it took the most

TABLE I

TIME COST OF THE SCANS ILLUSTRATED IN FIG. 7

scan	arm shape	# config.	avg. # tries	# collision checks	total plan. time	avg. plan. time/config.
forward for G_1	$Ssec_{1,3}$	173	0	173	0.34 (s)	2.0 (ms)
	$Ssec_{1,2}$ and sec_3	95	1.55	385	0.77 (s)	8.1 (ms)
	sec_1, sec_2 and sec_3	232	2.43	1866	3.73 (s)	16.0 (ms)
backward for G_2	$Ssec_{1,3}$	173	0	173	0.34 (s)	2.0 (ms)
	$Ssec_{1,2}$ and sec_3	95	0.75	194	0.38 (s)	4.1 (ms)
	sec_1, sec_2 and sec_3	232	1.07	930	1.86 (s)	8.0 (ms)

amount of time to find a feasible configuration on average for the most relaxed shape with three single sections sec_1, sec_2 and sec_3 (and no supersection) (16.0 ms and 8.0 ms in the forward and backward scans respectively). Note that the average $\#tries$ to search a suitable base position for each arm segment (i.e., a section or a super-section) is quite low. In the case the arm forms a single super-section, no search of base point is needed (due to the fixed base of the manipulator).

The attached video shows the complete process of inspection by alternating forward and backward scans of 10 task curves with the found feasible paths of arm configurations by our algorithms.

VII. CONCLUSIONS

In this paper, we proposed a general and efficient approach for planning an n-section continuum manipulator to perform task-constrained manipulation in cluttered space. A task constraint is defined as a curve that must be followed by the tip of the manipulator. Our algorithms find a path of feasible arm configurations for the manipulator to satisfy the task constraint while avoid obstacles in the cluttered environment. We applied our approach to an inspection task, and the testing results show that our approach generated feasible paths correctly and efficiently.

Because of its fast computation for each feasible configuration (in a few ms), our approach can be used to conduct not only off-line path planning but real-time simultaneous planning and execution of a path by a continuum manipulator. The latter option is more important as it has the potential to enable inspecting an object that is only partially visible initially, and as the inspection proceeds, more of the object can become visible. Thus, inspecting an object in real time with partial information will be a next step. Another extension is to consider changing environments with dynamic obstacles. In addition, we will consider constraining the shape of the arm sections to enable task-constrained whole arm manipulation. It is also important to study the effect of gravity and the resulting shape deformation.

ACKNOWLEDGMENT

This work is supported by the US National Science Foundation grant IIS-0904093 and by funding from EPRI. The authors appreciate discussions with Robert Grizzi and John Lindberg of EPRI on inspection tasks.

REFERENCES

[1] G. Robinson and J. B. C. Davies, "Continuum robots - a state of the art," *Proc. CDEN Design Conf.*, pp. 2849–2854, 1999.
[2] D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied Bionics and Biomechanics*, vol. 5(3), pp. 99–117, 2008.
[3] W. McMahan, B. A. Jones, I. D. Walker, V. Chitrakaran, A. Seshadri, and D. Dawson, "Robotic manipulators inspired by cephalopod limbs," *Proc. CDEN Design Conf.*, 2004.
[4] R. Kang, D. Branson, T. Zheng, E. Guglielmino, and D. Caldwell, "Design, modeling and control of a pneumatically actuated manipulator inspired by biological continuum structures," *Bioinspiration&Biomimetics*, 2013.
[5] R. Cieslak and A. Moreck, "Elephant trunk type elastic manipulator a tool for bulk and liquid type materials transportation," *Robotica*, 17, pp. 11–16, 1999.
[6] R. J. Webster, J. M. Romano, and N. J. Cowan, "Mechanics of precurved-tube continuum robots," *IEEE Trans. Robot.*, 25(1), 2009.
[7] J. Furusho, T. Katsuragi, T. Kikuchi, T. Suzuki, H. Tanaka, Y. Chiba, and H. Horio, "Curved multi-tube systems for fetal blood sampling and treatments of organs like brain and breast," *J. Comput. Assist. Radiol. Surg*, vol. 1, pp. 223–226, 2006.
[8] W. McMahan, V. Chitrakaran, M. Csencsits, D. Dawson, I. Walker, B. Jones, M. Pritts, D. Dienno, M. Grissom, and C. Rahn, "Field trials and testing of the octarm continuum manipulator," *Proc. IEEE Int. Conf. on Robot. and Auto. (ICRA)*, 2006.
[9] W. McMahan and I. Walker, "Octopus-inspired grasp synergies for continuum manipulators," *IEEE Int. Conf. on Robot. & Biomim.*, 2009.
[10] L. G. Torres and R. Alterovitz, "Motion planning for concentric tube robots using mechanics-based models," *IEEE/RSJ Int. Conf. on Intel. Robot. and Sys. (IROS)*, 2011.
[11] L. G. Torres, R. J. Webster, and R. Alterovitz, "Task-oriented design of concentric tube robots using mechanics-based models," *IROS*, 2012.
[12] J. Li and J. Xiao, "Determining "grasping configurations for a spatial continuum manipulator," *IROS*, 2011.
[13] J. Li and J. Xiao, "Progressive generation of force-closure grasps for an n-section continuum manipulator," *ICRA*, 2013.
[14] J. Xiao and R. Vatcha, "Real-time adaptive motion planning for a continuum manipulator," *IROS*, 2010.
[15] J. Li, Z. Teng, J. Xiao, A. Kapadia, A. Bartow, and I. Walker, "Autonomous continuum grasping," *IROS*, 2013.
[16] J. Li and J. Xiao, "Progressive, continuum grasping in cluttered space," *IROS*, 2013.
[17] O. Brock, O. Khatib, and S. Viji, "Task-consistent obstacle avoidance and motion behavior for mobile manipulation," in *ICRA*, 2002.
[18] M. Stilman, "Global manipulation planing in robot joint space with task constraints," *IEEE Trans. on Robot.* , vol. 26, pp. 576–584, 2010.
[19] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *The Int. J. of Robot. Res.*, 2011.
[20] I. S. Godage, E. Guglielmino, and D. T. Branson, "Novel modal approach for kinematics of multisection continuum arms," *IROS*, 2011.
[21] S. Neppalli, M. A. Csencsits, B. A. Jones, and I. Walker, "A geometrical approach to inverse kinematics for continuum manipulators," *IROS*, 2008.
[22] L. E. Kavraki, P. Svestka, J. C.Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. on Robot. and Auto.*, vol. 12, pp. 566–580, 1996.
[23] J. Li and J. Xiao, "Exact and efficient collision detection for a multi-section continuum manipulator," *ICRA*, 2012.