Progressive Object Modeling with a Continuum Manipulator in Unknown Environments

Huitan Mao[1], Zhou Teng[1,2], and Jing Xiao[1]

***Abstract*— It is important to enable a robot to manipulate a target object that has no 3-D model information and is situated in an environment with other unknown objects nearby. This poses an open problem of how to combine perception and manipulation to enable the robot to build an appearance-based model of the target object on the spot to facilitate further manipulation of the object while avoiding the other unknown obstacles in the way. In this paper, we introduce an approach to enable a continuum manipulator, which is apt to maneuver through a crowded environment, to gradually build a 3-D surface model of the target object by moving an RGB-D sensor around the object while also detecting and avoiding surrounding unknown obstacles.**

Our approach interleaves perception and manipulation such that perception guides the manipulator movement, which in turn allows more perception of the target object for object model building and further manipulator motion. Our approach is characterized by a progressive strategy to register RGB-D images of the target object to build and extend a partial model of the object and the corresponding motion planning strategy for the continuum robot to carry out model building and avoid obstacles at the same time. To demonstrate the effectiveness of our approach, experiments on progressive model building of real objects from real RGB-D images are conducted, where a simulated continuum robot plans and executes its motion to carry the RGB-D camera around a target object for taking those images in an augmented reality setting.

I. Introduction

Although object models are not needed for people to perform many manipulation tasks, they are often necessary for robots to perform manipulation tasks. In order to expand the application of robotic manipulation, especially in unstructured and unknown environments, it is important to equip a robot with the capability of perceiving and building the object model of a target object on the spot to facilitate further manipulation of the object. For example, in order to remove a piece of suspecious foreign object on a cluttered desk, the robot needs to find ways to grasp it by constructing some model of the object through perception to facilitate grasping. In order to construct the object model, the robot also needs to move a sensor around the object and find suitable locations to collect sensory data, for instance, visual images of the object, while avoiding other objects. In such a scenario, perception and manipulation planning have to facilitate each other.

In related work, researchers have studied view planning [1], [2] to find the best set of views to capture the surface appearance of an isolated object, object model building [1], [3] of an isolated object, and object manipulation through perception and learning [4], compliance with soft hands [5] or through exploiting environment constraints [6] without explicit model building. Some recent work of object modeling [7]–[9] uses a robot hand to hold the object for pose change, but the hand can also occlude the object. However, a common underlying assumption is that the object is stand alone in isolation so that there is no need to consider obstacles. UAVs are also considered to model open and large environments [10], [11], but they are not suitable to maneuver in a narrow and cluttered space to model small daily objects. For the vast literature on motion planning for manipulation, object model is assumed known and so are obstacle models to some extent. How to manipulate an unmodeled object in a cluttered environment with unknown obstacles is an open problem.

Continuum manipulators [12]–[14] can deform continuously and are inherently compliant [15], [16], which are more suitable to maneuver in a cluttered environment. So far most existing work on autonomous object manipulation using a continuum manipulator assumes that the object model and the environment are known (i.e., with known models) [17]–[20] or fully visible through some external sensor (such as an overhead camera) [21], and only recently, manipulating a known object situated in an unknown, cluttered environment is addressed [22]. There is no work on handling an object without a known model in a cluttered space with a continuum manipulator.

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

In this paper, we address autonomous and progressive model building of an object in an unknown, cluttered environment using a multi-section continuum manipulator (see Fig. 1 for an example). We consider that an RGB-D camera is mounted on the tip of the continuum manipulator for perception. Our approach plans the robot arm motion to position the camera at suitable spots around the target object to take images and register those images to build and extend a 3-D model of the object gradually, while avoiding obstacles. Section II presents an overview of our approach. Section III describes the strategy of perception-guided planning and

[1]The authors are with the Department of Computer Science, University of North Carolina, Charlotte, NC 28223, USA, {hmao4, xiao}@uncc.edu

[2]The author is with ABB US Corporate Research Center, Bloomfield, CT 06002, USA, zhou.teng@us.abb.com

execution of robot arm motion for modeling an object. Section IV presents our progressive strategy to build partial object models based on RGB-D sensing. Section V provides experimental results and analyses. Section VI concludes the paper.

II. Overview

A. Environment and Task

The environment considered in this paper is an unknown cluttered space containing multiple obstacles and a single target object. The task is to use a continuum manipulator to approach the target object and build a model for it, while avoiding obstacles. We further assume the following:

(1) The continuum manipulator has a fixed base.
(2) All the objects in the environment are static.
(3) The target object to be modeled can be wrapped around by the robot arm either in one direction, clockwise or counter-clockwise, or both directions to form a closed loop coverage of the side surface[1].
(4) An RGB-D camera is attached to the tip of the robot so the robot is able to sense the surroundings as it moves, called a tip camera in this paper.
(5) The robot arm is initially positioned, by human assistance, outside of the unknown task space with its tip camera facing the target object.

Some typical environments and tasks are shown in Fig. 6.

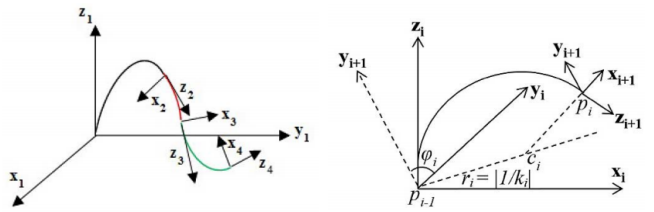
A general n-section continuum manipulator is used. Each section sec_i, i=1, ..., n is characterized by its three controllable variables: length s_i, curvature κ_i, and orientation ϕ_i, as illustrated in Fig. 2(b). Each sec_i can be positioned by specifying its base point p_{i-1}. The tip point p_i of sec_i is computed using its corresponding base point and controllable variables. Adjacent sections are connected tangentially as shown in Fig. 2(a). The *arm configuration* of an n-section continuum manipulator can be formulated as $C = \{(s_1, \kappa_1, \phi_1), ..., (s_n, \kappa_n, \phi_n)\}$ [20], [23]. This kinematic model is widely used in the literature thanks to its generality [23]–[26]. Note that the arm can in fact have infinite degrees of freedom because it can deform upon contact. However, for the task of object modeling, the arm is not planned to contact the target object or obstacles. The finite number of controllable variables are sufficient to describe the collision-free motion of the arm.

B. Approach

In order to address the autonomous and progressive modeling of an object in an unknown and cluttered environment, sensing and robotic manipulation have to be interleaved closely in that sensing guides the manipulation and the manipulation in turn enables further sensing in cluttered space. Fig. 3 presents an overview of our proposed approach.

Starting with an initial RGB-D image of the target object taken at the initial configuration of the robot (see assumption (5) in Section II.A), the robot's tip camera is adjusted to first sense the environment and distinguish the target object among the sensed obstacles (Section III.A). Then the tip camera is steered towards the target object again to take an RGB-D image, and the captured target object point cloud is registered with the previous partial model (or image) of

[1]This notion indicates the projected directions of the arm's spatial motion on a plane.



(a) Frames are attached to each robot section.

(b) Length s_i, curvature κ_i, and orientation ϕ_i are controllable variables per section.

Fig. 2: The robot frames and the controllable variables [22].

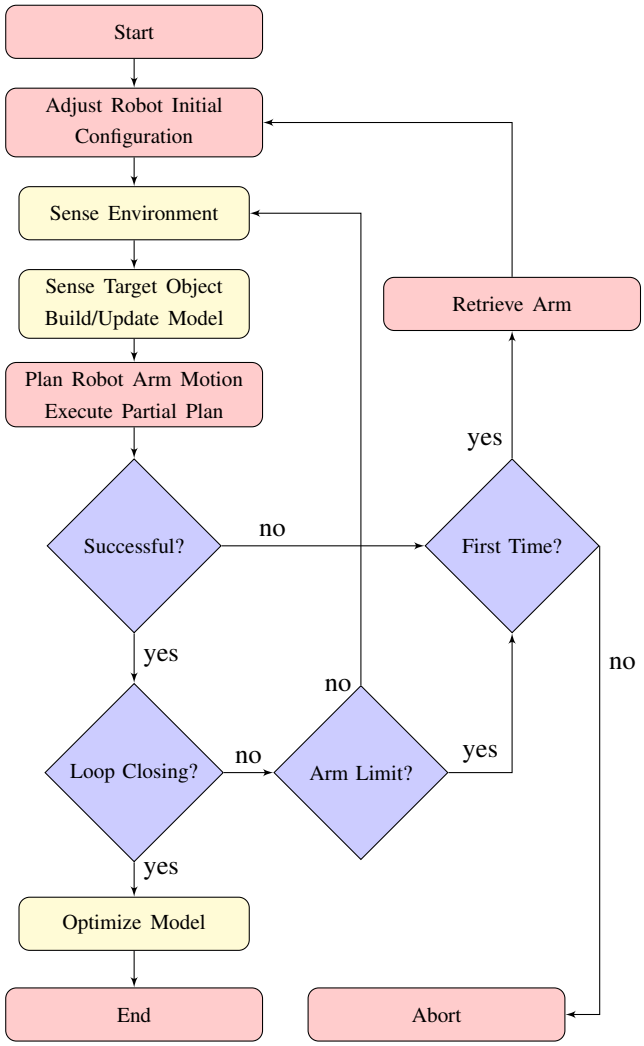


Fig. 3: Overview of the progressive modeling approach. The blocks for perception are in yellow, and those for robot motion are in red.

the object (see Section IV). Next, the robot arm motion is planned and executed based on the updated partial object model for a short distance (See Section III.B).

As the above procedure is iterated, more and more images of the target object are captured, and if the number of captured images is sufficient to make a closed loop coverage of the side surface of the object, the obtained object model is refined by a global optimization algorithm.

There are two cases when a closed loop cannot be reached in the current direction of wrapping the target object: (1) the arm extends to its physical limit before closing the loop, and (2) there is not enough space between the target object and obstacles to allow the arm move forward (i.e., no more collision-free motion forward for the arm can be found). In both cases, the arm is retrieved and re-positioned to allow for moving in the other direction around the object to continue object model building.

If loop closing cannot be achieved from both directions around the object, the program ends with a partial object model (or disconnected partial models) without optimization.

III. Perception-based Motion Planning and Execution

A. Distinguishing Target Object and Obstacles

Distinguishing the target object from surrounding obstacles is the first task for the continuum robot arm to accomplish towards perception-guided object modeling. By initially making the tip camera facing the target object, a point cloud of the target object can be obtained and marked. When it is time to sense the environment (as shown in the first yellow box of the flowchart in Fig. 3), the camera is turned to face the space between the target object and obstacles, and the obstacle point clouds are sensed, clustered, and distinguished from the object point cloud based on Euclidean distance. For each subsequent sensing of the environment as the robot arm moves, the clusters in the newly sensed point clouds are either merged (registered) with the existing ones or saved as new clusters (i.e. new obstacles are discovered). Clustering based on Euclidean distance is computed efficiently using K-d tree data structure.

B. Planning and Execution of Robot Arm Motion

Starting from an initial configuration of the continuum manipulator, **Algorithm 1** first plans a collision-free arc for the robot arm to follow. Such an arc is computed through searching in the sensed cluttered space, taking into account the partial object model, obstacles, and robot arm constraints [22]. Collision detection between the arc and obstacles or object is checked efficiently based on the algorithm in [27].

Next, the robot executes a small portion of the planned motion by making the tip follow the arc and conform one or multiple sections to the curvature and orientation of the planned arc. The resulting arm configuration can be found using constrained inverse kinematics [19]. The short distance Δs, for which the robot arm moves along the planned arc, can be scaled to guarantee that there is enough overlap between two consecutive images taken of the target object.

As the robot arm moves along the first arc, the visible region of the target object and environment grows. Thus, robot motion is further planned. After each new sensing, our **Algorithm 1** searches another collision-free arc to replace the current arc, such that the updated arc starts from the same position as the current arc but is flatter (smaller curvature) and longer to take advantage of the updated sensed space. The robot arm is then made to follow and conform to the updated arc. If no such replacement arc can be found due to inevitable collision with the obstacles or exceeding the arm limits, **Algorithm 1** plans a new arc for the robot arm to follow once the robot's tip reaches the end of the current arc, and the new arc maintains tangential continuity at its connection to the end of the current arc.

Therefore, motion planning alternates between finding a replacement for the current arc or a new arc. By first exhausting the possibilities of updating the current arc that the robot arm followed before adding a new arc, **Algorithm 1** generates an efficient motion for the continuum arm to observe a target object with as few robot sections mobilized as possible. To move the robot along a planned arc can be done either with a closed-loop controller [28] or without one because our model building strategy (see Section IV) is robust to motion uncertainty.

If the space between the object and obstacles is too small at some point so that no collision-free path can be found for the arm to move forward, **Algorithm 1** exits and reports no motion.

Algorithm 1: Planning and Execution of Robot Arm Motion

```
if no previous path for the robot arm then
    Plan the first arc for the robot arm;
else
    Update the current arc the robot arm follows;
    if updating fails then
        Plan a new arc;
if no collision-free path can be found then
    Set flag "No Motion" and exit;
Move the robot arm forward along the newly updated
or planned arc for a short distance Δs;
```

IV. Progressive Object Modeling

The objectives of progressive object modeling are (1) to build a 3D point cloud model of the target object, and (2) to obtain the 6D pose (i.e., position and orientation) of the target object, both reasonably accurately, in the presence of uncertainties in robot pose (e.g., caused by payload and motion error) and camera pose w.r.t. the robot. In order to achieve both objectives, we use a two-step approach:

- forward pairwise registration of the current point cloud of the partial object model (starting from the first object image) to each newly obtained object image as the robot arm moves;

- global optimization of the registration results from closed-loop images captured to increase accuracy of both the 3D object model and its 6D pose.

A. Forward Pairwise Registration

To start, the object point cloud C_1 of the first object RGB-D image is registered (forward) to the object point cloud C_2 of the next object image to form a partial object model with the corresponding transformation matrix ${}^{2}\hat{T}_1$ from the frame of C_1 to that of C_2. As the robot arm moves and takes a new image of the object and obtains the object point cloud C_3, C_2 is then registered to C_3, with the transformation matrix ${}^{3}\hat{T}_2$, and so on. In general, each time a new object point cloud C_{i+1} is obtained from a new image, the point cloud C_i (of the partial object model) is registered to C_{i+1}, $i = 1, 2, ...$ Note that the images of C_i and C_{i+1} have sufficient overlap regions to facilitate the registration of the two point clouds, and C_i and C_{i+1} are called *neighboring* point clouds.

We use the following strategy to register one object point cloud C_i to the next one C_{i+1}:

- Extract and match ASIFT [29] keypoints between the two RGB-D images, as shown in Fig. 4; if the matched keypoint pairs are sufficient, compute the transformation matrix ${}^{i+1}\hat{T}_i$ based on their 3D coordinates, using the RANSAC [30], [31] and SVD [32] algorithms.
- Apply the ICP algorithm [33]: if a transformation matrix is computed from the previous step, use it as the initial input to the ICP algorithm; otherwise, use an identity matrix as the initial estimate. Note that as the robot arm motion between taking two object images is small, our approach avoids large jumps in coordinates from one image to the other to ensure the effectiveness of the ICP algorithm. Each motion step can always be further reduced to guarantee robust registration if necessary.

Fig. 4: Two images of neighbouring object point clouds and their keypoints matching results.

With an initial transformation estimate from the keypoints matching, the ICP algorithm converges much faster and also achieves registration results in better quality. It takes into account all the following factors in a weighted sum to evaluate the quality of matching point pairs: distance between the two points, color similarity of the two points based on RGB values, distance between normals at the two points, and the distance from one point to the approximate plane of the other point.

A round of forward pairwise registration ends when either (i) a closed loop is achieved such that the object point cloud of the last image C_m sufficiently overlaps with the object point cloud C_1 of the first image or (ii) the robot arm has reached its limit to extend further. In the case (ii), if loop closure cannot be achieved with moving the arm around the object in one direction (e.g., clockwise), the arm is retrieved after images have been taken along one motion direction and is commanded next to move around the object in the opposite motion direction (e.g., counter-clockwise) to close the loop.

Once loop closure is achieved, the registration results are optimized globally (indicated in Fig. 3) as detailed below.

B. Global Optimization

After forward pairwise registration that registered m object point clouds C_1, C_2, ... C_m, the built partial object model is subject to accumulation error of registration. Specifically, ${}^{1}\hat{T}_m$ from registering C_m to C_1 directly to close the loop can be significantly different from the inverse of the product ${}^{m}\hat{T}_{m-1}{}^{m-1}\hat{T}_{m-2}...{}^{2}\hat{T}_1$, even though they should be the same if there were no registration error. See Fig. 5 for the pairwise coordinate transformations obtained from forward pairwise registration.

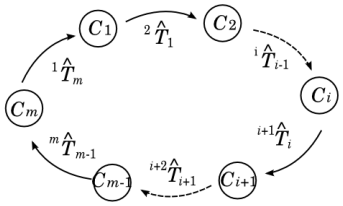


Fig. 5: The frames of all the object point clouds and their relations, illustrated as a closed loop.

Let O indicate the frame for the partial object model after the loop is closed, which can be the same as the frame for point cloud C_m. Then, the pose of each object point cloud with respect to the object model frame O can be expressed by the transformation transition as follows:

$$ {}^{O}\hat{T}_m = I, \tag{1} $$

$$ {}^{O}\hat{T}_i = {}^{O}\hat{T}_{i+1}{}^{i+1}\hat{T}_i, \quad i = m-1, m-2, ..., 1. \tag{2} $$

In order to reduce the accumulated registration error embedded in the transformation transition result ${}^{O}\hat{T}_i$, $i = 1, ..., m$, our approach is to further refine ${}^{O}\hat{T}_i$, starting from C_m, by registering the point cloud C_i to both its neighbours C_{i-1} and C_{i+1}, $i = 1, ..., m$, based on virtually generated matched point pairs [34]. Note that $C_{m+1} = C_1$, and $C_0 = C_m$.

The virtually generated matched points purely based on the transformation matrix are also called *virtual mates*. To generate virtual mates between C_i and its neighbours C_{i-1} and C_{i+1}, we first sample points from C_i in the overlap regions between C_i and C_{i-1} and between C_i and C_{i+1} respectively.

Suppose p is a sampled point from C_i, with its position in the object point cloud frame denoted as $\mathbf{p}$, and let q be the virtually generated mate of p:

- if p is sampled from the overlap region between C_i and C_{i+1}, then the position of its virtual mate ${}^{i+1}\mathbf{q}$

is obtained by transforming $\mathbf{p}$ from the frame i to the frame $i+1$ based on ${}^{i+1}\hat{T}_i$;

- if p is sampled from the overlap region between C_i and C_{i-1}, then the position of its virtual mate ${}^{i-1}\mathbf{q}$ is obtained by transforming $\mathbf{p}$ from the frame i to the frame $i-1$ based on ${}^{i-1}\hat{T}_i$, and ${}^{i-1}\hat{T}_i$ is the inverse matrix of ${}^{i}\hat{T}_{i-1}$.

Notice that virtual mates do not really exist in the point clouds C_{i-1} or C_{i+1}. They are introduced as the constraints of both ${}^{i}\hat{T}_{i-1}$ and ${}^{i+1}\hat{T}_i$ in the optimization to refine the pose ${}^{O}\hat{T}_i$. ${}^{i}\hat{T}_{i-1}$ and ${}^{i+1}\hat{T}_i$ cannot be directly used as constraints, because it is difficult to directly measure the change of the registration result from the change of the rigid 3D transformation matrix in the optimization.

Finally, we further transform all the positions of the generated virtual mates ${}^{i+1}\mathbf{q}$ and ${}^{i-1}\mathbf{q}$ to be in the object model frame O, based on the current poses of their corresponding object point cloud frames ${}^{O}\hat{T}_{i+1}$ and ${}^{O}\hat{T}_{i-1}$, respectively. The refined pose ${}^{O}\hat{T}_i$ is computed based on all of those virtually generated point pair positions $\mathbf{p}$ and $\mathbf{q}$ with respect to the object model frame O.

Algorithm 2 starts from the refinement of ${}^{O}\hat{T}_m$. If the difference of the pose ${}^{O}\hat{T}_i$ (in terms of the sum of the absolute differences of all matrix elements between two transformation matrices) before and after the optimization is greater than a tolerance threshold, then ${}^{O}\hat{T}_{i-1}$ and ${}^{O}\hat{T}_{i+1}$ also need to be refined; each pose ${}^{O}\hat{T}_i$ may be refined more than once, depending on C_{i-1} and C_{i+1}. Our **Algorithm 2** ends when no more refinement is needed or a maximum number of iteration is reached, and the globally optimized estimate ${}^{O}T_i$ for each object point cloud C_i is obtained. **Algorithm 2** is implemented using a double-ended queue.

Algorithm 2: *GlobalOptimization*

```
1  ^O T̂_m = I, Queue = ∅;
2  for i = (m − 1) to 1 do
3  |  ^O T̂_i = ^O T̂_{i+1} ^{i+1} T̂_i;
4  end
5  ^O T̂_{m+1} = ^O T̂_1, C_{m+1} = C_1;
6  ^O T̂_0 = ^O T̂_m, C_0 = C_m;
7  Queue ← ^O T̂_m;
8  while Queue ≠ ∅ and num_iterations <
   maximum_num_iterations do
9  |  ^O T̂_i = Queue.pop_front(i);
10 |  ^O T̂_i = Align(C_i, C_{i−1}, C_{i+1}), subject to ^O T̂_{i−1},
   |  ^O T̂_{i+1}, ^i T̂_{i−1}, and ^{i+1} T̂_i;
11 |  if change of ^O T̂_i > tolerance_threshold then
12 |  |  Queue.push_back(^O T̂_{i−1}, ^O T̂_{i+1});
13 end
```

C. Multiple Closed Loops

If the target object is very tall so that one closed-loop of image taking cannot result in a complete model of its entire side surface, multiple loops of image taking can be conducted, depending on the need of the manipulation task. That is, the continuum robot arm can be used to wrap around the object in different routes, and each route will result in a partial object model. Once each partial object model is obtained from each closed loop and optmized, the next task is to register such partial models together. We still apply the ICP algorithm. The transformation matrix for changing the initial robot pose for different routes can be used as the initial estimate of the transformation for the ICP algorithm. Notice that there should be a large overlap between two partial object models, which makes the registration robust.

V. Experiments and Analyses

We have implemented and tested our approach of progressive object modeling in an augmented reality scenario, where a simulated continuum manipulator with a fixed base is situated in a sensed real environment with a real target object of unknown model and real surrounding unknown obstacles. A small and light-weight RGB-D camera (such as [35]) is assumed carried by the continuum manipulator (such as the OctArm) at its tip to sense the target object and the environment. All images of the target object are real images taken by a Microsoft Kinect camera, where the relative configuration of the camera with respect to the object (or vice versa) for each image is determined by planning the simulated robot.

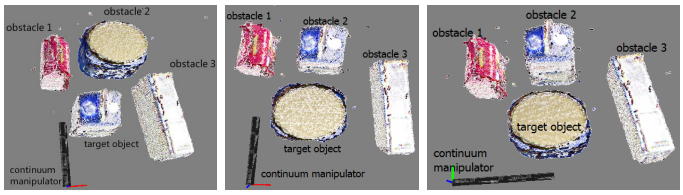
A. Progressive Object Modeling with a Continuum Arm

From each real RGB-D image of the target object, the corresponding 3D object point cloud is obtained[2] for object model building. On the other hand, sensing of the obstacles as the robot moves is simulated. We first obtain a real 3D point cloud for each obstacle offline, which is unknown to the motion planner of the continuum arm. Then, at each environment sensing step of the robot's operation, the camera is turned[3] to view the space between the target object and the obstacles, and the portion of each obstacle point cloud in the viewing frustum of the simulated camera is extracted for robot motion planning. The robot arm only needs to avoid the obstacle points during its motion without caring about differentiating the obstacles.

Fig. 6 shows two table-top task environments in overhead views *unknown* to the continuum robot manipulator, where a three-section simulated continuum robot is at its initial configuration in the sensed real world. The target object is a milk box and a coffee can in **Task 1** (Fig. 6(a)) and **Task 2** (Fig. 6(b) and (c)) respectively. In order to achieve the closed-loop coverage of the side surface in **Task 2**, the robot needs to move around the target object in two directions, first clockwise and then counter-clockwise, which we call subtasks **Task 2-1** (Fig. 6(b)) and **Task 2-2** (Fig. 6(c)) respectively.

[2]Since the tip camera is close to the target object, only the target object (no obstacles) appears in the real RGB-D image, which simplifies segmentation.

[3]The simulated camera at the tip of the continuum arm has an additional actuated degree-of-freedom to turn left and right.



(a) Task 1 in Env. 1 (b) Task 2-1 in Env. 2 (c) Task 2-2 in Env. 2

Fig. 6: Top view of environments with different target objects, obstacles, and robot arm initial configurations.

In **Task 1**, the robot arm is initially positioned outside the unknown cluttered space. With the initial model of the target object and the sensed obstacles, the robot arm starts moving onto the first planned collision-free arc (Fig. 7(b)). As more sensings are enabled, an updated arc is planned (Fig. 7(c)) for the robot arm to follow until a new arc is necessary (Fig. 7(d)). In the end, the robot arm is able to make a closed loop coverage of the side surface of the milkbox (Fig. 7(e)). The final refined model after global optimization is shown in Fig. 11(a). Note that the milk box is too tall for the tip camera to sense the top surface.

In **Task 2-1**, the arm gradually moves around the object from the concave side in clockwise direction (Fig. 8). As shown in Fig. 9(b), the arm initially follows the red arc, which is then updated to be the yellow arc when another sensing is made. Eventually, the arm is able to maneuver through the concave side by following the green arc. Due to the section length limit, the arm stops after it covers the concave side, resulting in an incomplete coverage of the side surface of the coffee can.

TABLE I: Total number of images captured, the short distance Δs used, and the total time for motion planning T_{mp}.

Task	# images	Δs(cm)	T_{mp}(ms)
1	11	3	20
2-1	5	6	23
2-2	7	6	31

To enable the tip camera to sense the coffee can from the other direction, the initial configuration of the arm is reset as displayed in **Task 2-2**. As shown in Fig. 10, the arm is able to observe the unmodeled part of the coffee can by moving in the counter-clockwise direction. By combining the images captured in **Task 2-1** and **Task 2-2**, the complete model (Fig. 11(b)) of the side surface is obtained, which also includes some portion of the top surface.

Table I shows the number of images captured, the short distance Δs used, and the total time for planning the robot arm motion in the experiments. Δs is set to be both sufficiently small to ensure a sufficient overlap between the images captured in two consecutive sensings and also large enough to be efficient. Choosing Δs as 3cm in Task 1 and 6cm in Task 2 provides roughly a 25° to 30° change of viewing angle between two consecutive images. The total time T_{mp} is the sum of the planning time for every arc for the arm to move along to capture all the images of the target object, which is very short and indicates that our algorithm is a real-time motion planning algorithm.

The most time-consuming process is image registration for model building. Depending on the size of the sensed point clouds and whether the ASIFT keypoint matching provides a good initial estimate for the ICP algorithm, the time cost for conducting one pairwise registration varies from 2 to 3 minutes. Refining the entire model using the global optimization algorithm is much faster and typically requires a total of 1 to 1.5 minutes.

The attached video shows the 3D arm motion of our presented experiments here.

B. Refining Models by Global Optimization

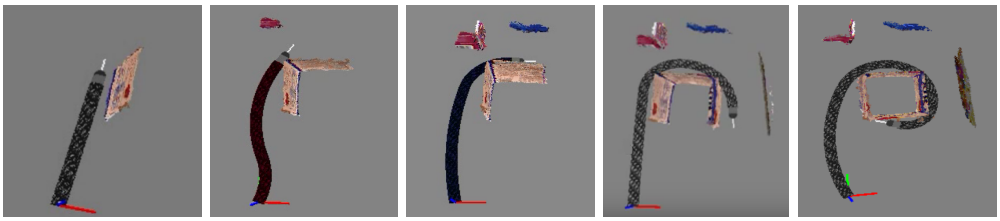
With an initial object model obtained after a sequence of pairwise registration, the model is further refined using the global optimization algorithm. Using the milk box as an example, we next explain how the model is improved both visually and statistically. As shown in Fig. 12, the initial model is subject to the artifact like the loose connection between the two surfaces as indicated in (a), which is corrected in the refined model (Fig. 12(b)).

Two statistical analyses are conducted to quantify the improvement of the model quality by applying the global optimization algorithm to reduce the accumulative error of the pairwise registration. In the first analysis, the mean distances of all the points to their K-nearest neighbors are computed for the models obtained before and after the global optimization. As shown in Fig. 13(a), the plotted curve is shifted to the left after the optimization, which indicates that the points are distributed closer to their neighbors. This reflects that for the same object points appearing in different images, the distances between their 3D positions obtained from those different images are further reduced in the model after optimization (with the ideal case being zero distance, see Fig. 12 again for example).

The second analysis is based on the number of neighbors of all the points, which are searched in the radius of $3mm$. As seen from Fig. 13(b), the plotted curve of the refined model is shifted to the right compared to the model before the optimization. Therefore, the points in the refined model have more neighbors within $3mm$ radius, which also shows that the points are located in the closer proximity of their neighbors.

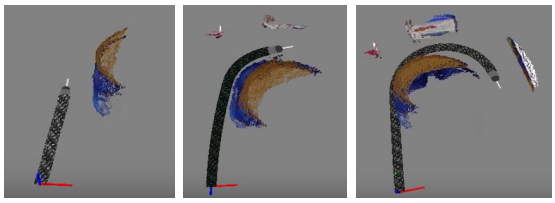
VI. Conclusions

This paper presents a general approach of progressive object modeling with a continuum manipulator in unknown and cluttered environments. By interleaving manipulation and perception, a continuum robot with a fixed base is able to gradually maneuver through the unknown space without colliding with the objects and sense the unmodeled target object from different viewpoints. The model of the target object is progressively built as the robot arm moves. The obtained model, which might be partial, is further refined using a global optimization algorithm after the closed-loop



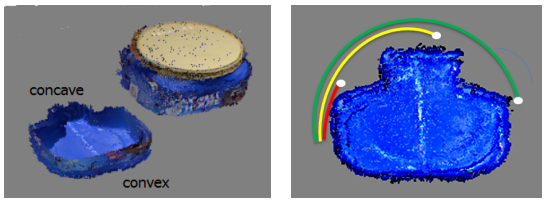
(a) Initial arm configuration. (b) The arm follows the planned arc. (c) The planned arc is updated. (d) The arm follows a new arc. (e) The arm finishes observing the object.

Fig. 7: Snapshots of the arm motion and the sensed object surfaces in **Task 1**.



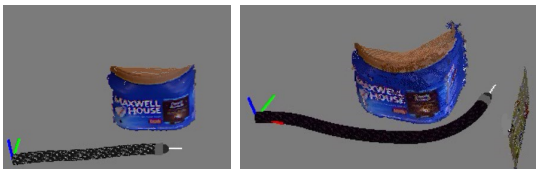
(a) Initial arm configuration. (b) The arm moves and senses. (c) The arm reaches its physical limit.

Fig. 8: Snapshots of the arm motion in **Task 2-1**.

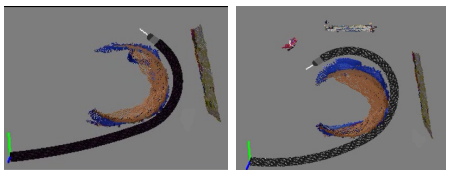


(a) Coffee can has both concave and convex sides. (b) Cross section top view. The arm sequentially follows the red, yellow and green arc.

Fig. 9: Illustration of the shape of the coffee can and the arm motion. The white dots in (b) correspond to three goal points the arm tip tries to reach in three consecutive sensings.

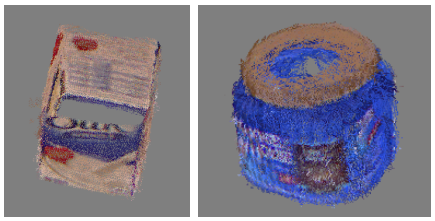


(a) Initial arm configuration. (b) The arm moves on the first planned arc.



(c) The arm moves on the second planned arc. (d) The arm moves on the third planned arc.

Fig. 10: Snapshots of the arm motion in **Task 2-2**.



(a) Milk box. (b) Coffee can.

Fig. 11: The built models.

coverage of the side surface is achieved. The experiments of modeling real objects from real RGB-D images taken based on the planned motion of the continuum robot demonstrate the effectiveness of the introduced approach, which can be applied to a real continuum robot.

One possible future extension is to consider a continuum robot with a mobile base. With a mobile base, the assumption (3) (Section II.A) can be relaxed as the robot can move forward with no limit. As the continuum robot needs to map the environment and localize its base at the same time, SLAM techniques [36] can be used. However, a mobile base usually requires larger space between the object and obstacles to move around.

ACKNOWLEDGMENT

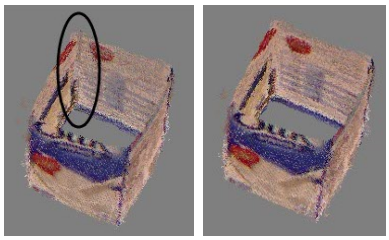
This work is supported by the US National Science Foundation (NSF) grant IIP-1439695 and by funding from the Electrical Power Research Institute under the NSF I/UCRC grant.

REFERENCES

[1] G. R. W. R. Scott and J. F. Rivest, "View planning for automated 3D object reconstruction inspection," *ACM Computing Surveys (CSUR)*, vol. 35, pp. 64–96, 2003.

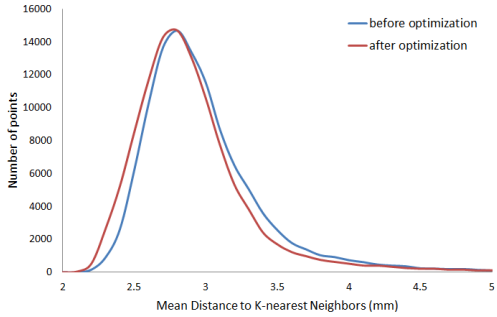
[2] J. Vasquez-Gomez, L. Sucar, R. Murrieta-Cid, and E. Lopez-Damian, "Volumetric next-best-view planning for 3D object reconstruction with positioning error," *International Journal of Advanced Robotic Systems*, vol. 11, pp. 1–13, 2014.

[3] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 127–136, 2011.

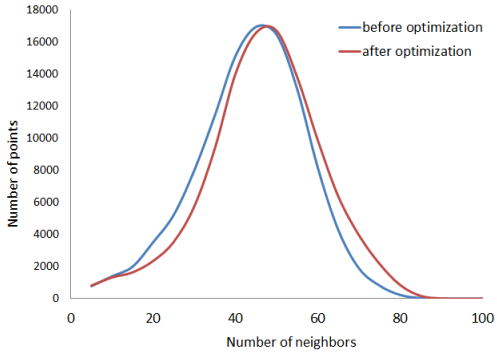


(a) Before (b) After

Fig. 12: Comparison of the models obtained before and after global optimization. As indicated in (a), the two surfaces circled in the model before global optimization are not tightly connected, which is corrected in (b).



(a) The mean distance to K-nearest neighbours of all the points ($K = 100$).



(b) The number of neighbors of all the points searched within $3mm$ radius.

Fig. 13: The statistical analyses of the models obtained before and after the global optimization.

[4] S. Ivaldi, N. Lyubova, A. Droniou, V. Padois, D. Filliat, P.-Y. Oudeyer, O. Sigaud, *et al.*, "Object learning through active exploration," *IEEE Transactions on Autonomous Mental Development*, 6(1):56-72, 2014.

[5] M. Bonilla, E. Farnioli, C. Piazza, M. Catalano, G. Grioli, M. Garabini, M. Gabiccini, and A. Bicchi, "Grasping with soft hands," in *IEEE International Conference on Humanoid Robots*, 2014.

[6] C. Eppner, R. Deimel, J. Álvarez-Ruiz, M. Maertens, and O. Brock, "Exploitation of environmental constraints in human and robotic grasping," *The International Journal of Robotics Research*, 2015.

[7] M. Krainin, B. Curless, and D. Fox, "Autonomous generation of complete 3d object models using next best view manipulation planning," in *International Conference on Robotics and Automation(ICRA)*, 2011.

[8] L. Ma, M. Ghafarianzadeh, D. Coleman, N. Correll, and G. Sibley, "Simultaneous localization, mapping, and manipulation for unsupervised object discovery," in *IEEE ICRA*, pp. 1344–1351, 2015.

[9] B. Browatzki, V. Tikhanoff, G. Metta, H. H. Bülthoff, and C. Wallraven, "Active in-hand object recognition on a humanoid robot," *IEEE Transactions on Robotics(TRO)*, 2014.

[10] F. Remondino, L. Barazzetti, F. Nex, M. Scaioni, and D. Sarazzi, "Uav photogrammetry for mapping and 3d modeling–current status and future perspectives," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38(1):C22,2011.

[11] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, "Autonomous, vision-based flight and live dense 3d mapping with a quadrotor micro aerial vehicle," *Journal of Field Robotics*, vol. 1, 2015.

[12] G. Robinson and J. B. C. Davies, "Continuum robots-a state of the art," in *IEEE ICRA*, vol. 4, pp. 2849–2854, 1999.

[13] 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, no. 3, pp. 99–117, 2008.

[14] R. J. Webster III, J. M. Romano, and N. J. Cowan, "Mechanics of precurved-tube continuum robots," *IEEE TRO*, 2009.

[15] W. McMahan, V. Chitrakaran, M. Csencsits, D. Dawson, I. D. Walker, B. A. Jones, M. Pritts, D. Dienno, M. Grissom, and C. D. Rahn, "Field trials and testing of the octarm continuum manipulator," in *IEEE ICRA*, 2006.

[16] W. McMahan and I. D. Walker, "Octopus-inspired grasp-synergies for continuum manipulators," in *IEEE International Conference on Robotics and Biomimetics*, pp. 945–950, 2009.

[17] K. Trovato and A. Popovic, "Collision-free 6d non-holonomic planning for nested cannulas," in *SPIE Medical Imaging*, pp. 72612H–72612H, International Society for Optics and Photonics, 2009.

[18] L. G. Torres, C. Baykal, and R. Alterovitz, "Interactive-rate motion planning for concentric tube robots," in *IEEE ICRA*, 2014.

[19] J. Li and J. Xiao, "A general formulation and approach to constrained, continuum manipulation," *Advanced Robotics*, 29(13):889-899,2015.

[20] J. Li and J. Xiao, "Progressive planning of continuum grasping in cluttered space," *IEEE TRO*, 2016.

[21] J. Li, Z. Teng, J. Xiao, A. Kapadia, A. Bartow, and I. Walker, "Autonomous continuum grasping," in *IEEE International Conference on Intelligent Robots and Systems(IROS)*, pp. 4569–4576, 2013.

[22] J. Li, Z. Teng, and J. Xiao, "Can a continuum manipulator fetch an object in an unknown cluttered space?," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 2–9, 2017.

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

[24] P. E. Dupont, J. Lock, B. Itkowitz, and E. Butler, "Design and control of concentric-tube robots," *IEEE TRO*, 2010.

[25] R. Kang, D. T. Branson, T. Zheng, E. Guglielmino, and D. G. Caldwell, "Design, modeling and control of a pneumatically actuated manipulator inspired by biological continuum structures," *Bioinspiration & biomimetics*, vol. 8, no. 3, p. 036008, 2013.

[26] S. Tully, G. Kantor, M. A. Zenati, and H. Choset, "Shape estimation for image-guided surgery with a highly articulated snake robot," in *IEEE IROS*, pp. 1353–1358, 2011.

[27] J. Li and J. Xiao, "An efficient algorithm for real time collision detection involving a continuum manipulator with multiple uniform-curvature sections," *Robotica*, vol. 34, no. 07, pp. 1566–1586, 2016.

[28] C. Frazelle, A. Kapadia, and I. Walker, "Teleoperation of planar extensible continuum robots utilizing nonlinear control," in *submission to American Control Conference*, 2017.

[29] G. Yu and J. M. Morel, "ASIFT: An algorithm for fully affine invariant comparison," *Image Processing On Line*, 2011.

[30] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, 1981.

[31] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *IEEE ICRA*, pp. 3212–3217, 2009.

[32] D. W. Eggert, A. Lorusso, and R. B. Fisher, "Estimating 3-D rigid body transformations: A comparison of four major algorithms," *Machine Vision and Application*, vol. 9, pp. 272–290, 1997.

[33] P. J. Besl and H. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 239–256, 1992.

[34] K. Pulli, "Multiview registration for large data sets," in *1999. Proceedings. Second International Conferences on 3-D Digital Imaging and Modeling*, pp. 160–168, 1999.

[35] Heptagon, "Mora: World smallest all-in-one 3D module." `http://hptg.com/product/#imaging`. [Online; accessed 1-Dec-2016].

[36] J. Aulinas, Y. R. Petillot, J. Salvi, and X. Lladó, "The slam problem: a survey.," in *CCIA*, pp. 363–371, Citeseer, 2008.