Shape-based Object Classification and Recognition through Continuum Manipulation

Huitan Mao* and Jing Xiao
Department of Computer Science
University of North Carolina at Charlotte

Mabel M. Zhang* and Kostas Daniilidis
GRASP Laboratory
University of Pennsylvania

***Abstract*— We introduce a novel approach to shape-based object classification and recognition through the use of a continuum manipulator. Noticing the fact that when a continuum manipulator wraps around an object in a whole-arm grasping, its own shape is indicative of the shape of the object, our approach enables learning and recognition of object classes based on the shapes of continuum wraps. It offers the following advantages: (1) recognition of objects that are not easily detected by vision, such as transparent objects, and (2) highly efficient recognition of such objects of varied sizes due to high-level and rich shape information in each wrap, unlike recognition based on tactile sensing via conventional grasping. Simulation and experiments demonstrate the effectiveness of our approach.**

I. Introduction

Object classification and recognition is a key requirement for many autonomous robot systems. Over the years, object classification has seen major progress from approaches mainly relying on visual and depth information of object appearance. However, vision-based approaches can be rendered ineffective due to poor illumination, transparent object surfaces, and heavy occlusion. Tactile sensing is used as a complementary means to object recognition.

In this paper, we address the problem of automatic object recognition based on shape information obtained with a continuum manipulator guided by tactile and vision sensing. Humans and animals often rely on touching an object and exploring its shape to recognize it when vision cannot be effective. There exists research on detecting and identifying objects through grasping with tactile sensors attached to a robotic hand or gripper, but it usually requires a lot of grasps to capture the shape information of an object [1]. With a continuum manipulator, whole-arm grasping or wrapping of an object can be conducted to capture more contact points at once, and when the continuum manipulator wraps around an object, its own shape, being compliant to the shape of the object, is indicative of the shape of the object.

Hence, our study is focused on using continuum robot manipulation for shape-based object modeling and recognition, which has not been studied before. We introduce a strategy to enable a continuum manipulator wrap around a target object based on sensed contact points between the continuum manipulator and the object (Fig. 1), interleaving contact sensing and manipulator motion planning and execution.

*H. Mao and M. M. Zhang contributed equally to this work.

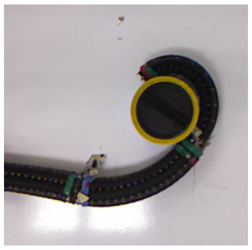


Fig. 1: A continuum robot wraps a cylindrical object.

Once a whole-arm wrapping of the target object is achieved by the continuum manipulator, the shape of the manipulator is captured and encoded as a shape feature of the object. That is, our strategy uses the continuum arm as a tool to “measure” the object shape. Such a strategy has the advantage of capturing shapes of objects that are hardly visible, such as transparent objects, as long as the continuum manipulator itself is visible. We further present an algorithm to systematically generate whole-arm wraps of objects and train a support vector machine (SVM) for classification of object categories based on the shape information of the wraps.

For object recognition, we introduce an active algorithm of conducting wrappings of a target object selectively by maximizing the probability of recognition and minimizing the movement cost of the continuum robot. Simulation and real experiments demonstrate the effectiveness of our approach.

Section II reviews literature related to continuum manipulation and sensing for object modeling and recognition. Section III describes the research objective and the problem. Section IV describes the proposed methods. Simulation and real robot experiment results are presented in Section V. Section VI concludes the paper.

II. Related Work

We provide a review of related work in continuum manipulation and shape-based object classification and recognition.

Due to the inherent compliance in a continuum manipulator [2]–[6], it can wrap around an object in a whole-arm grasp and deform its shape to comply to the object shape. There exist several autonomous algorithms of generating continuum graspings [7]–[9] and conducting task-constrained inspection tasks [10] of known objects in known environments. More recently, the study in [11] analytically formulates the constraints that have to be satisfied to fetch an object in an unknown cluttered environment perceived through an RGBD camera on the tip of the continuum manipulator, and in [12], an approach is introduced to model an unknown object automatically on-site with an RGBD sensor carried by a continuum manipulator in a cluttered environment.

Traditionally, object shape has been recognized by vision. The closest related work to this paper from the vision literature is [13], which used chords within object contours to characterize the shape of a 2D object. For the purpose of manipulation, work has been done in object detection for grasping [14], grasping by contour [15], grasping unknown objects by shape [16], [17], object classification from single grasps [18], and detecting grasping points on novel objects [19]. A survey explores various vision-based methods for grasping [20].

Touch sensing has been shown to be effective in exploring the shape of an object. There is work on guiding compliant motion of a robot finger or end-effector by tracking tactile features [21], [22] and for grasping unknown objects [23]. There is also work on recognition and reconstruction of curved surface patches through touch [24], [25].

In order to ensure a complete coverage of a target area by touch, a grid is often used to enumerate the poses for a robotic hand equipped with tactile sensors to visit the target area [1], [26], [27]. However, these methods do not take advantage of the adjacency information of contacts.

In [28], dynamic potential fields are used to guide touch-based exploration of an object to build an object model as a contact point cloud. Initialized as a uniformly attractive grid, the field is updated as the sensed contacts increase and generate repulsive forces to drive the robotic hand to explore unvisited areas.

To deal with the noise and non-uniform distribution of tactile data, filtering based on Gaussian Process (GP) [29] is effective in active tactile exploration to reject any measurements that do not reduce the uncertainty significantly. Originated from GP, a probabilistic model of uncertainty based on Gaussian Process Implicit Surfaces (GPIS) [30] is used in [31]–[33] to guide the haptic exploration towards high uncertainty.

In contrast to greedy exploration approaches, [34] used a lookahead policy to predict a sequence of actions optimal for high recognition certainty and low cost in a few future steps. It used a triangle histogram descriptor [1] for tactile recognition.

III. Research Objective and Problem Formulation

Our goal is to recognize the category of an object by conducting just a few continuum wraps around the object. For training, we systematically let the continuum arm wrap around each object in many different ways, and we use a chord histogram to describe the shape of the continuum arm wrapping around the object each time. For testing, given an object of unknown category, we actively select continuum wraps to collect its shape information until the object category can be recognized with high confidence.

We consider a continuum arm with a fixed base and n-sections. Each section $sec_i, i = 1, \dots, n$ is a circular arc when intact, which can be described by three controllable variables: length s_i, curvature κ_i, and orientation ϕ_i. Each section is bounded by its base point and end point. A frame is attached to the base point of sec_i with the z axis tangential to sec_i as illustrated in Fig. 2. The *arm configuration* of an n-section continuum manipulator can be represented as $\{(s_1, \kappa_1, \phi_1), ..., (s_n, \kappa_n, \phi_n)\}$ [4], [9].

We assume that the continuum arm is covered by tactile sensors to detect the contact made with an object. A contact region is denoted as $contact = \{\mathbf{p}, \mathbf{n}, \mathbf{t}, \mathbf{b}\}$, where $\mathbf{p}$ is its center position, $\mathbf{n}$, $\mathbf{t}$, and $\mathbf{b}$ are the normal, tangential, and binormal unit vectors respectively. In simulation, contacts between a continuum manipulator and an object represented in polygonal mesh can be efficiently detected using the algorithm presented in [35].

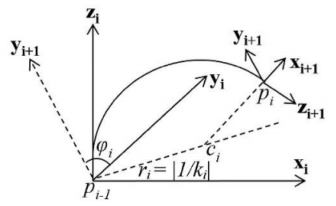


Fig. 2: Illustration of the frame of sec_i and the control variables s_i, κ_i and ϕ_i [11].

The target objects considered in this paper are mostly rigid objects (represented in meshes) with sizes that can be partially wrapped around by the continuum manipulator with a fixed base. However, the approach can be extended to a continuum manipulator with a mobile base.

IV. Methodology

In this section, we present our approaches in detail.

A. Touch-driven Whole Arm Wrapping

We use a progressive strategy to generate a whole arm wrap of an unknown object based on touch. The arm is initially placed near the target unknown object in a straight-line configuration (see Fig. 6 for an example). This strategy then alternates between generating two types of motion in small steps: **enclosing** and **advancing**, to start and gradually deepen the wrap around the object until the arm conforms to the shape of the object sufficiently. Our algorithm is outlined in **Algorithm** 1.

For **enclosing**, the arm tries to make contact with the target object by curving its section(s). First, sec_1 is curved through increasing its curvature κ_1 by a small amount $\Delta\kappa$ (if sec_1 is not at its curvature limit). As the result, if the arm is in contact with the object at d locations, the set of contact regions is saved as $C = \{contact_1, \dots, contact_d\}$. Next, if sec_i $(1 < i \leq n)$ is the arm section immediately after the current furthest contact (i.e., closer to the arm tip than the furthest contact), sec_i is also curved in order to form more contacts until it is either stopped by a contact or reaches its curvature limit, and the process repeats until either sec_n is in contact or reaches its curvature limit. Note that in curving a section, the section length s is also increased by a small amount Δs to reach further as the curvature κ is increased. By curving and extending the arm sections, the arm closes in upon the object until it contacts the object as much as possible.

Next, for **advancing** the arm, our algorithm finds a new arm configuration that moves the robot a small step forward. It first finds new positions for the endpoints of the arm sections using the following strategy. For all the m contacts that happen on sec_n (last section), the position of the sec_n's endpoint (arm tip) is extrapolated m times along the direction of $(\mathbf{n_i} + \mathbf{t_i} + \mathbf{b_i}), i = 1, \ldots, m$ by a small distance δ each time. The obtained new position will enable the robot arm move forward to facilitate the further motion of enclosing. Similarly, we use the contact points that happen on other sections of the arm to extrapolate the positions of the section endpoints closest to them. If there are endpoints whose positions are not modified based on the contact information, their new positions are obtained by moving the current endpoint position by a small amount δ along the tangential direction of the section arc at this endpoint, i.e., along the z-axis of the local frame at the endpoint. Note that always using the contacts on sec_n to obtain the new position of the arm tip prevents the arm tip from penetrating into the object, which could happen if the tip's position is simply extended along the tip's z-axis.

Once the new positions of all section endpoints are obtained, the corresponding new arm configuration can be solved by constrained inverse kinematics [10]. The robot is then moved to the new arm configuration, preparing itself for the next **enclosing** motion step.

Algorithm 1 terminates when the z-axis of the tip frame has rotated more than a threshold θ from the initial direction $\mathbf{z}_s$ to its current direction or when it fails to solve the whole arm configuration due to the robot's physical limits. θ can be used to control how much coverage of the object surface is needed in a wrap of the object.

Algorithm 1: Touch-driven Whole Arm Wrapping

```
1  $tipRotatedAngle = 0$;
2  while $tipRotatedAngle < \theta$ do
3      $Contacts = \emptyset$; $\mathbf{P}_{new} = \emptyset$; $i = 1$; $k = 0$;
       // enclosing motion step
4      repeat
5          if $sec_i$ is not at curvature limit then
6              Curve $sec_i$ by increasing $\kappa_i$ by $\Delta\kappa$;
7          end
8          else if $i < n$ then
9              $i = i + 1$;
10         end
11         if $sec_i$ is not at length limit then
12             Extend $sec_i$ by increasing $s_i$ by $\Delta s$;
13         end
14         if arm is in contact with the object at
             $C = \{contact_1, \ldots, contact_d\}$ then
15             $Contacts \leftarrow Contacts \cup C$;
16             if $sec_j$ is the closest in-contact sec to the
                 tip and $j < n$ then
17                 $i = j + 1$;
18             end
19         end
20     until $sec_n$ is in contact with the object or reaches
         its curvature limit;
       // advancing motion step
21     while $k < |Contacts|$ do
22         if $contact_k$ is on $sec_n$ then
23             $m \leftarrow n$;
24         end
25         else
26             $m \leftarrow$ index of the section endpoint closest
                 to $contact_k$;
27         end
28         $\mathbf{P} \leftarrow$ the position of the $i$th endpoint;
29         $\mathbf{P}_m \leftarrow \mathbf{P} + \delta(\mathbf{n_k} + \mathbf{t_k} + \mathbf{b_k})$;
30         $\mathbf{P}_{new} \leftarrow \mathbf{P}_{new} \cup \mathbf{P}_m$; $k = k + 1$;
31     end
32     for each remaining section endpoint do
33         $m \leftarrow$ its section index;
34         $\mathbf{P} \leftarrow$ its current position;
35         $\mathbf{z} \leftarrow$ the $z$ axis at the endpoint; $\mathbf{P}_m \leftarrow \mathbf{P} + \delta\mathbf{z}$ ;
36         $\mathbf{P}_{new} \leftarrow \mathbf{P}_{new} \cup \mathbf{P}_m$;
37     end
38     $armConfig = constrainedIK(\mathbf{P}_{new})$;
39     if Failed to find a valid $armConfig$ then
40         Set Flag "No IK found" and exit;
41     end
42     else
43         Move the arm to $armConfig$;
44         $tipRotatedAngle \leftarrow$ the angle from the initial
             $\mathbf{z}_s$ to the current $\mathbf{z}$ axis at the tip;
45     end
46 end
```

B. Chord Histogram Descriptor

We use a chord histogram to characterize the shape of the continuum manipulator when it wraps around an object in some way. The chord histogram we propose is inspired by chordiograms [13] used for object recognition in 2D images. Chordiogram is a histogram computed from geometric relationships described by chords, which are segments connecting point pairs on the contour of an object. It had been extended to 3D tactile recognition in [1], which used 3D triangles instead of planar 2D chords – the disadvantage is that the surface normals are dropped from the original chordiogram. However, without normals, the lengths and angles used in [1] can only capture the size and shape, and not surface curvature. As we believe that the curvature information is an important feature, we extend the 2D chordiograms into 3D Chord historgrams in this paper.

In our 3D extension, we use the length, endpoint angles, and endpoint surface normals of the 3D chords. We parameterize each chord as $(l, c_\theta, c_\phi, n_{0\theta}, n_{0\phi}, n_{1\theta}, n_{1\phi})$. l is the length of a chord. c, n_0, n_1 are the chord and the normals at its two endpoints. (θ, ϕ) parameterizes an angle on a sphere, represented as polar coordinates for the fewest parameters.

At each wrap, a set of points are sampled along the medial axis of the continuum arm contacting the object, and each pair of points makes a chord. Fig. 3 shows chords collected on example wraps. The sampling density of points cannot be too low to lose information of the arm shape and cannot be too high to introduce redundancy in the chords. In practice, we filter the chords that are too similar to other chords.

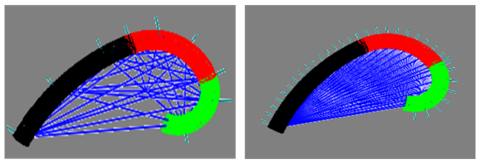

(a) Too few chords. (b) Too many chords.

Fig. 3: Chords collected on example wraps are shown in blue. The normals at the sampled points are shown in green.

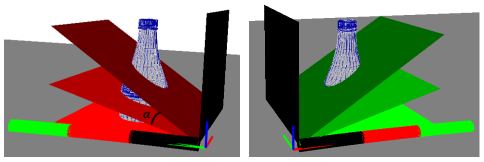
After an object is wrapped around several times, the sets of chords collected are binned into a 7D histogram. Each dimension of the histogram represents a parameter. Principal Component Analysis (PCA) is applied to the data to reduce redundancy.

C. Shape-based Classification of Object Classes

We train a support vector machine (SVM) [36] to classify object classes based on the shapes of the continuum robot when it wraps around each object, represented as chord histograms.

During training, each object is placed in the workspace of the arm. A number of touch-driven whole arm wraps are conducted to capture the object shape. Each of them is a wrap on a certain plane, and it captures the shape of the object cross section cut by this plane. Different planes for wrapping are systematically enumerated in the robot workspace to capture the object shape as completely as necessary. Each plane is determined by rotating the tabletop plane by α angle about the x-axis and y-axis respectively (Fig. 4). Angle α can be used to determine how densely these planes are enumerated. Note that a wrap is only generated if the cross section of the object shape cut by the plane is not empty.

At the end of each wrap, a set of chords formed by pairs of points on the arm is recorded as an *observation*, later used for active guidance of object recognition. After all wraps are conducted on an object, the chords are used to obtain the chord histogram, one per object.



(a) The robot wraps in clockwise directions to capture one side of the object geometry. (b) The robot wraps in counter-clockwise directions to capture the other side of the object geometry.

Fig. 4: A few planes for a bottle are shown, which are generated by rotating from the tabletop plane every α angle about x-axis (subfigure (a)) or y-axis (subfigure (b)) until α reaches 90 degs(black planes). These planes all pass through the robot base frame since the robot base is fixed.

D. Object Recognition with Active Guidance

Given a new object, object recognition can be conducted by having the continuum robot wrap around the object to obtain chord histograms and supplying the chord histograms to the trained SVM classifier. One important question is how to conduct the wrapping and how many wraps are needed for recognition. We propose a strategy to actively guide the process of determining wraps of the object for recognition.

1) Active Formulation: The core idea is that observations of an object's shape and the actions taken to wrap around the object are related. In contrast to popular greedy approaches, we choose to optimize an objective function that trades off between minimum movement cost and maximum recognition certainty (Eqn. 1). A similar formulation has been used for a different descriptor and robot platform [34].

The difference of our approach from information gain approaches is that the latter is greedy and optimizes the best short-term move, which can miss beneficial features that may not be immediately rewarding. For example, a feature that is the most discriminating in the short term can be a strenuous pose for the robot, costing a large movement cost before and after. Compare it with a feature that is the second-most discriminating but minimal-cost movement. A greedy approach would select the first feature, while our approach would prefer the second.

We formulate the problem as a Markov Decision Process (MDP), which searches for a policy π, or a sequence of actions, that minimizes the objective:

$$\min_\pi C_T(\pi) \triangleq \frac{\lambda}{T}\mathbb{E}\left[\sum_{t=0}^{T-1} c_m(\pi(x_t))\right] + (1-\lambda)\mathbb{P}(\hat{y}_T \neq y) \quad (1)$$

The first term is the movement cost, and the second is the cost for incorrect recognition. Minimizing both trades off between the two terms. In the first term, c_m is the movement cost, calculated on an action returned by some policy π. x_t is the current state, which we define as the chord histogram h_t computed from all observations so far up to time t. T is the total number of time steps.

In the second term, $\hat{y}_T$ is the predicted class, y is the true class. $\mathbb{P}(\hat{y}_T \neq y)$ is the probability of misclassification, or one minus the confidence in the predicted class, which is simply the maximum probability from a classifier.

2) Test Stage: The observations of all objects from training (Section IV-C) are loaded. The chords are discretized to allow for repeatability across different objects, thereby creating a probability distribution. The discretization step is each histogram bin center, which exists for all objects. This discretized observation z is later used to compute the probability relating two consecutive observations to an action, which is defined as a movement between two wraps.

To find the policy in the objective function (1), we use the Monte Carlo tree search [37]. Tree search is a lookahead policy, which in contrast to greedy policies, explicitly formulates future information and future cost [38]. The Monte Carlo approximation enables practical lookahead search time, which would otherwise be expensive.

A tree representing an MDP is similar to a Markov chain, with actions added to the edges. The purpose of a tree search is to generate the nodes and edges of a tree using known data, and finding an optimal sequence of edges from the root to a leaf.

We define our tree with a node representing each state x_t, and an edge representing each action a_{t+1}. Since x_t is a histogram, directly estimating the state space would be high dimensional and slow. Instead, we only estimate and store the raw data, observation z_t, which is the collection of chords that compose the histogram, at each node. In addition, z_t is directly sampled from training data, eliminating the need to model it in a high-dimensional (7D chords) space. To do this, we will formulate a probability independent of the state and only depends on the observations and actions.

Action a is defined as a movement between two cross section planes for wraps. A plane can be represented by a unit vector normal to the plane, and the movement to change one plane to the other can be represented by the rotational transformation between the two unit vectors, which defines the movement cost in Eqn. (1). h_t is the chord histogram computed from observations $z_{1:t}$ up to time t. h_0 is initialized to empty.

A state has multiple possible outgoing actions, each associated with a reward C_{t_a}. The root is at time step $t = 0$ and leaves are at $t = T$ (a defined horizon). During the tree search, each simulation starts at the root, and at each node, an action edge and child node must be chosen (and created if they do not yet exist) to continue downward traversal until the horizon, where reward is computed and backpropagated.

At each node, the next best action a_{t+1}, i.e., the move to the next best wrap, is chosen by the upper confidence bound for trees (UCT) [39], which addresses the exploration-exploitation dilemma of which action to visit when multiple actions are available:

$$a_{t+1} = \arg\max_a \left((1 - C_{t_a}) + c\sqrt{\frac{2 \ln N}{N_a}} \right) \tag{2}$$

where C_{t_a} is the objective cost in Eqn. (1) computed in a previous simulation and stored in the node, N is the number of visits to a tree node, N_a is the number of visits to the outgoing action a from the node, and c is a constant to balance between the exploration and exploitation terms.

After the next action a_{t+1} is chosen, it is created as a child edge, leading to a child node. The child node is created by selecting the next observation z_{t+1} that follows from the next action, given the observation z_t at the current node.

We define the core probability that relates two consecutive observations and the action between them as $p(z_{t+1}|z_t, a_{t+1}, y)$. It is computed by sampling directly from the tallies stored during training stage. Each pair of wraps in training yield an action a, and the observations at the two wraps are assigned as z_t and z_{t+1}. Given n wraps, n^2 such relationships can be defined.

To select the observation z_{t+1} for the next node, we sample from the core probability, marginalized over class y as the true y is unknown:

$$z_{t+1} \sim p(z_{t+1}|z_t, a_{t+1}) = \sum_y p(z_{t+1}|z_t, a_{t+1}, y) p(y|h_t) \tag{3}$$

In each simulation of the action selection process, the traversal starts at the root, and the node- and edge-selection process is repeated until a given horizon tree depth. When the traversal reaches a dead end with no more child nodes before the horizon, a new node is created using the a_{t+1} and z_{t+1} selection process. One new node is created in each simulation. The remaining path from the new node to the horizon is traversed by a rollout policy, which randomly chooses an action a at each node and does not store the new nodes.

At the end of a simulation, the reward $1 - C_{t_a}$ is calculated at the leaf at the horizon and backpropagated up to the root of the tree. The backpropagation is done in the form of discounted rewards at each node, to keep each node updated with the newly seen edges and their reward yields. Each node's rewards account for the rewards in all the nodes on its child subpath. Backpropagation and discounted rewards are standard parts of policy search in MDP and reinforcement learning, and we refer the reader to [37].

For each tree search, we define a number of simulations. The more simulations, the more thoroughly the tree is searched, and the more bushy. At the end of all simulations, we look up the optimal path from the root to the leaf, by taking the maximum-reward action edge at each node. We call each tree search an *iteration*, as multiple iterations may be needed.

V. Experiments

We implemented our algorithms in C++ and Python under ROS on a 3.4GHz CPU, and tested them on a 3-section continuum manipulator. We use the dataset from [1]. In total, there are 185 objects from 10 categories: 12 apples, 6 bananas, 51 bottles, 21 bowls, 10 cups, 10 donuts, 28 hammers, 32 mugs, 6 teapots and 9 toilet paper rolls. Each object has the same relative pose to the robot base in the training and testing stage.

A. Capturing Object Shape

Table I shows the average number of wraps and chords conducted per object in each category for training the classifier. Each object requires only 10 or fewer wraps, and depending on the object dimension, there can be $1,500$ to $3,000$ chords to describe the shapes of the wraps of each object in our dataset. Some objects, which are flat (such as donuts), typically need fewer wraps, while objects that are tall, such as hammers, need more wraps. We use a bin size 7 in constructing the 7D chordiogram. PCA extracted 101 principal components out of 7^7 to cover 95% of variance, which further reduces the redundancy in the chordiogram and speeds up the computation. The time for each wrapping motion, which involves planning and collision checking for detecting contacts, is typically about a few hundreds ms.

In contrast, using a conventional robotic hand to capture the shape of an object through touch requires $364 - 760$ wrist poses per object [1] and an average of 20 mins to perform grasping from those poses to collect contact points in physical simulation, not including the time necessary for grasp planning to enable a manipulator reach those desired wrist poses around each object. Clearly, the introduced novel method with a continuum manipulator is far more efficient in capturing shape information of an object due to the rich information content of each wrap and the efficient planning algorithm for wrapping. The classifier training time is significantly smaller with our approach.

Fig. 5 shows some wraps achieved using **Algorithm 1** to capture the object shape in different cross sections. For some objects, such as apples and bowls, the wraps are closely conformed to the object contour, and for some others, the wraps also successfully encode the shapes of critical part information (such as the handle in cups, mugs and teapots). Fig. 6 shows the snapshots of a few example robot motions from the initial configuration to the final wrapping configuration. Each row shows a wrap. Note that in the last row, the wrap actually stopped inside the mug, which may capture that the mug is hollow. The parameters in **Algorithm 1** used are: $\theta = 270$ degrees, $\Delta k = 1e-3$ (1/cm) , $\Delta s = 1e-4$ cm, $\delta = 0.1$ cm. Animated robot motion can also be found in the attached video.

TABLE I: The average number of chords and wraps used per object in each category for training the classifier, and the average time T_{wrap} per wrap for motion planning + collision detection.

Category	average # chords	average # wraps	T_{wrap}(ms)
apple	2676	10	400
banana	1816	9.7	190
bottle	1760	10	130
bowl	2090	9.6	230
cup	2317	10	540
donut	1365	5.1	200
hammer	1711	10	110
mug	2557	10	530
teapot	2350	9.8	1200
toilet paper	2944	10	260

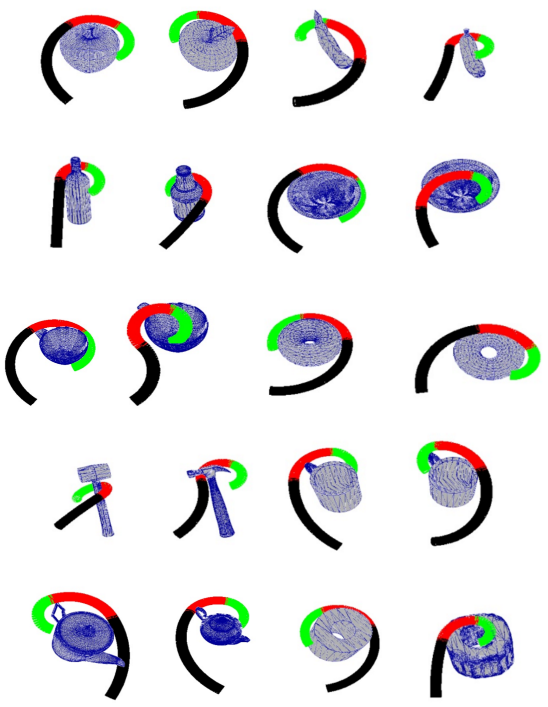


Fig. 5: Example wraps used to encode object cross section shape into the arm shape. Each category has two example wraps (from top left to lower right): apple, banana, bottle, bowl, cup, donut, hammer, mug, teapot, toilet paper rolls.

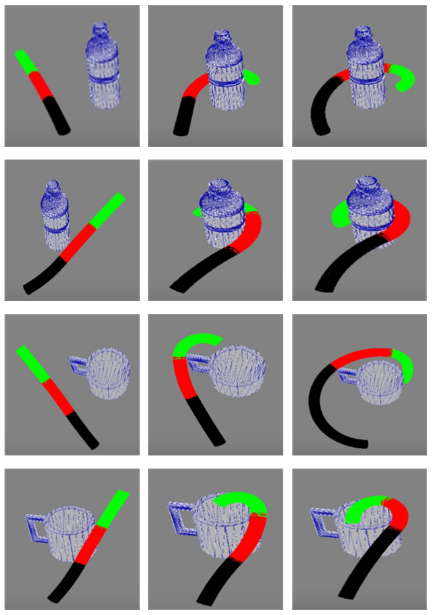


Fig. 6: Snapshots of example motions of wrapping. Each row shows one wrap from the initial configuration (left sub-figure) to the final wrapping configuration (right sub-figure).

B. Classification Performance

The average classification accuracy across 100 random splits of 50% training and 50% testing set is 75.9% using a linear SVM. The accuracy is slightly higher than the 74.7% reported in [1], where a triangle histogram is built from many contacts sampled on the object using a conventional robotic hand. As shown from the confusion matrix in Fig. 7, the classifier does well on classifying apples, bottles, bowls, donuts, hammers, and mugs. However, it does poorly on bananas, cups, teapots and toilet paper rolls, which is likely due to the lack of enough training objects in those categories.

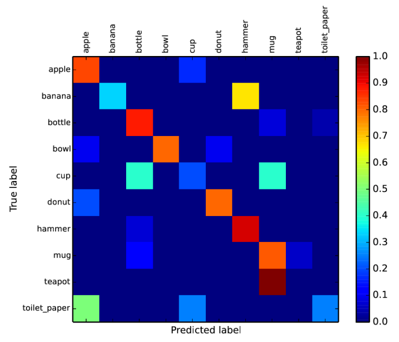


Fig. 7: SVM confusion matrix.

C. Recognition with Active Guidance

Five objects (a bottle, a hammer, a cup, a bowl, and a mug) are used to test our active recognition algorithm. The results are summarized in Table II. Since we used a maximum of 10 wraps per object during training, we set the horizon to be 5 at testing. The top 3 predictions sorted by their probabilities are reported.

For each object, each iteration is a new tree search and outputs a sequence of wrapping planes for the robot. After the robot conducts wrapping of the object along each of the plane, the chords describing the shape of the arm in the final wrap are collected. A histogram is built with all the chords collected and fed to the classifier to make the prediction. If the prediction is strong, i.e., the probability for the predicted category is more than 0.5 and at least 3 times higher than the probability of the next category, the recognition process is terminated. Otherwise, a new tree search is conducted in the next iteration. The search results guide the robot to conduct additional wraps, and the corresponding new chords are added to the existing histogram to make a new prediction, i.e., the histogram is accumulated in iterations. This process is repeated until a strong prediction is made.

As the histogram becomes more filled in with more iterations, the recognition probability also increases. For objects that the classifier does well, such as hammers and bowls, they may be recognized correctly as soon as iteration 1. The results from 3 iterations for all the tested objects are presented.

Note that a tree with depth 5 can output 5 wrapping planes to be used by the robot at most. We filter the duplicate planes as they do not provide new shape information. In any iteration, if all the wrapping planes found have already been used in the previous iterations, we put a 0 as the number of wraps, as shown in iteration 3 for object mug.

D. Real Robot Wraps

We tested using the real OctArm robot to wrap two real objects, one cylindrical and one rectangular, by teleoperation [6], and captured the wraps using a Microsoft Kinect. We also save the arm configurations of the wraps. Fig. 8 compares the image of a wrap for each object by the real robot to the corresponding simulated version at the same arm configuration. Even though each real wrap is slightly deformed upon contacting the object, the similarity between the real wrap and the simulated wrap, which does not consider deformation, is still very high. This means that the shape descriptor of the simulated wrap is very similar to that of the real one in capturing the real object shape, which indicates the real-world feasibility of the introduced method for object recognition based on continuum arm wraps.

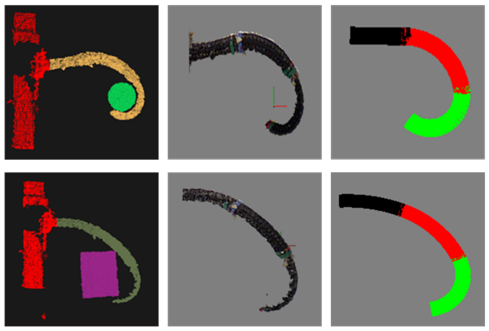


Fig. 8: Comparing real wraps and the corresponding simulated wraps around a cylindrical object (top) and a rectangular object (bottom) respectively. In each row from left to right: the segmented RGBD point cloud of the OctArm wrapping the object, segmented OctArm shape, reconstructed OctArm shape in simulation.

VI. Conclusion

In this paper, we present a shape-based object classification and recognition approach through continuum manipulation. The main idea is that the shape of an object can be effectively and efficiently captured by the shapes of a continuum manipulator wrapping around the object. A real-time, progressive touch-based motion planning algorithm enables a continuum manipulator to wrap around an object based on tactile sensing. The shapes of different continuum wraps around different objects are used to train an classifier of object categories very efficiently, and the effectiveness has been tested with 185 objects of 10 categories. An algorithm for active guidance of object recognition allows an object to be recognized with just a few continuum wraps. As the next step, we plan to conduct more real experiments to further verify the approach.

TABLE II: Active recognition performance (horizon=5).

Object	iteration1			iteration2			iteration3		
	prediction	probability	# wraps	prediction	probability	# wraps	prediction	probability	# wraps
bottle	bottle	0.39	3	bottle	0.42	1	bottle	0.7	2
	mug	0.19		mug	0.14		mug	0.05	
	banana	0.08		banana	0.1		teapot	0.04	
hammer	hammer	0.77	2	hammer	0.81	2	hammer	0.83	1
	bottle	0.04		bottle	0.03		banana	0.03	
	cup	0.03		banana	0.03		bottle	0.02	
cup	cup	0.51	3	cup	0.56	2	cup	0.57	3
	mug	0.09		mug	0.11		mug	0.12	
	bowl	0.08		bowl	0.06		bowl	0.06	
bowl	bowl	0.52	5	bowl	0.65	1	bowl	0.75	2
	apple	0.12		toilet paper	0.09		toilet paper	0.05	
	toilet paper	0.12		apple	0.07		apple	0.04	
mug	mug	0.46	3	mug	0.69	2	mug	0.69	0
	apple	0.14		cup	0.08		cup	0.08	
	cup	0.12		teapot	0.07		teapot	0.07	

ACKNOWLEDGMENT

We would like to thank Chase Frazelle, Apoorva Kapadia and Ian Walker from Clemson University for their help with using the real OctArm. The work is supported by NSF grant IIP-1439695.

REFERENCES

[1] M. M. Zhang, M. Kennedy, M. Hsieh, and K. Daniilidis, “A triangle histogram for object classification by tactile sensing,” in *International Conference on Intelligent Robots and Systems(IROS)*, IEEE, 2016.

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

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

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

[5] I. D. Walker, “Continuous backbone continuum robot manipulators,” *ISRN Robotics*, vol. 2013, 2013.

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

[7] J. Li and J. Xiao, “Determining grasping configurations for a spatial continuum manipulator,” in *IEEE IROS*, 2011.

[8] J. Li, Z. Teng, J. Xiao, A. Kapadia, A. Bartow, and I. Walker, “Autonomous continuum grasping,” in *IEEE IROS*, 2013.

[9] J. Li and J. Xiao, “Progressive, continuum grasping in cluttered space,” in *IEEE IROS*, 2013.

[10] J. Li and J. Xiao, “A general formulation and approach to constrained, continuum manipulation,” *Advanced Robotics*, vol. 29(13), 2015.

[11] J. Li, Z. Teng, and J. Xiao, “Can a continuum manipulator fetch an object in an unknown cluttered space?,” *IEEE Robotics and Automation Letters*, 2017.

[12] H. Mao, Z. Teng, and J. Xiao, “Progressive object modeling with a continuum manipulator in unknown environments,” in *ICRA*, 2017.

[13] A. Toshev, B. Taskar, and K. Daniilidis, “Shape-based object detection via boundary structure segmentation,” *Int. J. of Computer Vision*, 2012.

[14] M. Zhu, K. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis, “Single image 3d object detection and pose estimation for grasping,” in *ICRA*, 2014.

[15] J. Bohg and D. Kragic, “Grasping familiar objects using shape context,” in *ICAR*, 2009.

[16] M. Popović, D. Kraft, L. Bodenhagen, E. Başeski, N. Pugeault, D. Kragic, T. Asfour, and N. Krüger, “A strategy for grasping unknown objects based on co-planarity and colour information,” *RAS*, 2010.

[17] R. Detry, C. H. Ek, M. Madry, and D. Kragic, “Learning a dictionary of prototypical grasp-predicting parts from grasping experience,” in *ICRA*, 2013.

[18] M. V. Liarokapis, B. Calli, A. J. Spiers, and A. M. Dollar, “Unplanned, model-free, single grasp object classification with underactuated hands and force sensors,” in *IEEE IROS*, 2015.

[19] A. Saxena, J. Driemeyer, and A. Y. Ng, “Robotic grasping of novel objects using vision,” *IJRR*, 2008.

[20] J. Bohg, A. Morales, T. Asfour, and D. Kragic, “Data-driven grasp synthesis - a survey,” *TRO*, 2014.

[21] A. M. Okamura and M. Curkosky, “Feature-guided exploration with a robotic finger,” in *IEEE ICRA*, 2001.

[22] Q. Li, C. Schürmann, R. Haschke, and H. J. Ritter, “A control framework for tactile servoing.,” in *Robotics: Science and systems*, 2013.

[23] N. Sommer, M. Li, and A. Billard, “Bimanual compliant tactile exploration for grasping unknown objects,” in *IEEE ICRA*, 2014.

[24] R. Ibrayev and Y.-B. Jia, “Recognition of curved surfaces from one-dimensional tactile data,” *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 3, pp. 613–621, 2012.

[25] Y.-B. Jia and J. Tian, “Surface patch reconstruction from one-dimensional tactile data,” *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 2, pp. 400–407, 2010.

[26] M. Meier, M. Schopfer, R. Haschke, and H. Ritter, “A probabilistic approach to tactile shape reconstruction,” *IEEE TRO*, vol. 27(3), 2011.

[27] P. K. Allen and P. Michelman, “Acquisition and interpretation of 3-d sensor data from touch,” in *Workshop on Interpretation of 3D Scenes*, pp. 33–40, IEEE, 1989.

[28] A. Bierbaum, M. Rambow, T. Asfour, and R. Dillmann, “A potential field approach to dexterous tactile exploration of unknown objects,” in *International Conference on Humanoid Robots*, IEEE, 2008.

[29] C. E. Rasmussen, “Gaussian processes for machine learning,” MIT Press, 2006.

[30] O. Williams and A. Fitzgibbon, “Gaussian process implicit surfaces,” *Gaussian Proc. in Practice*, 2007.

[31] M. Björkman, Y. Bekiroglu, V. Högman, and D. Kragic, “Enhancing visual perception of shape through tactile glances,” in *IEEE IROS*, pp. 3180–3186, 2013.

[32] J. Ilonen, J. Bohg, and V. Kyrki, “Fusing visual and tactile sensing for 3-d object reconstruction while grasping,” in *IEEE ICRA*, 2013.

[33] Z. Yi, R. Calandra, F. Veiga, H. van Hoof, T. Hermans, Y. Zhang, and J. Peters, “Active tactile object exploration using gaussian processes,” in *IEEE IROS*, 2016.

[34] M. M. Zhang, N. Atanasov, and K. Daniilidis, “Active end-effector pose selection for tactile object recognition through monte carlo tree search,” to appear at IEEE IROS, 2017.

[35] 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.

[36] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2001.

[37] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, “A survey of Monte Carlo tree search methods,” *IEEE Transactions on Computational Intelligence and AI in Games*, 2012.

[38] W. B. Powell, *Approximate Dynamic Programming*, ch. 6, pp. 197–221. John Wiley & Sons, Inc, 2010.

[39] L. Kocsis and C. Szepesvári, “Bandit based Monte-Carlo Planning,” in *European Conference on Machine Learning (ECML)*, pp. 282–293, 2006.