MANIPULATION AND PERCEPTION SYNERGY FOR AUTONOMOUS ROBOTS IN UNKNOWN ENVIRONMENTS

by

Huitan Mao

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2019

Approved by:

Dr. Jing Xiao

Dr. Srinivas Akella

Dr. Jianping Fan

Dr. Andrew Willis

ABSTRACT

HUITAN MAO. Manipulation and Perception Synergy for Autonomous Robots in Unknown Environments. (Under the direction of DR. JING XIAO)

Intelligent robots have been increasingly used in unstructured and unknown environments rather than being limited in well-controlled settings. The key to successful autonomous robot operations in such environments is to combine robot manipulation and perception in a synergy, such that the perception guides robot motion, and the robot motion in turn enables better perception.

This dissertation first addresses how to perceive an unknown object effectively and efficiently in an unknown environment through robot and object contact interaction. The novel approaches introduced efficiently generate continuum wraps around unknown objects based on contact interaction and use the resulting robot shape to capture the object shape information to achieve effective object classification, recognition and shape estimation. Additionally, experimental results also demonstrate that object classification can be achieved through simulation-to-real-world transferable learning.

This dissertation further considers appearance-based object modeling in cluttered environments. Leveraging flexible continuum manipulation, an approach is introduced to plan robot motion that positions a tip camera at suitable spots around the target object to take RGBD images and register them to build and extend the object 3D model progressively, while avoiding obstacles in unknown and cluttered environments.

This dissertation also addresses how to achieve more flexible and autonomous robotic manipulation based on perception. A real-time adaptive motion planning approach is introduced to enable automatic conflict resolutions between task constraints and obstacle avoidance based on real-time visual sensing. More natural robot motion that seamlessly switches between task-constrained and non-task-constrained modes is achieved for improved motion adaptiveness in dynamically unknown environments.

Last but not least, pose uncertainty reduction under complex contacts for fine manipulation is also investigated. A novel force forecast approach that relates real-world force sensing to a simulated world to enable pose uncertainty reduction is introduced. This approach does not require knowing contact locations or pre-define any contact types, and it can be directly applied to reduce pose uncertainty in real-world contact-rich assembly tasks.

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to my research advisor Dr. Jing Xiao for her continuous support of my Ph.D. study. Her patience, enthusiasm, inspiration, immense knowledge and rich experience in robotics make her a great mentor. Without her patient guidance and persistent help, this dissertation would not have been possible.

I would also like to thank the rest of my Ph.D. dissertation committee members: Dr. Srinivas Akella, Dr. Jianping Fan, and Dr. Andrew Willis for their insightful comments and encouragement, but also for their hard questions, which inspired me to broaden my research horizons.

I am also grateful to Liqin Zhu, Jinglin Li, Zhou Teng, Mabel M. Zhang and Junius Santoso for their excellent research collaborations. I also thank my labmates and friends: Sterling McLeod, David Vutetakis, Rongcheng Lin, Andre Z. Sanchez, Dapeng Chen, Saurav Agarwal, Sayantan Datta, Haofeng Jia, Qiuyu Chen and others for all the inspiring discussions and happy moments we shared together.

Last but not least, I am especially grateful to my family: my caring mother Caimei Si, my encouraging father Baiqing Mao, my pretty girlfriend Jing Chen for their love, companionship and support throughout this dissertation research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Robots are ubiquitous nowadays. On industrial manufacturing floors, robot arms with payloads ranging from a few milligrams to hundreds of kilograms have relieved people from repetitive and intensive labour work. In hospitals, surgical robots have also been revolutionizing how surgeries are done with the benefits of faster recovery and shorter hospital stay. In e-commerce warehouses, logistics dispatch centers, and marine ports, hundreds of unmanned ground vehicles (UGVs) are running continuously to fetch packages or cargo containers, and transport them to their destinations with motions coordinated to maximize time efficiency. In vast farmlands, unmanned aerial vehicles (UAVs) also find their great use of detecting plant diseases or agricultural pests with their on-board cameras and computer vision techniques. In lakes, water reservoirs, and the ocean, unmanned surface vessels (USVs) are used effectively for rescue, water quality inspection, ocean garbage collection, remote sensing, and so on. In our daily lives, many robotic devices, from mobile robots for vacuum cleaning to UAVs for personal photo shooting, also improve our quality of life.

Most of the aforementioned applications require a robot to operate in unstructured environments with unknowns and uncertainties, which require a robot to be able to perceive its environment through sensing and decide its motion in real-time adaptively. For example, when a mobile robot enters an unknown space, visual sensors are often necessary to guide the robot's initial movement. As the robot moves, visual

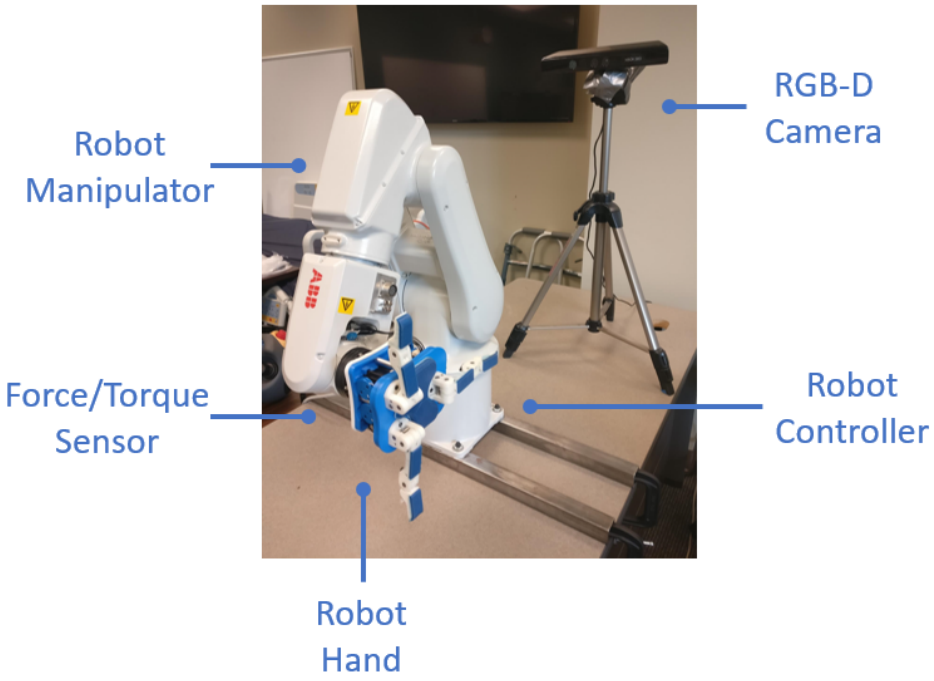


Figure 1: A common setup of an autonomous robot system includes a robot manipulator, a robot end-effector (hand), a robot controller (partially seen in the image), a force/torque sensor and an RGB-D camera.

sensors (carried by the robot) can also gather new information and hence can be used to further guide the robot motion. Therefore, interleaving robot perception and motion is key to performing a task in unknown and unstructured environments.

Fig. 1 shows a common setup of an autonomous robot system, where a few sensors (force/torque and RGBD) are used to perceive the environment and guide the robot manipulator motion. A dexterous robot hand is often equipped for object and environment interaction. Such an autonomous system is characterized by synergistic integration of perception and manipulation (action) to function effectively in unknown and unstructured environments. Robot perception provides information to guide or initiate the manipulation, and manipulation can be planned and controlled intelligently to facilitate perception.

1.1 Perception for Manipulation

Visual perception captures appearance of an environment using cameras, depth sensors, laser range finders, and other sensors. Such information is often interpreted to provide explicit meanings for object recognition, pose estimation [118], scene understanding [111], or even the entire reconstruction and mapping of 3D environments [89]. For robotic manipulation tasks, visual perception often deals with scenes cluttered with multiple objects stacking together or occluding each other [117, 134]. In GPS-denied environments, vision is used as the main sensing modality for robot localization, which is often considered in simultaneous localization and mapping (SLAM) problems [7].

More recently, with the development of deep neural networks, end-to-end visuomotor control policies also provide new capabilities for robot perception and manipulation, for instance, for in-hand manipulation [6] or bin picking in cluttered environments [54]. These end-to-end control policies take raw visual images as input and directly output manipulator motor toques. However, training such systems end-to-end cannot be easily done in real world, and training in simulation cannot be transferred easily to real world. Additionally, the obtained policies may be difficult to interpret, and safety measures are hard to be incorporated.

Tactile perception complements visual perception to provide local contact information. The most common form is a single contact sensor working as a bump switch [75]. Tactile arrays extend this by placing many single contact sensors near one another and forming an array [65, 66], or integrating sensors with different modalities

(force, proximity, temperature and accelerometer) into an array [43]. However, these tactile arrays can only form rigid contacts when they are touching other objects or environments. More recently, Gelsight [141] sensors overcome this problem by building passive compliance into the sensors, so that they can passively conform to the shape of the contact region to obtain contact information more accurately. Perceiving contact information with sensors directly mounted on robotic hands facilitates manipulation of grasped objects, such as detection of object slippage and compliant grasping [21, 55, 108].

Force/torque sensors are used to sense external contact forces and torques, and they are usually installed between the last link of the robot arm and its end-effector [140]. Similarly, torques at each robot arm joint can be used to provide a measure of contact impact by equipping a torque sensor at each joint or estimating the torque value from the motor currents [103]. Such indirect contact sensing is often used to facilitate robot control in contact-rich tasks (such as robotic assembly [103]).

1.2 Manipulation-enabled Perception

Manipulation can also facilitate perception. Touch-based manipulation can aid object perception thanks to the rich information embedded in contacts. For example, recognition and reconstruction of local surface patches were achieved in [41, 34] based on touch-enabled exploration motion. Object (or environment) models were estimated in [20, 137]. The robot end-effector tool calibration and its kinematics estimation can be achieved by making and maintaining contacts [46, 66]. Many object fine property perception problems have been investigated by using Gelsight sensors to

directly capture contact information, such as surface texture recognition [67], hardness estimation [145, 143], clothing material perception [142]. Object localization and pose estimation is achieved effectively through touch-based robot contact interaction (for example [95, 88]), especially in environments where vision sensing is ineffective. However, the aforementioned literature only uses a robot end-effector to touch an object at discrete locations, and thus it is a slow process to collect sufficient contact information to capture the global shape information of the object. Unlike conventional robot arms, continuum robots [101, 130, 42, 128] are suitable for whole-arm object manipulation, and therefore, rich contact information can be obtained efficiently. The existing manipulation planning literature for continuum robots [63, 62, 60] mainly considers how to plan collision-free robot motion for manipulating an object with a known model.

Therefore, there exists an open problem on how to use touch information to automatically and progressively wrap around an unknown object using a flexible continuum robot, and how to use the result of such continuum wrapping manipulation to achieve object perception in terms of classification, recognition, and shape estimation. Chapter 2, 3 and 4 further motivate this open problem and introduce novel approaches to address them.

Vision-based object perception is often facilitated by using robots to move cameras to desired locations. For example, in view planning problems [127, 126], vision sensors can be moved to locations where the information gain in an unknown environment is optimized. Instead of moving a camera, object modeling can also be achieved with robot motion that moves a grasped object in front of a camera [73, 33, 117].

Therefore, through coupling robot action and visual perception, sensing ranges can be expanded, and more efficient and effective sensing can be achieved. However, the aforementioned literature on vision-based object modeling commonly assumes that the object is stand alone in isolation so that there is no need to consider obstacles. For the vast literature on motion planning for manipulation, the target object model is assumed known and so are obstacle models to some extent.

Therefore, there exists an open problem on how to manipulate an unmodeled object in a cluttered environment with unknown obstacles. Chapter 5 further motivates this problem and introduces an approach on progressive object modeling with a continuum robot in unknown and cluttered environments.

1.3 Perception-guided Manipulation

Visual perception guides many robot actions, especially for robot real-time motion planning in dynamically changing environments [125]. For instance, to bring a glass of water to a customer in a crowded restaurant, a robot waiter will have to avoid unforeseen moving obstacles based on visual or range sensing. In such dynamically changing environments, motion planners not only need to react promptly based on perception information but also generate smooth and safe robot motion that satisfies other constraints, such as robot kinematics constraints, obstacle avoidance, task constraints and so on. Although there exist algorithms for planning task-constrained motion (such as [48, 109, 148, 23]) and for planning robot motion in dynamic environments (such as [30, 92, 136]), they are not best equipped to tackle the general challenge of resolving conflicts between satisfying task constraints and avoiding un-

foreseen dynamic obstacles.

Therefore, there exists an open problem about how to design a mechanism that resolves the conflicts on-the-fly for task-constrained manipulation in dynamically unknown environments. Chapter 6 further motivates this problem and introduces approaches to address it.

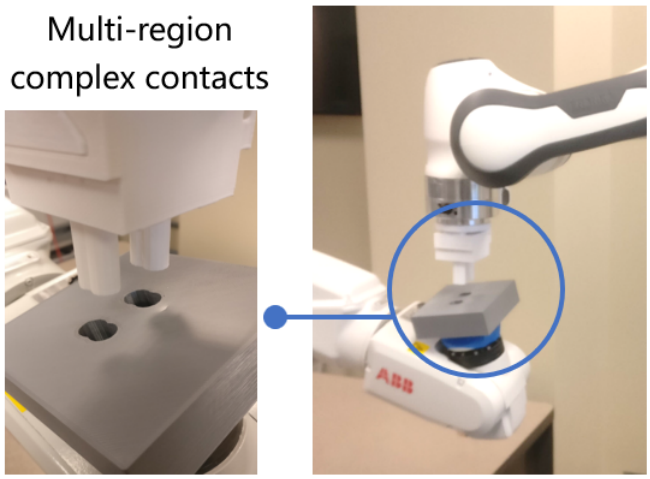


Figure 2: An example of a two-pin peg-in-hole robotic assembly. Contacts may occur at multiple regions at the same time, and the arbitrary/irregular shape of the peg and hole further makes contacts complex.

Tactile and contact sensing are crucial for robotic grasping, in-hand manipulation, assembly [22, 103, 132], etc. Contacts for those tasks are typically complex and occur at multiple regions at the same time (see Fig. 2 for an example). The complex nature of such contacts makes many contact-rich manipulation tasks challenging. Moreover, uncertainties associated with an object pose can easily make any predefined manipulation procedures invalid. The existing literature on contact representation and state computation [135, 116, 40, 71] are based on primitive contact types. However, identifying such primitive and well-formed contacts is practically difficult. For sensor-based contact-rich manipulation (such as local adjustments or compliant motion [108, 65]),

existing methods either require tactile sensors to directly perceive local contact information [37] or have restrictive assumptions on contact locations [139, 20] or predefined contact types [140].

Therefore, there exists an open problem about how to use complex contact information that can only be sensed indirectly to reduce pose uncertainty. Chapter 7 further reviews literature on this and presents a novel approach to address it.

1.4 Summary of Open Problems

The open problems addressed in this dissertation are summarized as follows,

1. **Manipulation-enabled perception**

   (a) *There exists an open problem on how to automatically and progressively wrap around an unknown object using a flexible continuum robot based on touch information, and how to use the result of such continuum wrapping manipulation to achieve object perception* (Chapter 2, 3 and 4).

   (b) *There exists an open problem on how to manipulate an unmodeled object in a cluttered environment with unknown obstacles* (Chapter 5).

2. **Perception-guided manipulation**

   (a) *There exists an open problem on how to design a mechanism that resolves the conflicts on-the-fly for task-constrained manipulation in dynamically unknown environments* (Chapter 6).

   (b) *There exists an open problem on how to actively use complex contact information that can only be sensed indirectly to reduce pose uncertainty*

(Chapter 7).

Each subsequent chapter is further described below.

Object classification and recognition with touch-based continuum manipulation is presented in Chapter 2. 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 is presented, and it 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 a classifier of object categories very efficiently. An algorithm for active guidance of object recognition is also introduced, and it allows an object to be recognized with just a few continuum wraps.

Object shape estimation with touch-based continuum manipulation is presented in Chapter 3. Unlike shape estimation using contact points collected through the end-effector of a conventional, articulated manipulator, the shape estimation through continuum manipulation is both more efficient and effective by deliberate use of the robot proprioception data in addition to contact information collected at each robot intermediate motion step.

Chapter 4 presents experiments demonstrating that the shape-based classifier trained solely from simulation is able to generalize to classifying real-world objects. Since conducting many real-world continuum wraps can be time-consuming, it is significant that a classifier trained purely in simulation shows considerable effectiveness in classifying real objects. This could make classifier training more efficient and feasible

for classifying a large number of categories of many real objects from touch-based continuum wrapping.

Chapter 5 presents a general approach to simultaneously planning collision-free continuum robot motion and achieving effective visual perception for on-site model building in unknown 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 obstacles and sense the unmodeled target object from different viewpoints. The model of the target object is progressively built as the robot arm moves.

Chapter 6 introduces a real-time motion planning approach that enables automatic conflict resolutions between task constraints and obstacle avoidance based on real-time visual sensing information. More natural robot motion that seamlessly switches between task-constrained and non-task-constrained motion is achieved for better adaptiveness in dynamically unknown environments.

Chapter 7 presents a novel force forecast approach that relates the real world force sensing to a simulated world to enable pose uncertainty reduction in real-world contact-rich tasks. This approach does not require information of contact locations or pre-define any contact types. Real-world experiments demonstrate that challenging multiple-pin peg-in-hole assembly tasks can be accomplished with reduced pose uncertainty using the introduced approach.

Chapter 8 summarizes the contributions of this dissertation and discusses the future work.

CHAPTER 2: SHAPE-BASED OBJECT CLASSIFICATION AND RECOGNITION THROUGH CONTINUUM MANIPULATION

In this chapter, we address the problem of automatic object classification and recognition based on shape information obtained with a continuum manipulator guided by tactile sensing. Humans and animals often rely on touching an object and exploring its shape to recognize it when vision cannot be effective (due to poor illumination, transparent object surfaces, and heavy occlusion). 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 [147]. 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.

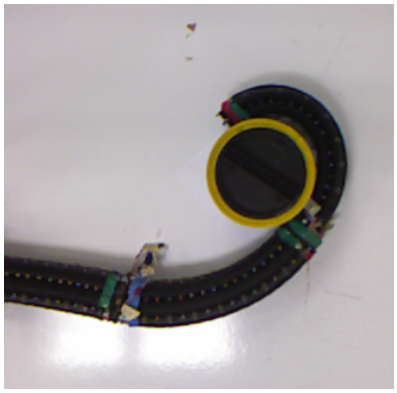


Figure 3: A continuum robot wraps a cylindrical object.

Hence, this 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. 3), interleaving contact sensing and manipulator motion planning and execution.

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.

2.1 Related Literature

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 [101, 130, 42, 128], 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 [58, 57, 59] and conducting task-constrained inspection tasks [60] of known objects in known environments. More recently, the study in [56] 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 [76], 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 described analytically (such as [87]). Whereas, in many robotics applications, it is difficult to obtain such analytical object shape information, and hence shape is mostly recognized by vision. The closest related work to this study from the vision literature is [121], 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 [149], grasping by contour [12], grasping unknown objects by shape [97, 25], object classification from single grasps [69], and detecting grasping points on novel objects [104]. A survey explores various vision-based methods for grasping [13].

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 [91, 65] and for grasping unknown objects [108]. There is also work on recognition and reconstruction of curved surface patches through touch [34, 41].

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 [85, 5, 147]. However, these methods do not take advantage of the adjacency information of contacts. In [10], 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) [98] 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) [131] is used in [11, 35, 137] to guide the haptic exploration towards high uncertainty.

In contrast to greedy exploration approaches, [146] 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 [147] for tactile recognition.

2.2 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 $sec_i, i = 1, \ldots, n$ 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. 4. The *arm configuration* of an n-section continuum manipulator can be represented as $\{(s_1, \kappa_1, \varphi_1), ..., (s_n, \kappa_n, \varphi_n)\}$ [42, 59].

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 [62].

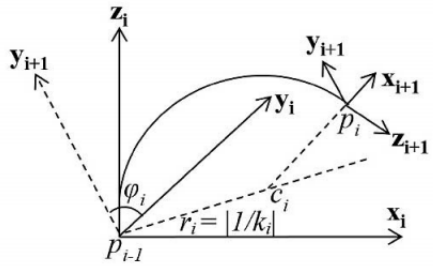


Figure 4: Illustration of the frame of sec_i and the control variables s_i, κ_i and φ_i [56].

The target objects considered in this work 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.

2.3 Methodology

In this section, we present our approaches in detail.

2.3.1 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. 8 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 $C = \{contact_1, \ldots, contact_d\}$. 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 [60]. 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

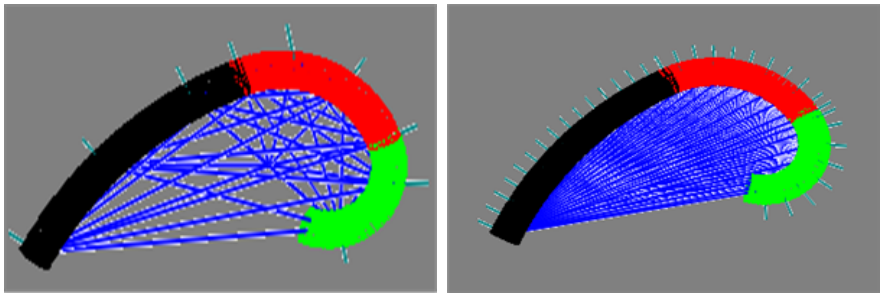
$tipRotatedAngle = 0$;
while $tipRotatedAngle < \theta$ **do**
  $Contacts = \emptyset$; $\mathbf{P}_{new} = \emptyset$; $i = 1$; $k = 0$;
  `// enclosing motion step`
  repeat
    if *sec_i is not at curvature limit* **then**
      Curve sec_i by increasing κ_i by $\Delta\kappa$;
    else if $i < n$ **then**
      $i = i + 1$;
    if *sec_i is not at length limit* **then**
      Extend sec_i by increasing s_i by Δs;
    if *arm is in contact with the object at* $C = \{contact_1, \ldots, contact_d\}$ **then**
      $Contacts \leftarrow Contacts \cup C$;
      if *sec_j is the closest in-contact sec to the tip and* $j < n$ **then**
        $i = j + 1$;
  until *sec_n is in contact with the object or reaches its curvature limit*;
  `// advancing motion step`
  while $k < |Contacts|$ **do**
    if *$contact_k$ is on sec_n* **then**
      $m \leftarrow n$;
    else
      $m \leftarrow$ index of the section endpoint closest to $contact_k$;
    $\mathbf{P} \leftarrow$ the position of the ith endpoint;
    $\mathbf{P}_m \leftarrow \mathbf{P} + \delta(\mathbf{n_k} + \mathbf{t_k} + \mathbf{b_k})$;
    $\mathbf{P}_{new} \leftarrow \mathbf{P}_{new} \cup \mathbf{P}_m$; $k = k + 1$;
  end
  for *each remaining section endpoint* **do**
    $m \leftarrow$ its section index;
    $\mathbf{P} \leftarrow$ its current position;
    $\mathbf{z} \leftarrow$ the z axis at the endpoint; $\mathbf{P}_m \leftarrow \mathbf{P} + \delta\mathbf{z}$;
    $\mathbf{P}_{new} \leftarrow \mathbf{P}_{new} \cup \mathbf{P}_m$;
  end
  $armConfig = constrainedIK(\mathbf{P}_{new})$;
  if *Failed to find a valid $armConfig$* **then**
    Set Flag "No IK found" and exit;
  else
    Move the arm to $armConfig$;
    $tipRotatedAngle \leftarrow$ angle from initial $\mathbf{z}_s$ to current $\mathbf{z}$ axis at tip;
end

2.3.2 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 [121] 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 [147], 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 [147] 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 chapter.

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. 5 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.

Figure 5: Chords collected on example wraps are shown in blue. The normals at the sampled points are shown in green.

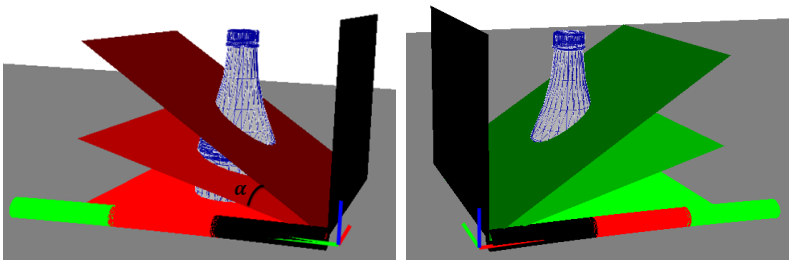
After an object is wrapped around several times, the sets of chords collected are binned into a 7D histogram. Each dimension of the histogram represents a parameter. Principal Component Analysis (PCA) is applied to the data to reduce redundancy.

2.3.3 Shape-based Classification of Object Classes

We train a support vector machine (SVM) [105] 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. 6). 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.



(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.

Figure 6: 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.

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.

2.3.4 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. The core idea is that observations of an object's shape and the actions taken to wrap around the object are related. We formulate the problem

as a Markov Decision Process (MDP), which searches for a sequence of actions, that minimizes an objective function that trades off between minimum movement cost and maximum recognition certainty. We use the Monte Carlo tree search [17] to find such policy. The robot action is defined as a wrapping plane in which a new continuum wrap is about to be generated. The output of the tree search is a sequence of such actions for the robot to execute. My collaborator Mabel Zhang and Kostas Daniilidis contributed to the design of the method used in object recognition with active guidance. Please see original paper [80] for detailed description of this part.

2.4 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 [147]. 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.

2.4.1 Capturing Object Shape

Table 1 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 [147] 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. 7 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. 8 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 video attached to [80].

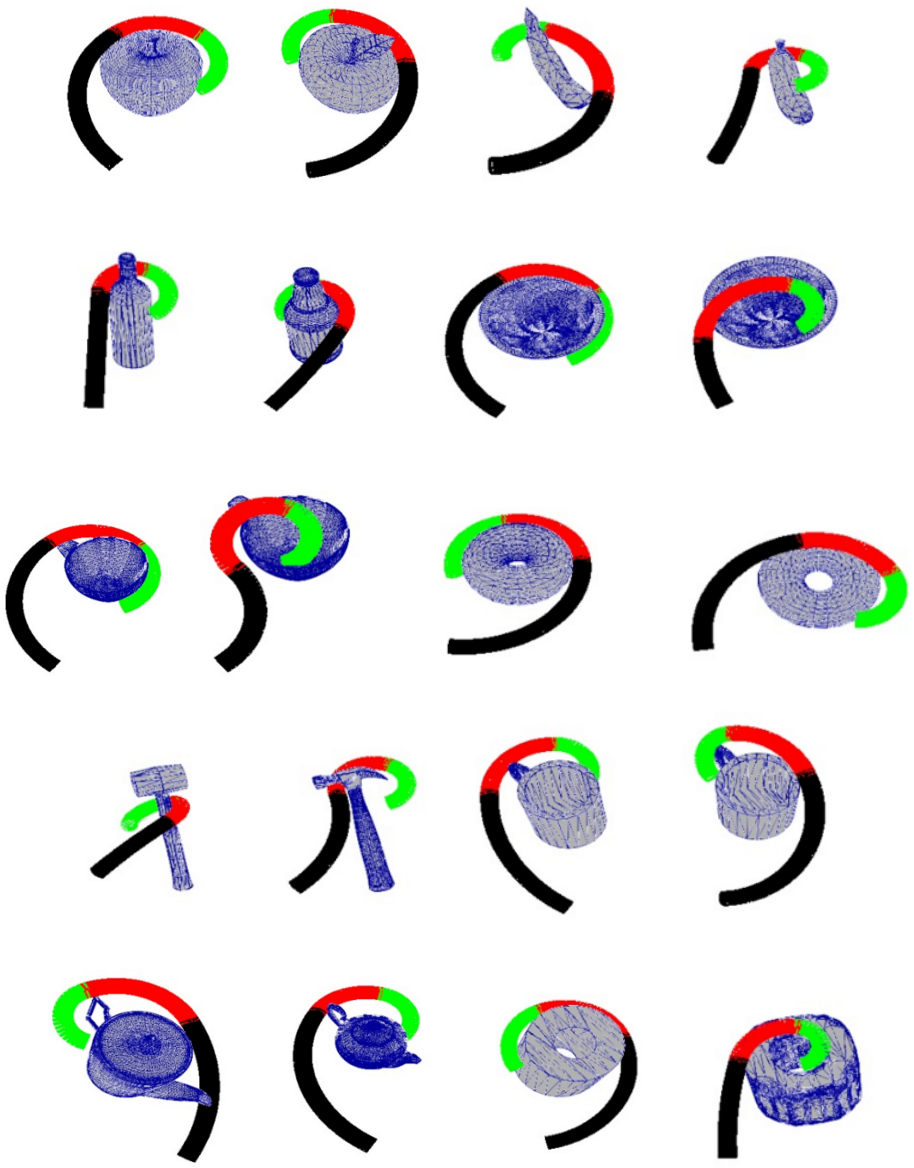


Figure 7: Example wraps used to encode object cross section shape into the arm shape. Each category has two example wraps (from top left to lower right): apple, banana, bottle, bowl, cup, donut, hammer, mug, teapot, toilet paper rolls.

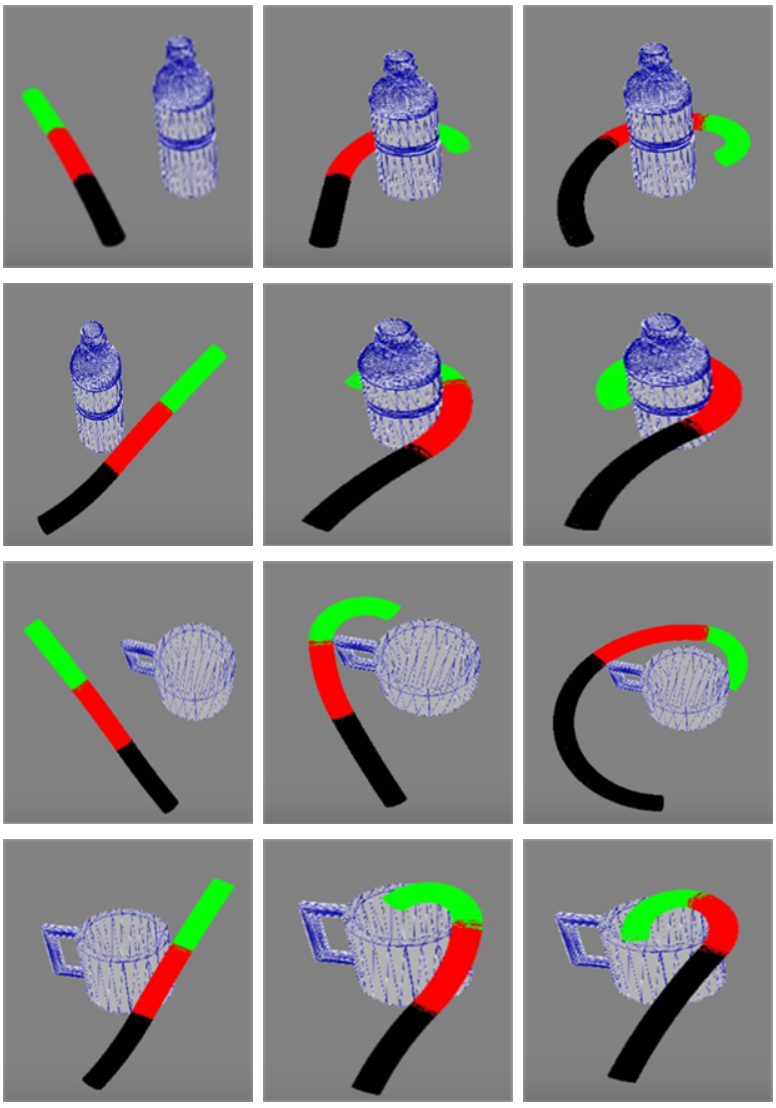


Figure 8: Snapshots of example motions of wrapping. Each row shows one wrap from the initial configuration (left sub-figure) to the final wrapping configuration (right sub-figure).

Table 1: 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

2.4.2 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 [147], 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. 9, 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.

2.4.3 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 2. 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.

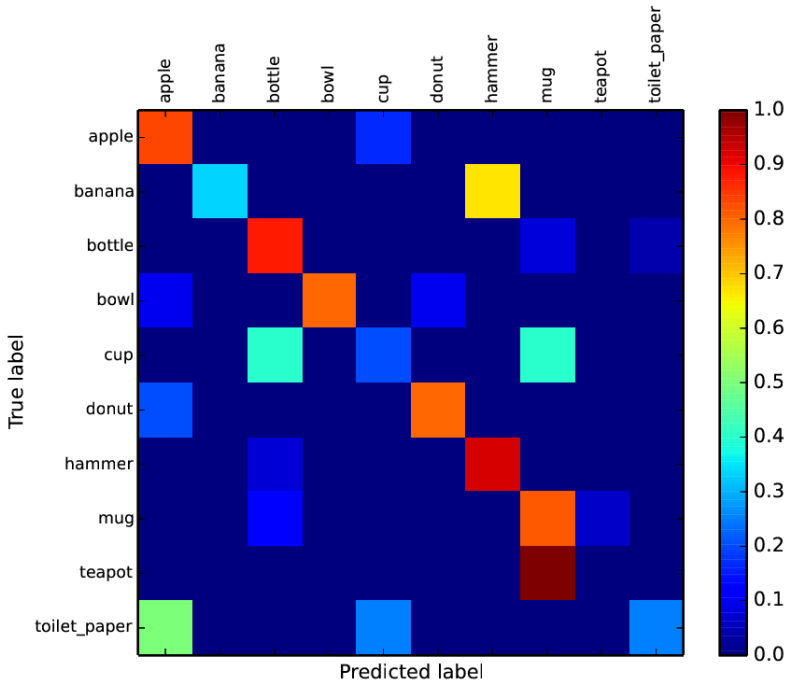


Figure 9: SVM confusion matrix.

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.

Table 2: Active recognition performance (horizon=5). Pred. indicates the predication. Prob. indicates the probability. Object paper indicates toilet paper.

Object	iteration1			iteration2			iteration3		
	pred.	prob.	# wraps	pred.	prob.	# wraps	pred.	prob.	# 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		paper	0.09		paper	0.05	
	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	

2.4.4 Real Robot Wraps

We tested using the real OctArm robot to wrap two real objects, one cylindrical and one rectangular by teleoperation, and captured the wraps using a Microsoft Kinect. We also save the arm configurations of the wraps. Fig. 10 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.

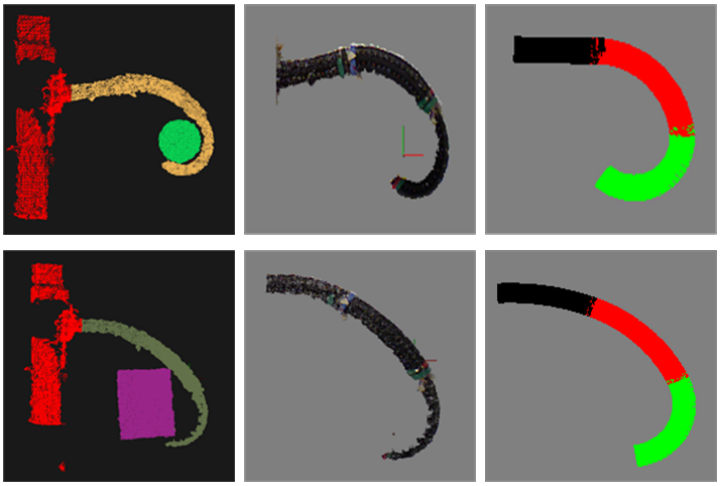


Figure 10: Comparing real wraps and the corresponding simulated wraps around a cylindrical object (top) and a rectangular object (bottom) respectively. In each row from left to right: the segmented RGBD point cloud of the OctArm wrapping the object, segmented OctArm shape, reconstructed OctArm shape in simulation.

2.5 Summary

In this chapter, 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.

CHAPTER 3: OBJECT SHAPE ESTIMATION THROUGH TOUCH-BASED CONTINUUM MANIPULATION

In this chapter, we address the problem of estimating the shape of an unknown object through obtaining object shape information from touch-based continuum manipulation, which has not been studied before. Information of object shape is crucial for many robotic tasks. A grasp can be planned to fetch an object with a known shape. When planning a path, collision can be checked against an object using its shape and configuration. Information of object shape also facilitates object detection, recognition, and pose estimation. Usually, the shape of an object is either provided as a priori knowledge or acquired by object model building through sensing, especially vision and tactile sensing.

By wrapping around an object (see Fig. 3 for an example), a continuum manipulator makes many more contacts with an object than a conventional robot end-effector. Moreover, the manipulator shape itself in a wrapping configuration is also indicative of the object shape. Our approach, called *progressive object shape estimation through continuum manipulation* (POSE-CoM), extends the Gaussian Process Implicit Surfaces (GPIS) [131] method by explicitly incorporating the continuum robot arm shape in addition to the contact points made between the robot and the object in each wrap to estimate the overall object shape. The approach is shown to be both more efficient and more accurate over existing methods for touch-based object shape estimation.

3.1 Related Literature

Vision sensing has been widely used for object model building. An object appearance model can be built by either moving an RGB-D camera around the target object [89] or moving the object with a turntable [24, 107]. Through robotic manipulation to change the bottom surface of a table-top object, an automatic approach interleaving perception and manipulation [117] is able to build the entire surface model of the object. However, vision sensing can be ineffective for transparent objects or in environments with poor illumination and specularity conditions.

Tactile sensing is useful for exploring object shapes and building object models when vision is ineffective, for example through touch-based guarded moves [20] and compliant motion [91, 65]. In [34], recognition of curved surfaces through touch is achieved by matching contact points to principal-curvature-based local geometry features. Similar features are also shown to be useful in reconstructing the local surface patch by fitting a high-order polynomial [41].

In order to guide touch-based exploration, i.e., to decide where to touch next to collect contact points, there are several methods. One method uses a dynamic potential field [10], where a uniformly attractive potential field is updated as more contacts are made and generates repulsive forces to push the touch-enabled hand to visit unexplored areas. Gaussian Process (GP) [98] is used to drive active exploration into uncertain areas. In [137], discrete touch probings of the end-effector are progressively generated to reduce the uncertainty of the interest area using GP regression. In [39], it is shown that GP classification can also effectively bias the exploration towards the

boundaries of the objects, which are more informative of the object shape. Extended from GP, a probabilistic model of uncertainty based on Gaussian Process Implicit Surfaces (GPIS) [131] is used to guide the active exploration and modeling of an object [26, 20] or serve as a framework of data fusion from sensors with different modalities [31].

However, using only the robot end-effector to touch a target object usually only makes a couple contact points per probing, and thus it is a slow process to collect sufficient contact points to capture the global shape of the object by changing the end-effector pose after each probe. The process also does not explicitly utilize the adjacency information of nearby contact points. Unlike conventional robotic manipulators, continuum robots [101, 130, 42, 128] are suitable for whole-arm manipulation to wrap around an object, so that rich contact information can be obtained efficiently.

3.2 Progressive Object Shape Estimation through Continuum Manipulation (POSE-CoM)

An overview of our approach is presented in Fig. 11. First, **touch-based continuum wrapping** progressively moves the continuum robot to a wrapping configuration based on contact points made between the robot and the object. Next, **object shape data collection** gathers contact points and the shape information from the continuum wrap. **Shape estimation** uses the data collected so far in a probabilistic framework based on Gaussian Process Implicit Surfaces(GPIS) to estimate the overall shape of the object. **Active guidance** uses the result of estimation to decide the next continuum wrap to cover the most uncertain region of the object and collect

more data for estimation. The process repeats until either the estimated object shape has low uncertainty or the robot has exhausted the possible wraps.

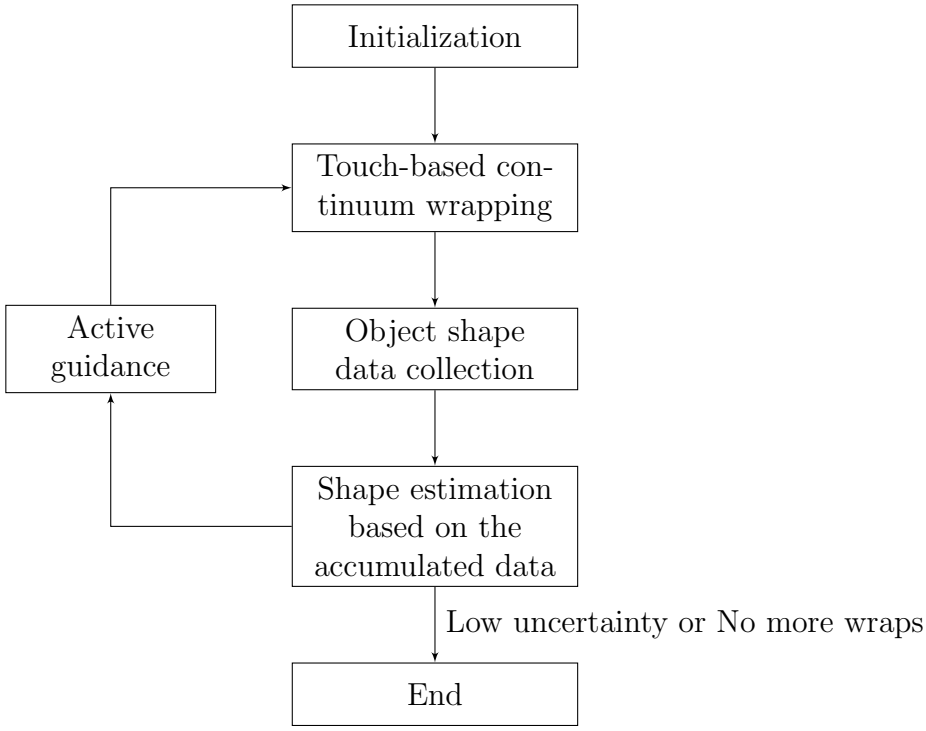


Figure 11: Overview of the POSE-CoM approach.

3.2.1 Touch-based Continuum Wrapping

A continuum *wrap* of an object here is defined as a planar whole-arm grasp by the continuum manipulator around a cross section of the object, which further defines a *wrapping plane*.

Given a continuum manipulator with a fixed base, its workspace can be decomposed into discrete wrapping planes, as shown in Fig. 12. We denote the set of wrapping planes as $WP = \{wp_1, \ldots, wp_c\}$. On each wrapping plane, the continuum robot can generate a wrap of an object around the cross section on the plane. See Fig. 13 for some example wraps on different wrapping planes of an object.

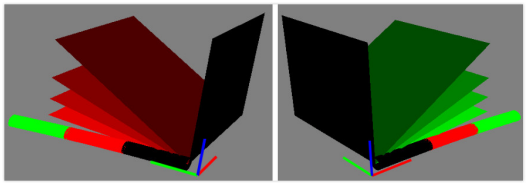


Figure 12: The wrapping planes are systematically enumerated between the table plane (grey background) and the black planes orthogonal to the table plane. **Left:** each red plane is used to generate a clockwise continuum wrap. **Right:** each green plane is used to generate a counter-clockwise continuum wrap. Note that the initial configurations of the robot in **Left** and **Right** are different. The frame at the robot base is the global coordinate system.

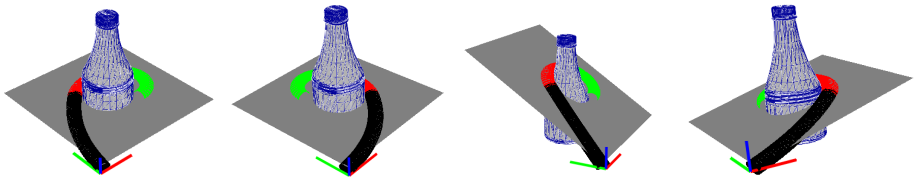


Figure 13: A few examples of the wrapping planes (grey) and their corresponding continuum wraps. These planes also uniquely cut through the object (a bottle) and pass through the fixed robot base. The frame at the robot base is the global coordinate system.

For initialization, an initial set of wrapping planes $WP_0 \subset WP$ are randomly selected, and the continuum robot generates wraps on those wrapping planes one by one, while accumulating shape data of the object from each wrap. The data are then used to conduct shape estimation. The result is further used to guide the selection of the next wrapping plane by **Active guidance** (see Section 3.4), and so on.

Now, given a wrapping plane and an initial arm configuration, we use a motion planning strategy to generate a continuum wrap of an object progressively as guided by the contact points made along the way. Detailed description is presented in **Algorithm 1**. Overall, the continuum manipulator alternates between the **enclosing**

motion step and the **advancing motion step** on the wrapping plane until a wrap is achieved. Illustration of the two types of motion is presented next.

Enclosing motion step brings the robot into contact with the object as much as possible to create contact points. This is achieved by having the robot curve its sections one by one to make contact with the object until no further contact points can be made. Fig. 14 shows an example of such an enclosing motion.

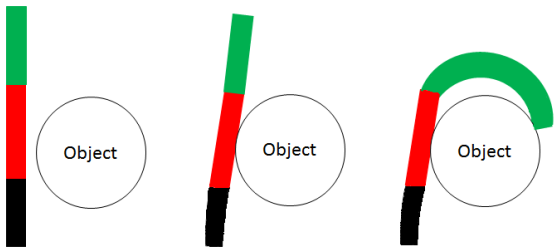


Figure 14: Illustration of an enclosing motion. **Left**: Robot at the initial configuration. **Middle**: The robot makes first contact by curving section 1 (black). **Right**: The robot makes the second contact by curving section 3 (green) without penetrating into the object.

Advancing motion step takes advantage of the contacts made in the **enclosing motion step** to move the robot forward to a new arm configuration towards wrapping around the object. The new arm configuration is achieved by (1) moving the endpoint of each robot section a small distance from its current position along the direction $\mathbf{n}+\mathbf{t}$ of the closest contact point, where $\mathbf{n}$ and $\mathbf{t}$ denote the normal and the tangential unit vectors of the contact point respectively[1]; (2) solving the resulting arm configuration by the constrained inverse kinematics [60] corresponding to the new endpoint positions

[1]The direction of $\mathbf{t}$ is flipped if the dot product between $\mathbf{t}$ and the z-axis of the local frame on the section endpoint closest to the contact point is negative, to ensure that the robot moves towards a wrapping configuration.

from (1).

Fig. 15 shows an example of advancing motion. The robot backbone(dashed line) is colored using black, red and green for sections 1, 2 and 3 respectively. As the result of the previous enclosing motion step, the robot is in contact with the object at the red points on sections 2 and 3. Now, for the end point of each section in contact, its new position (green) is determined by a translation of a small distance from its current position (yellow) along $\mathbf{n} + \mathbf{t}$ of the closest contact point. For section 1 that is not in contact, its new endpoint position is obtained by a small translation from the current endpoint position along the z axis of its local frame.

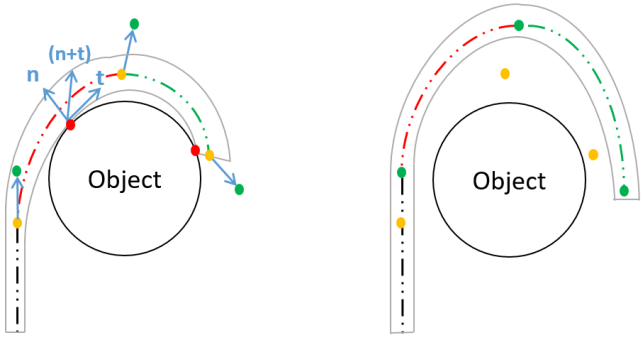


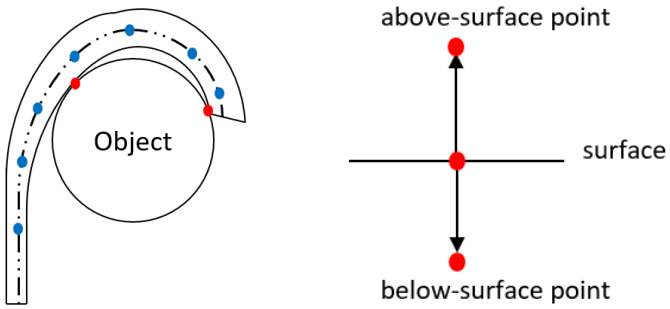
Figure 15: Illustration of an advancing motion. **Left**: the robot section end points are indicated in yellow, the contact points are in red, and the new endpoint positions are indicated in green. **Right**: the new arm configuration solved using constrained inverse kinematics based on the new endpoint positions (green).

3.2.2 Object Shape Data Collection and Generation

Two types of shape data are collected in our approach (see Fig. 16(a)):

1. contact points made between the robot and the object during wraps, and

2. arm points sampled along the backbone of the robot when the robot is in the



(a) (b)

Figure 16: **Left**: the dashed line is the backbone of the continuum robot. The blue dots are the arm points systematically sampled on the arm backbone. The red dots are the contact points between the robot and the object. **Right**: the interpolated above-surface and below-surface points from the contact point along the contact normal.

configuration of a complete wrap, i.e., points indicative of the robot shape.

We denote the collected points from all wraps made (both the contact points and the arm points) as $X_{col} = \{\mathbf{x}_i\}$, where $\mathbf{x}_i \in \mathbb{R}^3$, $i = 1, 2, \ldots$ A potential function value $y_i \in Y \subset \mathbb{R}$ associated with $\mathbf{x}_i$ is defined as follows, based on GPIS[131].

$$y_i = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ is above the surface} \\ 0 & \text{if } \mathbf{x}_i \text{ is on the surface} \\ -1 & \text{if } \mathbf{x}_i \text{ is below the surface} \end{cases} \tag{1}$$

The contact points that happen on the object surfaces have $y = 0$, while the arm points sampled on the backbone of the continuum robot have $y = 1$ (since the arm does not penetrate into the object).

We further generate the above-surface and below-surface points for each contact point along the contact normal (see Fig. 16(b) for an illustration) and denote the

set of such generated points $X_{gen} \subset \mathbb{R}^3$. Now let T be the set of all points that are collected and generated so far: $T = \{\mathbf{x}_i, y_i\}, \mathbf{x}_i \in X_{col} \cup X_{gen}, y_i \in Y, i = 1, 2, \ldots$

Note that initially T only contains points collected and generated from the few initial wraps. As each new wrap is conducted (based on active guidance – see Section 3.4), more points are added to T to facilitate more accurate shape estimation.

3.2.3 GPIS-based Shape Estimation

Estimation of an object's shape is done by finding the points with zero y value (i.e., the isosurface) in a 3D region of interest. GPIS is used to learn such a mapping $f(\mathbf{x})$: $\mathbf{x} \in \mathbb{R}^3$ to $y \in \mathbb{R}$ based on the data in T. It is fully defined by a Gaussian Process (GP) [98] with a mean function $\mu(\mathbf{x})$ and a covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$, where $j = 1, 2, ...$ The prior $\mu(\mathbf{x})$ is zero. $k(\mathbf{x_i}, \mathbf{x_j})$ is chosen to be the commonly used squared exponential kernel, and a noise $\epsilon \sim N(0, \sigma_n^2)$ is also included:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 exp(-\frac{(\mathbf{x}_i - \mathbf{x}_j)^2}{2l^2}) + \sigma_n^2 \delta_{ij} \tag{2}$$

where δ_{ij} is the Kronecker delta, which is 1 iff $i = j$ and 0 otherwise.

The hyper parameters $\{\sigma_f, l, \sigma_n\}$ are then optimized by maximizing the log-marginal likelihood[98] using the data in T, i.e., the training data.

Next, the zero-mean isosurface is extracted as the current estimation of the object shape by querying the GPIS model with τ testing points from a 3D region, which is known to contain the object or is within the reachable region of the robot.

For a testing point $\mathbf{x}_\star$, the predicted distribution is a Gaussian with the mean $\mu(\mathbf{x}_\star)$ in Eq. (3) and the variance $\sigma^2(\mathbf{x}_\star)$ in Eq. (4)

$$\mu(\mathbf{x}_\star) = \mathbf{k}_\star^T \mathbf{K}^{-1} Y \tag{3}$$

$$\sigma^2(\mathbf{x}_\star) = \mathbf{k}_{\star\star} - \mathbf{k}_\star^T \mathbf{K}^{-1} \mathbf{k}_\star \tag{4}$$

where $\mathbf{k}_\star$ is a covariance matrix between m training points and τ testing points $[\mathbf{k}_\star]_{i=1...m,j=1...\tau} = k(\mathbf{x}_i, \mathbf{x}_{\star j})$, $\mathbf{K}$ is a covariance matrix between training points $[\mathbf{K}]_{i,j=1...m} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{k}_{\star\star}$ is a covariance matrix between testing points $[\mathbf{k}_{\star\star}]_{i,j=1...\tau} = k(\mathbf{x}_{\star i}, \mathbf{x}_{\star j})$. Note that the number of training points m increases after new wraps are conducted.

Each testing point corresponding to a zero mean of Eq. (3) is a point on the estimated object shape, and the associated posterior variance of Eq. (4) defines the uncertainty of the point due to few data points nearby or large shape change.

3.2.4 Active Object Exploration

Our approach actively determines the next wrapping plane to conduct another touch-based continuum wrap based on a measure of uncertainty. Denote the set of the available wrapping planes WP_a as $WP_a = WP \setminus WP_0$.

For each available plane $wp_i \in WP_a$, let P_i be the set of points on wp_i that are also on the zero-mean isosurface, then the uncertainty measure u_i of wp_i is computed as the average standard deviation $\overline{\sigma}(q), \forall q \in P_i$.

Let wp_j be the wrapping plane with $u_j = \max_i(u_i)$, then wp_j is chosen as the next wrapping plane to conduct a continuum wrap, and a new iteration in the POSE-CoM process (Fig. 11) starts. The newly collected and generated data from the new wrap

are added to the training set T, and the GPIS model is in turn updated. WP_a is also updated by removing wp_j (after it has been used).

The POSE-CoM process (Fig. 11) is repeated until the maximum uncertainty u_j is below a predefined threshold or $WP_a = \emptyset$.

3.3 Experiments

We implemented our approach in C++ and Python under ROS and tested it on a 3.4GHz CPU with 16GB RAM. Objects of various shapes are used (Fig. 17).

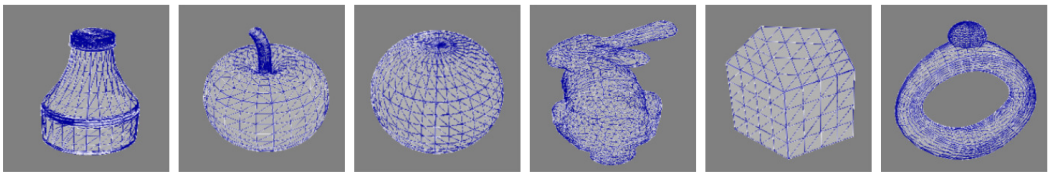
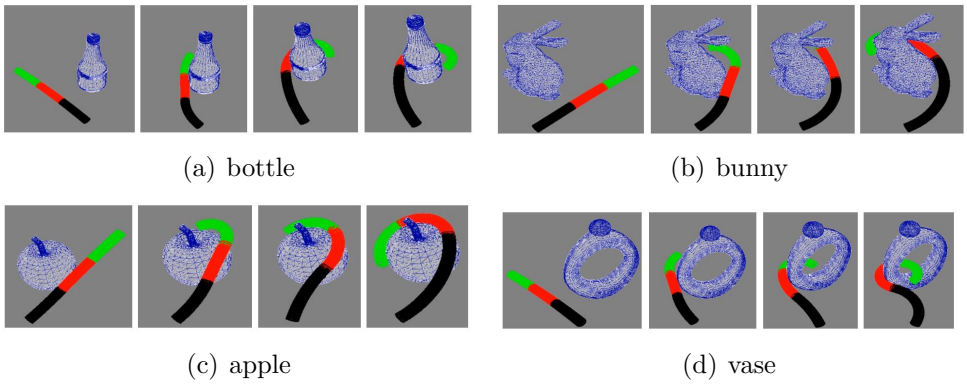


Figure 17: Object mesh models: (left to right) bottle, apple, sphere, bunny, pentagon, vase.

To initialize the POSE-CoM process, we randomly select 4 wrapping planes in the robot workspace for the continuum robot to generate touch-based wraps and collect data. Next, a GPIS model is trained and queried in the region of $[(0.0-1.0), (0.0-1.0), (0.0-0.8)]$(dm). The simulated continuum robot has the following parameters: the width of each section is 0.5 (cm), the length of each section can vary from 2 to 15 (cm), the curvature of each section can vary from 0 to 0.1 (cm^{-1}), and the orientation of each section can vary from $-\pi$ to π. It typically takes 3 to 15 mins to train a GPIS model. The next wrapping plane is selected according to the active strategy in Section 3.2.4, then touch-based continuum wrapping is conducted, and the POSE-CoM process repeats. The process is terminated when 1) the maximum uncertainty

measure u_j is below a predefined threshold 0.15, or 2) all the wrapping planes in the robot workspace have been used at least once to generate continuum wraps. Fig. 18 shows the snapshots of robot motion for example continuum wraps. The video accompanied to [77] shows the animated motion of the example continuum wraps.



(a) bottle (b) bunny

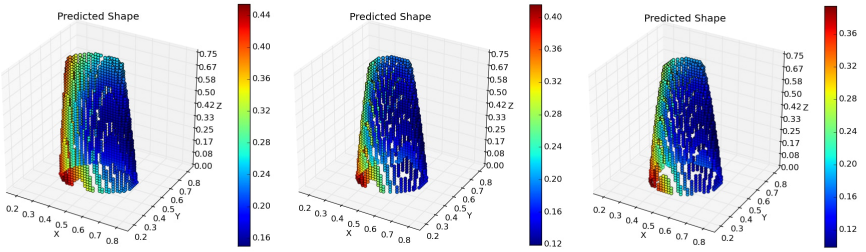
(c) apple (d) vase

Figure 18: Snapshots of the motion of example continuum wraps. Each subfigure is a wrap from the initial configuration (left) to the wrapping configuration (right).

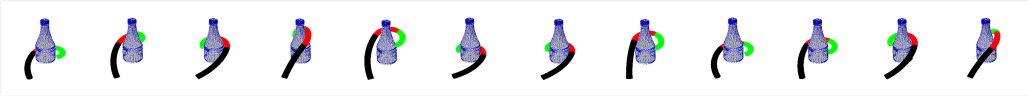
3.3.1 Shape Estimation Results

For objects with simple shape geometry, such as the bottle (Fig. 19) and the sphere (Fig. 20), with just a few wraps, the estimated shape is already similar to the actual shape. In Fig. 19, the object shape uncertainty keeps decreasing (i.e., the blue parts in Fig. 19 keep increasing) as more wraps are conducted until all available wraps are exhausted. The maximum uncertainty measure u_j starts being 0.38 and reaches 0.32 at the end. Note that the part near the origin (where the robot arm base is) is more uncertain (colored mostly red), which is due to that the robot cannot reach this area as it is too close to the robot base. In Fig. 20, note how the fifth wrap (counted from left) in Fig. 20(d) helps reduce the uncertain (red) area in Fig. 20(a). The

POSE-CoM process is terminated after 7 wraps as u_j reaches 0.14 (below threshold 0.15).

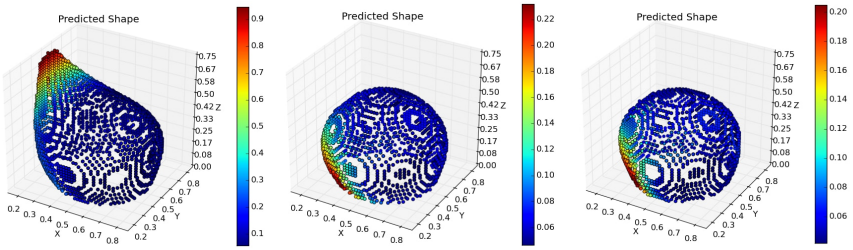


(a) 4 wraps (b) 8 wraps (c) 12 wraps

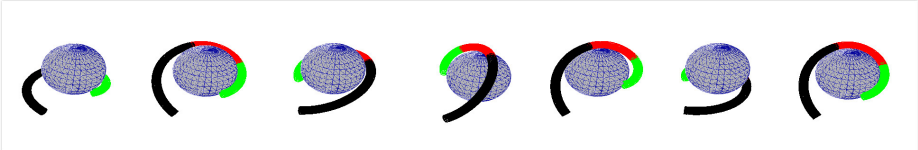


(d) Left to right: the first conducted wrap to the last conducted wrap.

Figure 19: Shape estimation results for the **bottle**, colored by uncertainty.



(a) 4 wraps (b) 5 wraps (c) 7 wraps

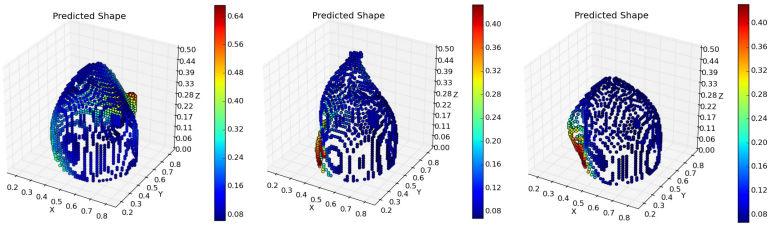


(d) Left to right: the first conducted wrap to the last conducted wrap.

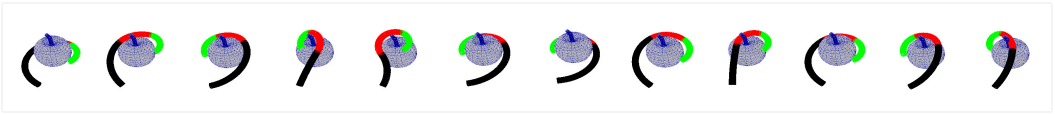
Figure 20: Shape estimation results for the **sphere**, colored by uncertainty.

For objects with more complex shapes, such as the apple (Fig. 21), bunny (Fig. 22), pentagon (Fig. 23), and hollow vase (Fig. 26), more varied wraps are needed to better capture certain details for more accurate shape estimation results. For

example, wraps are needed around the stalk of the apple, the ears of the bunny, and the entrance of the hollow vase, to better capture those details. However, the GPIS modeling tends to blur the connection between a detail and the main shape because the prediction assumes smooth connection. Thus, the estimation results of a complex shape tend to resemble certain bounding envelopes of the actual shape.

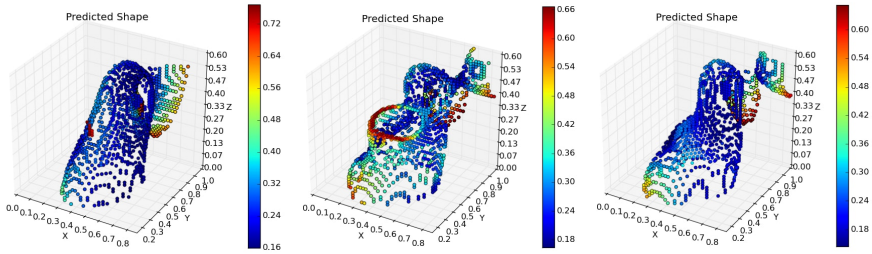


(a) 4 wraps (b) 8 wraps (c) 12 wraps

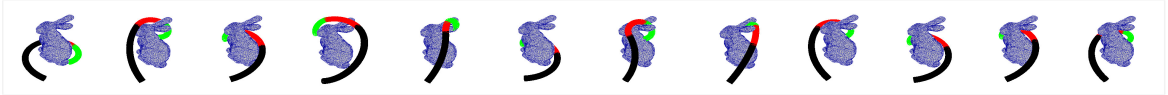


(d) Left to right: the first conducted wrap to the last conducted wrap.

Figure 21: Shape estimation results for the **apple**, colored by uncertainty.



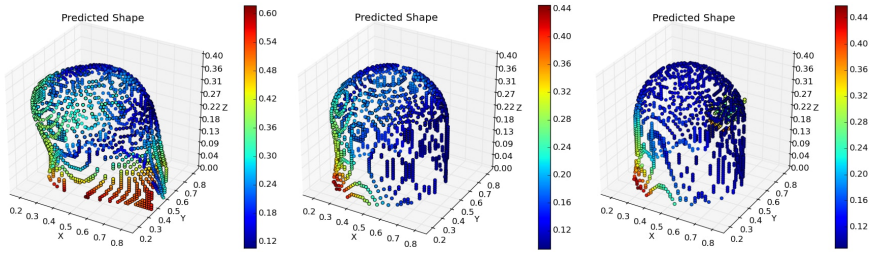
(a) 6 wraps (b) 9 wraps (c) 12 wraps



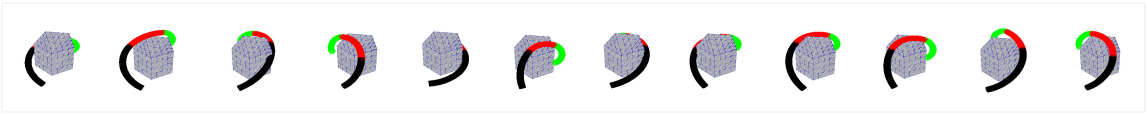
(d) Left to right: the first conducted wrap to the last conducted wrap.

Figure 22: Shape estimation results for the **bunny**, colored by uncertainty.

Table 3 shows that a total of 500 to 900 points are used to estimate the shape



(a) 4 wraps (b) 8 wraps (c) 12 wraps

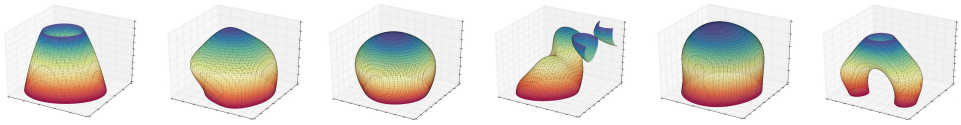


(d) Left to right: the first conducted wrap to the last conducted wrap.

Figure 23: Shape estimation results for the **pentagon**, colored by uncertainty.

Table 3: The total number of contact points and arm points collected on each object, the total number of wraps, and the average time T_{wrap} per wrap for (motion planning + collision detection).

Object	# contact points	# arm points	# wraps	T_{wrap}(ms)
bottle	143	150	12	135
apple	259	172	12	425
sphere	267	113	7	246
bunny	264	172	12	432
pentagon	141	167	12	132
vase	153	160	12	278



(a) bottle (b) apple (c) sphere (d) bunny (e) pentagon (f) vase

Figure 24: Object mesh models reconstructed from shape estimation results using Marching Cubes' algorithm [70], colored by elevation.

of an object[2]. It would be very time-consuming if those points were collected by a conventional manipulator making point contacts with the object [137, 20, 39, 147].

[2]Recall that one above-surface point and one below-surface point are generated for each contact point along the contact normal.

However, our approach is very efficient by using continuum wraps automatically generated from our motion planning algorithm. Table 3 also shows that it is very fast to plan a touch-based continuum wrap.

Fig. 24 shows the reconstructed mesh models from the shape estimation results using the Marching Cubes' algorithm [70]. As explained earlier, those models capture enveloping contours of the actual shapes because sharp details were blurred by the GPIS modeling under noisy observations.

If the reconstructed model is identical to the original mesh model of an object, given the same arm initial configuration, model location, and the wrapping plane, the final wrapping configuration C_r of the reconstructed model should be the same as the final wrapping configuration C_o of the original mesh model. Therefore, the distance of normalized configurations $\|C_r^{'} - C_o^{'}\|$, where $C_r^{'}$ and $C_o^{'}$ are obtained by normalizing the component values in C_r and C_o, is indicative of how accurate the shape estimation is. We use the average distance of multiple wraps as a parameter to measure the accuracy of shape estimation indirectly, as shown in Table 4. It is clear that model quality improves as more wraps are used to reconstruct the model. Fig. 25 compares example wraps visually.

3.3.2 Significance of the Arm Points

Fig. 27 shows the shape estimation results of the hollow vase using only contact points. Since the object is hollow, continuum wraps cannot make a large number of contacts, and thus the shape estimation result is inaccurate. However, the arm shape at a wrapping configuration itself is indicative of the object shape, and the arm points

Table 4: Average distances between the wrapping configurations of a reconstructed model and the original mesh model. # wraps indicates the number of wraps used to reconstruct the model.

Object	Intermediate Model 1		Intermediate Model 2		Final Model	
	# wraps	avg. dist.	# wraps	avg. dist.	# wraps	avg. dist.
bottle	4	0.31	8	0.27	12	0.27
apple	4	0.44	8	0.36	12	0.28
sphere	4	0.28	5	0.26	7	0.24
bunny	6	0.57	9	0.47	12	0.31
pentagon	4	0.34	8	0.19	12	0.16
vase	6	0.48	9	0.41	12	0.31

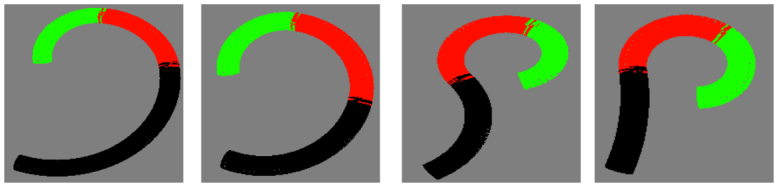
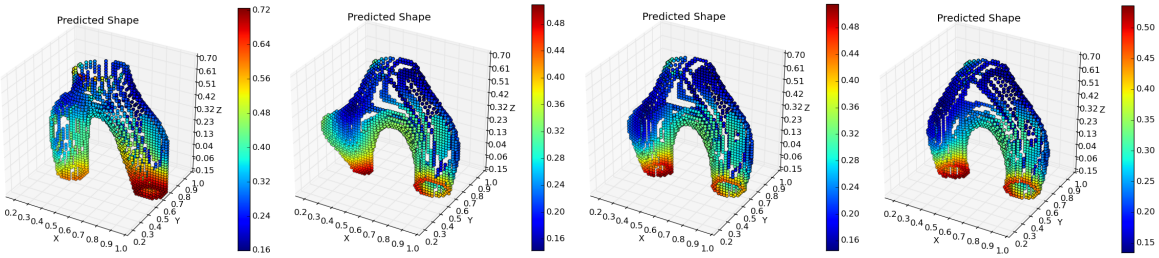


Figure 25: Comparing example wraps on the original mesh model and the reconstructed model. (a) wrap on the original apple model (b) wrap on a reconstructed apple model (c) wrap on the original vase model (d) wrap on a reconstructed vase model.

can be used to compensate for the lack of sufficient contact points and determine the object geometry. As shown in Fig. 26, our approach using both arm points (robot proprioception) and contact points better captured the vase shape and also captured the hollowness of the vase using as few as 6 wraps. Whereas, without using the arm points, as shown in Fig. 27, the hollowness of the vase could not be captured even with 12 wraps.

3.4 Summary

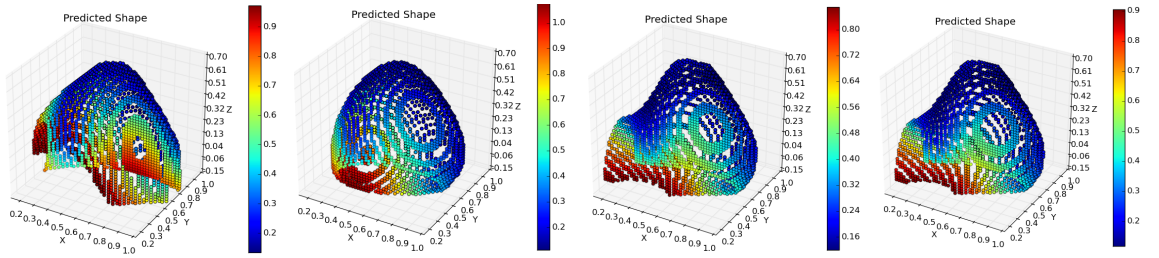
We have presented an approach POSE-CoM of progressive object shape estimation through touch-based continuum manipulation and a GPIS-based probabilistic model.



(a) 6 wraps (b) 9 wraps (c) 10 wraps (d) 12 wraps

(e) Left to right: the first conducted wrap to the last conducted wrap.

Figure 26: Shape estimation results of the **vase**, colored by uncertainty.



(a) 6 wraps (b) 9 wraps (c) 10 wraps (d) 12 wraps

Figure 27: Shape estimation results of the **vase** without using arm points, colored by uncertainty.

Unlike shape estimation based on contact points collected through the end-effector of a conventional, articulated manipulator, shape estimation through continuum manipulation is both more efficient and effective by deliberate use of the robot proprioception data in addition to contact information, as demonstrated in the experiments on objects of various shapes.

CHAPTER 4: SIM-TO-REAL TRANSFERABLE OBJECT CLASSIFICATION THROUGH TOUCH-BASED CONTINUUM MANIPULATION

In Chapter 2, a shape-based approach for object classification and recognition through continuum manipulation is introduced. The main idea is to use the continuum robot as a tool to indirectly "measure" the object shape. That is, the shape of a continuum arm during whole-arm wraps of a target object, which can be transparent (and thus not visible), is used to indirectly characterize the shape of the object and this information is used for classification and recognition. For an object of any shape, a continuum wrap is generated automatically by a touch-based approach. However, the work is tested only in simulation with the assumption that the continuum arm is covered by tactile sensors. In reality, existing continuum manipulators are often not equipped with tactile sensors. One difficulty is that the body of the robot is deformable. A kinematics-based contact detection and localization approach for continuum robots is presented in [8], for which an external tracking system is required but may not always be available in real-world scenarios.

In this chapter, we study touch-based identification of object categories using a new form of continuum manipulator consisting of origami-based modules [102] and tactile sensors attached at each section. Using these new manipulators with sparse tactile sensing, we aim to achieve touch-based wrapping of objects and experimentally validate the following conjecture: the shape-based classifier introduced in Chapter 2

and trained in simulation can be readily transferred to classifying real objects with real touch-based continuum manipulation. We envision the robot to be used in a search and rescue scenario where it could be exposed to low-light environment hence could benefit from the touch-based manipulation.

4.1 Approach

We next explain the manipulators we built for this study, sensors, and the touch-based motion planning strategy for generating continuum wraps, and the classification method of unknown objects.

4.1.1 Manipulators and Sensors

The continuum manipulators we built for this study consist of multiple origami continuum modules connected in series. Each continuum module contains a foldable origami body, three brushed DC micro-motors with pulley systems, and a controller board that offers on-board sensory measurements, feedback control, and module-to-module communication. The foldable body is made out of Polyethylene terephthalate (PET) films and constructed based on the Yoshimura crease pattern. This unique tubular structure with a diameter of 7 cm is capable of bending in two directions and extending/retracting, while maintaining its structure and resisting torsion. The foldable structure is connected to an acrylic plate on the top and to the PCB on the bottom where the motors are secured. Three nylon cables secured to the motor shafts and spanning the length of the structure along the edges are used to drive the segment. Each motor is equipped with a magnetic encoder for position control[3].

[3]My collaborator Junius Santoso and Cagdas Onal at WPI contributed to the design and fabrication of the continuum manipulator used in this study.

We built two continuum manipulators for this study. Fig. 28(a) shows the 3-section manipulator used in the experiments with planar wraps, and each section has an active white module with motors on it and a passive green foldable body to expand the robot workspace. Improved from the 3-section manipulator, Fig. 28(b) shows the 4-section manipulator used for generating spatial wraps benefiting to its more compact and lightweight modules and larger touch sensor contact areas.

Each section is characterized using three parameters (s, κ, φ), where s is the section length, κ is the curvature, and φ is the orientation angle [63, 102]. Using the inverse kinematics of a continuum section developed in [102], we can then find the required cable lengths (l_1, l_2, l_3), that will shape the module into the desired configuration $(s_d, \theta_d, \varphi_d)$. The cable lengths are converted into encoder positions, which are then sent to the low-level controller as reference signals.

We constructed the touch sensors acting as bumper switches using two copper sheets adhered onto a parallel plate structure made out of the same material used for the origami collapsible body. One copper sheet is connected to the control board's digital I/O pin while the other is connected to ground reference. When the copper sheets touch each other due to depression of the structure, an electrical circuit is completed hence signaling a touch on the continuum section. For each continuum section we place the touch sensor at the module and sandwiched between consecutive modules as shown in Fig. 28.

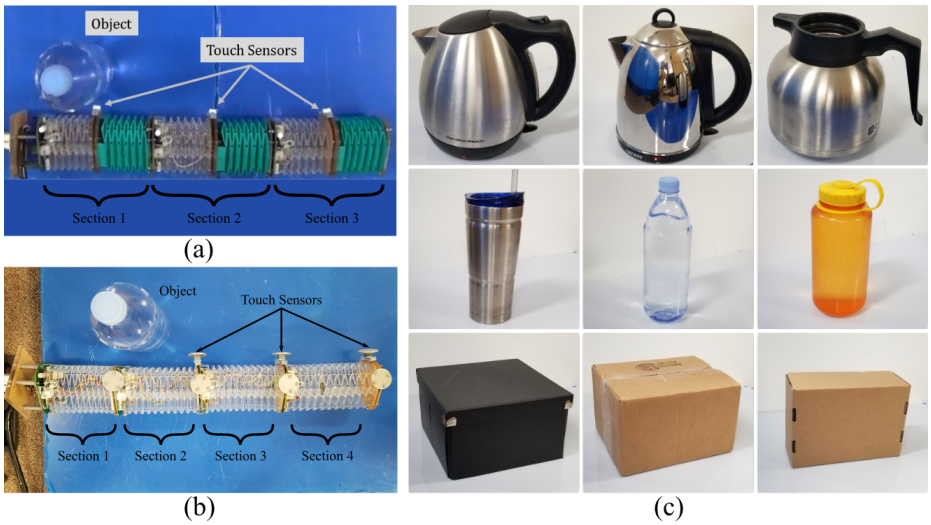


Figure 28: An object (a transparent water bottle) was placed near the base of arms for manipulation: (a) a touch sensor was mounted in the middle of each origami module in a 3-section arm, (b) a touch sensor was mounted at the distal end of each module in a 4-section arm. The diameter of the continuum section is 7 cm. (c) A list of objects used in the experiments.

4.1.2 Touch-based Continuum Wrapping

A general touch-based motion planning strategy is introduced in Chapter 2 for a continuum manipulator to progressively generate wraps around an object under the guidance of the contacts made along the way without knowing the object model. Starting from the initial configuration, the robot motion alternates between the *enclosing motion step* to make contacts with the object, and the *advancing motion step* to move forward towards wrapping around the object, until a wrap is formed or no further motion is feasible due to the physical limits of the manipulator. Such continuum wraps can be efficiently generated within hundreds of milliseconds (time of planning and collision checking combined) in simulation (Chapter 2).

To achieve an *advancing motion step*, as described in Chapter 2, contact localization

and estimation of the tangential and normal directions of the local contact patches are required; whereas, we relax this requirement in this chapter through extrapolating the robot section endpoints based on their local frames. Therefore, our planner only needs to know whether each manipulator section is in contact or not from the tactile sensing to plan the next move, which makes it more effective to guide the manipulator hardware to achieve touch-based continuum wraps.

4.1.3 Classification of Unknown Objects

As introduced in at the beginning of this chapter, we aim to experimentally validate a shape-based classifier using continuum wraps (Chapter 2). The shape of the continuum robot wrapping around an object is described by a chord histogram descriptor, which approximates the robot shape using many 3D chords and statistically captures its shape based on the chord parameters. We first trained a linear SVM classifier in simulation using wraps generated around the simulated objects and next applied the trained model to classify the real-world objects using real-world wraps. The objects in simulation were scaled to roughly match the dimensions of the real-world objects.

4.2 Experimental Results with Planar Wraps

In our experiments with planar wraps, we used the 3-section manipulator as shown in Fig. 28(a). The robot manipulator is initially set at a straight-line configuration with full contraction, and the testing object is placed near the arm base. For each section, $s \in [0.085, 0.145](m)$, $\kappa \in [-10.69, 10.69](1/m)$, and $\varphi \in [-\pi, \pi]$.

A linear SVM classifier is trained in simulation using wraps from 10 water bottles, 10 boxes, and 6 teapots. For each object one wrap was generated. Fig. 29 shows a

few planar continuum wraps around the objects in simulation. We next conducted classification of three different real-world objects: a teapot, a water bottle, and a card box, through real-world continuum wraps.

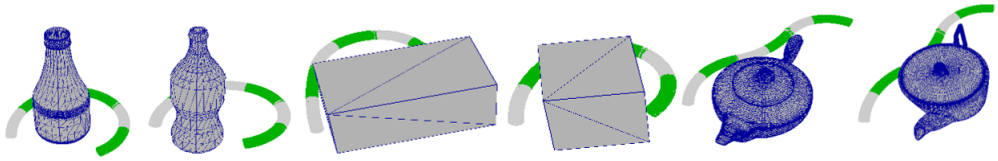


Figure 29: A few examples of the planar continuum wraps generated on different objects in simulation. The simulated arm has 3 sections and each section is colored in white for the first half and in green for the second half.

The video attached to [75] shows the wrapping process of the real-world objects, and Fig. 30 shows a few motion snapshots. Note that the goal of such wraps is to encode the shape of an unknown object into the shape of the manipulator, as opposed to achieving tight grasps of objects with known models [63]. Therefore, the wraps do not need to be enclosing.

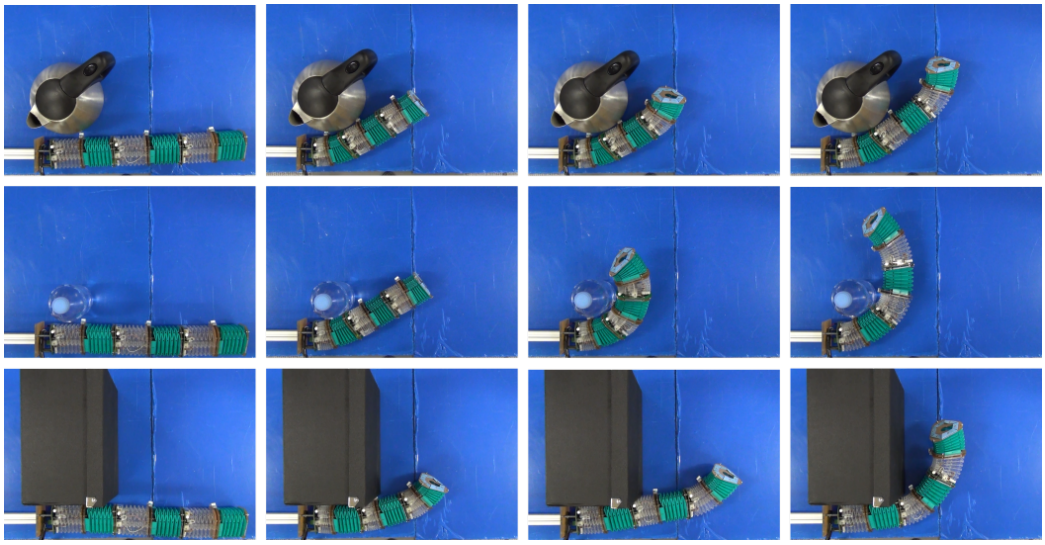


Figure 30: The motion snapshots of the continuum wraps generated on the real objects. The rightmost sub-figure in each row shows the final wrapping configuration.

The final robot configurations commanded by the planner were used for classification. Table 5 summarizes object and wrap information and the classification results. It can be seen that the classifier learned solely from simulation is already effective in classifying the real-world objects, as the probabilities of correct classification are more than two times higher than that of a random guess (about 0.33). However, the box was mis-classified to be a bottle because the contact on section 1, which was closest to the base, was missed by the sensor, and therefore the planner kept commanding section 1 to bend more while it was actually stopped by the contact. Since the curvature of section 1 is a distinctive feature for classification as shown in Fig. 30, the classifier with the inaccurate data of a larger curvature resulted in the misclassification.

Table 5: Object dimension, number of intermediate configurations to generate the wraps, SVM prediction and its probability using 1 planar wrap for each object.

Object	dimension(cm)	# of configurations	SVM Prediction	Probability
bottle	$8 \times 8 \times 24$	59	bottle	0.81
box	$27 \times 16 \times 26$	61	bottle	0.83
teapot	$24 \times 18 \times 21$	42	teapot	0.71

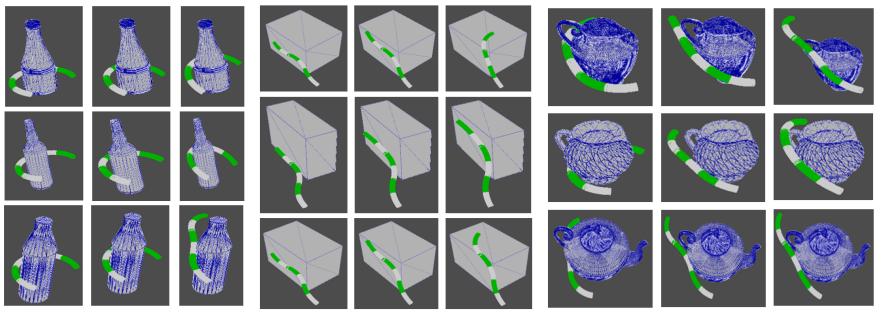
4.3 Experimental Results with Spatial Wraps

We use the 4-section robot arm (Fig. 28(b)) to conduct spatial continuum wraps around the objects to collect spatial shape information. The arm sections of this robot are more compact (rather than having two modules connected as one robot section) and lightweight, which makes it more suitable for spatial wraps. It also overcame the problems of missed contacts with increased contact areas of the touch

sensors. Different wraps covering different areas on the objects were conducted to collect object spatial shape information. The 3D chords generated from different wraps are accumulated into one histogram as an overall representation of the object shape. We next explain how the robot arm is lifted to conduct spatial wraps and present the object classification results.

4.3.1 Robot Arm Lifting

The arm is lifted up by keeping the first module of the robot at a certain configuration during the experiment and then changed when switching to other wrapping planes. See Fig. 32(b) for such examples when the robot arm is lifted up. The benefit of having this additional module is that the other modules (section 2, 3 and 4) can be fully used for generating the wraps, and they only need to undergo less strain and stress when wrapping around the objects. Alternatively, the modules could be mounted vertically to minimize the effect of gravity; however this configuration cannot always be possible in a real life scenario, hence we decided to proceed with the former.



(a) bottles (b) boxes (c) teapots

Figure 31: The spatial wraps in simulation. Each row in each subfigure is the three wraps around the same object. The simulated arm has 3 sections and each section is colored in white for the first half and in green for the second half.

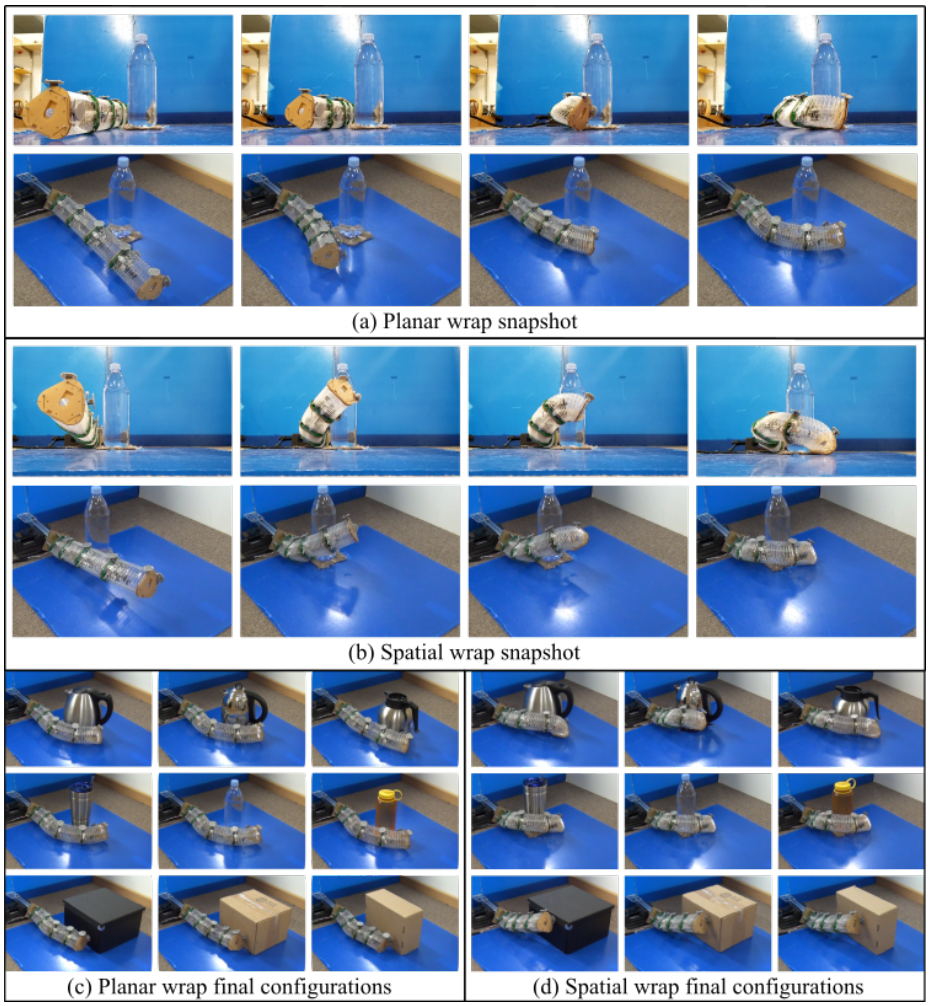


Figure 32: The motion snapshots of (a) planar and (b) spatial continuum wraps around the bottle, where the top and bottom rows show the motion of the same wrap from two view angles respectively, (c) final configurations for the remaining objects with planar wraps, (d) spatial wrap final configurations.

4.3.2 Classification Results

We trained a linear SVM classifier using the same set of training objects used in Section 4.2, and the shape of each object was captured using 3 wraps. Fig. 31 shows the wraps around the training objects generated in simulation. Fig. 32 shows examples of such planar and spatial wraps around the real world objects. For testing dataset, we considered 3 object categories and 3 objects from each category (Fig.

28(c)).

The video attached to the original paper [75] shows the motion process of the continuum wraps. As mentioned earlier, such wraps are the result of local motion generation, and they are just used to encode the object shape onto the robot arm shape. Therefore, some (for instance the wraps around the boxes) may only locally conform the robot shape to the object shape and achieve partial wrapping of the object.

We noticed that the robot final wrapping configurations deviate from the robot motion commands sent by the motion planner (major cause for the mis-classification of the box in Section 4.2). There are two main reasons. First, our robot arms currently do not have the proprioceptive sensors to achieve precise closed-loop control. Second, some contacts may be missed by the current sparse touch sensing on our robot arms. Therefore, in order to more precisely identify the final robot shape, we used an external vision tracking system to identify the robot final configuration when the wrapping process is terminated. This wrapping configuration is next used for generating the 3D chords and conducting the final object category classification.

Table 6 summarizes the objects used, the average number of robot configurations to generate the wraps, and the SVM prediction results. Overall, the classifier was able to correctly recognize 2 bottles, 3 boxes, and 2 teapots. The boxes are easier to be correctly identified since the robot arm conforms to the side surfaces of the boxes and therefore has distinctive straight sections in contact. The wraps of the bottles and the teapots typically have more curved robot sections but differ in lengths due to their dimensions. The confusion of classifying the bottles and the teapots is because

sometimes the handles of the teapots (more distinctive features) are not captured. This can be improved by using longer robot sections and more dense touch sensing.

Table 6: Object dimension, average number of intermediate configurations to generate one wrap, SVM prediction and its probability using 3 spatial wraps for each object.

Object	dimension(cm)	avg. # of config.	SVM Prediction	Probability
bottle1	$8 \times 8 \times 24$	45	bottle	0.52
bottle2	$9 \times 9 \times 21$	43	teapot	0.56
bottle3	$8 \times 8 \times 21$	40	bottle	0.51
box1	$27 \times 16 \times 26$	12	box	0.74
box2	$25 \times 21 \times 17$	10	box	0.71
box3	$27 \times 10 \times 18$	15	box	0.78
teapot1	$24 \times 18 \times 21$	36	teapot	0.46
teapot2	$21 \times 15 \times 21$	40	teapot	0.51
teapot3	$22 \times 16 \times 16$	38	bottle	0.51

4.4 Summary

Our results have demonstrated that the shape-based classifier trained solely from simulation is able to generalize to real-world objects. This confirms our two key insights. First, because object classification is based on the shapes of the continuum arm wrapping around the objects and not the shapes of the objects directly, the classifier has the advantage of avoiding direct sensing and perception of the shape of an unknown target object as well as the associated limitations (such as low object visibility) and all the sensing uncertainties involved that can negatively affect classification accuracy. Second, the continuum wraps generated on objects in the same category have similar shapes, which are captured by the intrinsic parameters of the continuum arm, no matter if the objects wrapped are virtual or real.

Since conducting many real-world continuum wraps can be time-consuming, it is

significant that the classifier trained purely in simulation showed considerable effectiveness in classifying real objects. This could make classifier training more efficient and feasible for classifying a large number of categories of many real objects from touch-based continuum wrapping.

CHAPTER 5: PROGRESSIVE OBJECT MODELING WITH A CONTINUUM MANIPULATOR IN UNKNOWN ENVIRONMENTS

In this chapter, we address autonomous and progressive model building of an object in an unknown, cluttered environment using a multi-section continuum manipulator. 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.

5.1 Related Literature

In related work, researchers have studied view planning [127, 126] to find the best set of views to capture the surface appearance of an isolated object, object model building[127, 89] of an isolated object, and object manipulation through perception and learning[36], compliance with soft hands [14] or through exploiting environment constraints [28] without explicit model building. Some recent work of object modeling[49, 73, 16] uses a robot hand to hold the object for pose change, but the hand can also occlude the object. More recently, interleaving visual perception and robotic manipulation is studied in [117] to achieve complete object modeling. 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 [100, 29], 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 [101, 122, 130] can deform continuously and are inherently compliant [82, 83], 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) [124, 120, 61, 64] or fully visible through some external sensor (such as an overhead camera) [57], and only recently, manipulating a known object situated in an unknown, cluttered environment is addressed [56]. There is no work on handling an object without a known model in a cluttered space with a continuum manipulator.

5.2 Overview

We first present the environments and tasks considered in this chapter, and next provides an overview of the introduced approach.

5.2.1 Environment and Task

The environment considered in this study 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[4].

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 study.

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. 34. The same kinematics model of an n-section continuum manipulator described in Chapter 2 is used.

5.2.2 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. 33 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 5.2.1), the robot's tip camera is adjusted to first sense the environment and distinguish the target object among the sensed obstacles (Section 5.3.1). Then the tip camera is steered towards the target

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

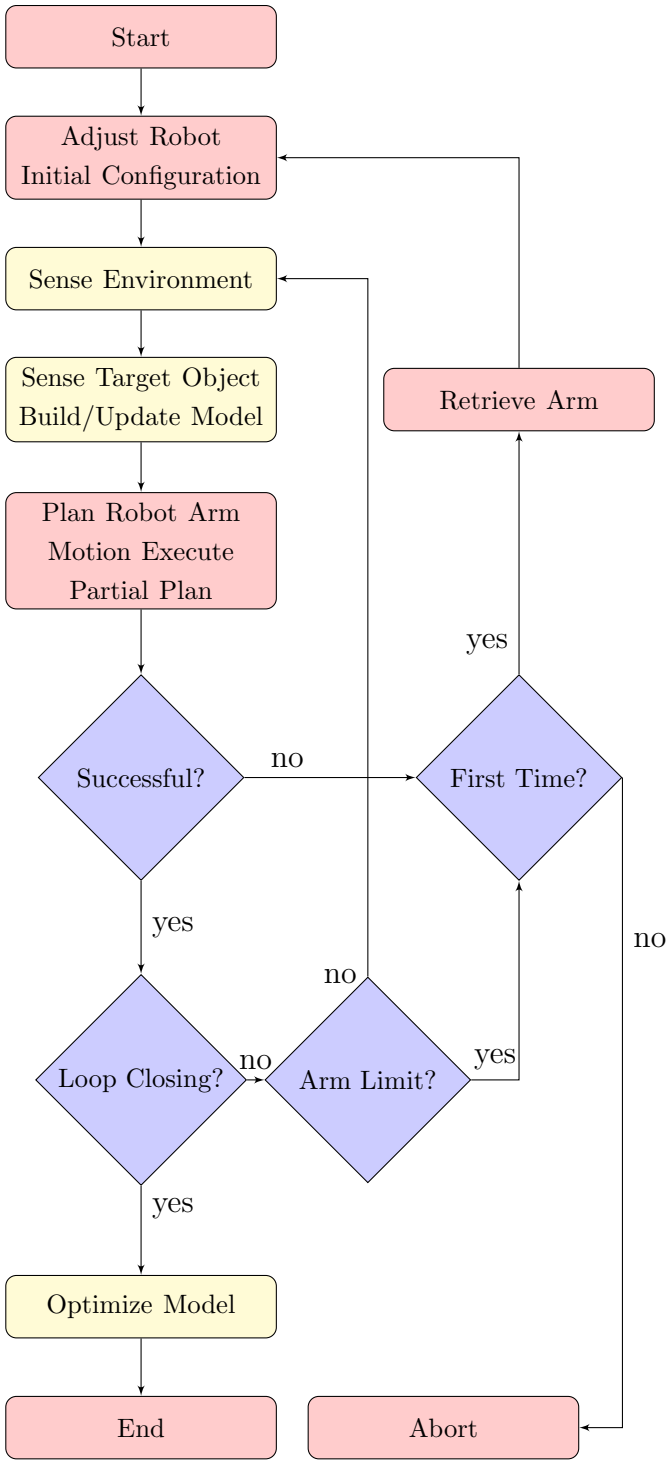


Figure 33: Overview of the progressive modeling approach. The blocks for perception are in yellow, and those for robot motion are in red.

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 the object (see Section 5.4).

Next, the robot arm motion is planned and executed based on the updated partial object model for a short distance (See Section 5.3.2).

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.

5.3 Perception-based Motion Planning and Execution

We first explain to how to distinguish target objects and obstacles, and next explain the approach of robot motion planning and execution.

5.3.1 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. 33), 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.

5.3.2 Planning and Execution of Robot Arm Motion

Starting from an initial configuration of the continuum manipulator, **Algorithm 2** 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 [56]. Collision detection between the arc and obstacles or object is checked efficiently based on the algorithm in [62].

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 [61]. 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 2** 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 2** 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 2** 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 or without one because our model building strategy (see Section 5.4) 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 2** exits and reports no motion.

5.4 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

Algorithm 2: 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;
```

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:

1. 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;

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

My collaborator Zhou Teng contributed to this section. Please see original paper for detailed description [76] of the method used in this section.

5.5 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 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.

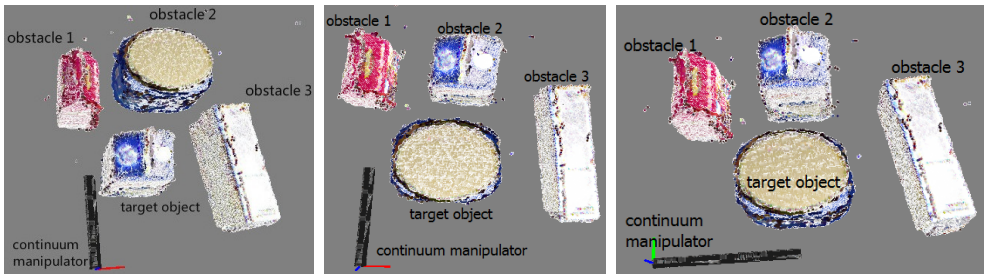
5.5.1 Progressive Object Modeling with a Continuum Arm

From each real RGB-D image of the target object, the corresponding 3D object point cloud is obtained[5] 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[6] 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. 34 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. 34(a)) and **Task 2** (Fig. 34(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

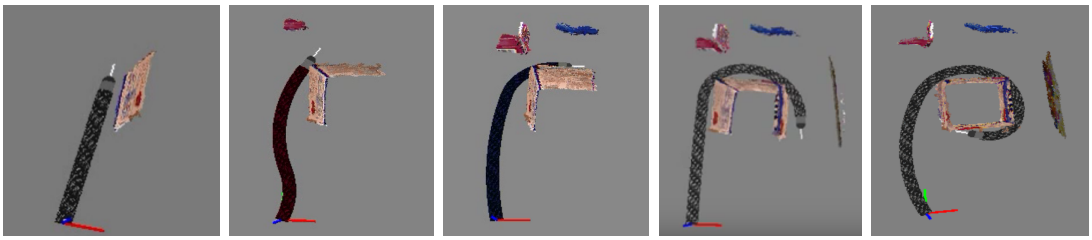
[5]Since the tip camera is close to the target object, only the target object (no obstacles) appears in the real RGB-D image, which simplifies segmentation.

[6]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

Figure 34: Top view of environments with different target objects, obstacles, and robot arm initial configurations.



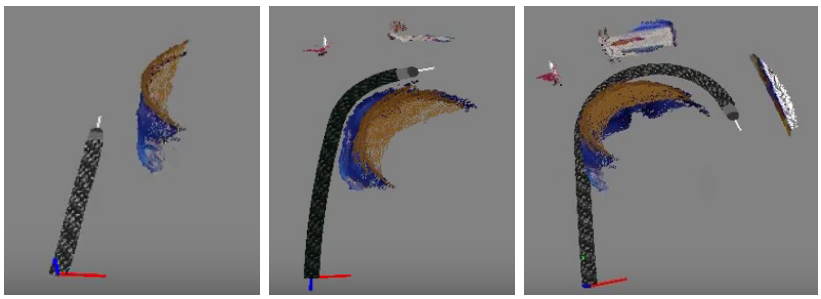
(a) Initial arm configuration. (b) The arm follows the planned arc. (c) The planned arc is updated. (d) The arm follows a new arc. (e) The arm finishes observing the object.

Figure 35: Snapshots of the arm motion and the sensed object surfaces in **Task 1**.

counter-clockwise, which we call subtasks **Task 2-1** (Fig. 34(b)) and **Task 2-2** (Fig. 34(c)) respectively.

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. 35(b)). As more sensings are enabled, an updated arc is planned (Fig. 35(c)) for the robot arm to follow until a new arc is necessary (Fig. 35(d)). In the end, the robot arm is able to make a closed loop coverage of the side surface of the milkbox (Fig. 35(e)). The final refined model after global optimization is shown in Fig. 39(a). Note that the milk

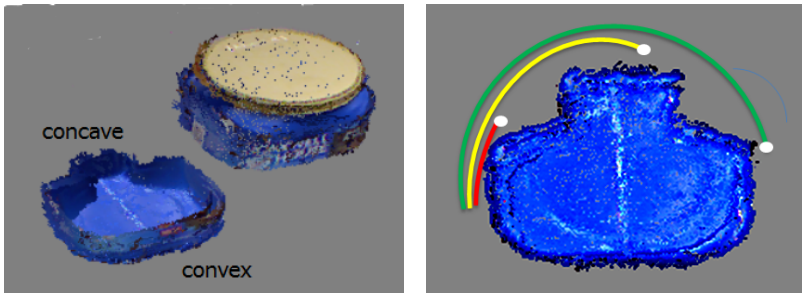
box is too tall for the tip camera to sense the top surface.



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

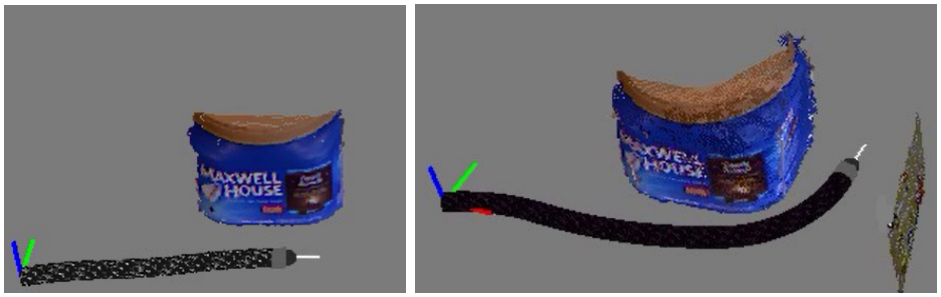
Figure 36: Snapshots of the arm motion in **Task 2-1**.

In **Task 2-1**, the arm gradually moves around the object from the concave side in clockwise direction (Fig. 36). As shown in Fig. 37(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.

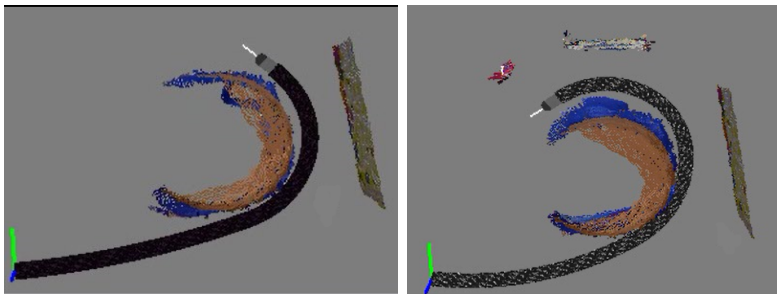


(a) Coffee can has both concave and convex sides. (b) Cross section top view. The arm sequentially follows the red, yellow and green arc.

Figure 37: 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.

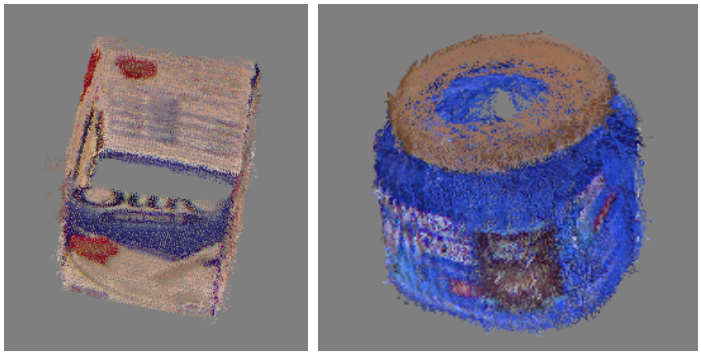
Figure 38: Snapshots of the arm motion in **Task 2-2**.

Table 7: 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. 38, 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. 39(b)) of the side surface is obtained, which also includes some portion of the top surface.

Table 7 shows the number of images captured, the short distance Δs used, and the



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

Figure 39: The built models.

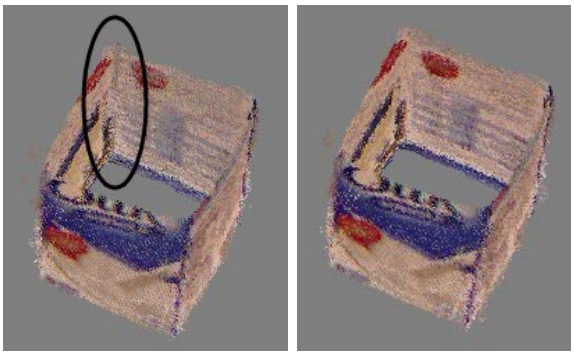
total time for planning the robot arm motion in the experiments. Δs is set to be both sufficiently small to ensure a sufficient overlap between the images captured in two consecutive sensings and also large enough to be efficient. Choosing Δs as 3cm in Task 1 and 6cm in Task 2 provides roughly a 25° 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 video attached to [76] shows the 3D arm motion of our presented experiments here.

5.5.2 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. 40, 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. 40(b)).



(a) Before (b) After

Figure 40: 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).

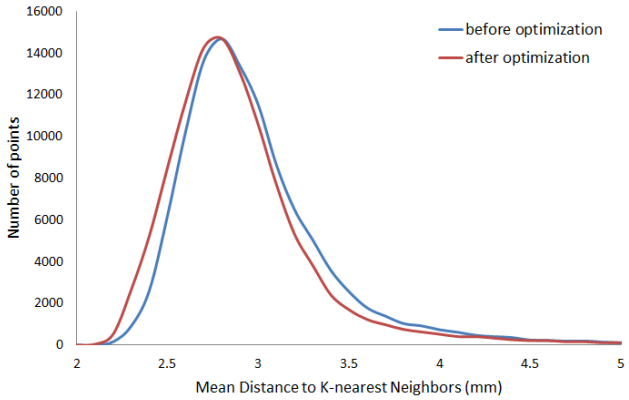
Two statistical analyses are conducted to quantify the improvement of the model quality by applying the global optimization algorithm to reduce the accumulative error of the pairwise registration. In the first analysis, the mean distances of all the points to their K-nearest neighbors are computed for the models obtained before and after the global optimization. As shown in Fig. 41(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. 40 again for example).

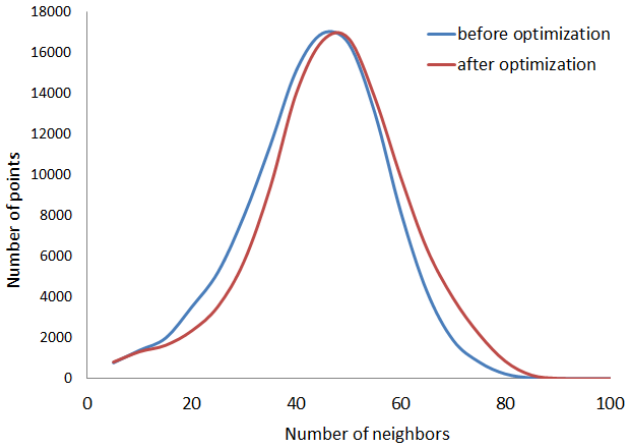
The second analysis is based on the number of neighbors of all the points, which are searched in the radius of $3mm$. As seen from Fig. 41(b), the plotted curve of the refined model is shifted to the right compared to the model before the optimization. Therefore, the points in the refined model have more neighbors within $3mm$ radius, which also shows that the points are located in the closer proximity of their neighbors.

5.6 Summary

This chapter 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 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.



(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.

Figure 41: The statistical analyses of the models obtained before and after the global optimization.

CHAPTER 6: REAL-TIME CONFLICT RESOLUTION OF TASK-CONSTRAINED MANIPULATOR MOTION IN UNFORESEEN DYNAMIC ENVIRONMENTS

In order for robots to work in human-centered environments, they must be able to perform tasks under constraints imposed by the tasks while avoiding unforeseen dynamic obstacles (such as moving human customers and moved chairs in a restaurant) at the same time. For instance, to bring a glass of water to a customer, a robot manipulator has to maintain the task constraints in terms of keeping orientation angles of the end-effector within a certain range to prevent spilling (see Fig. 42 for an example) while avoiding unforeseen moving obstacles. To open or close a drawer, the robot manipulator has to constrain both the position and orientation of the end-effector to move within the confinement of the cabinet while also avoiding unforeseen moving obstacles. Although there exist algorithms for planning task-constrained motion and for planning robot motion in dynamic environments, they are not best equipped to tackle the general challenge of resolving conflicts between satisfying task constraints and avoiding unforeseen dynamic obstacles.

In this chapter, we introduce an algorithm that incorporates task constraints in the general RAMP approach [125] to (1) planning task-constrained manipulator motion in environments with dynamically unknown obstacles, and (2) seamlessly resolving the conflicts between task constraints and obstacle avoidance by allowing the robot to change goals on-the-fly through releasing and resuming task constraints. The

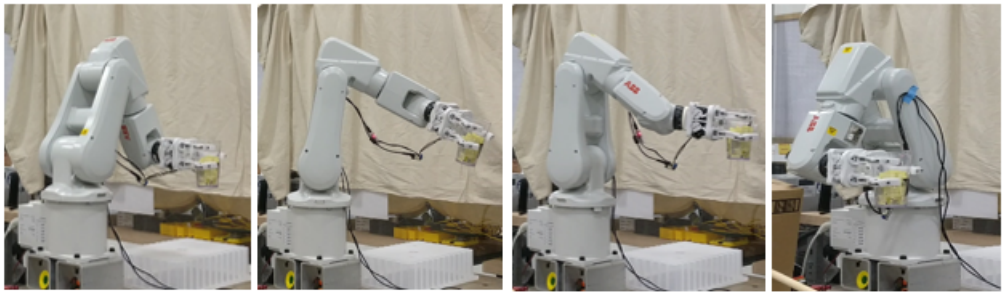


Figure 42: Transferring a water cup to a goal location. The end-effector has to constrain orientation angles to prevent spilling water.

new algorithm, which we call *task-constrained RAMP*, takes advantage of the core strength of RAMP to continuously improve diverse manipulator trajectories so that the robot can avoid unforeseen dynamic obstacles through rapidly switching to a better trajectory while performing task constrained motion safely. In doing so, the algorithm enables conflict resolution to allow the robot temporarily released from task constraints in a safe way for both the robot and the environments, which is adaptive to circumstances, and resume the task constrained motion whenever possible. In [23, 52, 93], as summarized in Table 8, such flexibility of allowing the tasks to be released and resumed again is not provided, hence the robot will have to come to an emergency stop due to imminent collision if the conflicts between task constraints and obstacle avoidance arise and persist. However, emergency stops can also make the robot more vulnerable to being hit by obstacles. Our *task-constrained RAMP* algorithm avoids undesired robot stops by enabling the manipulator to change goals on the fly to avoid the obstacles under reduced or no task constraints before resuming the task.

The effectiveness of *task-constrained RAMP* is verified on a six degrees-of-freedom

manipulator with a fixed base (i.e., a non-redundant manipulator) performing two tasks: transferring a cup of water (which imposes orientation constraint on the end-effector) and closing a drawer (which imposes position and orientation constraint on the end-effector) in an environment with a dynamic obstacle of unknown motion. For both tasks, our experiments show that the introduced algorithm resolves the conflicts between the task constraints and obstacle avoidance seamlessly. Both simulation and real-world robot experiments are conducted for validation.

6.1 Related Literature

Planning with task constraints: most of the literature does not consider dynamic environments. Sampling-based methods are focused on how to efficiently acquire configuration space samples that satisfy specified task constraints[48, 109, 9, 110, 38, 32, 84, 23]. Jacobian-based projection operators have been shown to be effective in achieving such goals[109, 9], where iterative procedures are used to gradually project a free configuration sample onto the constraint manifolds. There is also work to approximate or parameterize the constraint manifolds in different ways [110, 38, 32, 84]. However, the approximated or parameterized constraint manifolds may not be feasible in unforeseen dynamic environments of moving obstacles.

Optimization-based methods take an initial trajectory and try to locally optimize the trajectory under multi-criteria objective functions[148, 99, 44]. Jacobian pseudo-inverse based optimization[148] achieves this in a local reactive manner and may be stuck when different criteria or constraints are conflicting, which requires a careful design of mechanisms for switching priorities [106, 86]. Covariant Hamiltonian op-

timization for motion planning (CHOMP)[99], stochastic trajectory optimization for motion planning (STOMP)[44], and many of their variants [19, 94] require precomputed representation of the environments to conduct gradient-based or gradient-free local optimization, which may not be real-time for handling dynamically changing and uncertain environments.

Table 8: Comparison of literature on task-constrained planning in dynamic environments.

Ref.	Online sensing	Data structure	Methods	Real experiments
[23]	No	Tree	Sampling	No
[52]	Yes	Road map	Local controller	Yes
[93]	Yes	Road map	Optimization	Yes

Planning in dynamic environments without task constraints: Some variants of sampling-based methods, such as dynamic rapidly exploring random trees [30, 92, 112] and dynamic probabilistic road maps [53], maintain a group of trees or graphs and conduct fast replanning and rewiring in reaction to the dynamic changes in the environment. In the spirit of artificial potential field, the elastic strip method[15] allows the trajectories to be deformed locally in the same homotopy class to avoid dynamic obstacles. The elastic road map method[136, 52] maintains a low-dimensional workspace graph that can move the nodes (i.e., milestones) in response to dynamic changes. Reactive control methods (for example [133, 45]) have been successful in many visual-servo tasks (like playing ping-pang) by locally adapting to new changes. Reinforcement learning based control approaches (such as [96]) also provide a new way for online motion generation of 3D reaching tasks at the cost of long training time, generalization difficulty and non-smooth robot motion. The real-time adaptive

motion planning (RAMP) approach [125, 81] simultaneously conducts planning and execution to adapt to dynamic environments of unforeseen moving obstacles.

Planning in dynamic environments with task constraints: In [23], an RRT-based planner was presented and task constraints were satisfied through generating samples in joint space. However, this is an offline algorithm, and it requires that the obstacle trajectories are known beforehand. There is also work on planning the motion of a mobile manipulator with task constraints in dynamic environments [52], taking advantage of the many redundant degrees-of-freedom to facilitate obstacle avoidance under task constraints. This approach consists of local reactive control based on sensing and higher-level global planning integrated in a road map data structure. However, little data and discussion were reported on how sensing, local reactive control, and global planning interact to satisfy task constraints and obstacle avoidance. Moreover, local reactive control can fail to resolve the conflict between task constraints and obstacle avoidance especially if the robot does not have a sufficient number of redundant degrees of freedom. In [93], multiple trajectories extracted from a pre-built road map were optimized in parallel to search for a feasible solution. However, the planning times reported in static environments were about 10 to 25 seconds. Hence, it is not fast enough to deal with unknown dynamic obstacles in an environment.

6.2 Task Constraints

When a robot manipulator performs a task, it usually involves the end-effector holding an object or tool. Therefore, we consider the *goal of a task* as characterized by two aspects: (1) the constrained attachment of an object or tool of the task to the

robot end-effector, and (2) the constraints imposed on the pose of the end-effector because of the attachment, which we call the *task constraints*.

Task constraints are considered as kinematic restrictions on the pose of a robot end-effector [109]. Denote a pose $[x, y, z, \alpha, \beta, \gamma]$ that specifies the end-effector position $[x, y, z]$ and orientation roll α, pitch β, and yaw γ in the task frame. We adopt the concept of *motion constraint vector* C from [109] for specifying the task-specific constraints, where $C = [c_x, c_y, c_z, c_\alpha, c_\beta, c_\gamma]^T$, and each element c in C is 0 if the corresponding motion is free and 1 if constrained.

In this study, we also consider soft task constraints that can be violated to a small extent. For example, when the end-effector holds a half-filled cup of water, it is allowed to tilt the cup to some extent. Relaxing task constraints allows more motion solutions, especially for planning in dynamically unknown environments.

6.3 A Review of RAMP

The real-time adaptive motion planning (RAMP) framework is a general, stochastic framework inspired by evolutionary computation and applicable to anytime motion planning of high degrees-of-freedom robots (e.g., mobile manipulators) in configuration $\times$ time (CT) space with unknown dynamic obstacles. A robot trajectory in RAMP is represented, at the high level, as a sequence of configurations, called *knot* configurations, connecting the robot initial configuration to a goal configuration, with appropriate time stamps added to each knot configuration. At the lower level, a polynomial trajectory segment connects any two adjacent knot configurations. Each trajectory is associated with a cost function taking into account multiple optimization

criteria. Motion planning and execution are conducted simultaneously. A group of trajectories, called a population, is maintained and updated all the time as the robot moves. The population consists of trajectories that are diverse and different from one another to result in a global representation of the CT space. A trajectory is *feasible* if it is collision-free; otherwise, it is *infeasible*. RAMP allows the population of trajectories to include both feasible and infeasible ones. The core idea behind RAMP is to enable trajectory switching to rapidly adapt to unforeseen changes in dynamic environments: the trajectory being executed can always be changed (before its first collision arrives) to the new best trajectory as the environment changes. Simultaneous planning and execution are achieved through interactions of the following three types of cycles:

1. *Sensing cycles* repeatedly update perception of the environments to capture changes.

2. *Planning cycles* repeatedly update and improve the trajectories based on sensing.

3. *Control cycles* repeatedly determine the best trajectory to execute next and switch the robot to that.

In each planning cycle, a randomly selected trajectory is updated by one of the following randomly selected genetic operators [125]: randomly *Insert*, *Delete*, or *Change* a knot configuration, *Swap* the order of two randomly chosen adjacent knot configurations, *Crossover* two trajectories, and *Stop* robot motion at a randomly chosen knot configuration for a random duration.

6.4 Task-constrained RAMP

We now incorporate task constraints into the RAMP framework.

6.4.1 Task-constrained Genetic Operators

Consider a manipulator arm. We use the following two new task-constrained operators in addition to the original operators [125] to update its joint trajectories.

- *Task-constrained Insert*: Randomly generate and insert a new knot configuration that satisfies the task constraints into a path.
- *Task-constrained Change*: Randomly change a knot configuration to a new one that satisfies the task constraints.

To obtain manipulator joint configurations satisfying the task constraints, we simply sample end-effector poses that satisfy the task constraints in the workspace and use inverse kinematics to obtain the corresponding joint configurations. If there are many inverse kinematics solutions given one end-effector pose, we randomly select one among all the solutions. Unlike directly sampling task-constrained joint configurations using computationally expensive and iterative procedures[109, 9, 110], our strategy is simple and efficient but effective in achieving real-time adaptive performance in dynamic environments.

Since RAMP has many planning cycles and the genetic operators are used many times, it can tolerate sampling invalid end-effector poses with no inverse kinematics solutions, which sometimes happen, as shown in the experiments.

6.4.2 Three Types of Goals

We define the following three types of goals:

1. main task-constrained goal g_{tc_main}: task-constrained robot goal configurations.
2. temporary task-constrained goal g_{tc_temp}: task-constrained robot configurations designated as temporary goals.
3. non-task-constrained goal g_{ntc}: robot configurations not satisfying task constraints.

For example, in the task of transferring a water cup, g_{tc_main} means the final robot configurations to place down the cup and g_{tc_temp} means other robot configurations that are also suitable for putting down the water cup. g_{tc_temp} is useful for conflict resolution as further explained in Section 6.5.

At a g_{ntc} configuration, the robot is either separated from the constraining object/tool or the object/tool no longer constrains the end-effector, e.g., an empty cup or a paint gun not emitting paint. g_{ntc} configurations are initially randomly generated and updated later when the robot needs to release and resume task constraints (as further explained in Section 6.5).

Task-constrained trajectories end at configurations of g_{tc_main} or g_{tc_temp}; whereas non-task-constrained trajectories end at configurations of g_{ntc}.

6.4.3 Trajectory Generation and Evaluation

An initial population of arm trajectories are generated consisting of those from an initial robot configuration to either a specified task-constrained goal configuration

or a randomly sampled non-task constrained goal configuration. Intermediate knot configurations (which are task-constrained or non-task-constrained, depending on the type of goal) are randomly generated on each trajectory. Next, time stamps are added through connecting adjacent knot configurations by cubic splines, taking into account the acceleration and velocity limits of a robot manipulator and their continuity.

We use a universal evaluation function *Cost* to measure the goodness of either a feasible (i.e., collision-free) or an infeasible trajectory. A trajectory is evaluated by considering the following cost terms: total duration D, total kinetic energy $E = \sum e_i$, total cost $V = \sum v_i$ of task constraint violation, total cost $M = \sum m_i$ for manipulability, total cost $S = \sum ts_i$ for trajectory smoothness, and a collision cost H, where e_i, v_i, m_i, and ts_i are defined below for the ith configuration $\mathbf{q}_i$:

1. $e_i = \frac{1}{2} \sum_{j=1}^{j=n} I_j {w_j}^2$, where I_j is the moment of inertia of joint j, w_j is the angular velocity of joint j, and n is the number of joints.

2. v_i is the difference between the ith end-effector pose and the task-constrained pose expressed in position and Euler angle differences.

3. $m_i = \frac{1}{\sqrt{detJJ^T}}$, which is the inverse of the manipulability measure[138]. J is the Jacobian matrix at $\mathbf{q}_i$. The closer to a singularity, the larger is m_i.

4. $ts_i = \|\mathbf{q}_{i-1} - 2\mathbf{q}_i + \mathbf{q}_{i+1}\|^2$ [44].

H equals zero if the trajectory is collision-free and equals to $\frac{Q}{t_{collide}}$ if the trajectory is not collision-free and has an earliest predicted collision at time $t_{collide}$. Q is a very large penalty value.

The overall *Cost* of a trajectory is computed as a normalized weighted sum of D, E, V, M, S, and H. A trajectory is task-constrained if it ends at a task-constrained configuration, and its V cost is less than a threshold V_{th}. The overall cost function promotes trajectories that are time and energy efficient, deviate as little as possible from the hard task constraints, has high manipulability, and stay far away from predicted collisions.

6.4.4 Trajectory Subpopulations for Conflict Resolution

Key to the adaptive capability of RAMP to environmental changes and effective conflict resolution is to maintain a diverse population of trajectories, such that whatever changes occur in the environment or conflicts arise among the tasks, there are most likely always trajectories suitable to the newly changed environment. The original RAMP [125] maintains subpopulations of certain homotopic diversity to be adaptive to unexpected moving obstacles. In this work, we use the following three new types of subpopulations,

1. subpopulation P_{tc_main} containing task-constrained trajectories ending at g_{tc_main}.

2. subpopulation P_{tc_temp} containing task-constrained trajectories ending at g_{tc_temp}.

3. subpopulation P_{ntc} containing non-task-constrained trajectories ending at g_{ntc}.

P_{tc_temp} is used to promote the population diversity for better adaptability to the unknown environments and also prepares the manipulator for conflict resolution when no trajectories in P_{tc_main} are collision-free. For example, if the manipulator cannot reach the final goal to deliver the cup without spilling water, RAMP may drive the

robot to reach places corresponding to g_{tc_temp} configurations while still maintaining the task constraints. When the manipulator reaches such places, RAMP may even release the robot from task constraints by letting the manipulator place down the cup so that it is free to conduct non-task-constrained-motion to avoid obstacles.

We allow these different subpopulations to co-exist in RAMP for achieving conflict resolution seamlessly between task constraints and obstacle avoidance, as further explained in Section 6.5.

6.5 Summary of Task-constrained RAMP

After the initialization of trajectory population P, which includes a task-constrained subpopulation $P_{tc} = P_{tc\_main} \cup P_{tc\_temp}$ and a non-task-constrained subpopulation P_{ntc}, P is further evolved and improved through a number of iterations of offline planning cycles to increase the overall fitness of trajectories (similar to Algorithm 3 Offline Planner in [125]).

Two global flags F_{tc} and $F_{release}$ are used in our algorithm. F_{tc} indicates whether the robot is currently task constrained. It is initially set to true and may be toggled as the robot moves to enable the switch between task-constrained and non-task-constrained motions. $F_{release}$ indicates whether the task constraints can be released at any time. It is determined based on the task requirements and fixed during the task execution.

Next the robot starts to move along the current best trajectory and kicks off on-line simultaneous planning and execution, and the *sensing*, *planning*, and *control* cycles interact until the final goal of task-constrained operation is reached.

In each *planning* cycle, an operator is randomly selected, based on which, a tra-

jectory (or two – if the operator is crossover) is (are) randomly selected in the corresponding subpopulation (i.e., P_{tc_main}, P_{tc_temp} or P_{ntc}) and applied the operator to obtain a new trajectory τ_{new}, which is then evaluated (Section 6.4.3). If τ_{new} is worse than the worst trajectory in the subpopulation of its parent trajectory, it is discarded; otherwise, it replaces a non-best trajectory in the same subpopulation, and if τ_{new} is infeasible, it can only replace an infeasible and non-best trajectory.

In each *control* cycle, as summarized in Algorithm 1, each trajectory in P is first updated such that the beginning configuration is the current robot configuration with the current state (i.e., joint angles and velocities) and evaluated by taking into account the latest sensing information.

If the current best trajectory $\tau_{tc\_main\_best} \in P_{tc\_main}$ does not have an imminent collision (which is defined as having the earliest predicted colliding time less than a threshold), the robot is switched to follow $\tau_{tc\_main\_best}$.

If the $\tau_{tc\_main\_best}$ has imminent collision, conflict resolution is evoked (illustrated in Fig. 43).

If the best task-constrained trajectory $\tau_{tc\_temp\_best} \in P_{tc\_temp}$ does not have imminent collision, the robot is next switched to execute $\tau_{tc\_temp\_best}$. While the robot moves on $\tau_{tc\_temp\_best}$ and more control cycles are conducted, if conflicts are automatically resolved (for example the obstacles moved away), $\tau_{tc\_main\_best} \in P_{tc\_main}$ no longer has imminent collision. Hence the robot is switched to execute $\tau_{tc\_main\_best}$ as illustrated in the dashed line in Fig. 43. On the other hand, if the robot has already reached the end of $\tau_{tc\_temp\_best} \in P_{tc\_temp}$, the robot arm can move without the task constraints and then the robot is switched to $\tau_{ntc\_best} \in P_{ntc}$ (i.e. the robot motion is not subject to task

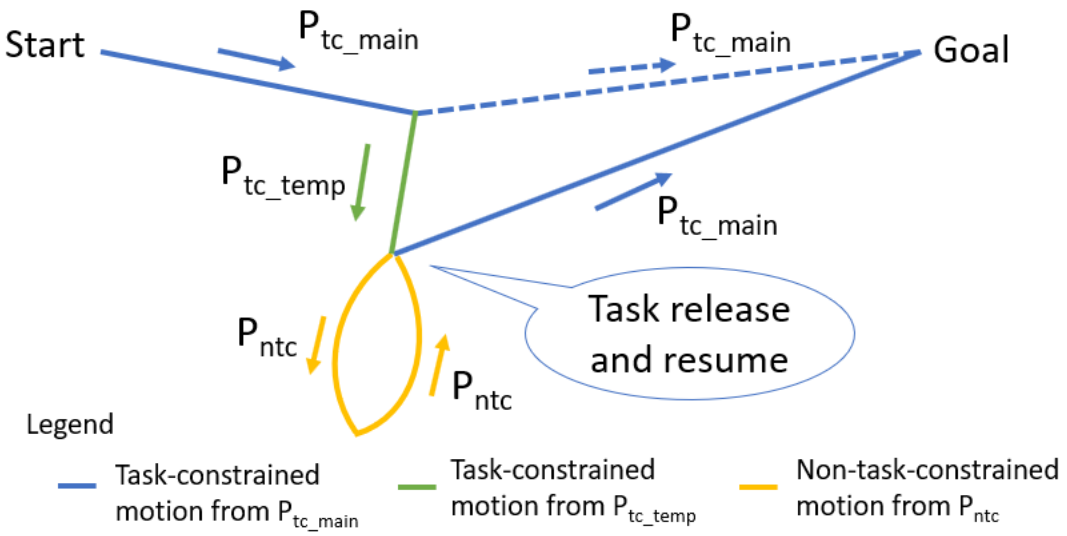


Figure 43: Illustration of conflict resolution mechanism. The blue and green solid lines indicate task-constrained motion from P_{tc_main} and P_{tc_temp} respectively, and the yellow solid lines indicate the non-task-constrained motion. The blue dashed lines indicate the motion that would be executed if the conflict between the tasks did not exist (any more). Task constraints are released or resumed at the intersection of the blue and yellow solid lines. The arrows indicate the motion direction.

constraints); g_{ntc} is set to reach the end-effector pose where the task constraints are released (Algorithm 4). While the robot moves, RAMP continuously improves P until the best task-constrained trajectory $\tau_{tc\_main\_best} \in P_{tc\_main}$ no longer has imminent collision. Then the robot arm moves back to g_{ntc} to resume the task constraints (Algorithm 5), and the robot follows $\tau_{tc\_main\_best} \in P_{tc\_main}$ to reach the final goal g_{tc_main}.

Overall, our approach achieves real-time conflict resolution in two levels. (1) *Intra-population conflict resolution by switching trajectories*: each subpopulation contains many and different trajectories and the robot arm can always switch to the best trajectory in the same subpopulation. (2) *Inter-population conflict resolution by switching subpopulations*: the robot arm can switch to other subpopulations if current subpop-

ulation is entirely infeasible. Both are demonstrated in the experiments presented next.

Algorithm 3: A Control Cycle

update trajectories in P with the current robot state and evaluate P;
update global flag F_{tc};
if F_{tc} **then**
    if $\tau_{tc\_main\_best}$ *has no imminent collision* **then**
        execute $\tau_{tc\_main\_best}$;
    else if $\tau_{tc\_temp\_best}$ *has no imminent collision* **then**
        execute $\tau_{tc\_temp\_best}$;
        if *end of* $\tau_{tc\_temp\_best}$ *is reached* **then**
            invoke Algorithm 4 to release task constraints;
    else
        if $F_{release}$ **then**
            invoke Algorithm 4 to release task constraints;
        else
            emergency stop;
else
    if τ_{ntc_best} *has no imminent collision* **then**
        execute τ_{ntc_best};
        if *end of* τ_{ntc_best} *is reached and* $\tau_{tc\_main\_best}$ *has no imminent collision* **then**
            invoke Algorithm 5 to resume task constraints;
    else
        emergence stop;

Algorithm 4: Releasing Task Constraints

detach tool/object from the robot end-effector;
add tool/object model to environment as a static obstacle;
set g_{ntc} to reach current end-effector pose;
$F_{tc} \leftarrow false$;

Algorithm 5: Resuming Task Constraints

attach tool/object to the robot end-effector;
remove tool/object model from environment;
$F_{tc} \leftarrow true$;

6.6 Overview of Experiments

We have implemented the introduced algorithm in C++ under ROS on a PC with an Intel i7-7700HQ 8-core 2.8GHz CPU. We considered a task of transferring a water cup (imposing end-effector orientation constraints) and a task of closing a drawer (imposing end-effector position and orientation constraints) under different testing scenarios. In all the experiments, an ABB IRB120 6-DOF manipulator was the one running our algorithm, and the motions of dynamic obstacles were unknown to the ABB robot.

Each trajectory in population P is initialized with 20 knot configurations. Each program run starts with a number of offline planning cycles (which takes about 1.5 mins) in the static environment to take into account static obstacles and improve overall trajectory quality based on the cost function of multiple optimization criteria. This is an offline initialization process and not part of the real-time motion planning. Next, the ABB robot and the dynamic obstacle start moving and the cycles of sensing, planning and control start interacting to achieve real-time adaptiveness of the ABB robot.

All the trajectories are discretized at every 0.3 secs for evaluation (Section 6.4.3). Such fine time resolution ensures that both the knot configurations on the trajectories and the cubic splines are evaluated. If a trajectory has task-constrained knot configurations but the interpolations between them violate the task constraints, this trajectory still has a high cost of task constraint violation.

For safety, we limit the joint velocities of the ABB robot to 20% of their max-

imum ranges in trajectory generation. The workspace of the ABB robot used for sampling task-constrained end-effector poses is a cubic volume with x=$[-0.4, 0.4]m$, y=$[-0.4, 0.4]m$ and z=$[0.2, 0.7]m$. The average success rate of the *Task-constrained Insert* and *Task-constrained Change* operator is about 75.5% since some sampled poses are not reachable by the robot under task constraints. Experimental results of each task are described in details below. The video accompanied to [78] shows the experiments presented.

6.7 Experiments of Transferring a Water Cup

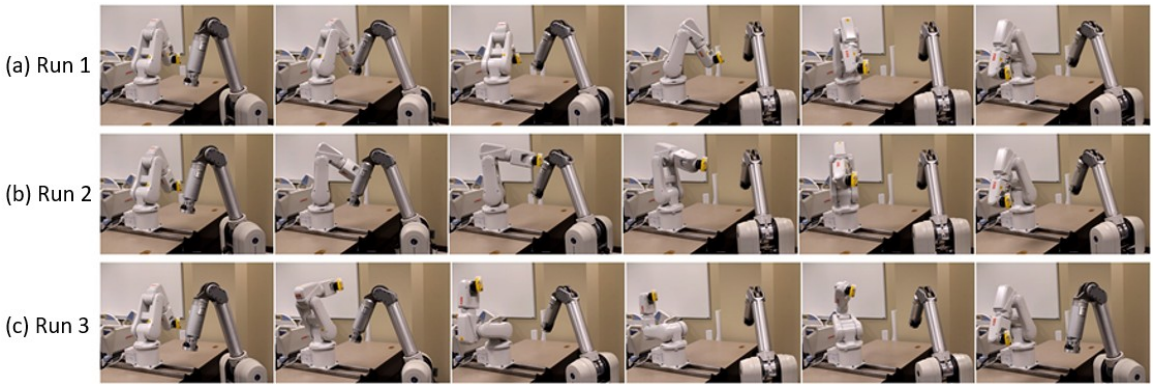


Figure 44: Motion snapshots of transferring a water cup. Each row corresponds to one program run. Only P_{tc_main} is used.

We considered two sets of experiments for this task. Subsection 6.7.1 presents experiments with only subpopulation P_{tc_main}, i.e., the robot arm is only allowed to switch trajectories in P_{tc_main} to avoid the obstacle and directly transfer the water cup to the final goal without placing down the cup and picking up again at some point (motion behavior from P_{tc_temp} and P_{ntc}). Subsection 6.7.2 presents the experiments using all three subpopulations P_{tc_main}, P_{tc_temp} and P_{ntc}, and therefore allowing the robot arm to switch among subpopulations in addition to switching trajectories inside

one subpopulation.

We considered soft task constraints by allowing the end-effector to tilt a maximum ± 15 degrees from the upright direction. The start end-effector pose is $[0.1, 0.3, 0.3, 0, 0, 0]$ and the goal pose is $[0.1, -0.3, 0.3, 0, 0, 0]$. Position and orientation are in m and rad respectively. The *motion constraint vector* C is $[0, 0, 0, 1, 1, 0]^T$.

6.7.1 Conflict Resolution with P_{tc_main}

Table 9: Statistics of the experiments of transferring a water cup and closing a drawer.

	Water cup runs (Section 6.7.1)					Drawer runs (Section 6.8)				
	1	2	3	4	5	1	2	3	4	5
Total run time(s)	7.08	6.97	10.80	10.55	27.46	23.60	17.81	18.50	18.77	24.64
# of planning cyc.	27	33	46	52	107	46	48	46	51	60
# of control cyc.	27	33	46	52	107	46	48	46	51	60
Avg. time of planning cyc.(s)	0.11	0.07	0.09	0.07	0.10	0.13	0.10	0.10	0.11	0.11
Avg. time of control cyc.(s)	0.15	0.14	0.14	0.14	0.15	0.38	0.27	0.31	0.26	0.29
Avg. control frequency(Hz)	3.82	4.74	4.26	4.93	3.90	1.95	2.70	2.49	2.72	2.44
Avg. tilted angle(deg)	7.73	6.63	8.93	8.99	13.99	NA	NA	NA	NA	NA
Total # of traj. switches	5	3	1	7	13	6	8	17	12	10
# of traj. switch(traj. infeas.)	2	0	0	2	5	2	3	6	4	3
# of traj. switch(better traj.)	3	3	1	5	8	4	5	11	8	7
# of control cyc. from P_{ntc}.	0	0	0	0	0	38	38	37	40	51

Here the ABB robot has to transfer a yellow water cup to the goal pose, avoid a static object (the table) and the Barrett WAM, which is in a repetitive motion mimicking a human worker doing pick-and-place tasks. The motion of Barrett WAM is unknown to the ABB robot. The joint angles and velocities of the ABB robot and Barrett WAM are updated at 10 Hz and 250 Hz respectively during sensing cycles.

Under the *task-constrained RAMP* with only one subpopulation P_{tc_main} of 20 (evolving) trajectories, the ABB robot successfully finished the task without collision in 16 out of 20 consecutive program runs. The failures were mostly because the robot did not switch the trajectories quickly enough, which could be improved with increased control frequencies. Fig. 44 shows the motion snapshots of 3 successful pro-

gram runs. Table 9 summarizes the experimental statistics. The accompanying video shows all 5 runs. It can be seen that the ABB robot has to adapt to the moving Barrett WAM by switching trajectories multiple times while maintaining the end-effector pose constraints. A trajectory switch can happen either because the trajectory being executed becomes infeasible as a result of the dynamic obstacles or because a better task-constrained trajectory with a lower overall cost appears in the population as the result of online trajectory optimization in planning cycles. It is also clear that in order to avoid the moving WAM, the ABB robot sometimes tilted its end-effector a bit further away from the vertical direction to take advantage of the soft constraints.

The actual trajectories that the ABB robot executed were quite different in different program runs, which reflected the global nature of the RAMP algorithm and demonstrated the intra-population real-time adaptiveness. For example, in run 1, the ABB robot first moved sideways to wait until the WAM passed before it started moving towards the goal. In run 2, the ABB robot avoided the WAM by elevating the end-effector. In run 3, the ABB robot operated in a space that was far away from the WAM, and as a result the number of trajectory switches is fewer than in the other program runs (Table 9).

As shown in Table 9, the task took about a total of 10 secs to finish. The average tilted angle (computed as $max\{roll, pitch\}$) is about 10 degs, within the specified range. One planning cycle and one control cycle took about 0.1 secs and 0.15 secs respectively, resulting in an average of 4 Hz control frequency.

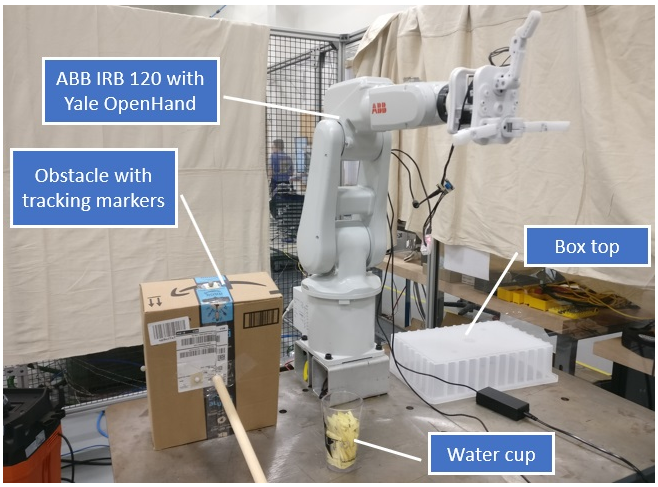


Figure 45: We used ABB IRB 6-axis robot with a 3-finger Yale OpenHand [74]. A person moves the box using the pole attached to the box to disrupt the robot motion. The top surface of the white box is designated as a safe spot to place down the cup.

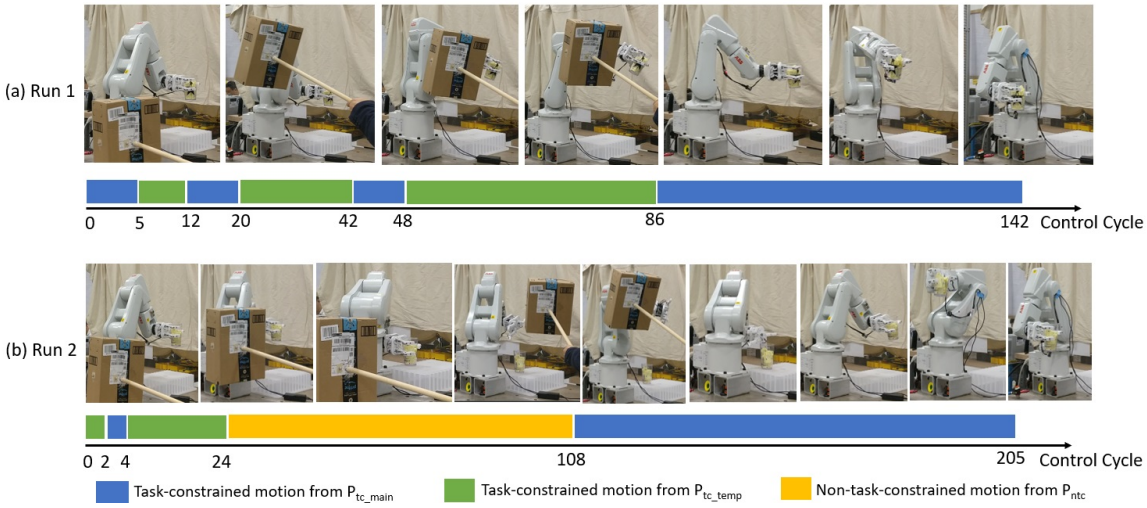


Figure 46: Transferring the water cup with a dynamic obstacle. P_{tc_main}, P_{tc_temp} and P_{ntc} are used. Non-task-constrained motion is not triggered in subfigure (a) and triggered in (b).

6.7.2 Conflict Resolution with P_{tc_main}, P_{tc_temp} and P_{ntc}

We constructed an experiment where the *task-constrained RAMP* had to use all three subpopulations, P_{tc_main}, P_{tc_temp} and P_{ntc}, to achieve conflict resolution. Each subpopulation had 10 trajectories. Fig. 45 presents the overall setup. A white box

was placed at a flat place near the robot starting pose, its top was designated as a temporary, safe place for the robot end-effector to put down the cup if needed, and the corresponding arm configurations were determined as those of task-constrained temporary goals g_{tc_temp}. A box moved by a person was used as a dynamic obstacle and its pose was streamed at 100Hz by an external vision tracking system.

When the dynamic obstacle was not moved by the person, the robot arm perceived this via online sensing, and hence just started moving the cup and reaching the final goal pose without any behavior of obstacle avoidance or conflict resolution. Fig. 42 shows the snapshots of the motion. This execution of the overall task serves as a baseline.

When the obstacle was being moved, the robot performed interesting motions of obstacle avoidance and conflict resolution using P_{tc_temp} and P_{ntc} as shown in Fig. 46. The axis with color bars in Fig. 46 indicates the decomposition of the subpopulation usage in the whole process of the motion. The blue bar indicates motion from P_{tc_main}, the green from P_{tc_temp} and the yellow from P_{ntc}. In Fig. 46(a), the robot arm sensed that the box was nearby, avoided the box and finally reached the final goal pose without temporarily placing down the cup or using the non-task-constrained trajectories from P_{ntc}. The robot started by using trajectories from P_{tc_main} to try to reach the final goal directly and it had to switch between P_{tc_main} and P_{tc_temp} to avoid the obstacle with the cup still grasped in hand. As the obstacle moved away after control cycle 86, the robot had a clear path to reach the final goal. On average, the control frequency is about 6.5 Hz, and the time of one control cycle is 0.12 secs. On average, the tilted angle of the cup is 13.4 degs.

Fig. 46(b) shows more interesting robot motion for conflict resolution using P_{ntc}. At the beginning of the motion, the robot perceived that the box obstacle was blocking all the trajectories in P_{tc_main}, and therefore started to head to the temporary goal with trajectories from P_{tc_temp}. Since the box was constantly blocking P_{tc_main}, the robot reached the temporary goal g_{tc_temp} (the end of trajectories in P_{tc_temp}) at control cycle 24, and chose to put down the cup temporarily. Hence, the robot was released from the task constraints and was able to conduct non-task-constrained motion to avoid the obstacle using P_{ntc} from control cycles 24 to 108. The robot next came back to pick up the cup and reached the final goal at control cycle 205. On average, the control frequency is about 4.1 Hz, and the time of one control cycle is 0.21 secs. The average tilted angle is 12.5 degs along the task-constrained part of the executed trajectory.

The different robot motion behaviors shown in Fig. 42 and Fig. 46 are results achieved by our algorithm based on online sensing and the status of the subpopulations. As shown in Fig. 42, if P_{tc_main} contains trajectories good for both obstacle avoidance and reaching the final goal, the robot will use it and directly go to the final goal without using P_{tc_temp} or P_{tc_ntc}. As shown in Fig. 46(a), if the coordination of P_{tc_main} and P_{tc_temp} helps avoid the obstacle and also maintain the task constraints, the robot will try to switch the trajectory or alternate between subpopulations and reach the final goal without having to place down the cup. As shown in Fig. 46(b), if no task-constrained trajectories satisfying both obstacle avoidance and task constraints at the same time, and the conflicts exist for a long time, the motion to put down the cup at the temporary goal is triggered, and the robot conducts

non-task-constrained motion to avoid the dynamic obstacle. *Task-constrained RAMP* continuously improves the trajectory population, and when the obstacle moves away, the robot resumes task-constrained motion from P_{tc_main} again.

6.8 Experiments of Closing a Drawer

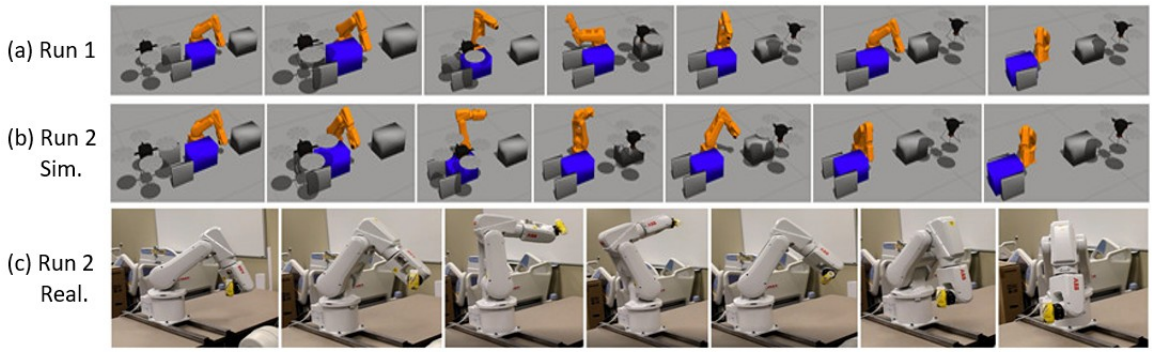
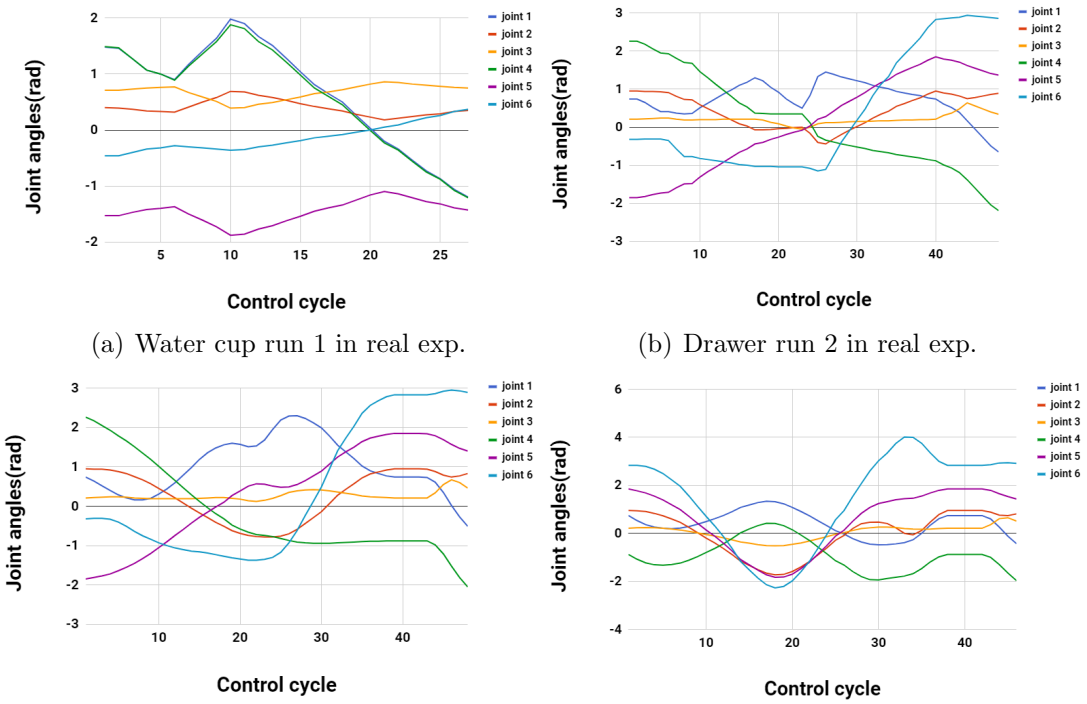


Figure 47: Motion snapshots in the task of closing a drawer. The simulated ABB robot is in orange. Each row corresponds to one program run. Last two rows show the same run in simulation and real world. The yellow bottle on the real robot indicates the end-effector pose.

The task is first run in simulation, where the ABB robot pushes the blue box until it is between two other objects to mimic the task of closing a drawer. During the whole motion, the robot has to avoid the static objects and a UAV of motion unknown to the robot, which flies from left to right and may interfere with the task of pushing the blue object. Gazebo broadcasts the states of the ABB robot, the UAV, and the object poses at 50 Hz. The start end-effector pose is $[0.3, 0.2, 0.2, 0, 0, -1.57]$, and the goal pose is $[0.3, -0.3, 0.2, 0, 0, -1.57]$. Position and orientation are in m and rad. The task constraint is $C = [1, 0, 1, 1, 1, 1]^T$. In the simulation runs, all three subpopulations (P_{tc_main}, P_{tc_temp} and P_{ntc}) were used by *task-constrained RAMP* and each contained 10 trajectories.



(a) Water cup run 1 in real exp.

(b) Drawer run 2 in real exp.

(c) Drawer run 2 in simulation.

(d) Drawer run 1 in simulation.

Figure 48: Joint angles. (a) and (b) were obtained from real experiments. (c) and (d) were from simulation.

In order to give the robot maximum manoeuvring space, the temporary task-constrained goal g_{tc_temp} configurations were chosen as those that enable the end-effector to reach the starting pose. However, as shown in the experiments, the robot directly switched to non-task-constrained motion to resolve the conflicts since they were so severe that all the task-constrained trajectories in P_{tc_main} and P_{tc_temp} had predicted collisions.

The ABB robot successfully finished the task without collision in 20 consecutive program runs. Fig. 47 shows the motion snapshots of two successful runs, and Table 9 summarizes the statistics from five runs of experiments. The accompanying video shows all 5 runs. It shows that the robot started by pushing the blue box, but quickly

it sensed that the UAV was flying towards it. As the result, the robot stopped pushing the box for the time being and switched to non-task-constrained trajectories to avoid the UAV. When the UAV was far away, the robot moved back to resume pushing the box.

Note that the blue box being pushed was also considered as a static obstacle that the robot had to avoid when the robot moved away from pushing and came back to continue the pushing. Fig. 47(c) shows the real robot executing the same trajectories executed in the simulation run 2 (Fig. 47(b)). We used the attached bottle to visualize the end-effector poses (the bottle should stay upright during pushing). It can also be seen from Fig. 47 and the accompanying video that *task-constrained RAMP* is able to find different motions in response to the same UAV motion pattern, thanks to the global and stochastic nature of our algorithm.

As shown in Table 9, the robot spent the majority of the control cycles executing non-task-constrained trajectories from P_{ntc} to avoid the UAV, which demonstrated the importance of conflict resolution. Without the subpopulation P_{ntc}, the whole population of robot trajectories would converge to the end-effector straight-line motion to only fulfill the task constraints, and the robot would not be able to adapt to the unexpected UAV motion.

6.9 Discussion and Performance Improvements

As shown in Fig. 44, Fig. 47, and the accompanying video, the ABB robot sometimes had some unnecessary motions rather than directly driving to the goal when it was close to the goal, or the robot took a relatively long time to come back and

continue pushing the drawer after it moved away. This could be due to that the population was still not diverse enough to contain more efficient trajectories. Since RAMP is inherently parallel, and by leveraging massive parallelization on GPUs, the processing speed can be vastly improved, and a larger population of trajectories can be accommodated to improve the performance.

Fig. 48 shows the joint angles of a few program runs. We can see that the joint motion obtained from real experiments is not as smooth as from the simulation in Gazebo – note that Fig. 48(b) and (c) are both from the same run 2 of the drawer case. There are two main reasons for this. (1) When switching a trajectory, currently the real robot controller has to cancel the current control set point and apply a new one, and this could result in the real robot slows down (or stops) before the new control set point takes effect [3]. Using Externally Guided Motion (EGM) from ABB can provide more smooth transitions when there are trajectory switches [1]. (2) Each control cycle took 0.1 to 0.3 secs to compute, and thus the robot might be at a noticeably different state compared to the state used for planning the trajectory switch earlier in the control cycle. Again, making *task-constrained RAMP* more efficient through parallel processing could resolve the issue. With greater computation speed, the time threshold for imminent collision can be reduced with more accurate prediction of obstacle motion and a strategy that fully exploits the robot acceleration capabilities.

6.10 Summary

In this chapter, we extend the RAMP framework by incorporating task constraints, which enables a robot manipulator to perform tasks with constraints on its end-

effector effectively in unforeseen dynamic environments. The *task-constrained RAMP* algorithm is validated in different example tasks and in different dynamically unknown environments. The introduced algorithm is demonstrated to be able to resolve conflicts between satisfying task constraints and avoidance of unexpected dynamic obstacles seamlessly through real-time adaptiveness.

CHAPTER 7: REDUCING POSE UNCERTAINTY UNDER COMPLEX CONTACTS VIA FORCE FORECAST

Robotic manipulation requires robots interacting with the physical world through making contacts. Contacts are important for robotic grasping, in-hand manipulation, assembly [22, 103, 132]. These contacts are typically complex and occur at multiple regions at the same time (see Fig. 2 for an example). The complex nature of such contacts makes many contact-rich manipulation tasks challenging. The uncertainty associated with the object pose can easily make any predefined manipulation procedures invalid. On the other hand, information of contacts is useful for reducing uncertainty.

In this chapter, we introduce a novel force forecast approach that relates the real world force sensing to the simulated world to enable reduction of pose uncertainty. This approach corrects the pose uncertainty in a real-world robotic assembly or manipulation task through finding the contact force in simulation that best matches the measured contact force in real-world and adjusting the corresponding relative pose of the parts to reduce uncertainty. Our approach does not make assumptions about contact locations or pre-define any contact types. It also does not have any restriction on the shape of peg and hole. Contact force forecast is achieved by first leveraging a model-based haptic simulation algorithm [129] that generates the initial contact force and next relating this force to the real contact force through calibration based on

force sensing. An interesting finding is that effective calibration can be achieved with a simple regression model trained using a small number of force signals collected in real world.

We apply this force forecasting approach to reducing pose uncertainty in challenging two-pin and three-pin peg and hole assembly tasks. See Fig. 2 for an example. The pegs and holes are of arbitrary and rather complex shape, and the assembly tasks often have complex multi-region contacts. Real-world experiments demonstrate that the assembly tasks can be accomplished with our approach and a compliant robot.

7.1 Related Literature

There have been continued research efforts on representing and computing contact states, sensing contacts, and reducing uncertainty in robotic assembly tasks.

Contact representation and state computation: How to systematically represent contacts and compute possible contact states has been an active research topic. Pioneering work investigated representations of contact states between polyhedral objects [135] and curved objects [116]. Such contact state information is useful for planning robot motion compliant to contacts [40] or simulating haptic response [71]. A graph-based contact transition representation can also be automatically generated [115, 50], and it can be used for contact-based state estimation [68] or reducing uncertainty in assembly tasks via contact state transitions [103]. These approaches are built upon the concepts of principal contacts, which are formed by combining certain primitive contacts. However, it is practically difficult to identify in real-time such contacts that may occur simultaneously at multiple regions.

Direct contact sensing: With touch sensors, robots can directly detect contacts, localize contact areas, and parse local contact information. For instance, a rigid touch pad installed on the robot end-effector enables a robot arm to track a rope for tactile servo [65]. More recently, the soft touch sensors (such as GelSight [141] and BioTac [113]) attached to the robotic hands can passively conform to contact areas and directly capture tactile images. Rich contact information embedded in these images has been shown to be useful for many tasks, for example grasping slip detection [21], object property perception [144], force estimation [72] and so on. However, when the robot holds an object (or a tool) to interact with other objects in assembly tasks or tool-based manipulation tasks, it is practically not feasible to cover the objects (or tools being held by the robot) with many touch sensors. In such cases, contact information can only be indirectly obtained.

Indirect contact sensing: Force/torque (F/T) sensors installed at the wrist of a robot arm or joint torque sensors are usually used for this purpose. External contacts or collisions with the environments can be detected if the sensor readings exceed a threshold. Assume knowing where contacts would happen, there are methods to achieve object tracking and state estimation [139], object modeling or reconstruction [20], and tool calibration [46, 66] with touch sensing alone or in combination with vision. However, the assumption of known contact locations is restrictive. An approach to relaxing such an assumption is presented in [140], where a data-driven method is used to classify the contact types online based on F/T sensor readings. From the contact type information, correct contact constraints can be imposed, and object pose estimation can be made effective. However, the contact types were prede-

fined based on some contact primitives, and it is difficult to generalize the approach to more complex contact scenarios.

Robotic assembly tasks: The rich contact information (mostly sensed indirectly) can be leveraged to reduce object pose uncertainty. Contact transitions via robot compliant motion (such as [27, 103, 132, 114]) have been shown to be useful to reduce uncertainty and accomplish some peg-in-hole tasks. Analytical approaches that actively control the robot motion based on force feedback are also effective [18, 47], but they require task-specific derivations and have restrictive assumptions on object geometry. Assume that a peg is aligned with the hole, localization of the hole areas (position uncertainty) can be achieved with visual servo [51] or exploration search motion [90, 123]. Reducing orientation uncertainty is investigated in [140] but it has restrictive assumptions on predefined contact types. More recently, reinforcement learning approaches (such as [51, 119]) also showed promising results. However, the learned policy is task-specific, and it typically requires significant training with robot hardware to be transferred to a new task.

There is also existing work focusing on how to estimate or reconstruct forces, for example through learning [113] or inverse finite element methods [72] to relate tactile sensing data (direct contact sensing) to contact forces. However, how to relate contact forces to reduction of relative pose uncertainty of contacting parts in multi-region complex contacts is still a largely open problem.

7.2 Uncertainty Reduction via Force Forecast

An overview of uncertainty reduction via force forecast is presented in Fig. 49. Measured relative pose T_m between objects (for assembly) or the robot hand and object (for manipulation) is subject to uncertainty, and our goal is to reduce its uncertainty based on the measured contact force f_m. Due to uncertainty in T_m, it may not be a contact pose (i.e., T_m can correspond to an interpenetration or non-contact between the objects). A contact pose T_c neighboring to T_m is thus generated in the model (i.e., simulated) world, and the corresponding contact force f_c is obtained by a model-based haptic simulation algorithm. f_s is next obtained by calibrating f_c using a data-driven learning approach. If f_s matches f_m, T_c is the more accurate estimate of object pose. Otherwise, another contact pose can be generated to continue searching for f_s that matches f_m. The model-based haptic simulation and data-based force calibration form the overall force forecast.

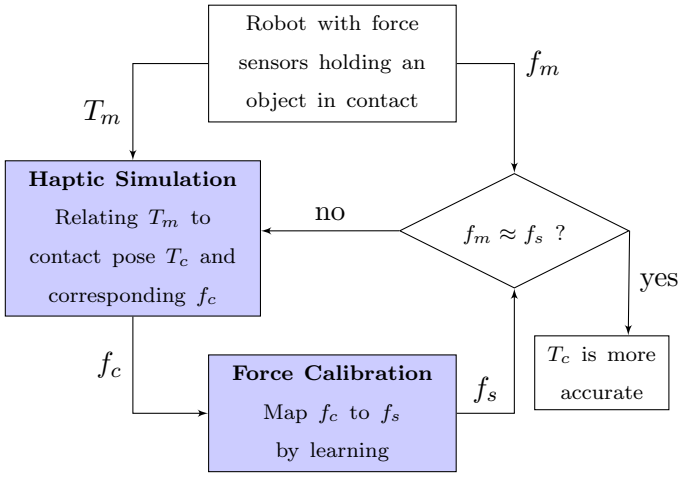


Figure 49: An overview of pose uncertainty reduction via force forecast, which includes haptic simulation and force calibration.

We next explain the detailed approach.

7.2.1 Force Forecast

Consider two objects p and h for contact interaction in simulation. Without loss of generality, we assume object p moves along trajectory τ and object h is fixed at a measured pose ${}^{w}T_{h}$, where w indicates the world frame and τ includes a sequence of poses of object p. ${}^{w}T_{h}$ is subject to sensing uncertainty. We further assume object mesh models of p and h are available and denote them as M_p and M_h respectively. Object mesh models of industrial parts or 3D printed parts typically are easy to obtain. Object mesh models can also be built automatically based on perceived appearance [117, 77].

Haptic Simulation: A haptic simulation algorithm is used to generate initial contact force f_c in simulation given τ, ${}^{w}T_{h}$ and object models M_p and M_h. We leverage recent progress in haptic simulation presented in [129] for this purpose. This approach uses object models represented as sphere trees, which provide a uniform representation for handling complex contact states. Sphere tree models can be automatically constructed from mesh models [4].

For simplicity, in the rest of the chapter, we further use M_p and M_h as general notations to indicate both the mesh models and the sphere-tree models built upon the mesh models of objects p and h respectively. See Fig. 50(a) for a sphere tree model of a two-pin peg.

As object p moves on trajectory τ, p contacts h (fixed at ${}^{w}T_{h}$) at time t with pose ${}^{w}T_{p}$. Due to uncertainty in pose ${}^{w}T_{h}$, M_p may penetrate into M_h (or not in contact with M_h) in simulation. A physically correct contact pose ${}^{w}T'_{p}$ is obtained by

minimizing elastic energy E stored in a virtual spring connecting object p at ${}^{w}T_p$ and at ${}^{w}T'_p$ while satisfying the non-penetration constraints [129]:

$$\textit{minimize:} \quad E({}^{w}T_p, {}^{w}T'_p) \tag{5}$$

$$\textit{subject to:} \quad M_p \cap M_h = \emptyset \tag{6}$$

With the sphere-tree models of objects, the non-penetration constraints in equation (6) are expressed uniformly as distance constraints between spheres in M_p and M_h, regardless of the shapes of the objects. This optimization problem is solved using quadratic programming with linearized constraints, and it can be computed efficiently in about 1 kHz even under complex contact states [129]. The resulting ${}^{w}T'_p$ corresponds to the object p staying in contact with object h at time t (instead of penetrating into h in simulation).

The position difference between ${}^{w}T_p$ and ${}^{w}T'_p$ is used to generate the contact force f_c based on the Hooke's law (with estimated stiffness coefficients). See Fig. 50(b) for an illustration.

Force Calibration: A data-driven regression model FC is used to take the simulated contact force f_c as an input and outputs a corresponding force f_s that matches a measured force f_m, when the objects p and h are in contact.

This calibration is necessary because a discrepancy exists between initial contact force f_c in simulation and the corresponding measured forces f_m in real world because of the following reasons:

1. Sphere tree models approximate the shapes of objects.

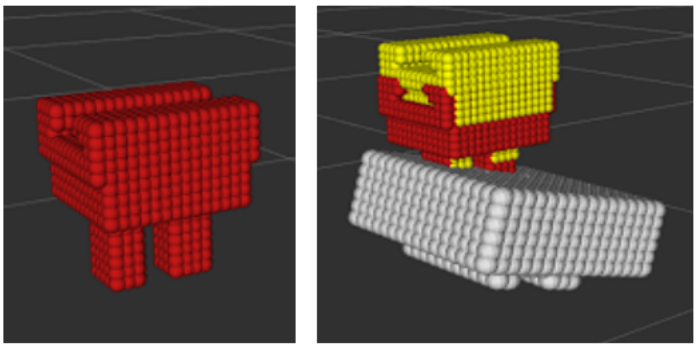


Figure 50: **Left:** A two-pin peg represented using Octree level 5. **Right:** The red object is peg at ${}^{w}T_{p}$ interacting with the hole (gray). The yellow object is a virtual peg at a physically correct contact pose ${}^{w}T'_{p}$ that satisfies the non-penetration and time history constraints. The pose difference between the red and the yellow pegs is used to compute initial contact force f_c.

2. Linearized contact constraints are used to solve haptic simulation with quadratic programming.

3. Stiffness coefficients are only estimated based on the Hooke's law.

We trained a simple regression model with a small number of sensed forces collected in real-world experiments to map initial contact force f_c to f_s, which better matches measured force f_m. Such calibration can be done effectively under complex contact states, as shown in our evaluation (Section 7.5 and 7.6).

We next explain how to apply force forecast to reduce pose uncertainty.

7.2.2 Uncertainty Reduction

As mentioned earlier, we assume ${}^{w}T_{h}$ is subject to uncertainty. ${}^{w}T_{h}$ can be further decomposed as ${}^{w}T_{h} = T_{nom}\ \Delta T$, where T_{nom} denotes the nominal pose of object h in the world frame, and ΔT denotes the uncertain transformation of the pose in the local frame of object h. T_{nom} is known, for example from initial system calibration or

vision-based pose estimation. The goal is to estimate $\Delta T \in SE(3)$.

Algorithm 6 summarizes how to estimate ΔT by searching in forces computed from different contacts. Consider a candidate set $\mathbb{C} = \{c_i\}$, $i = 1 \dots n$, where c_i is a set of parameter values representing the i-th candidate ΔT_i for ΔT. $\mathbb{C}$ can be systematically generated by discretizing the parameters in their ranges of uncertainty. For each $c_i \in \mathbb{C}$, the contact force f_c encountered during the execution of trajectory τ is computed in simulation. From force calibration, The corresponding f_s that matches the measured force f_m is selected, and the corresponding set of parameter values c^* is returned for the best estimation of ΔT. f_s is computed from FC as explained in Section 7.2.1.

Algorithm 6: Uncertainty Reduction

input : Pose candidate set $\mathbb{C}$, measured contact force f_m, object p trajectory τ, models M_p and M_h, calibration model FC;
output: estimated pose c^*;

$F \leftarrow \emptyset$;
for *each* $c \in \mathbb{C}$ **do**
  move object h in simulation to ${}^wT_h$ computed with uncertainty parameters c;
  move object p along τ in simulation until contact;
  compute f_c with ${}^wT_h, \tau, M_p, M_h$ according to Section 7.2.1;
  obtain f_s by calibrating f_c using FC;
  $F \leftarrow F \cup (f_s, c)$;
end
search in F for f_s closest to f_m and return its corresponding c as c^*;

7.3 Application to Assembly

We next apply pose uncertainty reduction via force forecast to complex multi-peg-in-hole tasks with pegs and holes of arbitrary shape.

7.3.1 Problem Definition

Let p be a peg structure and h be a hole structure, where p may have multiple pins and h may have multiple corresponding holes (for example, a two-pin or three-pin peg and hole case). We assume a robot arm is used to hold the peg structure for insertion, and the hole structure is at a fixed pose in the work space and will not move during insertion. The relative pose between the peg structure and the hole structure is computed as ${}^{h}T_{p} = {}^{w}T_{h}^{-1}\ {}^{w}T_{p}$.

We assume that the peg structure p is rigidly attached to the robot end-effector, and ${}^{w}T_{p}$ is accurately known through the robot arm configuration. Whereas, the configuration ${}^{w}T_{h}$ of the hole structure is subject to sensing uncertainty, since it can only be obtained through perception using external sensors.

As described in Section 7.2.2, the objective here is to estimate $\Delta T = \begin{pmatrix} R_u & P_u \\ \mathbf{0} & 1 \end{pmatrix} \in SE(3)$, which depicts the uncertainty of both position and orientation. In this study, we assume the peg structure is already near the hole area (i.e., P_u is small and close to $\mathbf{0}$) and focus on reducing the orientation uncertainty R_u (as motivated in related literature). We further express $R_u = R(\alpha)R(\beta)R(\gamma)$, where α, β, γ represent Euler angles respectively.

We use $c = \{\alpha, \beta, \gamma\}$ to represent the uncertainty parameters of ΔT for simplicity. Since the uncertainty is typically limited, α, β, γ are also within small ranges (such as $\pm 15^{\circ}$). Hence, a candidate set $\mathbb{C}$ can be generated by enumerating α, β, and γ with a fine discretization.

We further assume that the robot holding the peg structure is equipped with a

force sensor, and when the peg structure contacts the hole structure, a contact force f_m can be sensed. Table 10 summarizes what we assume and do not assume. No vision sensing is used for uncertainty reduction.

Table 10: Summary of what are assumed and not assumed on pose uncertainty reduction in multi-peg-in-hole tasks.

Assumed	**Not assumed**
Robot with force sensing	
M_p and M_h available	Contact location
${}^wT_p$ is accurate	Contact types
${}^wT_h$ is subject to uncertainty	
T_{nom} is available and P_u is $\mathbf{0}$	

7.3.2 Approach

See Algorithm 7. We denote the pose of the peg structure when the peg structure is inserted into the hole structure (i.e. when assembly is completed) as a goal pose ${}^hT_g$. The goal of a peg-in-hole assembly task is to put the peg structure at ${}^hT_g$. Note that ${}^hT_g$ is a sensed pose computed based on the current estimation of the hole structure pose, which is subject to uncertainty.

A nominal trajectory τ that moves the peg structure towards the sensed goal pose ${}^hT_g$ for insertion can be generated. The motion consists of orienting the peg structure to align it with the hole structure in its sensed pose and Cartesian space straight-line motion for directly inserting the peg structure. If the uncertainty is sufficiently small, insertion can be achieved directly. Otherwise, the motion of the peg structure will be blocked by some contacts, and the contact force f_m is obtained from robot sensor (after the contacts are stabilized). Algorithm 6 (Section. 7.2.2) is next invoked to analyze the uncertainty based on measured force f_m, uncertainty parameter candidate

set $\mathbb{C}$, trajectory τ, object models M_p and M_h, and the calibration model FC.

With the estimated pose uncertainty c^*, a new goal pose ${}^hT'_g$ is computed. The robot is now put in compliant motion mode to move the peg structure from the current contact configuration to the newly estimated goal pose ${}^hT'_g$. The robot motion (controlled in joint compliance mode), and the contacts already established between the peg structure and the hole structure will collectively orient the peg structure towards reaching the orientation of the new goal ${}^hT'_g$. This can be achieved as long as the external contact force does not exceed a safety limit. Insertion will be halted if it is again blocked by the contacts, and another round of uncertainty reduction and compliant motion can be repeated.

Algorithm 7: Multi-Peg-in-Hole Assembly with Uncertainty Reduction

```
1  move peg structure to default start pose;
2  generate uncertainty candidate set ℂ for the hole structure;
3  move peg structure to sensed nominal goal pose ^hT_g with trajectory τ;
4  if ^hT_g is reached then
5  |  insertion is finished
6  end
7  else // motion blocked by contacts.
8  |  obtain contact force f_m from robot sensors;
9  |  c* ← UncertaintyReduction(ℂ, f_m, τ, M_p, M_h, FC);
10 |  compute new goal pose ^hT'_g based on c*;
11 |  compliantly transit peg structure in place to reach the orientation of ^hT'_g;
12 |  move peg structure to reach position of ^hT'_g;
13 |  if ^hT'_g is reached then
14 |  |  insertion is finished
15 |  end
16 |  else
17 |  |  insertion is halted
18 |  end
19 end
```

7.4 Overview of Experimental Evaluation

Our experimental setup is shown in Fig. 2. A Franka Panda 7-DOF robot manipulator is used to hold the peg structure, and another ABB IRB 120 6-DOF robot manipulator is used to hold the hole structure. We 3D printed the pegs and the holes used in the experiments. See Fig. 51 for their shape. Note that the irregular shape of the pegs and the holes make the contacts complex. Each hole is $2mm$ larger than the peg. The two robot arm bases are calibrated.

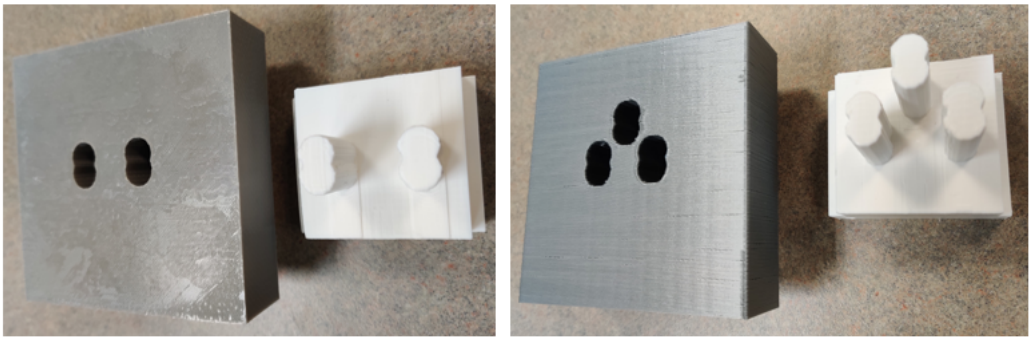


Figure 51: Pegs and holes used in the experiments.

The orientation of the hole structure can be controlled precisely by the ABB arm and used as the ground truth of the uncertainty for evaluating our approach. Note that this information is not available to the Panda robot before or during the insertion. The uncertainty range of α, β, γ considered in the experiments is $\pm 15°$. The pose of the hole structure is fixed (by the ABB arm) during each insertion.

The Panda robot is equipped with joint torque sensors for contact detection and force estimation [2]. The force threshold for detecting external contacts is 15 N throughout the experiments. The estimated forces are expressed in the Panda K frame (near the wrist) [2] and transformed to the world frame through robot kine-

matics. The force captured by the Panda robot sensor after the contacts are stabilized is used as the measured contact force f_m. In simulation, the force computed when the peg structure penetrated about $2cm$ into the hole structure is used as the contact force f_c.

7.5 Two-Pin Peg-in-Hole Assembly

We discretize α, β, γ each with a resolution of 2° over the range of $\pm 15^\circ$, which leads to $3,839$ uncertain poses of the hole structure, taking into account the inverse kinematics of the ABB arm holding the hole structure. The candidate set $\mathbb{C}$ used in Algorithms 1 and 2 are created from the majority of those uncertain poses, while the remaining smaller number of uncertain poses form the set Z for creating samples to train the force calibration model FC.

To train FC, we use the ABB robot to move the hole structure to an uncertain pose in Z and the Panda robot to move the peg structure to conduct a direct insertion based on the nominal pose of the hole structure. With the computed force f_c and the measured force f_m, we now have a tuple (c, f_c, f_m) for training. By exhausting the uncertain poses in Z, we have a list of tuples for training FC.

7.5.1 Contact Reasoning and Uncertainty Reduction

Table 11 shows the uncertainty reduction results. A fully connected neural network with 2 hidden layers (32 neuron each) with Rectified Linear Unit (ReLu) activation is used as the regression model for force calibration. Mean absolute errors between forces are used as the loss function. Stochastic gradient descent (SGD) with momentum is used for training. $\Delta\alpha$, $\Delta\beta$ and $\Delta\gamma$ are the absolute errors (in $^\circ$) between the predicted

α, β, and γ and their ground truth values computed on the test split of the dataset. It can be seen that $\Delta\alpha$ and $\Delta\beta$ are reduced from 30° (range of α and β is $\pm 15°$) to about $5° \pm 5°$ (mean $\pm$ standard deviation). This confirms the effectiveness of using computed realistic contact forces to reason about the actual orientation of the hole. As shown in Table 11, force sensing alone is not effective for reducing the uncertainty about the yaw ($\Delta\gamma$). This is because the contact forces are not distinctive enough, and can be improved by equipping the robot with a camera to capture the top-view images of the hole during insertion.

It is also interesting to see from Table 11 that the regression model can be trained with a small number of contact forces collected in the real world, while still achieving roughly the same level of uncertainty reduction. 1% split indicates about 40 data samples for training and the rest for testing.

Table 11: Uncertainty reduction in two-pin assembly. A fully connected neural network ($32 - 32 - 3$) with 2 hidden layers (32 neurons each) with ReLu activation is used as the regression model to predict $3D$ forces. Trained with 300 epoches.

Training split (%)	$\Delta\alpha(°)$	$\Delta\beta(°)$	$\Delta\gamma(°)$
1	5.58 ± 5.83	5.43 ± 5.28	11.10 ± 8.13
2	5.64 ± 5.92	4.21 ± 5.31	11.72 ± 7.86
10	5.50 ± 5.99	4.37 ± 5.64	10.56 ± 8.13
30	5.71 ± 6.31	4.99 ± 6.76	10.71 ± 7.94
50	5.49 ± 6.23	4.71 ± 6.13	11.01 ± 8.22

7.5.2 Reducing Regression Model Complexity

We tested regression models of different complexity to understand which model is sufficient to make the forces realistic. Table 12 shows the results with different neural network structures. 8-8-3 indicates a fully connected neural network with 8 neurons

for each hidden layer, and outputs $3D$ forces (using ReLu activation). As we can see from Table 12, roughly the same level of uncertainty reduction can be achieved even when the neural network degenerates to a simple linear regression model. Fig. 52 visualizes the total error of $(\Delta\alpha + \Delta\beta)$ for $\gamma = 7^\circ$ in a heat map (using the linear regression model and 1% of data for training). About 85.5% of the testing poses have total errors $< 15^\circ$. The poses still subject to large uncertainty can be further reduced by conducting another round of contact and reasoning.

Table 12: Comparison of different regression models in two-pin assembly. $\Delta\alpha$, Trained with 300 epoches.

Train split %	Structure	# of parameters	$\Delta\alpha(^\circ)$	$\Delta\beta(^\circ)$	$\Delta\gamma(^\circ)$
50	8-8-3	155	5.48 ± 6.48	4.79 ± 6.24	11.37 ± 8.31
50	8-3	99	5.48 ± 6.28	5.12 ± 6.77	11.08 ± 8.42
50	**3**	21	**5.45 ± 6.17**	**4.44 ± 5.80**	10.96 ± 7.68
10	3	21	5.36 ± 5.69	4.68 ± 6.19	11.50 ± 8.48
1	**3**	21	**5.49 ± 5.98**	**4.65 ± 5.65**	11.31 ± 8.18
0.5	3	21	6.64 ± 5.96	7.05 ± 6.38	11.50 ± 8.17

Our results presented here suggest an interesting finding. Force calibration (mapping from haptic simulation algorithm outputs to realistic forces) is as simple as a linear regression model and it can be trained effectively with a small number of forces collected in the real world (about 40 force signals). One reason is that forces computed by the haptic simulation algorithm are precise to the extent that they can be used as good initial guesses to predict the realistic forces.

Through combining a model-based haptic simulation approach and a data-driven regression model for capturing the unmodeled residual information synergistically, our approach is able to compute realistic contact forces from simulated contacts that correspond to real contacts, and the uncertainty associated with the roll and pitch of the hole is reduced.

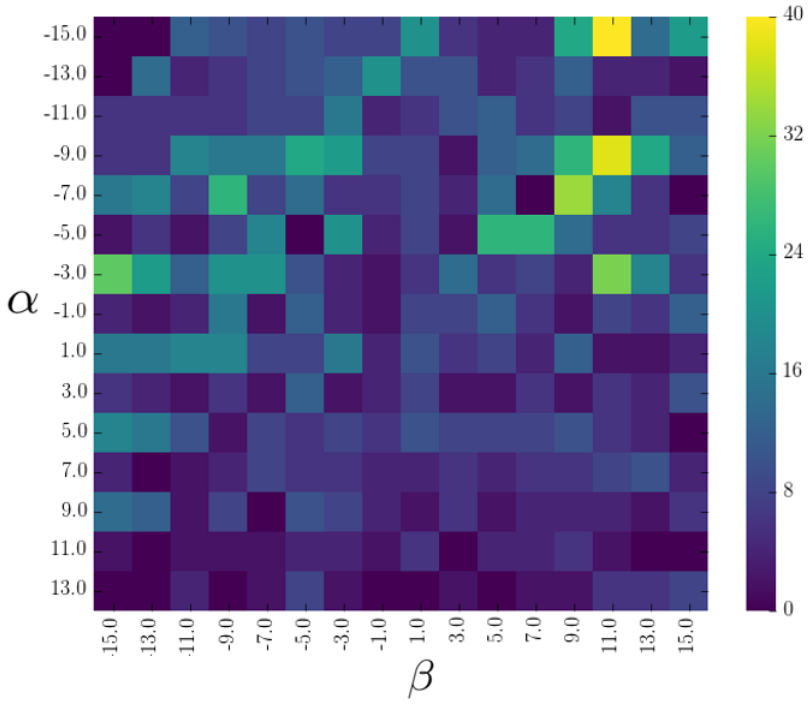


Figure 52: Total errors of $(\Delta\alpha + \Delta\beta)$ for $\gamma = 7$ (°) visualized in a heat map (using the linear regression model and 1% of data for training).

7.5.3 Robotic Insertion

By reducing pose uncertainty, multi-peg-in-hole insertion can be achieved via compliant transitions (with the Panda robot controlled in joint compliance mode). Fig. 53 shows snapshots of an insertion, and the video accompanied to [79] shows the robot motion. Table 13 summarizes ten robotic insertion experiments. It can be seen that, due to pose uncertainty of the hole structure, no direct insertion can be achieved based on the nominal goal pose. By estimating pose uncertainty through contact to obtain a new goal pose with reduced uncertainty, 8 out of 10 insertions were accomplished by compliant motion.

Two insertions failed for poses 9 and 10. The main reason is that the external force exceeded the safety limit (15 N) during the robot compliant transition. Note that the

joint compliance controller (of the Panda robot) that we used in these experiments only passively allows the contact forces to be within a fixed limit while pursuing a joint position-controlled goal. A more sophisticated force control strategy that actively regulates contact forces (such as [114]) can be integrated to achieve more robust compliant robot motion and make the insertion process more successful.

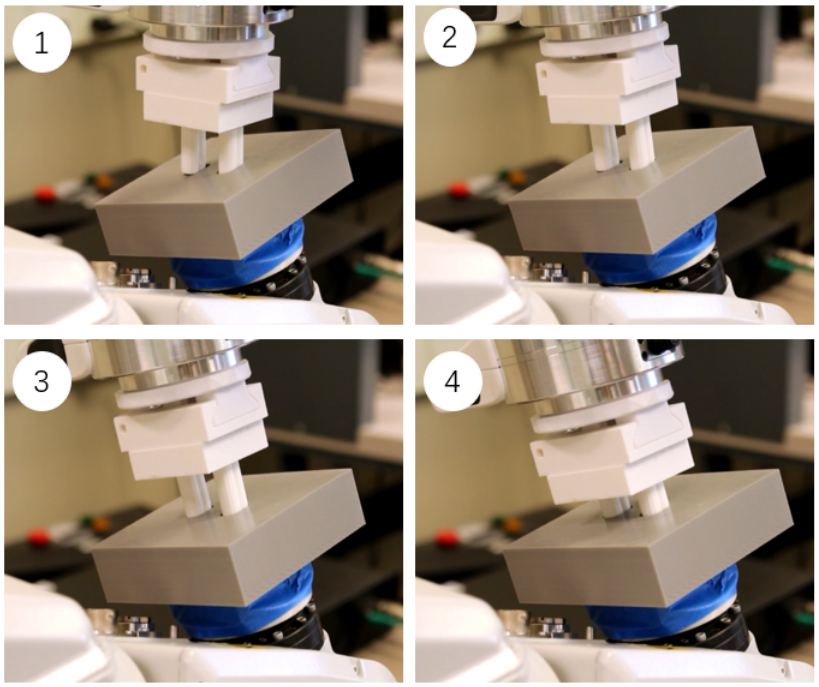


Figure 53: Motion snapshots of two-pin insertion with uncertainty reduction for the goal pose and compliant motion. Ground truth $(\alpha, \beta) = (9, -13)$. Predicted $(\alpha, \beta) = (7, -15)$. $\gamma = -1$ (°). (1) indicates the contact states encountered during direct insertion. (2) and (3) indicate the compliant execution of the robot motion under the contacts established in (1). (4) shows the inserted peg structure.

7.6 Three-Pin Peg-in-Hole Assembly

We further tested our approach on a three-pin peg-in-hole assembly task. We discretize α and β with a resolution of 2° over a range of ±15° for each and with $\gamma = 0°$. This results in 240 uncertain hole orientations, considering the inverse kinematics of the ABB robot, which are used to form the candidate set $\mathbb{C}$ and also the set Z for

Table 13: Summary of 10 robotic insertion experiments with goal-pose uncertainty reduction.

Case	Pose uncertainty (α, β in °)		Direct	Compliant
	True	Predicted		
1	$(9, -13)$	$(7, -15)$	×	✓
2	$(7, 1)$	$(3, 1)$	×	✓
3	$(11, -11)$	$(13, -15)$	×	✓
4	$(9, -9)$	$(9, -7)$	×	✓
5	$(-11, -7)$	$(-15, -3)$	×	✓
6	$(9, 1)$	$(5, 1)$	×	✓
7	$(-13, 7)$	$(-15, 1)$	×	✓
8	$(-7, -13)$	$(-13, -7)$	×	✓
9	$(7, 3)$	$(9, 11)$	×	×
10	$(-5, -7)$	$(-15, -5)$	×	×

training the force calibration model FC. Table 14 shows the pose estimation results using a fully connected neural network with 2 hidden layers and ReLu activation for force forecast. Each hidden layer has 16 neurons and the network outputs $3D$ forces. It can be seen that both $\Delta\alpha$ and $\Delta\beta$ are reduced using the introduced approach with calibrated force f_s, and such calibration regression model (FC) can be trained effectively with 48 (20%) force signals collected in the real world. About 81.6% of the testing poses have total errors ($\Delta\alpha + \Delta\beta$) fewer than 15° (using the regression model with structure 16-16-3 and 20% of data for training). This confirms the effectiveness of our approach.

Table 15 compares the effectiveness of different regression models. It can be seen that, more complex regressions models can better calibrate the forces in three-pin assembly tasks, which have more complex contacts comparing to two-pin assembly tasks. As shown in Table 15, $\Delta\alpha$ may still be too large. It is likely that the single

contact force signal captured (after the contacts stabilized) is not distinctive enough under such complex states. Force series data captured in the time window between initial contacts and the stabilized contacts can be further investigated to improve the performance. Fig. 54 shows an example three-pin peg-in-hole assembly through compliant insertion with reduced goal pose uncertainty. The video accompanied to [79] shows the robot motion.

Table 14: Uncertainty reduction in three-pin assembly. A fully connected neural network $(16 - 16 - 3)$ with 2 hidden layers (16 neurons each) with ReLu activation is used as the regression model to predict $3D$ forces. Trained with $1,000$ epoches.

Training split (%)	$\Delta\alpha(^\circ)$	$\Delta\beta(^\circ)$
10	7.62 ± 6.63	4.72 ± 6.25
20	6.94 ± 6.73	3.42 ± 4.22
40	7.21 ± 6.97	3.35 ± 4.16
50	7.02 ± 7.02	4.13 ± 4.77

Table 15: Comparison of regression models in three-pin assembly. Trained with $1,000$ epoches.

Train split (%)	Structure	$\Delta\alpha(^\circ)$	$\Delta\beta(^\circ)$
50	16-3	8.72 ± 7.62	3.66 ± 3.58
50	3	7.91 ± 6.62	3.41 ± 3.52
20	**16-16-3**	**6.94 ± 6.73**	**3.42 ± 4.22**
20	16-3	8.14 ± 7.77	4.15 ± 4.17
20	3	9.91 ± 8.81	5.31 ± 6.25

We also found that the regression model trained for the two-pin assembly case cannot be directly applied to the three-pin assembly case to achieve similar uncertainty reduction performance. This is because the regression model captures the information that is not modeled in haptic simulation but embedded in force data. Such residual information may be dependent on the object shape, maximum number of contact

regions, and also the mutual influence from different contact regions. One could train a universal and more complex regression model for force calibration based on a large number of contact states, so that the model might only need to be fine-tuned a bit when applying it to a new assembly task. However, this would front-load expensive training without guaranteeing universal effectiveness. Alternatively, we chose to train a simple and effective regression model for each task, which was done efficiently (with training time less than 1 min) using a small number of contact forces (about 40 forces) collected in the real world. Moreover, force data was collected automatically, and the speed was about 150 contact interactions per hour using our experimental setup.

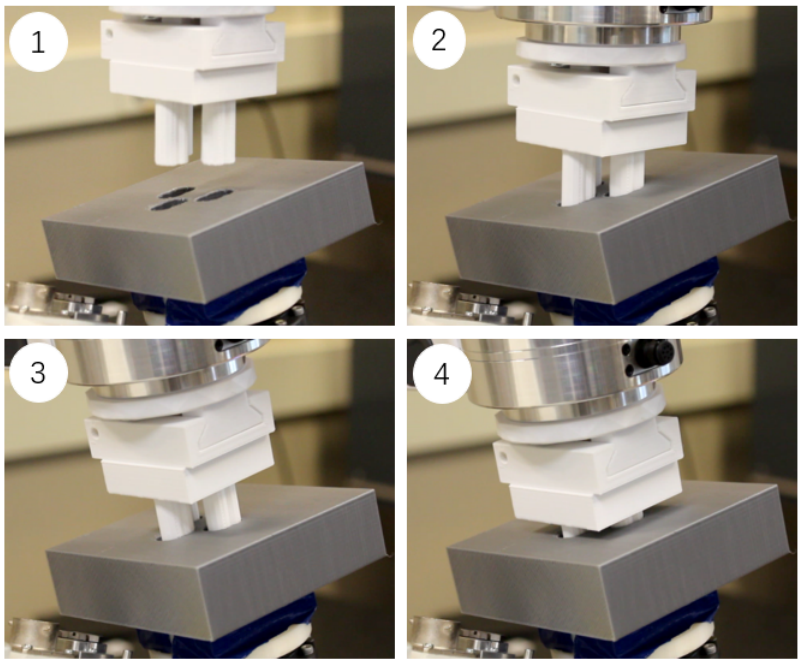


Figure 54: Motion snapshots of three-pin peg-in-hole insertion via uncertainty reduction of the goal pose and compliant transition. Ground truth $(\alpha, \beta) = (9, -7)$. Predicted $(\alpha, \beta) = (13, -3)$. $\gamma = 0$ (°). (1) shows the start of the insertion; (2) shows the motion blocked by the contacts; (3) shows the compliant transition; (4) shows the inserted peg structure.

7.7 Summary

In this chapter, a novel force forecast approach that relates real-world force sensing to a simulated world to enable pose uncertainty reduction of objects in contact-rich assembly/manipulation tasks. Our approach can handle multi-region complex contacts and makes no assumption about contact locations or any predefined contact types. It also does not put any restriction on object shapes. We applied this force forecast approach combined with compliant motion to achieve successful completion of challenging multi-peg-in-hole assembly tasks.

CHAPTER 8: CONCLUSIONS AND FUTURE WORK

To enable autonomous robotic operations in unknown and unstructured environments, it is important to have synergized robotic perception and manipulation. In this dissertation, novel approaches have been introduced to achieve autonomous robotic manipulation based on perception, and autonomous perception through manipulation.

8.1 Contributions

Manipulation-enabled perception: this dissertation presents a complete framework on how to use perception (tactile or visual) to enable continuum manipulation of an unknown object in an unknown environment, and also how to achieve object classification, recognition, shape estimation and appearance-based modeling based on continuum manipulation.

1. A novel approach is introduced on shape-based object classification and recognition with touch-based continuum manipulation. The introduced approach enables a continuum manipulator to automatically and progressively wrap around an unknown object based on tactile sensing of contacts in real time. The resulting shapes of the continuum manipulator wrapping around different objects are captured and used to train an classifier of object categories. In contrast to existing methods for robot and object interaction under discrete contacts,

the introduced method can capture object shape information more efficiently through whole-arm continuum manipulation.

2. An approach is introduced on estimating the shape of an unknown object based on touch-driven continuum manipulation. This approach efficiently utilizes the rich contact information in the motion as the robot progressively wraps around an unknown object. It fuses contact information and robot shape information (proprioception) to achieve more accurate shape estimation. The introduced approach also achieves shape estimation efficiently by actively guiding the robot to explore and cover unknown areas of the object. The approach is shown to be both more efficient and more accurate over existing methods for touch-based object shape estimation.

3. Experimental results presented demonstrate that the shape-based classifier trained solely from simulation is able to generalize to real-world objects, through using the robot shape to bridge the virtual world and real world. Since conducting many real-world continuum wraps can be time-consuming, it is significant that the classifier trained purely in simulation showed considerable effectiveness in classifying real objects. This could make classifier training more efficient and feasible for classifying a large number of categories of many real objects from touch-based continuum wrapping.

4. A general approach is introduced on progressive object modeling with a continuum manipulator in unknown and cluttered environments. By interleaving robot motion planning and perception, a continuum robot with a fixed base is

able to gradually maneuver through the unknown space without colliding with obstacles and sense the unmodeled target object from different viewpoints. The model of the target object is gradually built as the robot arm moves. The obtained object models can be used for further manipulation, such as grasping and object retrieval in cluttered environments.

Perception-guided manipulation: this dissertation also contributes to achieving more flexible and autonomous robotic manipulation based on perception in unknown or uncertain environments.

1. Real-time adaptive motion planning (RAMP) framework is extended by incorporating task constraints, which enables a robot manipulator to perform tasks with constraints on its end-effector effectively in unforeseen dynamic environments. The introduced algorithm is able to resolve conflicts between satisfying task constraints and avoidance of unexpected dynamic obstacles seamlessly by allowing the robot to change goals on-the-fly through releasing and resuming task constraints. Thus, more natural robot motion under task constraints is achieved for improved adaptiveness in dynamically unknown environments.

2. A novel force forecast approach is introduced to relate real world force sensing to a simulated world to enable pose uncertainty reduction. This approach corrects pose uncertainty in robotic assembly or manipulation tasks through finding the contact force in simulation that best matches the measured contact force in real world and adjusting the corresponding relative pose of the parts to reduce uncertainty. The introduced approach can handle multi-region complex contacts

and make no assumptions on contact locations or any predefined contact types. This force forecast approach has been applied in combination with robot compliant motion to achieve successful completion of challenging multi-peg-in-hole assembly tasks.

8.2 Future Work

Manipulation-enabled perception: one interesting extension to current shape-based object perception with touch-driven continuum manipulation is to relax the assumption that the target object is immobile, i.e., the object can be moved by the contacts made between the robot and the object. This introduces new challenges of how to simultaneously track the object in an unknown environment and still effectively wrap around it. Such extension is needed for practical object retrieval and manipulation in unknown and cluttered environments.

Moreover, it is also interesting to investigate planning compliant motion for continuum robots that can actively regulate the contact forces between the robot and the objects. This extension can further utilize the compliance embedded in continuum robots, and hence can enable continuum robots to manoeuvre in very cluttered and confined space, for more dexterous object manipulation in many applications such as robotic surgeries.

Continuum robots can also be equipped with a mobile base so that object modeling can be achieved for larger objects more efficiently. SLAM techniques [7] could be leveraged to simultaneously map the environment and localize the robot base.

The continuum robot hardware can be improved in multiple ways for better results

and robustness. For example, longer soft modules with more dense touch sensing can make the robot better explore the object shape. More sophisticated gravity compensation should be considered in order to better lift up the robot arm and form spatial wraps around different areas on the objects. Feedback control and robot proprioceptive sensors can be used to provide better control of the robot and better identify the robot's final shape.

Perception-guided manipulation: for real-time adaptive robot motion planning, one interesting extension is to explore accommodating very large trajectory population for enhanced real-time adaptiveness. Large-scale parallel processing techniques can be investigated to achieve real-time performance. Additionally, more efficient trajectory representation can be investigated to further improve the performance. For example, large trajectory population may be possibly embedded in neural networks, which may be an efficient data structure to capture a large number of diverse trajectories, and still maintain constant query time in online control cycles for real-time performance. More advanced sensing and real-time perception methods can be incorporated for autonomous selection of temporary task-constrained goals. Human-robot interaction can also be studied in this general and versatile framework.

The force forecast approach can be further extended to consider time series of force signals, which should contain more information about contact interaction to further improve the performance. Moreover, force forecast can be combined with visual perception for multi-modal uncertainty reduction of object pose uncertainty.

The introduced force forecast approach can be applied to other contact-rich robotic tasks. For example, robot motion planners can be informed by force forecast to avoid

contacts with excessive forces to protect the robot. This approach can also be applied to tasks involving deformable objects.

REFERENCES

[1] Abb externally guided motion. `https://github.com/ros-industrial/abb_libegm`. Accessed: 2018-08-29.

[2] Franka control interface. `https://frankaemika.github.io/docs/`. Accessed: 2019-07-16.

[3] Ros-industrial robot driver specification. `http://wiki.ros.org/Industrial/Industrial_Robot_Driver_Spec`. Accessed: 2018-08-29.

[4] Sphere-tree construction toolkit. `http://isg.cs.tcd.ie/spheretree/`. Accessed: 2019-07-20.

[5] P. K. Allen and P. Michelman. Acquisition and interpretation of 3-d sensor data from touch. In *Workshop on Interpretation of 3D Scenes*, pages 33–40. IEEE, 1989.

[6] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Bay, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

[7] J. Aulinas, Y. R. Petillot, J. Salvi, and X. Lladó. The slam problem: a survey. In *CCIA*, pages 363–371. Citeseer, 2008.

[8] A. Bajo, N. Simaan, et al. Kinematics-based detection and localization of contacts along multisegment continuum robots. *Transactions on Robotics*, 28(2):291–302, 2012.

[9] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner. Manipulation planning on constraint manifolds. In *International Conference on Robotics and Automation (ICRA)*, pages 625–632. IEEE, 2009.

[10] 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*, pages 360–366. IEEE, 2008.

[11] M. Björkman, Y. Bekiroglu, V. Högman, and D. Kragic. Enhancing visual perception of shape through tactile glances. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 3180–3186. IEEE/RSJ, 2013.

[12] J. Bohg and D. Kragic. Grasping familiar objects using shape context. In *2009 International Conference on Advanced Robotics*, pages 1–6. IEEE, 2009.

[13] J. Bohg, A. Morales, T. Asfour, and D. Kragic. Data-driven grasp synthesisa survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2013.

[14] M. Bonilla, E. Farnioli, C. Piazza, M. Catalano, G. Grioli, M. Garabini, M. Gabiccini, and A. Bicchi. Grasping with soft hands. In *International Conference on Humanoid Robots*, pages 581–587. IEEE, 2014.

[15] O. Brock and O. Khatib. Real-time re-planning in high-dimensional configuration spaces using sets of homotopic paths. In *International Conference on Robotics and Automation (ICRA)*, volume 1, pages 550–555. IEEE, 2000.

[16] 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*, 30(5):1260–1269, 2014.

[17] 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.

[18] H. Bruyninckx, S. Dutre, and J. De Schutter. Peg-on-hole: a model based solution to peg and hole alignment. In *International Conference on Robotics and Automation*, volume 2, pages 1919–1924. IEEE, 1995.

[19] A. Byravan, B. Boots, S. S. Srinivasa, and D. Fox. Space-time functional gradient optimization for motion planning. In *International Conference on Robotics and Automation (ICRA)*, pages 6499–6506. IEEE, 2014.

[20] S. Caccamo, Y. Bekiroglu, C. H. Ek, and D. Kragic. Active exploration using gaussian random fields and gaussian process implicit surfaces. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 582–589. IEEE/RSJ, 2016.

[21] R. Calandra, A. Owens, M. Upadhyaya, W. Yuan, J. Lin, E. H. Adelson, and S. Levine. The feeling of success: Does touch sensing help predict grasp outcomes? In *Conference on Robot Learning(CoRL)*, 2017.

[22] B. Calli, A. Kimmel, K. Hang, K. Bekris, and A. Dollar. Path planning for within-hand manipulation over learned representations of safe states. In *International Symposium on Experimental Robotics (ISER)*, 2018.

[23] M. Cefalo, G. Oriolo, and M. Vendittelli. Task-constrained motion planning with moving obstacles. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 5758–5763. IEEE/RSJ, 2013.

[24] C. Choi and H. I. Christensen. 3d pose estimation of daily objects using an rgb-d camera. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 3342–3349. IEEE, 2012.

[25] R. Detry, C. H. Ek, M. Madry, and D. Kragic. Learning a dictionary of prototypical grasp-predicting parts from grasping experience. In *International Conference on Robotics and Automation (ICRA)*, pages 601–608. IEEE, 2013.

[26] S. Dragiev, M. Toussaint, and M. Gienger. Gaussian process implicit surfaces for shape estimation and grasping. In *International Conference on Robotics and Automation (ICRA)*, pages 2845–2850. IEEE, 2011.

[27] S. H. Drake. *Using compliance in lieu of sensory feedback for automatic assembly.* PhD thesis, Massachusetts Institute of Technology, 1978.

[28] 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*, 34(7):1021–1038, 2015.

[29] 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*, 33(4):431–450, 2016.

[30] R. Gayle, K. R. Klingler, and P. G. Xavier. Lazy reconfiguration forest (lrf)-an approach for motion planning with multiple tasks in dynamic environments. In *International Conference on Robotics and Automation (ICRA)*, pages 1316–1323. IEEE, 2007.

[31] M. P. Gerardo-Castro, T. Peynot, and F. Ramos. Laser-radar data fusion with gaussian process implicit surfaces. In *Field and Service Robotics*, pages 289–302. Springer, 2015.

[32] L. Han, L. Rudolph, J. Blumenthal, and I. Valodzin. Convexly stratified deformation spaces and efficient path planning for planar closed chains with revolute joints. *The International Journal of Robotics Research*, 27(11-12):1189–1212, 2008.

[33] K. Huang and T. Hermans. Building 3d object models during manipulation by reconstruction-aware trajectory optimization. *arXiv preprint arXiv:1905.03907*, 2019.

[34] R. Ibrayev and Y.-B. Jia. Recognition of curved surfaces from one-dimensional tactile data. *IEEE Transactions on Automation Science and Engineering*, 9(3):613–621, 2012.

[35] J. Ilonen, J. Bohg, and V. Kyrki. Fusing visual and tactile sensing for 3-d object reconstruction while grasping. In *International Conference on Robotics and Automation (ICRA)*, pages 3547–3554. IEEE, 2013.

[36] 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, 2013.

[37] G. Izatt, G. Mirano, E. Adelson, and R. Tedrake. Tracking objects with point clouds from vision and touch. In *International Conference on Robotics and Automation (ICRA)*, pages 4000–4007. IEEE, 2017.

[38] L. Jaillet and J. M. Porta. Path planning under kinematic constraints by rapidly exploring manifolds. *IEEE Transactions on Robotics (TRO)*, 29(1):105–117, 2013.

[39] N. Jamali, C. Ciliberto, L. Rosasco, and L. Natale. Active perception: Building objects' models using tactile exploration. In *International Conference on Humanoid Robots (Humanoids)*, pages 179–185. IEEE, 2016.

[40] X. Ji and J. Xiao. Planning motions compliant to complex contact states. *The International Journal of Robotics Research (IJRR)*, 20(6):446–465, 2001.

[41] Y.-B. Jia and J. Tian. Surface patch reconstruction from one-dimensional tactile data. *IEEE Transactions on Automation Science and Engineering*, 7(2):400–407, 2009.

[42] B. A. Jones and I. D. Walker. Kinematics for multisection continuum robots. *Transactions on Robotics*, 22(1):43–55, 2006.

[43] M. Kaboli, D. Feng, K. Yao, P. Lanillos, and G. Cheng. A tactile-based framework for active object learning and discrimination using multimodal robotic skin. *IEEE Robotics and Automation Letters*, 2(4):2143–2150, 2017.

[44] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *International Conference on Robotics and Automation (ICRA)*, pages 4569–4574. IEEE, 2011.

[45] D. Kappler, F. Meier, J. Issac, J. Mainprice, C. G. Cifuentes, M. Wüthrich, V. Berenz, S. Schaal, N. Ratliff, and J. Bohg. Real-time perception meets reactive motion generation. *IEEE Robotics and Automation Letters*, 3(3):1864–1871, 2018.

[46] Y. Karayiannidis, C. Smith, F. E. Vina, and D. Kragic. Online contact point estimation for uncalibrated tool use. In *International Conference on Robotics and Automation (ICRA)*, pages 2488–2494. IEEE, 2014.

[47] C. H. Kim and J. Seo. Shallow-depth insertion: Peg in shallow hole through robotic in-hand manipulation. *IEEE Robotics and Automation Letters*, 4(2):383–390, 2019.

[48] Z. Kingston, M. Moll, and L. E. Kavraki. Decoupling constraints from sampling-based planners. In *International Symposium on Robotics Research*, 2017.

[49] 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)*, pages 5031–5037. IEEE, 2011.

[50] S. J. Kwak, T. Hasegawa, and S. Y. Chung. A framework for automatic generation of a contact state graph for robotic assembly. *Advanced Robotics*, 25(13-14):1603–1625, 2011.

[51] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg. Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8943–8950. IEEE, 2019.

[52] P. Lehner, A. Sieverling, and O. Brock. Incremental, sensor-based motion generation for mobile manipulators in unknown, dynamic environments. In *International Conference on Robotics and Automation (ICRA)*, pages 4761–4767. IEEE, 2015.

[53] P. Leven and S. Hutchinson. A framework for real-time path planning in changing environments. *The International Journal of Robotics Research*, 21(12):999–1030, 2002.

[54] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research (IJRR)*, 37(4-5):421–436, 2018.

[55] J. Li, S. Dong, and E. Adelson. Slip detection with combined tactile and visual information. In *International Conference on Robotics and Automation (ICRA)*, pages 7772–7777. IEEE, 2018.

[56] J. Li, Z. Teng, and J. Xiao. Can a continuum manipulator fetch an object in an unknown cluttered space? *IEEE Robotics and Automation Letters*, 2(1):2–9, 2017.

[57] J. Li, Z. Teng, J. Xiao, A. Kapadia, A. Bartow, and I. Walker. Autonomous continuum grasping. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 4569–4576. IEEE/RSJ, 2013.

[58] J. Li and J. Xiao. Determining grasping configurations for a spatial continuum manipulator. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 4207–4214. IEEE/RSJ, 2011.

[59] J. Li and J. Xiao. Progressive, continuum grasping in cluttered space. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 4563–4568. IEEE/RSJ, 2013.

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

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

[62] J. Li and J. Xiao. An efficient algorithm for real time collision detection involving a continuum manipulator with multiple uniform-curvature sections. *Robotica*, 34(7):1566–1586, 2016.

[63] J. Li and J. Xiao. Progressive planning of continuum grasping in cluttered space. *IEEE Transactions on Robotics(TRO)*, 32(3):707–716, 2016.

[64] J. Li and J. Xiao. Progressive planning of continuum grasping in cluttered space. *IEEE Transactions on Robotics*, 32(3):707–716, 2016.

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

[66] Q. Li, A. Ückermann, R. Haschke, and H. J. Ritter. Estimating an articulated tools kinematics via visuo-tactile based robotic interactive manipulation. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 6938–6944. IEEE/RSJ, 2018.

[67] R. Li and E. H. Adelson. Sensing and recognizing surface textures using a gelsight sensor. In *Conference on Computer Vision and Pattern Recognition*, pages 1241–1247. IEEE, 2013.

[68] S. Li, S. Lyu, and J. Trinkle. State estimation for dynamic systems with intermittent contact. In *International Conference on Robotics and Automation (ICRA)*, pages 3709–3715. IEEE, 2015.

[69] 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 *International Conference on Intelligent Robots and Systems (IROS)*, pages 5073–5080. IEEE/RSJ, 2015.

[70] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

[71] Q. Luo and J. Xiao. Contact and deformation modeling for interactive environments. *IEEE Transactions on Robotics*, 23(3):416–430, 2007.

[72] D. Ma, E. Donlon, S. Dong, and A. Rodriguez. Dense tactile force distribution estimation using gelslim and inverse fem. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[73] L. Ma, M. Ghafarianzadeh, D. Coleman, N. Correll, and G. Sibley. Simultaneous localization, mapping, and manipulation for unsupervised object discovery. In *International Conference on Robotics and Automation (ICRA)*, pages 1344–1351. IEEE, 2015.

[74] R. Ma and A. Dollar. Yale openhand project: Optimizing open-source hand designs for ease of fabrication and adoption. *IEEE Robotics & Automation Magazine*, 24(1):32–40, 2017.

[75] H. Mao, J. Santoso, C. Onal, and J. Xiao. Sim-to-real transferable object classification through touch-based continuum manipulation. In *International Symposium on Experimental Robotics (ISER)*. Springer, 2018.

[76] H. Mao, Z. Teng, and J. Xiao. Progressive object modeling with a continuum manipulator in unknown environments. In *International Conference on Robotics and Automation (ICRA)*, pages 5674–5681. IEEE, 2017.

[77] H. Mao and J. Xiao. Object shape estimation through touch-based continuum manipulation. In *International Symposium of Robotics Research (ISRR)*, 2017.

[78] H. Mao and J. Xiao. Real-time conflict resolution of task-constrained manipulator motion in unforeseen dynamic environments. In *IEEE Transactions on Robotics (TRO)*, 2019.

[79] H. Mao and J. Xiao. Reducing pose uncertainty under complex contacts via force forecast. In *in submission to Robotics and Automation Letters*. IEEE, 2019.

[80] H. Mao, J. Xiao, M. M. Zhang, and K. Daniilidis. Shape-based object classification and recognition through continuum manipulation. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 456–463. IEEE/RSJ, 2017.

[81] S. McLeod and J. Xiao. Real-time adaptive non-holonomic motion planning in unforeseen dynamic environments. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 4692–4699. IEEE/RSJ, 2016.

[82] 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 *International Conference on Robotics and Automation (ICRA)*, pages 2336–2341. IEEE, 2006.

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

[84] T. McMahon, S. Thomas, and N. M. Amato. Sampling-based motion planning with reachable volumes: Theoretical foundations. In *International Conference on Robotics and Automation (ICRA)*, pages 6514–6521. IEEE, 2014.

[85] M. Meier, M. Schopfer, R. Haschke, and H. Ritter. A probabilistic approach to tactile shape reconstruction. *Transactions on Robotics(TRO)*, 27(3):630–635, 2011.

[86] V. Modugno, G. Neumann, E. Rueckert, G. Oriolo, J. Peters, and S. Ivaldi. Learning soft task priorities for control of redundant robots. In *International Conference on Robotics and Automation (ICRA)*, pages 221–226. IEEE, 2016.

[87] D. Mumford, J. Fogarty, and F. Kirwan. *Geometric invariant theory*, volume 34. Springer Science & Business Media, 1994.

[88] A. Murali, Y. Li, D. Gandhi, and A. Gupta. Learning to grasp without seeing. *arXiv preprint arXiv:1805.04201*, 2018.

[89] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, volume 11, pages 127–136, 2011.

[90] H. Nguyen and Q.-C. Pham. A probabilistic framework for tracking uncertainties in robotic manipulation. *arXiv preprint arXiv:1901.00969*, 2019.

[91] A. M. Okamura and M. Curkosky. Feature-guided exploration with a robotic finger. In *International Conference on Robotics and Automation (ICRA)*, volume 1, pages 589–596. IEEE, 2001.

[92] M. Otte and E. Frazzoli. RRT^X: Real-time motion planning/replanning for environments with unpredictable obstacles. In *Algorithmic Foundations of Robotics XI*, pages 461–478. Springer, 2015.

[93] C. Park, F. Rabe, S. Sharma, C. Scheurer, U. E. Zimmermann, and D. Manocha. Parallel cartesian planning in dynamic environments using constrained trajectory planning. In *IEEE-RAS International Conference on Humanoid Robots*, pages 983–990. IEEE, 2015.

[94] D. Pavlichenko and S. Behnke. Efficient stochastic multicriteria arm trajectory optimization. In *International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2017.

[95] A. Petrovskaya and O. Khatib. Global localization of objects via touch. *IEEE Transactions on Robotics (TRO)*, 27(3):569–585, 2011.

[96] T.-H. Pham, G. De Magistris, and R. Tachibana. Optlayer-practical constrained optimization for deep reinforcement learning in the real world. In *International Conference on Robotics and Automation (ICRA)*, pages 6236–6243. IEEE, 2018.

[97] 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. *Robotics and Autonomous Systems*, 58(5):551–565, 2010.

[98] C. E. Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[99] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation(ICRA)*, pages 489–494. IEEE, 2009.

[100] 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.

[101] G. Robinson and J. B. C. Davies. Continuum robots-a state of the art. In *International Conference on Robotics and Automation (ICRA)*, volume 4, pages 2849–2854. IEEE, 1999.

[102] J. Santoso, E. H. Skorina, M. Luo, R. Yan, and C. D. Onal. Design and analysis of an origami continuum manipulation module with torsional strength. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2098–2104. IEEE/RSJ, 2017.

[103] A. Saric, J. Xiao, and J. Shi. Robotic surface assembly via contact state transitions. In *International Conference on Automation Science and Engineering (CASE)*, pages 954–959. IEEE, 2013.

[104] A. Saxena, J. Driemeyer, and A. Y. Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research (IJRR)*, 27(2):157–173, 2008.

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

[106] L. Sentis and O. Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 2(04):505–518, 2005.

[107] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel. Bigbird: A large-scale 3d database of object instances. In *International Conference on Robotics and Automation (ICRA)*, pages 509–516. IEEE, 2014.

[108] N. Sommer, M. Li, and A. Billard. Bimanual compliant tactile exploration for grasping unknown objects. In *International Conference on Robotics and Automation (ICRA)*, pages 6400–6407. IEEE, 2014.

[109] M. Stilman. Task constrained motion planning in robot joint space. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 3074–3081. IEEE/RSJ, 2007.

[110] C. Suh, T. T. Um, B. Kim, H. Noh, M. Kim, and F. C. Park. Tangent space rrt: A randomized planning algorithm on constraint manifolds. In *International Conference on Robotics and Automation (ICRA)*, pages 4968–4973. IEEE, 2011.

[111] Z. Sui, Z. Zhou, Z. Zeng, and O. C. Jenkins. Sum: Sequential scene understanding and manipulation. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 3281–3288. IEEE/RSJ, 2017.

[112] W. Sun, S. Patil, and R. Alterovitz. High-frequency replanning under uncertainty using parallel sampling-based motion planning. *IEEE Transactions on Robotics*, 31(1):104–116, 2015.

[113] B. Sundaralingam, A. Handa, B. Boots, T. Hermans, S. Birchfield, N. Ratliff, D. Fox, et al. Robust learning of tactile force estimation through robot interaction. In *International Conference on Robotics and Automation (ICRA)*, pages 9035–9042. IEEE, 2019.

[114] M. Suomalainen, S. Calinon, E. Pignat, and V. Kyrki. Improving dual-arm assembly by master-slave compliance. In *International Conference on Robotics and Automation (ICRA)*, pages 8676–8682. IEEE, 2019.

[115] P. Tang and J. Xiao. Automatic generation of a high-level contact state graph for assembly between curved objects. In *2007 IEEE International Symposium on Assembly and Manufacturing*, pages 197–202. IEEE, 2007.

[116] P. Tang and J. Xiao. Automatic generation of high-level contact state space between 3d curved objects. *The International Journal of Robotics Research (IJRR)*, 27(7):832–854, 2008.

[117] Z. Teng, H. Mao, and J. Xiao. Automatic object modeling through integrating perception and robotic manipulation. In *International Symposium on Experimental Robotics (ISER)*, pages 223–233. Springer, 2016.

[118] Z. Teng and J. Xiao. Surface-based detection and 6-dof pose estimation of 3-d objects in cluttered scenes. *IEEE Transactions on Robotics(TRO)*, 32(6):1347–1361, 2016.

[119] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, and P. Abbeel. Learning robotic assembly from cad. In *International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[120] L. G. Torres, C. Baykal, and R. Alterovitz. Interactive-rate motion planning for concentric tube robots. In *International Conference on Robotics and Automation (ICRA)*, pages 1915–1921. IEEE, 2014.

[121] A. Toshev, B. Taskar, and K. Daniilidis. Shape-based object detection via boundary structure segmentation. *International Journal of Computer Vision*, 2012.

[122] 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*, 5(3):99–117, 2008.

[123] J. C. Triyonoputro, W. Wan, and K. Harada. Quickly inserting pegs into uncertain holes using multi-view images and deep network trained on synthetic data. In *International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2019.

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

[125] J. Vannoy and J. Xiao. Real-time adaptive motion planning (ramp) of mobile manipulators in dynamic environments with unforeseen changes. *IEEE Transactions on Robotics*, 24(5):1199–1212, 2008.

[126] 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*, 11:1–13, 2014.

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

[128] I. D. Walker. Continuous backbone continuum robot manipulators. *ISRN Robotics*, 2013, 2013.

[129] D. Wang, X. Zhang, Y. Zhang, and J. Xiao. Configuration-based optimization for six degree-of-freedom haptic rendering for fine manipulation. *IEEE transactions on haptics*, 6(2):167–180, 2012.

[130] R. J. Webster III, J. M. Romano, and N. J. Cowan. Mechanics of precurved-tube continuum robots. *Transactions on Robotics*, 25(1):67–78, 2008.

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

[132] F. Wirnshofer, P. S. Schmitt, W. Feiten, G. v. Wichert, and W. Burgard. Robust, compliant assembly via optimal belief space planning. In *International Conference on Robotics and Automation (ICRA)*, pages 5436–5443. IEEE, 2018.

[133] J. Xiang, C. Zhong, and W. Wei. A varied weights method for the kinematic control of redundant manipulators with multiple constraints. *IEEE Transactions on Robotics*, 28(2):330–340, 2012.

[134] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[135] J. Xiao and X. Ji. Automatic generation of high-level contact state space. *The International Journal of Robotics Research (IJRR)*, 20(7):584–606, 2001.

[136] Y. Yang and O. Brock. Elastic roadmaps motion generation for autonomous mobile manipulation. *Autonomous Robots*, 28(1):113–130, 2010.

[137] Z. Yi, R. Calandra, F. Veiga, H. van Hoof, T. Hermans, Y. Zhang, and J. Peters. Active tactile object exploration with gaussian processes. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 4925–4930. IEEE/RSJ, 2016.

[138] T. Yoshikawa. Manipulability of robotic mechanisms. *The international journal of Robotics Research*, 4(2):3–9, 1985.

[139] K.-T. Yu and A. Rodriguez. Realtime state estimation with tactile and visual sensing. application to planar manipulation. pages 7778–7785, 2018.

[140] K.-T. Yu and A. Rodriguez. Realtime state estimation with tactile and visual sensing for inserting a suction-held object. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 1628–1635. IEEE/RSJ, 2018.

[141] W. Yuan, S. Dong, and E. H. Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017.

[142] W. Yuan, Y. Mo, S. Wang, and E. H. Adelson. Active clothing material perception using tactile sensing and deep learning. In *International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[143] W. Yuan, M. A. Srinivasan, and E. H. Adelson. Estimating object hardness with a gelsight touch sensor. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 208–215. IEEE/RSJ, 2016.

[144] W. Yuan, S. Wang, S. Dong, and E. Adelson. Connecting look and feel: Associating the visual and tactile properties of physical materials. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5580–5588, 2017.

[145] W. Yuan, C. Zhu, A. Owens, M. A. Srinivasan, and E. H. Adelson. Shape-independent hardness estimation using deep learning and a gelsight tactile sensor. In *International Conference on Robotics and Automation (ICRA)*, pages 951–958. IEEE, 2017.

[146] M. M. Zhang, N. Atanasov, and K. Daniilidis. Active end-effector pose selection for tactile object recognition through monte carlo tree search. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 3258–3265. IEEE/RSJ, 2017.

[147] 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)*, pages 4931–4938. IEEE/RSJ, 2016.

[148] L. Zhu, H. Mao, X. Luo, and J. Xiao. Determining null-space motion to satisfy both task constraints and obstacle avoidance. In *IEEE International Symposium on Assembly and Manufacturing (ISAM)*, pages 112–119. IEEE, 2016.

[149] 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 *International Conference on Robotics and Automation (ICRA)*. IEEE, 2014.

PUBLICATIONS

[1] Huitan Mao and Jing Xiao, "Reducing Pose Uncertainty under Complex Contacts via Force Forecast", in submission to IEEE Robotics and Automation Letters (RAL), 2019.

[2] Huitan Mao and Jing Xiao, "Task-constrained Real-time Adaptive Motion Planning of Robot Manipulators in Unforeseen Dynamic Environments", IEEE Transactions on Robotics (TRO), 2019.

[3] Sean McGovern, Huitan Mao and Jing Xiao, "Learning to Estimate Centers of Mass of Arbitrary Objects", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019.

[4] Huitan Mao, Junius Santoso, Cagdas Onal, and Jing Xiao, "Sim-to-real Transferable Object Classification through Touch-based Continuum Manipulation", International Symposium on Experimental Robotics (ISER), 2018.

[5] Huitan Mao and Jing Xiao, "Object Shape Estimation through Touch-based Continuum Manipulation", International Symposium on Robotics Research (ISRR), 2017.

[6] Huitan Mao, Mabel M. Zhang, Jing Xiao, and Kostas Daniilidis, "Shape-based Object Classification and Recognition through Continuum Manipulation", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017.

[7] Huitan Mao, Zhou Teng, and Jing Xiao, "Progressive Object Modeling with a Continuum Manipulator in Unknown Environments", IEEE International Conference on Robotics and Automation (ICRA), 2017.

[8] Zhou Teng, Huitan Mao, and Jing Xiao, "Automatic Object Modeling through Integrating Perception and Robotic Manipulation", International Symposium on Experimental Robotics (ISER), 2016.

[9] Liqin Zhu, Huitan Mao, Xiang Luo, Jing Xiao,"Determining Null-space Motion to satisfy Both Task Constraints and Obstacle Avoidance", IEEE International Symposium on Assembly and Manufacturing (ISAM), 2016.