Functional Contour-following via Haptic Perception and Reinforcement Learning

Randall B. Hellman, *Member, IEEE*, Cem Tekin, *Member, IEEE*,
Mihaela van der Schaar, *Fellow, IEEE*, and Veronica J. Santos, *Member, IEEE*

Abstract—Many tasks involve the fine manipulation of objects despite limited visual feedback. In such scenarios, tactile and proprioceptive feedback can be leveraged for task completion. We present an approach for real-time haptic perception and decision-making for a haptics-driven, functional contour-following task: the closure of a ziplock bag. This task is challenging for robots because the bag is deformable, transparent, and visually occluded by artificial fingertip sensors that are also compliant. A deep neural net classifier was trained to estimate the state of a zipper within a robot's pinch grasp. A Contextual Multi-Armed Bandit (C-MAB) reinforcement learning algorithm was implemented to maximize cumulative rewards by balancing exploration versus exploitation of the state-action space. The C-MAB learner outperformed a benchmark Q-learner by more efficiently exploring the state-action space while learning a hard-to-code task. The learned C-MAB policy was tested with novel ziplock bag scenarios and contours (wire, rope). Importantly, this work contributes to the development of reinforcement learning approaches that account for limited resources such as hardware life and researcher time. As robots are used to perform complex, physically interactive tasks in unstructured or unmodeled environments, it becomes important to develop methods that enable efficient and effective learning with physical testbeds.

Index Terms—Active touch, contour-following, decision making, haptic perception, manipulation, reinforcement learning

1 Introduction

Contour-following is often necessary for tasks requiring fine manipulation skills such as tying shoelaces, searching pockets, and buttoning shirts. The ability to perform touch-driven contour-following enables specific dexterous abilities that are necessary for capable robotic systems. We present a functional form of contour-following in which the contour is the zipper on a deformable ziplock bag to be closed.

Early work in contour-following used contact forces to track edges [1]. Scene and tactile images of the contacted interface were filtered to determine edges for vision-based contour-following [2]. Techniques integrating vision, force sensing, and accelerometers have also been used to track contours [3]. The effectiveness of these computer vision approaches can be limited if target objects are deformable, occluded, transparent, or otherwise optically ambiguous [4]. In such scenarios, it becomes necessary to additionally rely on haptic and proprioceptive feedback in order to perform functional tasks. Active perception is critical during contour-following as it is necessary for closed-loop repositioning of the manipulator due to error propagation and changes in the contour [5].

Even with the addition of proprioception, manipulation is often challenging due to compliance and uncertainty in robotic systems [6] and inadequate models of the environment. Robotic systems have various levels of proprioceptive precision. For instance, lash in tendon-driven systems or static friction in joints can lead to noisy torque estimates. Compliance in soft robot arms can mask finger-object interactions that would otherwise be observable with precise, direct-drive robot arms [7]. As robots become more commonplace they will begin to interact in ways similar to their human counterparts. Interactions will involve planned and incidental contact with areas of the robotic system beyond the end-effector [8]. For example, proprioceptive data cannot solely be used to estimate localized contact information at a fingertip when a robot's forearm also interacts with the environment.

In this work, we rely heavily upon tactile percepts that are localized near regions of finger-object contact, as visual and proprioceptive feedback can be noisy or intermittent. In this way, we minimize error propagation in forward calculations from visual and proprioceptive systems, and reduce uncertainty due to incidental contact of other regions of the robot with the environment.

Tasks that require contour-following are not limited to the manipulation of rigid objects. The manipulation of deformable objects such as rope, cloth, and sponges have been a focus of extensive research due to their applications in surgical and service robotics [9], [10], [11], [12]. Both physics-based and model-free approaches have been

- *R.B. Hellman and V.J. Santos are with the Mechanical and Aerospace Engineering Department, University of California, Los Angeles, CA 90095. E-mail: {rhellman, vjsantos}@ucla.edu.*
- *C. Tekin is with the Department of Electrical and Electronics Engineering, Bilkent University, Ankara 06800, Turkey. E-mail: cemtekin@ee.bilkent.edu.tr.*
- *M. van der Schaar is with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095. E-mail: Mihaela@ee.ucla.edu.*

Manuscript received 12 Dec. 2016; revised 5 July 2017; accepted 5 Sept. 2017. Date of publication 18 Sept. 2017; date of current version 26 Mar. 2018. (Corresponding author: Veronica J. Santos.)
Recommended for acceptance by M. Harders.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TOH.2017.2753233

successfully implemented [12], [13], [14], [15], [16]. In the work presented here, we use a model-free approach in which the stochastic nature of physical interactions with a deformable object is learned from action-driven, haptic and proprioceptive sensory feedback.

To learn a novel manipulation task, it is typically necessary to use physical robot systems as opposed to simulations alone. Models of deformable objects [13], [14], [16] and friction under soft contact conditions [17], [18] are still under development. Additionally, no model currently exists for accurate simulation of the electrical response of the deformable tactile sensor used in this work [19]. Thus, it is important to employ efficient learning algorithms, as brute force approaches that might work in simulation would be time consuming and induce unnecessary wear on physical robot systems.

Reinforcement learning has been successfully used to teach robots complex skills that would be difficult to code directly (e.g., [20]) in both real-world and simulated environments. Multi-Armed Bandit (MAB) models, a variant of reinforcement learning, have been developed for grasp selection [21], [22] and rearrangement planning [23] for robot interactions with rigid objects. However, these methods do not select or adapt actions based on information provided by real-time sensory feedback. An extension of MAB models called Contextual MABs can effectively utilize immediate sensory feedback to select actions [24], [25], [26], [27]. To the best of our knowledge, we are the first to apply reinforcement learning with Contextual MABs to robotic systems for haptics-driven manipulation. Additionally, the system is the first real-time hardware implementation of Contextual MABs and demonstrates a practical advantage for hardware-based experiments that are wear and time intensive.

2 Methods

2.1 Functional Contour-Following Task

The closure of a plastic ziplock bag is an everyday task that presents unique challenges for robotic systems. The task requires the manipulation of a transparent, deformable object whose geometric features are visually occluded by artificial fingertips. Computer vision could be useful for planning, but will likely be insufficient for task execution that requires carefully maintained finger-object relationships. While the closure of a plastic bag is typically a bimanual task, we simplified the experimental set-up to focus on the control of a single sensorized robot hand, wrist, and arm.

To successfully complete the task, we will need to determine the current orientation of the zipper in the grasp and then take a goal-directed action. Ideally, the action chosen would result in a new state that brings the system closer to achieving its goal: zipper closure without losing grasp of the bag.

2.2 Robot Testbed

The robot testbed consisted of a 7 degree-of-freedom (DOF) Barrett Whole Arm Manipulator (WAM) with a 4 DOF BarrettHand (Barrett Technology, Cambridge, MA) (Fig. 1). The BarrettHand can independently flex/extend three digits and adduct/abduct the outer two digits. BioTac sensors (SynTouch LLC, Los Angeles, CA) were attached to two opposing digits on the hand for the recording of internal fluid pressure, vibration, skin deformation, and temperature [19]. Due to the redundant information from the two BioTacs in this particular experiment, only a single BioTac was used for analysis. Electrode impedance was sampled at 100 Hz and internal fluid pressure was sampled at 100 Hz and 2200 Hz.

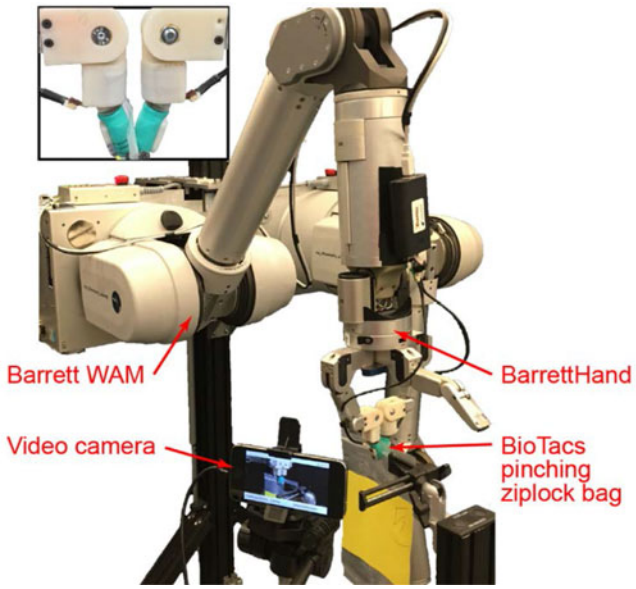

Fig. 1. Experimental set-up used to develop online haptic perception and decision-making capabilities for a functional contour-following task. *Inset:* A custom adapter enabled a 20 degree contact angle.

To achieve a precision pinch with ample fingerpad contact areas for bag closure, a custom adapter was used to reduce the BarrettHand-BioTac system to a sensorized parallel gripper. The variable adapter enabled the selection of a task-appropriate fingertip contact angle of 20 degree per digit (Fig. 1, inset).

For reliable control and sensing, command signals and sensor sampling were handled by hard real-time modules. Hard real-time processes were controlled from a Linux computer running Ubuntu 12.04 modified with a Xenomai kernel. Computational tasks such as haptic perception, decision-making, and reinforcement learning were implemented by soft real-time processes on a 2013 MacBook Pro.

Sensory data from the hard real-time module were sent to the external soft real-time module for processing. MATLAB (MathWorks, Natick, MA) and TensorFlow [28] were used for the online analysis of tactile (BioTac) and proprioceptive (WAM) data.

2.2.1 Motion Planning and Data Collection

An online motion planner was used to generate joint space trajectories for each action. The Robot Operating System [29] was used to communicate between computational nodes and the motion planner. MoveIt! was used for motion planning, and collision and self-collision avoidance [30]. MoveIt! incorporates various planning algorithms; we selected the commonly used sampling-based Optimally Rapidly-exploring Random Trees (RRT*) planner from the Open Motion Planning Library [31].

Close-up videos of the artificial fingertips and zipper were recorded for each action for the training of supervised

learning algorithms. A custom iOS application (Apple Inc., 2014) was developed for an iPhone 5s for video and image capture during reinforcement learning. In addition to these videos, synchronized multimodal tactile sensor data, WAM joint positions, velocities, and torques, were recorded during each action.

2.3 Reinforcement Learning

Reinforcement learning is a well-established approach for learning a policy to select actions in order to maximize expected rewards. During reinforcement learning, state and action spaces are explored and goal-directed rewards are given after each action. The immediate reward is then used to update the expected reward from the action taken.

2.3.1 Q-Learning

Q-learning was used as a baseline for comparison with the Contextual Multi-Armed Bandits approach described later. As a temporal difference method, Q-learning results in delayed rewards. Q learning will converge to the optimal policy asymptotically as time goes to infinity, but cannot provide performance guarantees of optimality for finite time periods. The policy is in the form of a "Q matrix" that contains the expected rewards for each "state-action pair" [32], [33]. Rows and columns of the Q-matrix correspond to states and actions, respectively. A single entry in the Q-matrix corresponds to the expected reward of taking an action given the current state (i.e., the reward for a specific "state-action pair"). It is necessary for the Q-learning algorithm to comprehensively explore the state space in order for rewards from goal success to back-propagate through the system. Given enough exploration, the expected reward of each state will converge to the optimal solution. However, exploration and exploitation must be balanced. Excessive exploration to reduce uncertainty wastes time and increases regret, which is the difference in rewards earned by an agent acting optimally versus those earned by an agent that receives rewards randomly.

Designers of Q-learning algorithms must set three parameters: learning rate α, discount rate γ, and exploration rate ε. The *learning rate* determines how quickly expected rewards are updated based on the difference in the expected reward and actual reward observed. If the learning rate is too high, then stochastic responses can excessively influence expected rewards. The learning rate was set to 0.25 in order to account for the fact that rewards will be stochastic and little is known about the expected rewards. The *discount rate* can be used to specify a decrease in value of future rewards. A low discount rate causes the learner to behave in a short-sighted and greedy manner by favoring immediate rewards. The discount rate is typically below one because future rewards can be troublesome, as they have the ability to propagate large stochastic errors. Based on the small state space of the system in this study, the discount rate was set to 1 such that future rewards were not discounted. The *exploration rate* is used to adjust the probability of exploring the state space to reduce uncertainty versus exploiting current knowledge and taking the action expected to return the maximum reward based on the current state. In order to learn more about unknown expected rewards, the *exploration rate* was set to 0.8. While Q-learning does not provide any guarantees on regret, it is possible to tune performance via an exponentially decaying exploration rate and empirically improve the rate of convergence to the optimal policy.

While Q-learning will converge to the optimal policy, it does not provide any guarantees of optimality in balancing exploration versus exploitation. All tuning of the learning rate α, discount rate γ, and exploration rate ε are done manually. Manual tuning could be sidestepped through the use of more advanced learners, such as Contextual Multi-Armed Bandits.

2.3.2 Contextual Multi-Armed Bandits

Multi-Armed Bandit algorithms are often explained using the analogy of a room of slot machines that each return a random reward from a unique, unknown probability distribution. Given a limited set of resources (e.g., money, hardware life, researcher time), how and when should the resources be spent on exploration (learning the payout structure) versus exploitation (playing a specific machine)? MABs are used to determine the best sequence and choice of "pulls" on slot machine "arms" (or actions) in order to maximize cumulative rewards. Learning the optimal policy (sequence and choice of actions) that maximizes cumulative rewards requires probability theory.

Early works on MABs include Thompson sampling [34] and the Gittins index policy [35]. The goal in these works was to maximize the total expected reward by repeatedly selecting one of the many finite actions over a sequence of trials. In general, each action taken in a given state results in a reward based on the action's outcome (i.e., new state). MABs can be designed to maximize the expected total reward over a certain finite number of trials. Simply put, MABs balance exploration versus exploitation of the state-action pairs in order to maximize the total reward. MABs track the number of pulls per arm (i.e., how frequently each action has been taken) and estimate the expected reward of each arm. Once a computed threshold of exploration has been met, the MAB will begin exploiting its knowledge by choosing the arm with the highest expected reward.

Contextual Multi-Armed Bandits (C-MABs) are a particularly useful variant of MABs. C-MABs take advantage of information encoded in "context vectors" of raw and/or pre-processed observations that get associated with actions through experience. The C-MAB problem is well known and has been an area of comprehensive study [24], [26], [36], [37]. Most C-MAB algorithms assume access to a similarity metric, which relates the difference between the expected reward of an action between two neighboring (related) contexts and the distance between these contexts. These algorithms are proven to achieve sublinear-in-time regret with respect to a hypothetical benchmark that selects its actions based on the complete knowledge of the expected reward for each action given each context. Sublinear-in-time regret implies that the C-MAB algorithm is "average reward optimal," which implies that the frequency of the expected number of suboptimal actions converges to zero and, hence, the algorithm is on par with the hypothetical benchmark ("ground truth") in terms of the expected average reward.

The C-MAB algorithm used in this work is called Contextual Learning with Uniform Partition (CLUP), which is

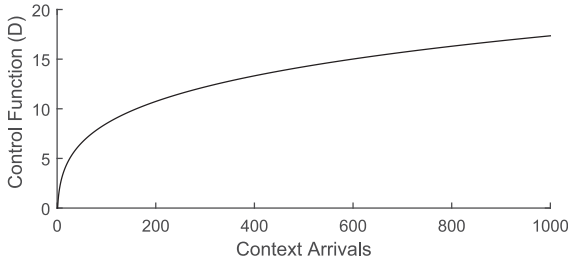

Fig. 2. The control function D versus the number of context arrivals $N_{f,p}$ for an assumed similarity between partitions of 0.25. If the number of context arrivals for all actions of a given state exceed the control function, the action with the highest expected reward is taken (i.e., exploitation is chosen over exploration).

taken from [27] and adapted for a single agent/learner setting. CLUP can be optimized to maximize the cumulative rewards given a specific number of trials. Given the inherent cost of collecting haptic data with robotic systems and an indeterminate number of trials, the maximization of cumulative rewards seemed appropriate. The C-MAB algorithm allows us to control how we learn and understand how well we perform over a finite time period. This is unlike Q-learning, for which there are no bounds on performance for a finite period of time.

The parameters of CLUP are defined as follows. Let each arm f be an element from the set of arms $\mathcal{F}$ available to the learner. CLUP partitions the context set into a finite number of sets, where each set is composed of similar contexts. For simplicity, it is assumed that each set in the partition is identical in size and shape (e.g., a hypercube with a specific edge length). This is called uniform partitioning since, given a random sample of tactile data, the system is equally likely to be in any of the partitions in a partition set $\mathcal{P}$.

Given uniform partitioning, let $p(t)$ be a partition in a partition set $\mathcal{P}$ to which the context $x(t)$ belongs. Let $N_{f,p}(t)$ be the number of "context arrivals" (i.e., how many times a context has been visited) for a given arm and partition. In this work, there were five arms (Fig. 4b) and three partitions (Fig. 4a), as will be described in Section 2.4.

For each discrete time step $= 1, 2, \ldots, T$, the learner observes a context $x(t)$ and then chooses an arm $f \in \mathcal{F}$ given $x(t)$. The choice of arm f depends on a parameter $D(t)$, which is a deterministic, increasing function of t called the control function Eq. (1). The parameter z defines the similarity of the partitions and affects the growth of the control parameter $D(t)$. In this work, z was empirically set to 0.086, with an assumed similarity between partitions of 0.25 for a 5-dimensional action space (Fig. 2). The reader is referred to [27] for a rigorous proof of CLUP, z, and the control function $D(t)$.

$$D(t) = t^z \log t \tag{1}$$

The number of context arrivals is compared to the control function in order to determine the set of under-explored arms $\mathcal{F}^{ue}(t)$ Eq. (2).

$$\mathcal{F}^{ue} := \left\{ f \in \mathcal{F} : N_{f,p}(t) \leq D(t) \right\} \tag{2}$$

If the set $\mathcal{F}^{ue}$ is non-empty, the learner selects an arm from $\mathcal{F}^{ue}$ at random in order to explore. When the set $\mathcal{F}^{ue}$ is empty, the learner enters the exploitation phase and selects the arm $a(t)$ with the highest mean estimated reward $\bar{r}_{f,p}(t)$ Eq. (3).

$$a(t) \in \underset{f}{\operatorname{argmax}}\ \bar{r}_{f,p}(t) \tag{3}$$

$$\bar{r}_{f,p} = \frac{\bar{r}_{f,p}\, N_{f,p} + r}{N_{f,p} + 1} \tag{4}$$

The mean estimated reward is updated for each arm as context arrivals occur and a reward r is received Eq. (4), (Fig. 4a). The pseudocode for CLUP for a single agent learner is shown in Fig. 3.

CLUP-variant of C-MAB learner adapted for a single agent:

Input: $D(t)$, T
Initialize sets: Create partition set $\mathcal{P}$ of contexts
Initialize counter: $N_{f,p} = 0,\ \forall f \in \mathcal{F}, \rho \in \mathcal{P}$
Initialize estimates: $\bar{r}_{f,p} = 0,\ \forall f \in \mathcal{F}, \rho \in \mathcal{P}$
While $t \geq 1$ **do**
  Find the partition $p(t)$ from set $\mathcal{P}$ to which context $x(t)$ belongs
  Let $p = p(t)$
  Compute set of underexplored arms $\mathcal{F}^{ue}(t)$
  if $\mathcal{F}^{ue}(t) \neq 0$ **then**
    Select arm $a(t)$ randomly from $\mathcal{F}^{ue}(t)$
  else
    Select arm $a(t)$ randomly from $\underset{f}{\operatorname{argmax}}\ \bar{r}_{f,p}(t)$
  end
  $\bar{r}_{f,p} = \frac{\bar{r}_{f,p} N_{f,p} + r}{N_{f,p} + 1}$
  $N_{f,p}++$
  $t = t + 1$
end while

Fig. 3. Pseudocode for the Adapted CLUP algorithm for a single agent C-MAB learner [27].

2.4 State Space, Action Space, and Rewards

For the haptics-intensive task of closing a ziplock bag, we defined a state space based on the spatial relationship between the artificial fingerpad and the zipper (contour to be followed). Based on findings from a pilot study that used Q-learning only [38], the state space was partitioned into "high," "center," and "low" states that described the location of the zipper along the length of the fingerpad (Fig. 4a). Although coarsely defined, the limited size of

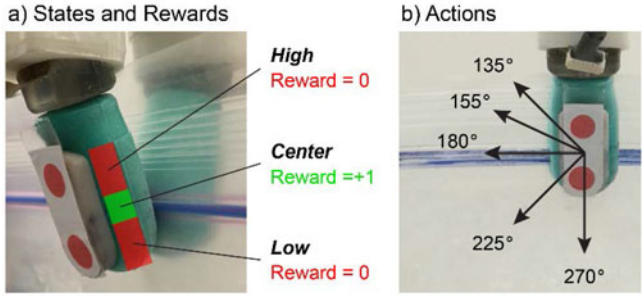

Fig. 4. a) The state space was defined according to spatial relationships between the fingerpad and zipper. The reward is shown for each state. b) The action space consisted of 0.75 cm trajectories in five directions.

the state space helps to keep the exploration process tractable.

The action space consisted of 0.75 cm trajectories in a 135, 155, 180, 225, or 270 degree direction from the current fingertip location (Fig. 4b). Fingertip movements of 0.75 cm were used to ensure that the elastic, silicone fingerpad would slip and slide along the zipper under kinetic friction conditions rather than simply stretching under static friction conditions. All trajectories were executed with a trapezoidal velocity profile with a maximum speed of 0.5 cm/s.

Based on the pilot study [38], the action space was designed to have trajectories that were not equally spaced with respect to the proximal and distal directions (Fig. 4b). During the pilot study, it was observed that the shape of the round fingertip led to large normal forces for distally directed actions; these movements tended to squeeze the zipper out of the robot's grasp. In this work, the action space was defined with more aggressive angles of attack in the distal directions.

The reward structure was designed to provide positive reinforcement for desired fingerpad-contour relationships, namely that the contour should ideally remain in the center of the fingerpad for successful closure of the ziplock bag. Thus, the "center" state received a $+1$ reward while the "high" and "low" states each received a reward of 0 (Fig. 4a). The state space, action space, and reward structure were identical for the Q- and C-MAB learners.

2.5 Context Vector for the C-MAB Learner

In defining a context vector for the C-MAB learner, tactile sensor data were trained with a classifier. Based on prior work that showed that geometric features could be encoded in artificial fingerpad deformation [39], [40], we considered only impedance data from the electrode array on the core of BioTac tactile sensor.

The classifier takes as input normalized, individual electrode impedance data and returns one of three discrete states describing the spatial relationship between the fingerpad and zipper (Fig. 4a). It is assumed that, due to the random exploration along the contour, the classifier will return a state that is equally likely to be in one of the three partitions, as described in Section 2.3.2.

2.6 Training of the State Classifier

2.6.1 Automated Labeling of States

In order to train a classifier to estimate the state of the zipper along the length of the fingerpad, all tactile sensor data needed to be labeled as "high," "center," or "low". In the pilot study, states were manually labeled according to marks drawn directly on the deformable sensor skin [38]. This approach was negatively affected by skin strain and was not representative of zipper position relative to the core of the tactile sensor that contains the spatial array of electrodes.

In this work, tactile sensor data were autonomously labeled by a computer vision system developed using OpenCV [41]. A Python node processed camera images sent from a custom iPhone 5s iOS application. High contrast colors and strict color thresholding were used to maximize the accuracy of the automated labeling. The BioTac fingernail was used to track the position of the sensor's rigid core relative to the zipper. To locate the fingernail visually, red circles were placed over each fingernail screw (Fig. 5). K-means clustering was used to identify the center of each red circle (green dots in Fig. 5). The inner surface of the zipper was colored blue for visual identification and robustness to wear over thousands of trials. A best fit line was applied to the zipper contour (yellow line in Fig. 5). The intersection of the center line of the fingernail (line connecting the green dots) and the zipper (yellow line) was marked by a blue dot.

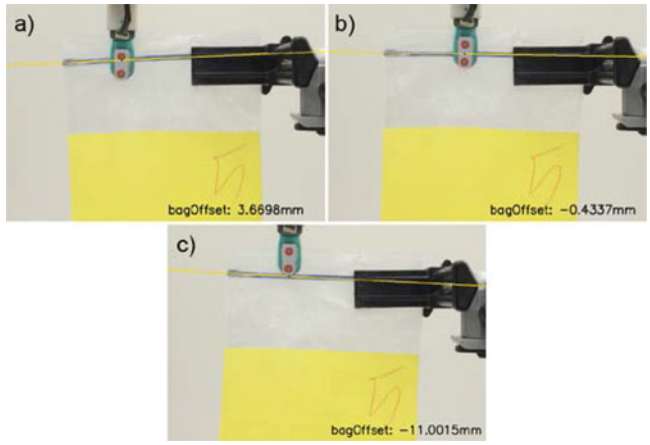

Fig. 5. Using image processing techniques, tactile sensor data were autonomously labeled according to a) "high," b) "center," and c) "low" states that described the location of the zipper along the length of the fingerpad.

The output of the image processing pipeline was a measurement of the offset (shortest distance) between the center line of the fingernail and the zipper along the center line of the finger. This offset value was taken as the difference between the location of the zipper in the grasp (blue dot) and the center of the fingernail (halfway between the two green dots).

Tactile sensor data were labeled according to the zipper offset value according to predefined thresholds. The zipper was considered to be "high" for an offset value greater than 0 mm (defined at the center of the two red circles) (Fig. 5a). The label "center" was applied for an offset value less than or equal to 0 mm and greater than -2.5 mm (Fig. 5b). The label "low" was used for all other offset values less than or equal to -2.5 mm (Fig. 5c). Tactile sensor data were excluded from model training if the fingertip lost contact with the zipper.

2.6.2 Input Vector of Tactile Sensor Data

Each trial was initialized by grasping the ziplock bag and performing one 180 degree fingertip movement (Fig. 4b), which initializes the strain in the BioTac skin. A complete trial was defined as the set of actions from an initialized grasp to successful zipper closure or loss of grasp of the zipper. In the pilot study, sensor data from a no-contact condition was used for each trial-specific baseline reference prior to grasp initialization [38]. However, this approach made the classifier unnecessarily susceptible to stochasticity inherent in the tactile sensor data from each multi-action trial. Furthermore, many tactile sensors, including the BioTac used in this experiment, are susceptible to sensor drift. Despite using the sensor per the vendor's instructions, slight variations in internal fluid volume, external temperature, and wear of the sensor skin can cause drift in BioTac

readings, which can also affect day-to-day classification of trained algorithms [42].

In this work, in order to regularize classifier inputs, we focused on changes in sensor data between adjacent fingertip actions. Each action corresponded with approximately 1.5 seconds of data. Immediately prior to each action, 100 ms of impedance data (batches of 10 samples) for each of the BioTac's 19 electrodes were collected for use as an action-specific baseline. After each action, the baseline data were subtracted from the most recent 100 ms of electrode impedance data. The L2-Norm of each of these action-generated differences in electrode impedance values were used to build a 19 x 1 input vector for the classifier. Thus, the classifier was trained on changes in tactile sensor data relative to the prior state.

2.6.3 State Classification via Deep Neural Nets

A Deep Neural Net (DNN) was used to learn a mapping between the tactile sensor data (inputs) and state labels (outputs). DNNs consist of multiple layers in which each layer is comprised of a set of nodes. By applying rectified linear units (ReLUs) on the inputs of each hidden layer, DNNs are able to fit nonlinear data, thereby allowing for robust classification of nonlinear finger-object interactions. The DNN in this work was comprised of three hidden layers with 512 nodes per hidden layer.

Classifier training and online classification were performed using TensorFlow [28]. Training of the DNN was accomplished through stochastic gradient descent (SGD) with an exponentially decaying learning rate, regularization, and dropout. SGD is a method of adjusting the weights and biases of each layer of the DNN to minimize a loss function. Instead of using the full set of training data for every update to the model, SGD randomly selects a small subset of data, which significantly reduces training time. Even though each training step does not use the full set of training data, by stochastically varying the training set the full set is represented. L2 Regularization [43], which adds terms to the loss function in order to penalize large weights, was used to reduce overfitting of the model to training data. Dropout, a technique for stochastically turning weights on and off during training [44], was used to prevent overfitting and distribute activation weights across the nodes of the DNN. By randomly turning a set percentage of the weights on and off during training, the model no longer relies on a single activation node for classification. Randomly removing weights ensures that the activation of nodes are distributed throughout the network. In this work, a dropout rate of 50 percent was used.

3 Results

3.1 Classifier Training

The multimodal tactile sensor data for a representative action are shown in Fig. 6. The input to the DNN state classifier was a 19 x 1 vector of normalized tactile sensor electrode impedance values (Fig. 6a). This vector represents the change in skin deformation due to the action that has just been taken. The time histories of the electrode impedance, internal fluid pressure, and vibration data are shown in Fig. 6b, 6c, 6d for completeness.

The DNN classifier was trained with 7,200 data samples (90 percent of 8,000 samples collected). The trained model

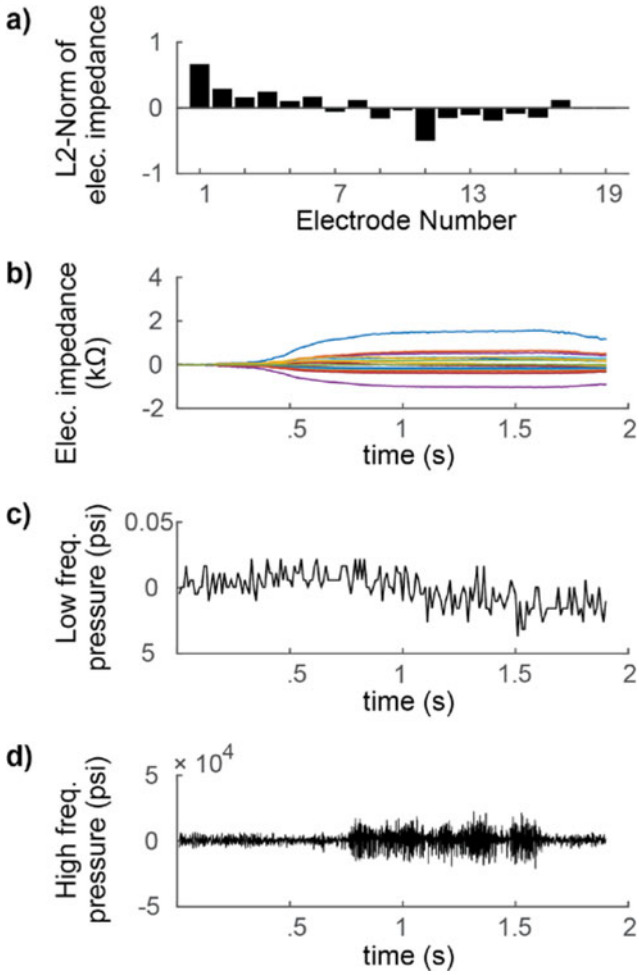

Fig. 6. Representative multimodal tactile data for a single action. a) DNN input vector values are shown as a histogram of the 19 normalized electrode values. Time histories for tactile sensor data for a complete action: b) electrode impedance (100 Hz), c) low frequency fluid pressure (100 Hz), and d) high frequency fluid pressure (2200 Hz).

was validated with the remaining 800 samples. The model was trained after approximately 15,000 iterations and an average training time of less than 10 minutes (2013 MacBook Pro, TensorFlow). The DNN classifier performed with accuracies of 89 and 86 percent for the training and validation datasets, respectively.

3.2 Learned Policies (Expected Reward Structures)

Two different reinforcement learning algorithms were used to determine the expected reward for each of the 15 state-action pairs (3 states x 5 actions). Both Q-learning and C-MAB were run independently for just over 750 actions in order to compare each algorithm's expected reward structure. Fig. 7 shows the expected rewards after 757 actions have been taken with Q-learning and 758 actions have been taken with C-MABs. All reinforcement learning was performed online so that the two learning algorithms could be properly compared, as each learner determines actions sequentially.

Of the three states, only one state ("high") was associated with a state-action pair ("high" state, 155 degree action; see orange pie slices in Fig. 7) that would yield the maximum expected reward for both learning algorithms. That is, both the Q- and C-MAB learners arrived at a policy in which a 155 degree action should be taken to maximize rewards if a

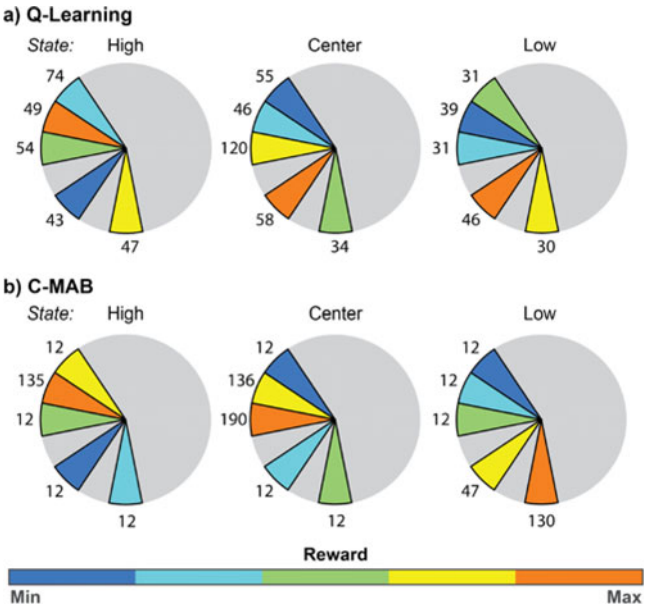

Fig. 7. Expected rewards for each state-action pair for the a) Q-learning and b) C-MAB learning algorithms after 757 and 758 consecutive actions, respectively. Each pie chart represents a state (Fig. 4a) while pie slices correspond to actions (Fig. 4b). As indicated by the scale at the bottom, colors correspond to the magnitude of the expected reward for each state-action pair. Numbers next to each slice indicate the number of context arrivals.

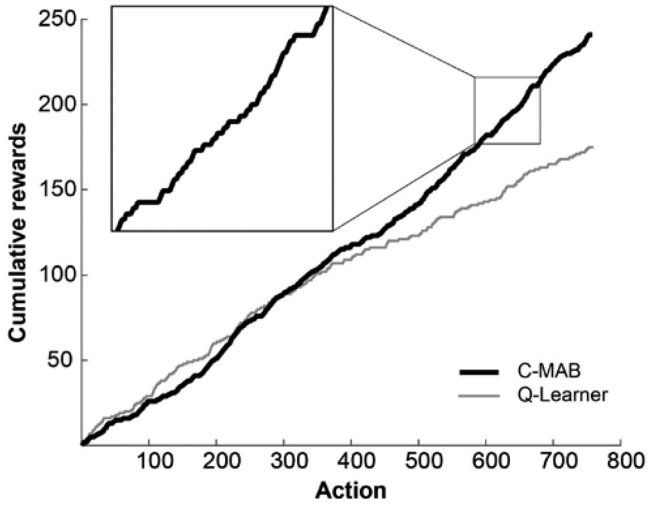

Fig. 8. Cumulative rewards for over 750 consecutive actions per learner. C-MAB (thick line) outperforms Q-learning (thin line) by optimizing exploration versus exploitation. The inset highlights two flat regions during which the control function (Fig. 2) dictated additional exploration of the expected rewards and the C-MAB was forced to select actions that did not yield rewards.

"high" state were estimated. The other two states ("center" and "low") resulted in similar, but not identical, action policies for maximizing expected rewards. For instance, for the "low" state, the Q-learner policy would dictate a 225 degree action while the C-MAB learner policy would use a 270 degree action to maximize expected rewards.

3.3 Cumulative Reward Performance

To compare the Q-learning and C-MAB reinforcement learning algorithms, cumulative rewards were plotted for all consecutive actions during the supervised learning process (Fig. 8). Cumulative rewards are the sum of all rewards received over all actions ever taken. Both reinforcement learning algorithms have the identical state and action spaces available to them. What differentiates each learner is how actions are chosen during exploration and how expected rewards are updated.

Q-learning initially outperforms C-MAB learning until approximately 300 actions have been taken. This is because, based on the control function D, the C-MAB learner is purely exploring in order to gather information about the distribution of expected rewards across the state-action pairs. However, once the C-MAB learner has sufficient information about expected rewards, it begins to exploit the policy it is learning. The trajectories for the cumulative rewards quickly diverge between the two learners, with the C-MAB learner significantly outperforming the Q-learner after only 750 actions (Fig. 8). The C-MAB learner continues to earn rewards at a higher rate than the Q-learner as time passes (Table 1). The Q-learner reward rate does not improve due to the fact that it is unable to understand or balance the exploration of the context space before attempting to exploit what is known about expected rewards. This reflects an underdeveloped Q-learner policy for the finite time period shown.

There are also short segments in Fig. 8 where the C-MAB cumulative reward function appears to be flat. These flat segments correspond to periods when the time-dependent control parameter D increases to the next integer value of context arrivals required for the exploration of the state-action pairs [27].

3.4 Task Performance

An empty, industrial ziplock bag (Fig. 5) with a closed zipper thickness of 1.7 mm was used for classifier training and reinforcement learning due to the large number of data collection trials required. However, the learned C-MAB policy was tested on a thinner, more flexible test bag with a closed zipper thickness of 1.4 mm (Fig. 9). All testing was conducted without computer vision, and with tactile and proprioceptive feedback only. Rewards were still received by the robot according to the DNN state classifier estimates resulting from the tactile sensor data.

The C-MAB policy resulted in successful bag closure in all 10 trials with the empty, novel test bag. The average time required to close the test bag was approximately 2.5 min. During each trial, 80 percent of the time was spent on motion planning. The elapsed time fluctuated by a small amount from trial to trial due to the efficiency of the actions chosen online.

4 Discussion

4.1 Challenges of Real-Time Haptic Perception during Task Execution

In order to complete a goal-driven manipulation task, a robotic system has to perceive the current state and decide

TABLE 1
Sum of Rewards for Each Quarter of Learning

Quarter	1st	2nd	3rd	4th
Q-Learning	61	49	33	37
C-MAB	48	65	58	70

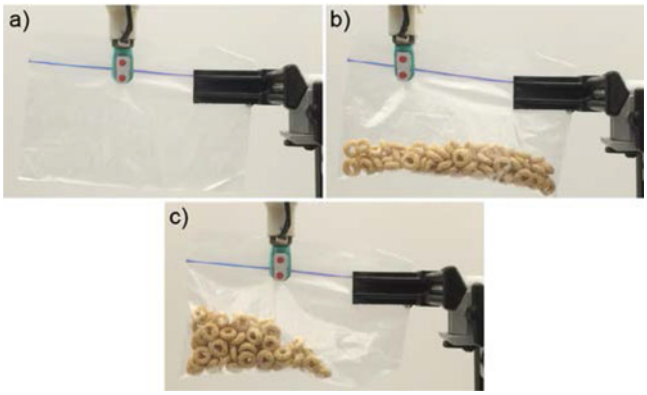

Fig. 9. The learned C-MAB policy was tested on a thinner, more flexible ziplock bag than that used during classifier training or policy learning. The policy was tested with the bag in three ways: a) empty, and containing 10 g of cereal b) evenly distributed and c) toward the end of the bag.

on the best action to take based on the task goal and prior knowledge. Corrective actions can be taken more frequently by perceiving the state of the system in real-time or by finely discretizing large movements with extended timescales, such as exploratory procedures (EPs) [45]. EPs are used to relate known fingertip trajectories to unknown object properties, such as texture and shape.

However, many tasks preclude the use of EPs as the sole method for planning. For instance, there may not always be time to perform a large suite of exploratory movements. In the case of closing a ziplock bag with a two-digit grasp, it would be impractical to perform a single-digit EP to estimate the orientation of the zipper, in the style of [39].

A few challenges arise from real-time haptic perception during task execution, especially with compliant tactile sensors. First, the initial movement of a sensorized fingertip may begin with the sensor in free space and not in contact with any object. When undeformed, the strain in a compliant fingerpad is mechanically reset to baseline conditions. Once in contact with an object, however, the strain in the fingerpad at the current state will affect the strain at the next state when a new action is taken and the finger remains in contact with the object [46]. As a result, mapping measured changes in fingerpad strain to known finger-object relationships is not a straightforward task. Second, drawing inferences from short fingertip trajectories (for frequent state updates) makes it difficult to distinguish causal changes in tactile data from those resulting from noise and/or transient events.

4.2 Importance of Robust State Classification

The robustness of a state classification model to variations in the robot (e.g., sensor wear) and the environment is critical to the success of any reinforcement learning algorithm. Maximally rewarding actions cannot be identified if the state itself is uncertain. In this work, the DNN classifier's ability to correctly classify the state of the zipper within the pinch grasp will minimize the stochasticity of the rewards during reinforcement learning. In turn, this will allow both Q- and C-MAB learners to more efficiently converge to an optimal policy.

State classification is challenging when using deformable tactile sensors. Compliant tactile sensors, such as the BioTac, feature advantages such as (i) increased contact area between fingertips and objects, and (ii) the ability to encode complex finger-object interactions across an entire fingerpad, including regions of the fingertip that are not in contact with the object [39], [40]. Such unique advantages make the use of deformable sensors worth the extra machine learning effort to achieve robust state classification.

Improving accuracy for compliant tactile sensors continues to be a challenge [42]. The tactile sensor electrode signals resulting from deformation of the artificial fingerpad were highly dependent on the strain induced by the previous action. Thus, we chose to use an action-specific baseline reference for the sensor electrode data such that the DNN classifier was trained on changes in tactile sensor data relative to the prior state.

All classification accuracies were determined by comparing the DNN classifier estimates with the results of the computer vision labeling process. The trained DNN classifier had an initial accuracy of 86 percent on out-of-sample data not used for training. This DNN model was used to classify states for both reinforcement learning algorithms. Despite the fact that the DNN classifier was trained on data collected 84 days prior to the reinforcement learning experiments, the classification accuracy during learning with new data was 71 percent for Q-learning and 74 percent for C-MAB learning. The state classification accuracy for both Q-learning and CLUP should be identical, as the trained classifier should degrade uniformly due to wear of the sensor and systemic changes in the environment. An average loss in classifier accuracy of 13.5 percent over an 84 day period suggests that the DNN classifier was fairly robust over time.

Future work could explore the robustness of DNN classification accuracy to changes in action variables (e.g., fingertip speed, movement distance) and other sensing modalities. However, it is unlikely that perturbations of fingertip velocity would affect the current classifier, as the classifier relies on action-specific baselines. Instead, providing additional sensing modalities to the classifier, such as high frequency vibration, could improve classifier performance. Additional classifiers could be designed to output local contour orientation with respect to the fingertip. Such information would provide additional context and could improve robustness to variations in contour stiffness, width, and shape.

4.3 Advantages of Contextual Multi-Armed Bandit Reinforcement Learning

When learning, rewards are stochastically received and the learner must appropriately update the expected reward based on the difference between an observed reward and the current expected reward. While Q-learning will converge to the optimal expected reward, this will only happen as time goes to infinity. Q-learning also does not provide any finite guarantees on optimality. The Q-learner parameters could likely be manually tuned to improve performance beyond that shown in Fig. 8. However, such laborious manual parameter tuning is avoided altogether by using the more advanced C-MAB learner. Importantly, C-MAB learning is characterized by bounded regret and enables an optimal search of the state-action space in order to maximize cumulative rewards given the number of actions already taken.

C-MABs employ a controlled exploration of the state-action pairs and are able to hone in on the actions that it learns to have the maximum expected reward [27]. By selecting actions purposefully in order to reduce uncertainty, a C-MAB learner is able to increase its confidence in the expected reward structure. For example, for the "low state," the C-MAB learner has spent the majority of its exploration efforts on the 225 and 270 degree actions (see context arrival counts of 130 and 47 adjacent to the orange and yellow pie slices, respectively, in Fig. 7b). In contrast, the Q-learner has spent more time and effort on randomly exploring all five actions (see similar context arrival counts for the "low" state in Fig. 7a); this results in a less efficient exploration of the state-action space. The relative efficiency of the C-MAB learner derives from its average reward optimality, which results in the selection of actions that coincide most of the time with the hypothetical benchmark. Since the reward structure is stochastic, the C-MAB learner alternates between beliefs that either the 225 or 270 degree action could yield the maximum expected reward. Due to the design of the CLUP-variant of the C-MAB learner, it focuses its exploration efforts to further differentiate between the 225 and 270 degree actions.

Efficient exploration during learning helps to reduce uncertainty for actions suspected of returning larger rewards while not being wasteful of resources (e.g., hardware life, researcher time). Based on the context arrival counts indicated next to each pie slice, it is apparent that the C-MAB learner exhibits a more controlled exploratory behavior than a Q-learner (Fig. 7). If a state-action pair is not believed to return high rewards, it will only be explored in order to reduce uncertainty as dictated by the control parameter $D(t)$ in the C-MAB learner algorithm. This control parameter continues to grow with time (number of actions), but its growth rate slows. The ever-increasing control function D (Fig. 2) guarantees that the robot never stops exploring the state-action pairs, but that it does so less frequently as time passes and both experience and confidence increase. In contrast, Q-learning randomly chooses arms during exploration and does not consider the uncertainty of the expected rewards to refine its exploration efforts.

4.3.1 Importance of Reinforcement Learning for Hard-to-code Behaviors

To demonstrate that a learned, closed-loop policy was necessary to accomplish the task of ziplock bag closure, naïve fingertip trajectories were tested. For each test, the zipper was placed at a known angle and assumed to follow a straight line. Upon grasp initialization (identical to that during reinforcement learning, the fingertips were moved in a straight line without stopping and without the use of closed-loop haptic perception or decision-making. All combinations of three trajectories and three zipper orientations (5, 0, and -5 degree, relative to the horizontal, for both fingertip trajectories and actual zipper orientations) were tested twice. With the exception of one trial (zipper and fingertip trajectory both at 0 degree), the robot failed to track the zipper and seal the bag during these 18 test trials. The single successful trial occurred when the preplanned trajectory and orientation of the zipper were, apparently, initialized perfectly.

4.4 Generalizability of the Learned C-MAB Policy

4.4.1 Novel Ziplock Bag Scenarios

Given that the system should perform best on the empty industrial ziplock bag used during training, we tested the ability of the robot to generalize its learned C-MAB policy to novel deformable contours. As reported in Section 3.4, the industrial bag used for training and reinforcement learning (Fig. 5) was exchanged for a thinner, more flexible bag for testing (Fig. 9). While the industrial bag lends itself to thousands of classifier training and policy learning trials, the test bag was the more familiar sandwich bag. The test bag was successfully closed 10 out of 10 times when empty (Fig. 9a).

The thin test bag was also filled with different distributions of 10 g of cereal to test the C-MAB policy on a weighted condition that the robot had never before seen. In one configuration, the cereal and its weight were evenly distributed along the bottom of the bag (Fig. 9b). In another configuration, the weight was forced to the end of the bag (Fig. 9c), which caused the zipper to slope downward toward the bulk of the cereal at the start of each trial. The DNN classifier was able to perceive this downward slope, identify the zipper in the "low" state, and take corrective actions to correctly track the zipper and close the bag. The robot was able to aggressively correct for the undesirable "low" state due to the C-MAB policy that dictated a 180 degree action for such situations (see the orange pie slice in the pie chart corresponding to the "low" state in Fig. 7b).

The C-MAB policy resulted in successful closure of a weighted bag 5 out of 7 times (71 percent success rate). The evenly distributed cereal (Fig. 9b) resulted in a zipper shape that was most similar to that of the empty bag (Fig. 9a) and had a high success rate. When the cereal was distributed toward the end of the bag, the success rate was only 60 percent (3 out of 5 times). As the robot's fingertips traveled closer to the weighted end of the bag, a single misclassification of the state could lead to an inappropriate action from which the robot was unable to recover. Failure could have resulted because the test conditions (e.g., contour flexibility, weight distribution) were challenging, the robot did not have time or travel distance to fix its error, the robot was instructed to abort the trial once grasp of the zipper was lost, or the DNN classifier was not trained on steep angles of the zipper. Incorporating a memory of sequential actions might enable the learner to make use of the overall shape of the contour. However, without visual feedback, there is no way to confirm the accuracy of the estimated shape of the contour.

4.4.2 Wire and Rope

The learned C-MAB policy was also tested on novel, deformable contours that were not zippers: thick electrical wire (3.5 mm diam.), thin electrical wire (1.5 mm diam.), and nylon rope (4 mm diam.) (Fig. 10). The DNN classifier trained on the industrial ziplock bag was able to correctly classify the state of the wire or rope within the pinch grasp. As such, the C-MAB policy appeared to select appropriate corrective actions. However, the C-MAB policy for the flat zipper did not generalize to the round wires and rope. The robot was unable to maintain the contours in the desired "center" state in order to track the wires or rope for any significant distance. The robot was only able to track the wire and rope for

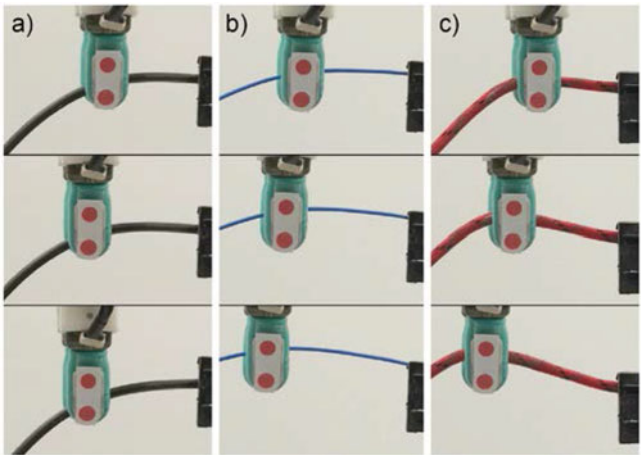

Fig. 10. The learned C-MAB policy was tested on three novel, deformable contours that were not zippers: a) thick electrical wire (3.5 mm diam.), b) thin electrical wire (1.5mm diam.), and c) nylon rope (4 mm diam). Each column shows sequential actions during an individual trial.

approximately three sequential actions before loss of contact. The failure to follow the wire and rope contours may be due to a combination of the round contour shapes, low friction (at least for the wires), and the steep angle of the contour for which the C-MAB policy was not learned.

It should be noted that grasp pressure was held constant for all actions, trials, and contours during classifier training, policy learning, and testing. The generalizability of the policy could be improved by increasing the action space to include the online modulation of grasp pressure, and by exposing the C-MAB learner to a larger variety of contours during the online learning process. Even with the addition of new contexts, the C-MABs are well suited to deal with continuous and uncountable contexts [47]. The C-MAB policy in this work could be used to seed the beliefs of the next-generation policy, thereby reducing the time and cost of learning a more robust policy.

5 Conclusion

In this work, a robot learned how to perform a haptics-intensive manipulation task through a resource-conscious reinforcement learning technique called Contextual Multi-Armed Bandits that improved the speed of learning and increased the cumulative rewards received over a finite period. The model-free approach employed a DNN classifier to learn the nonlinear mapping between fingerpad deformation and the relative location of a deformable contour along the length of the fingerpad. A policy was learned through trial and error such that exploration of the state-action space was performed purposefully to reduce uncertainty in the expected reward structure. Such an efficient use of exploratory effort is particularly important for the development of new complex skills that require repeated interactions between the robot and the environment, as such interactions are costly in terms of both time and wear.

The task of closing a ziplock bag introduced a few interesting twists on the standard contour-following task. Since the contour was deformable and visually occluded by the robot's fingertips, the robot had to learn what it felt like to interact with the ziplock bag. The robot learned the statistics of the consequences of its actions through hands-on experience. We used computer vision to automate the reward process during supervised learning, but the contour-following task could be accomplished through the use of tactile and proprioceptive feedback alone once the policy was learned.

While minimizing task execution time was not the primary focus of this work, it is important for systems in the real world. In this work, significant time delays resulted from the 3D motion planning performed for the 7 degree-of-freedom robot arm. A simpler inverse kinematic solver could have been used to reduce planning time for this work. However, 3D motion planning was implemented for scalability to more complex tasks in the future. Calculations for the DNN classifier and C-MAB learner actually required very little processor time. Parallelizing the motion planning process and using graphical processing units could significantly reduce the total task execution time.

For reinforcement learning, it was important to define the state and action spaces in a way that allows the robot learner to find interesting solutions that might not be apparent to the human designer. Non-intuitive solutions can be found by exploring the vast state-action space and minimizing uncertainty in the distribution of expected rewards. It was also important to limit the dimensionality of the state-action space in order to make the exploration problem tractable, especially given the high cost of collecting tactile sensor data. Tactile sensor data are difficult to simulate, time consuming to collect, and cause wear of the entire hardware system during collection.

The DNN state classifier was robust to changes in the sensor over time and generalized to novel contours such as thin, flexible zippers, wires, and rope. While the learned C-MAB policy was able to generalize to a novel test bag and different weight distributions of bag contents, the policy did not work for low friction, round contours such as the wires and rope. Potential future improvements include the expansion of the action space to include online modulation of grasp pressure, adjustments to fingertip travel length or velocity based on confidence, and out-of-plane movements and rotations of the fingertips.

Additional reinforcement learning techniques could also be considered in order to improve the robustness of the policy. In this work, uniform partitioning was used, but partitions can also be adapted over time. For instance, the partition set $\mathcal{P}$ can be refined over the regions of the context space with frequent context arrivals. This will result in a partition set composed of more similar contexts, for which rewards can be estimated more accurately. Effectively, such adaptive algorithms can be used to zoom in on high occupancy regions of the state-action space (i.e., those pie slices with high context arrival counts) in order to refine the action space and reduce the uncertainty of expected rewards while keeping the exploration problem tractable [27].

Acknowledgments

The authors wish to thank Peter Aspinall for assistance with the construction of the robot testbed. This work was supported in part by National Science Foundation Awards #1461547, #1463960, and #1533983, and the Office of Naval Research Award #N00014-16-1-2468.

REFERENCES

[1] N. Chen, H. Zhang, and R. Rink, "Edge tracking using tactile servo," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. "Hum. Robot Interact. Coop. Robots"*, 1995, vol. 2, pp. 84–89.

[2] D. Nakhaeinia, P. Payeur, and R. Laganiere, "Adaptive robotic contour following from low accuracy RGB-D surface profiling and visual servoing," in *Proc. Can. Conf. Comput. Robot Vis.*, 2014, pp. 48–55.

[3] H. Koch, A. Konig, A. Weigl-Seitz, K. Kleinmann, and J. Suchy, "Multisensor contour following with vision, force, and acceleration sensors for an industrial robot," *IEEE Trans. Instrum. Meas.*, vol. 62, no. 2, pp. 268–280, Feb. 2013.

[4] M. Irani, B. Rousso, and S. Peleg, "Computing occluding and transparent motions," *Int. J. Comput. Vis.*, vol. 12, no. 1, pp. 5–16, Feb. 1994.

[5] U. Martinez-Hernandez, G. Metta, T. J. Dodd, T. J. Prescott, L. Natale, and N. F. Lepora, "Active contour following to explore object shape with robot touch," in *Proc. World Haptics Conf.*, 2013, pp. 341–346.

[6] M. Klingensmith, S. S. Sirinivasa, and M. Kaess, "Articulated robot motion for simultaneous localization and mapping," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 1156–1163, Jul. 2016.

[7] G. A. Pratt and M. M. Williamson, "Series elastic actuators," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. "Hum. Robot Interact. Coop. Robots"*, 1995, vol. 1, pp. 399–406.

[8] T. Bhattacharjee, A. A. Shenoi, D. Park, J. M. Rehg, and C. C. Kemp, "Combining tactile sensing and vision for rapid haptic mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Rob Syst.*, 2015, pp. 1200–1207.

[9] Z. Lazher, B. Belhassen-Chedli, L. Sabourin, and M. Youcef, "Modeling and analysis of 3D deformable object grasping," in *Proc. 23rd Int. Conf. Robot. Alpe-Adria-Danube Reg.*, 2014, pp. 1–8.

[10] M. Salzmann, J. Pilet, S. Ilic, and P. Fua, "Surface deformation models for nonrigid 3D shape recovery," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 8, pp. 1481–1487, Aug. 2007.

[11] A. Kapusta, W. Yu, T. Bhattacharjee, C. K. Liu, G. Turk, and C. C. Kemp, "Data-driven haptic perception for robot-assisted dressing," in *Proc. 25th IEEE Int. Symp. Robot Hum. Interact. Commun.*, 2016, pp. 451–458.

[12] J. Schulman, A. Lee, J. Ho, and P. Abbeel, "Tracking deformable objects with point clouds," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 1130–1137.

[13] D. Berenson, "Manipulation of deformable objects without modeling and simulating deformation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 4525–4532.

[14] T. Bretl and Z. McCarthy, "Mechanics and quasi-static manipulation of planar elastic kinematic chains," *IEEE Trans. Robot.*, vol. 29, no. 1, pp. 1–14, Feb. 2013.

[15] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 2308–2315.

[16] A. J. Shah and J. A. Shah, "Towards manipulation planning for multiple interlinked deformable linear objects," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 3908–3915.

[17] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 5026–5033.

[18] Robot Locomotion Group at the MIT Computer Science and Artificial Intelligence Lab, "DRAKE: A planning, control, and analysis toolbox for nonlinear dynamical systems." (2016). [Online]. Available: http://drake.mit.edu/

[19] N. Wettels, V. J. Santos, R. S. Johansson, and G. E. Loeb, "Biomimetic Tactile Sensor Array," *Adv. Robot.*, vol. 22, pp. 829–849, Aug. 2008.

[20] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," in *Proc IEEE Int. Conf Robot. Autom.*, 2011, pp. 3828–3834.

[21] J. Mahler, et al., "Dex-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a multi-armed bandit model with correlated rewards," presented at the *Int. Conf. Robot. Autom.*, Stockholm, Sweden, 2016.

[22] M. Laskey, et al., "Multi-armed bandit models for 2D grasp planning with uncertainty," in *Proc. IEEE Int. Conf. Autom. Sci. Eng.*, 2015, 2015, pp. 572–579.

[23] M. C. Koval, J. E. King, N. S. Pollard, and S. S. Srinivasa, "Robust trajectory selection for rearrangement planning as a multi-armed bandit problem," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 2678–2685.

[24] M. Dudik, et al., "Efficient optimal learning for contextual bandits," *arXiv preprint arXiv:1106.2369*, Jun. 2011.

[25] J. Langford and T. Zhang, "The Epoch-Greedy algorithm for multi-armed bandits with side information," in *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds. Red Hook, NY, USA: Curran Associates, Inc., 2008, pp. 817–824.

[26] A. Slivkins, "Contextual bandits with similarity information," *J. Mach. Learn. Res.*, vol. 15, pp. 2533–2568, 2014.

[27] C. Tekin and M. van der Schaar, "Distributed online learning via cooperative contextual bandits," *IEEE Trans. Signal Process.*, vol. 63, no. 14, pp. 3700–3714, Jul. 2015.

[28] M. Abadi, et al., "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv: 1603.04467*, Mar. 2016.

[29] M. Quigley, et al., "ROS: An open-source robot operating system," 2009. [Online]. Available: http://www.ros.org/

[30] I. A. Sucan and S. Chitta, "MoveIt!," Oct. 2015. [Online]. Available: http://moveit.ros.org

[31] "Open Motion Planning Library (OMPL)," Nov. 2015. [Online]. Available: http://ompl.kavrakilab.org/

[32] C. J. C. H. Watkins and P. Dayan, "Technical note: Q-Learning," *Mach. Learn.*, vol. 8, no. 3/4, pp. 279–292, May 1992.

[33] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Univ. Cambridge, Cambridge, U.K., 1989.

[34] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.

[35] J. Gittins, K. Glazebrook, and R. Weber, *Multi-Armed Bandit Allocation Indices*. Hoboken, NJ, USA: Wiley, 2011.

[36] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvari, "X-Armed Bandits," *J. Mach. Learn. Res.*, vol. 12, pp. 1655–1695, 2011.

[37] R. Kleinberg, A. Slivkins, and E. Upfal, "Multi-armed bandits in metric spaces," in *Proc. Fortieth Annu. ACM Symp. Theory Comput.*, May 2008, pp. 681–690.

[38] R. B. Hellman and V. J. Santos, "Haptic perception and decision-making for a functional contour-following task," in presented at the *IEEE Haptics Symp., Work-in-Prog. Paper*, Philadelphia, PA, USA, 2016.

[39] R. D. Ponce Wong, R. B. Hellman, and V. J. Santos, "Spatial asymmetry in tactile sensor skin deformation aids perception of edge orientation during haptic exploration," *IEEE Trans. Haptics*, vol. 7, no. 2, pp. 191–202, Apr.-Jun. 2014.

[40] R. D. Ponce Wong, R. B. Hellman, and V. J. Santos, "Haptic exploration of fingertip-sized geometric features using a multimodal tactile sensor," in *Proc. SPIE 9116, Next-Generation Robots and Systems*, vol. 9116, Jun. 2014, pp. 1–15, doi: 10.1117/12.2058238

[41] "Open Computer Vision (OpenCV) Library (SourceForge.net)," 2015. [Online]. Available: http://sourceforge.net/projects/opencvlibrary/, Accessed on: Oct. 14, 2011.

[42] J. C. T. Hui, A. E. Block, C. J. Taylor, and K. J. Kuchenbecker, "Robust tactile perception of artificial tumors using pairwise comparisons of sensor array readings," in *Proc. IEEE Haptics Symp.*, 2016, pp. 305–312.

[43] A. Ng, "Feature selection, L_1 versus L_2 regularization, and rotational invariance," in *Proc. Int. Conf. Mach. Learn.*, 2004, Art. no. 78.

[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[45] S. J. Lederman and R. L. Klatzky, "Hand movements: A window into haptic object recognition," *Cogn. Psychol.*, vol. 19, no. 3, pp. 342–368, Jul. 1987.

[46] V. Levesque and V. Hayward, "Experimental evidence of lateral skin strain during tactile exploration," in *Proc. Eurohaptics*, 2003, vol. 2003, pp. 261–275.

[47] C. Tekin and M. van der Schaar, "RELEAF: An algorithm for learning and exploiting relevance," *IEEE J. Sel. Top. Signal Process., special issue on signal processing and big data*, vol. 9, no. 4, pp. 716–727, 2015.

Randall B Hellman received the BS degree in mechanical engineering from Purdue University, and the MS and PhD degrees in mechanical engineering from Arizona State University, in 2008 and 2016, respectively. He is currently a postdoctoral researcher with the University of California, Los Angeles. His research interests include artificial haptic perception, decision-making, reinforcement learning, manipulation, robotic hands, tactile sensors, and haptic interfaces for sensory feedback. He is a member of the IEEE.

Cem Tekin (S'09-M'13) received the PhD degree in electrical engineering systems from the University of Michigan, Ann Arbor, in 2013. He is an assistant professor in Electrical and Electronics Engineering Department, Bilkent University, Turkey. From 2013 to 2015, he was a postdoctoral scholar with the University of California, Los Angeles. His research interests include big data stream mining, machine learning, data science, multi-armed bandit problems, and recommender systems. He is a member of the IEEE.

Mihaela van der Schaar is Chancellor's Professor of electrical engineering with the University of California, Los Angeles. She was a distinguished lecturer of the Communications Society from 2011 to 2012, the editor in chief of the *IEEE Transactions on Multimedia* from 2011 to 2013, and a member of the editorial board of the *IEEE Journal on Selected Topics in Signal Processing,* in 2011. Her research interests include engineering economics and game theory, multi-agent learning, online learning, decision theory, network science, multi-user networking, big data and real-time stream mining, and multimedia. She is a fellow of the IEEE.

Veronica J. Santos received the BS degree in mechanical engineering with a music minor from the University of California, Berkeley, in 1999 and the MS and PhD degrees in mechanical engineering with a biometry minor from Cornell University, in 2007. She was a postdoctoral researcher with the University of Southern California from 2007 to 2008 and an assistant professor of mechanical and aerospace engineering with Arizona State University from 2008 to 2014. She is currently an associate professor of mechanical and aerospace engineering with the University of California, Los Angeles. She received the 2010 NSF CAREER Award and currently serves on the editorial boards for the *ASME Journal of Mechanisms and Robotics* and the IEEE International Conference on Robotics and Automation.