2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Robot Learning from Demonstration with Tactile Signals for Geometry-Dependent Tasks

Isabella Huang[1] and Ruzena Bajcsy[1]

Abstract— **Deploying robot learning frameworks in unconstrained environments requires robustness and tractability. We must not only equip the robot with a sufficient range of sensing capabilities, but also provide training data in a sample-efficient manner. To this end, we identify and address a need specifically in robot learning from demonstration (LfD) literature to account for not only end-effector pose and wrench signals, but also tactile signals for contact. While traditional pose and wrench signals have proven to be sufficient for robots to learn basic position and force-control behaviors, they are inherently too constraining for the learning of general manipulation tasks. In particular, useful manipulation tasks often rely on the geometry of the contact interaction. To explore the value of geometry-based tactile signals, we utilize a LfD framework built upon hidden Markov models and Gaussian mixture regression, adapt it to our robotic system equipped with a soft tactile sensor, and validate its performance with an edge-following task and a manipulation task involving different object geometries.**

I. INTRODUCTION

For robots to be useful in the real world, such as in the unconstrained environments of our homes, they should reliably meet the vast needs of users. Because these needs are ever-changing and cannot be fully anticipated, these robots should be able to *adapt* to different requirements. Furthermore, there should be an intuitive and simple interface through which robots can *learn* these desired behaviors. Robot learning from demonstration (LfD), also known as programming by demonstration and imitation learning, allows users avoid directly programming these behaviors onto the robot [1]. Rather, robots by means of LfD can observe human-provided demonstrations, learn to generalize the task to new situations, and then execute the proper behavior itself.

LfD has historically been successful in the encoding and reproduction of robot motion. However, accounting purely for propioception is not sufficient for many tasks, which often require physical interaction with objects and/or other agents. Thus, end effector wrench data has also been used in an LfD framework to capture salient interaction signals and enable task reproduction [2]. However, tasks cannot be fully represented with just one time-dependent wrench reading without a priori assumptions. One such assumption is that the *geometry* of the object with which the robot interacts is known and unchanging. In general, simple wrench signals cannot encode the rich tactile profiles that arise during robotic contact. Humans are equipped for dextrous manipulation through sensitive shape and force-sensing in our fingertips— for robots to perform similarly, their own tactile sensing must contribute to their learning. In this work, we explore how to perform LfD with manipulation tasks that are dependent on the *geometries* of the objects being manipulated. We employ the use of a custom tactile sensor and discuss how to effectively featurize its readings to best enable successful LfD with a low number of demonstrations. In addition, we validate the framework applied to two exemplary geometry-dependent tasks. The first is an edge-following task, and the second is a mixed manipulation task with objects of different shapes.

*This work was supported by NSF Award #1545126

[1]The authors are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94270, USA
{isabella.huang, bajcsy}@eecs.berkeley.edu

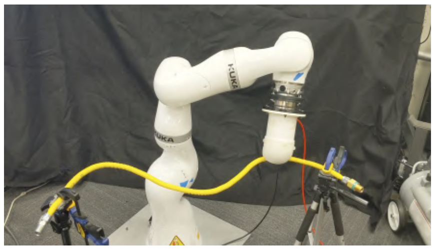
Fig. 1: Geometric tactile features for interaction-based tasks, eg. edge-following, cannot be encoded by wrench signals.

II. RELATED WORKS

Robot learning from demonstration is comprised of three building blocks— observation, encoding, and execution [3]. Though LfD is useful on a higher level of abstraction, such as in decomposing a complicated task into a sequence of subtasks [4], much work including ours performs learning on the lower sensory and motor levels. Over time, LfD works have advanced from motion replication to robust frameworks that use vision and force-torque sensing for interaction with the environment. Learning force-control tasks in particular has been successful for tasks such as ironing a shirt or pushing open a door [2]. While force sensing is imperative for the proper learning and execution of any interactive task, all prior LfD works have only recorded end effector wrenches, either via an attached force torque sensor or by reconstruction using joint torques. However, wrench readings yield no insight about contact geometry, which is a crucial element for interaction in general.

There are various interfaces through which human teachers can provide demonstrations to the robot. A natural method is for the teacher to perform the task themselves and have their actions recorded by a motion-capture system [5]. When additional sensing is required of the task, force and pose sensing have been combined through a force-sensing glove worn by the demonstrator [6]. However, there is a correspondence problem in mapping the sensing-action spaces between the teacher and robot [7]. This correspondence problem is removed when demonstrations are performed on the robot itself via teleoperation or kinesthetic teaching. Haptic devices have been used to successfully teach force-based tasks through teleoperation while allowing the demonstrator to feel the forces sensed by the robot [8]. On the other hand, while the demonstrator physically guides the robot in kinesthetic teaching, the contact forces acting on the robot are physically relayed to the teacher as well. However, when end effector wrenches are inferred using joint torques, torques applied during kinesthetic teaching can confound this inference. This is not an issue for our particular setup, and thus all our demonstrations were performed kinesthetically.

Compared to deep learning techniques, where high-dimensional data is fed as input at the cost of requiring much of it, simpler LfD models achieve generalization with less data with low dimensionality. The dimensionality of the demonstration data can be reduced by either dropping sensor readings that contain the least mutual information with the action outputs [8], or by projecting them into a latent space [9]. Dynamic motion primitives (DMP) is an LfD approach first proposed in [10] that, after having broken down a complex task into a sequence of motion primitives, models them as a set of mass-spring-damper systems. It is a general framework for all motor and sensor trajectories, time-invariant, and guaranteed to converge. Meanwhile, the hidden Markov model (HMM) provides a unified approach that can encode multiple motion alternatives in one model and can be updated with partial demonstrations [11]. Though HMM does not have a convergence guarantee, it is advantageous in that it does not require the a priori representation of the task structure as with DMP. Thus, we chose to adopt the HMM approach for its flexibility.

III. LEARNING FROM DEMONSTRATION FRAMEWORK

Our LfD framework closely follows the proposed approach by Calinon et al [11], which models the task as an HMM and predicts best actions during execution with a modified version of Gaussian mixture regression (GMR) [12] that accounts for hidden state transitions.

A. Hidden Markov Model Setup

The crux of the HMM approach in a LfD setting is to estimate the joint distributions of the sensor input and action outputs observed in a demonstration. Each teacher demonstration is comprised of of a sequence of physically observable sensor input-action output pairs $O = (O_1 = o_1, ..., O_T = o_T)$ of length T. Under a hidden Markov model, we assume that the system at each step t is in one of N unobservable, or hidden states $Q_t \in \{1...N\}$. We assume that the probability of an observation o_t at time t given that the hidden state is j, denoted as $b_j(o_t)$, is normally distributed according to parameters associated to j:

$$b_j(o_t) := p(O_t = o_t | Q_t = j) = \mathcal{N}(o_t | \mu_j, \Sigma_j) \tag{1}$$

We denote the entire set of observation distribution parameters by $B = \{\mu_j, \Sigma_j\}$. Furthermore, the chain of hidden states is assumed to be both time-homogeneous and Markovian, so that the state transitions can be modeled by a time-independent stochastic transition matrix $A = \{a_{i,j}\} := p(Q_t = j | Q_{t-1} = i)$. At the initial state of $t = 1$, the hidden state distribution is described by $\pi_1 = p(Q_1 = i)$. Thus, every hidden Markov model is defined by a set of parameters $\Lambda = (A, B, \pi)$. These parameters are found by employing the Baum-Welch algorithm to find the maximum likelihood estimate Λ^* given the observation sequence O:

$$\Lambda^* = \underset{\Lambda}{\operatorname{argmax}}\, p(O|\Lambda) \tag{2}$$

B. Baum-Welch Algorithm

The HMM forward variable $\alpha_i(t)$ is the probability at time t of having observed the partial starting sequence $\{O_1 = o_1, ..., O_t = o_t\}$ and being in state i, and is defined recursively as such:

$$\alpha_i(t) = p(O_1 = o_1, ..., O_t = o_t, Q_t = i | \Lambda) \tag{3}$$

$$\alpha_i(1) = \pi_i b_i(o_1) \tag{4}$$

$$\alpha_j(t+1) = b_j(o_{t+1}) \sum_{i=1}^{N} \alpha_i(t) a_{ij} \tag{5}$$

The backward variable $\beta_i(t)$ is the probability of observing the partial ending sequence $\{O_{t+1} = o_{t+1}, ..., O_T = o_T\}$ having been in state i at time t, and is defined recursively:

$$\beta_i(t) = p(O_{t+1} = o_{t+1}, ..., O_T = o_T | Q_t = i, \Lambda) \tag{6}$$

$$\beta_i(T) = 1 \tag{7}$$

$$\beta_i(t) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_j(t+1) \tag{8}$$

We further define $\gamma_i(t)$ as the probability of being in state i at time t given observation sequence O:

$$\gamma_i(t) := p(Q_t = i | O, \Lambda) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)} \tag{9}$$

Also, $\xi_{ij}(t)$ is defined as $p(Q_t = i, Q_{t+1} = j | O, \Lambda)$, the probability of being in state i at time t and then j at $t+1$:

$$\xi_{ij}(t) = \frac{\gamma_i(t) a_{ij} b_j(o_{t+1}) \beta_j(t+1)}{\beta_i(t)} \tag{10}$$

Similarly to typical expectation-maximization algorithm for Gaussian mixtures, the HMM parameter update equations are as follows, where m indexes the M demonstrations $\{O_1, ...O_M\}$:

$$\pi_i = \frac{\sum_{m=1}^{M} \gamma_i^m(1)}{M} \tag{11}$$

$$\mu_i = \frac{\sum_{m=1}^{M}\sum_{t=1}^{T_m}\gamma_i^m(t)o_t^m}{\sum_{m=1}^{M}\sum_{t=1}^{T_m}\gamma_i^m(t)} \tag{12}$$

$$\Sigma_i = \frac{\sum_{m=1}^{M}\sum_{t=1}^{T_m}\gamma_i^m(t)(o_t^m-\mu_i)(o_t^m-\mu_i)^T}{\sum_{m=1}^{M}\sum_{t=1}^{T_m}\gamma_i^m(t)} \tag{13}$$

$$a_{ij} = \frac{\sum_{m=1}^{M}\sum_{t=1}^{T_m-1}\xi_{ij}^m(t)}{\sum_{m=1}^{M}\sum_{t=1}^{T_m-1}\gamma_i^m(t)} \tag{14}$$

In our work, we initialize π to be uniform and μ_i to be the i^{th} mean of N-means clustering on the demonstration data. Covariances Σ_i are initialized randomly.

C. Action Prediction during Task Replication

Each observation from demonstration m at elapsed time t is comprised of a sensor input-action output pair. That is, $o^m(t) = \begin{bmatrix} S^m(t) & A^m(t)\end{bmatrix}^T$. Thus, parameters in B can be broken into components corresponding to the input and output spaces of the observations, such that:

$$\mu_i = \begin{bmatrix}\mu_i^S & \mu_i^A\end{bmatrix}^T \tag{15}$$

$$\Sigma_i = \begin{bmatrix}\Sigma_i^{SS} & \Sigma_i^{SA} \\ \Sigma_i^{AS} & \Sigma_i^{AA}\end{bmatrix} \tag{16}$$

One method by which to predict the most likely action output $\hat{A}$ from a sensor input S is to treat the hidden states as constituents of a Gaussian mixture model and apply Gaussian mixture regression [12]. However, this regression does not leverage the sequential encoding of the learned HMM. Calinon et al [11] proposed an extension to GMR, called GMRa, which instead considers not only the state vector itself, but also the sequential information captured through the forward variable in the HMM. Under this formulation, the mixing weight for each state becomes the relative likelihood of the corresponding forward variable component in the sensor input space. That is,

$$h_i(S_t) = \frac{\mathcal{N}(S_t|\mu_i^S,\Sigma_i^{SS})\sum_{j=1}^{K}h_j(S_{t-1})a_{ji}}{\sum_{k=1}^{K}\left[\mathcal{N}(S_t|\mu_k^S,\Sigma_k^{SS})\sum_{j=1}^{K}h_j(S_{t-1})a_{jk}\right]} \tag{17}$$

$$h_i(S_1) = \frac{\pi_i\mathcal{N}(S_1|\mu_i^S,\Sigma_i^{SS})}{\sum_{k=1}^{K}\left[\pi_k\mathcal{N}(S_1|\mu_k^S,\Sigma_k^{SS})\right]} \tag{18}$$

IV. EXPERIMENTAL SETUP

A. System Design

Our robotic platform comprises a soft tactile sensor mounted as the end effector of a KUKA LBR iiwa 14 R820 (Fig. 2). This sensor contacts the environment with a hemispherical palm-like 3 mm thick membrane molded from Ecoflex$^{\text{TM}}$ 50 silicone. It is sealed onto an air-pressurized cylindrical capsule, and inflates slightly upon internal pressurization. The sensor was fixed at one pressure state, at which the membrane's inner radius was inflated to 50 mm compared to its neutral 47 mm radius when free air exchange is permitted. Contact states are measured by imaging the inside of the membrane using an embedded miniature time-of-flight Camboard pico-flexx depth-sensing camera from PMD Technologies. Demonstrations were performed kinesthetically by the researcher and collected at 5 Hz.

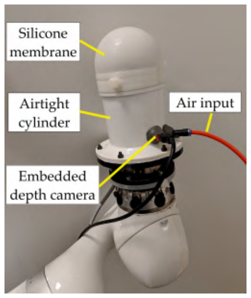

Fig. 2: Tactile sensor mounted onto the KUKA end effector.

Fig. 3: Example point cloud reading of the tactile sensor, with black deformed portion and red principal component vectors.

B. Input and Output Featurization

The inputs of the learning model are derived from the high-dimensional 224×171 sized point cloud readings of the tactile sensor. As shown in the example point cloud reading in Fig. 3, we can isolate the portion of the point cloud that is deformed upon contact through comparison with the undeformed reading. From this deformed point cloud, signals correlated with the physical contact state related to forces and geometry can be obtained. We extract a a set of signals including the 3D centroid $\vec{c}$, the three variance values $\vec{\lambda}$ along its three principal component vectors, as well as the three principal component vectors $\vec{e_1}$, $\vec{e_2}$, and $\vec{e_3}$. The subset of these signals to be used as features in the HMM were handpicked based on domain knowledge, rather than through quantitative methods such as mutual information analysis. Consistency in the direction of the component vectors is maintained by setting their signs so their largest element is positive. All sensor input quantities are expressed in the world frame rather than in the camera frame. When the deformed point cloud is empty, as is the case when nothing contacts the sensor, by default the centroid is the origin of the camera frame, and the principal component vectors are the camera frame coordinate axes. When transformed to the world frame, these default quantities during non-contact contain propioceptive information regarding the pose of the end effector. By not including explicit robot position and orientation signals in the input, we ensure that during contact, the robot responds solely to tactile signals rather than overfits to the demonstrated trajectories. The action output of the model is a 6-dimensional instantaneous velocity of the end effector with respect to the world frame with both linear $\vec{v}$ and angular $\vec{\omega}$ components approximated by the rate of change of time-adjacent readings. After collecting demonstrations, all datapoints for both inputs and outputs were first linearly normalized from 0 to 1 before learning the HMM. During task execution, outputs are predicted for normalized inputs, then sent to the robot as velocity commands at 5 Hz.

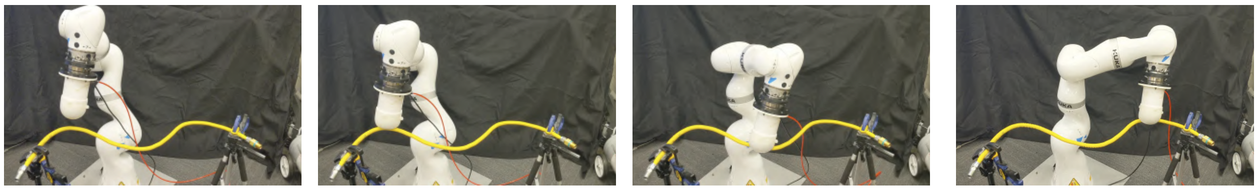
(a) Robot begins its policy rollout above the obstacle. (b) Robot lowers itself onto the pipe and follows the edge. (c) Contact is maintained during a dip in the $-z$ direction (d) Robot continues to contact the pipe along a lateral bend.

Fig. 4: Policy rollout of edge following HMM on a curved pipe.

V. EDGE-FOLLOWING TASK

A. Task Demonstrations

The first task taught to the robot was to follow an edge. During the demonstration phase, we lowered the robot end effector until the palm sensor contacted an edge positioned directly below the starting point, and then slid the sensor along that edge. The physical edge used in the demonstration was the length of a straight copper pipe with a 0.5” diameter. We performed 6 different demonstrations with the pipe fixed at different orientations and positions in space. By convention, the traversal direction was such that the y component in the world frame was positive. We considered the edge-following task successful as long as the sensor consistently made contact with the edge, meaning that the end effector orientation was not critical to the task. The six demonstration end effector position trajectories are plotted in Fig. 5, with arrows along the trajectory showing the $\vec{e_1}$ reading at that position. The demonstration consists of two intuitive phases. The first is the lowering phase, during which no contact with the edge is made. The next is the phase during which the sensor contacts the edge and traverses along it.

B. Learned HMM

For both tasks in this work, sensor input features were handpicked to minimize dimensionality but maximize relevance to the action outputs. In edge-following, it is important to detect only whether the sensor is in contact with something, and if so, along which direction the longest edge is “pointing.” Thus, we chose the inputs to be the principal variances and the first principal component vector, such that $S(t) = \begin{bmatrix}\lambda_1(t) & \lambda_2(t) & \lambda_3(t) & e_{1x}(t) & e_{1y}(t) & e_{1z}(t)\end{bmatrix}^T$. The centroid was excluded from the feature vector because the location of contact is not important to the task. Although the demonstrations were performed on a straight pipe, we wanted the robot to be able to follow an edge with arbitrary curvature. To this end, we learned an HMM with $N = 2$ states— one that corresponds to the robot lowering onto the edge, and one to perform edge following. We tried to avoid overfitting to the demonstration by limiting the number of hidden states to be learned. We plot the same demonstrations in $\vec{\lambda}$ space in Fig. 6 with each reading colored according to the most likely hidden state it is in according to Eq. 17. Matching up to our intuition, the two HMM states do correspond to the physical pre-contact (Q_2) and edge traversal conditions (Q_1), where the yellow star marks the reading in the absence of contact, ie. $\vec{\lambda} = \vec{0}$.

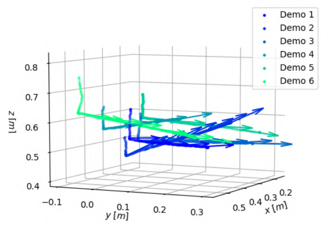

Fig. 5: Demonstration trajectories for the edge-following experiment along a straight pipe. Round markers are end effector positions, and are $\vec{e_1}$.

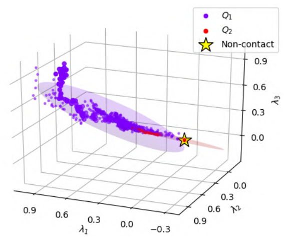

Fig. 6: Demonstration data alongside HMM states in normalized $\vec{\lambda}$ space. Ellipsoids depict the 95% confidence region of the corresponding Σ_i.

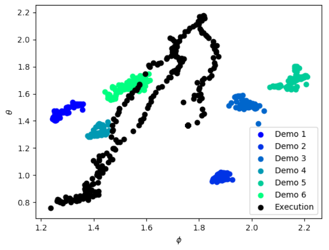

Fig. 7: Principal component $\vec{e_1}$ observed in demonstration and execution phases, mapped onto spherical coordinates.

C. Task Execution

To verify generalizability of the edge-following HMM, we performed the task execution on a flexible gas connector also with a 0.5” diameter that was bent to be curved in 3D space. As pictured in the rollout in Fig. 4, the robot was successful in initiating and maintaining contact throughout the length of the pipe. We can visualize how the HMM was able to generalize to new observations in Fig. 7, where the spherical coordinates of $\vec{e_1}$ observed in both the demonstration and rollout phases had little overlap. Note that in the teaching phase, each demonstration performed edge traversal in only one direction. Thus, during the execution phase, the robot is unable to follow an edge that goes back in an opposite direction it has experienced before, such as along a closed path. Furthermore, since no stopping condition was observed in the demonstrations, as long as the robot senses an edge, it will move. If contact is broken, it believes it is again in Q_2 and lowers itself until a new edge is found.

VI. MIXED OBJECT MANIPULATION

A. *Task Demonstrations*

The robot was also taught to manipulate objects with different geometries with the following desired behavior:

- Starting from a point above a predefined quadrant of the table, lower the robot palm and make contact with the object directly underneath.
- If the object is capsule-shaped, *push* it through a predefined corridor and off the table into a bin.
- If the object is a rectangular prism, *rotate* it in place until it its length is parallel to the table edge closest to the robot.

Example demonstrations for both subtasks are in Fig. 9 and 8 for the capsule and prism subtasks respectively. Nine different objects pictured in Fig. 10 were used in demonstrating this task. Of the nine, eight were capsules (with diameters and heights ranging from $[1...5]$ and $[1.5...1.7]$ cm), and one was a rectangular prism of dimensions $7.5 \times 1 \times 2$ cm. Ten demonstrations of each type (pushing and rotating) were performed on the robot, and a 4-state HMM was learned from the data. The sensor input used here was the 9-dimensional feature vector containing $\vec{c}$, $\vec{\lambda}$, and $\vec{e_1}$.

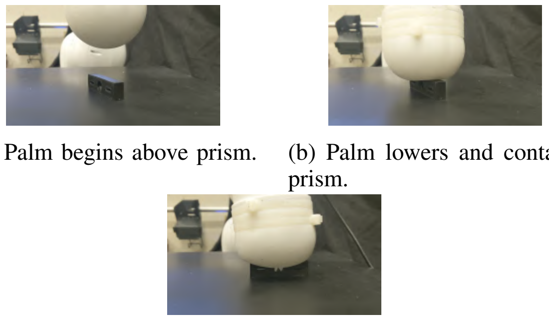
(a) Palm begins above prism.
(b) Palm lowers and contacts prism.
(c) Palm rotates until desired orientation is achieved.

Fig. 8: Kinesthetic demonstration showing how to rotate a prism to its desired orientation.

B. *Learned HMM*

A 4-state HMM was learned from the collected demonstrations, and the most likely exit transitions are depicted in the finite state machine in Fig. 11. Figs. 12 and 13 plot the centroids observed in the demonstrated manipulations with the capsules and prism respectively, colored according to the most likely hidden state. Both subtasks start off in the same state, Q_3, which corresponds to the pre-contact lowering phase. Upon contact, the system moves into either

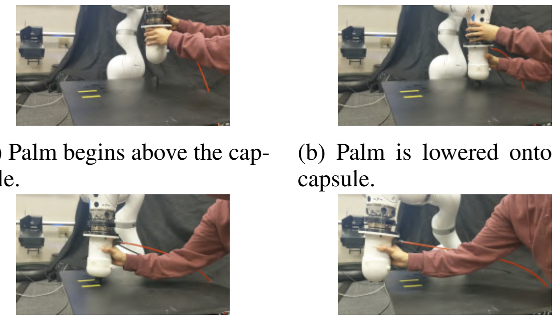
(a) Palm begins above the capsule.
(b) Palm is lowered onto the capsule.
(c) Palm pushes the capsule through a corridor.
(d) Palm pushes capsule off of table.

Fig. 9: Kinesthetic demonstration showing how to push a capsule through a corridor and off the table.

Fig. 10: Set of objects used in the mixed manipulation task.

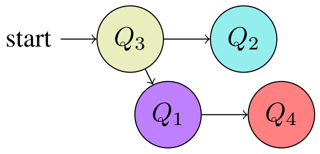
$$\begin{tikzcd}
\text{start} \arrow[r] & Q_3 \arrow[r] \arrow[d] & Q_2 \\
& Q_1 \arrow[r] & Q_4
\end{tikzcd}$$

Fig. 11: 4-state HMM with the most likely exit transitions depicted as as edges.

the pushing phase Q_1 or the rotating phase Q_2 depending on the shape of the object. Fig. 14 depicts the demonstrations in normalized $\vec{\lambda}$ space and shows that there is a clear distinction between $\vec{\lambda}$s when contacting the capsules versus the prism that the HMM was able to differentiate. Once the robot pushes the object off the table so that it is no longer sensed by the palm, it immediately transitions from Q_1 to the post-pushing phase Q_4.

C. *Task Execution*

In the task execution phase, we gave the robot 10 unseen scenes (5 for each subtask type) from randomly sampled starting poses in the predefined table quadrant, and found that it was successful in completing its task in all 10 trials. For the capsule subtasks, the executed trajectory successfully eased the capsule through the corridor and off the table just like in the demonstrations (Fig. 15). During the prism subtask executions, the rectangle's orientation also converged correctly (Fig. 16), where the final orientations from the execution matched up well with that which was observed in demonstration. Granted, the starting states in the execution phase were not vastly different from what was observed in the demonstration phase. For example, if the task was initialized at the opposite side of the table, around which the robot never recorded any demonstration, we found that it either did not manage to squeeze through the corridor properly, or even veered off in a wrong direction. Because the $\vec{c}$ centroid readings were highly correlated with the actual position of the end effector, we would have had to ensure a higher variance in positions during the demonstration phase to encourage generalization.

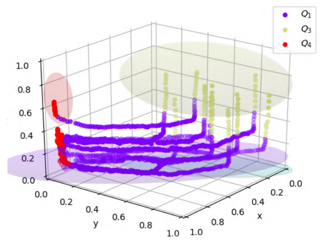
Fig. 12: Capsule demonstrations alongside HMM states in normalized $\vec{c}$ space. Ellipsoids depict 95% confidence region.

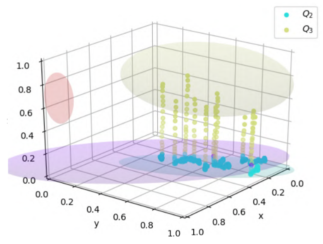
Fig. 13: Prism demonstrations alongside HMM states in normalized $\vec{c}$ space. Ellipsoids depict 95% confidence region.

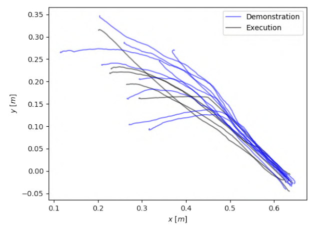
Fig. 15: Centroid trajectories recorded in demonstration and execution phases during contact with the capsule mapped onto the normalized xy space.

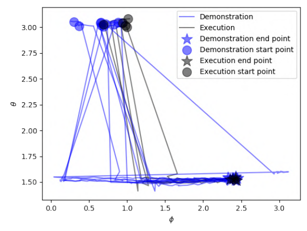
Fig. 16: Principal components $\vec{e_1}$ recorded in demonstration and execution phases during contact with the prism, mapped onto spherical coordinates.

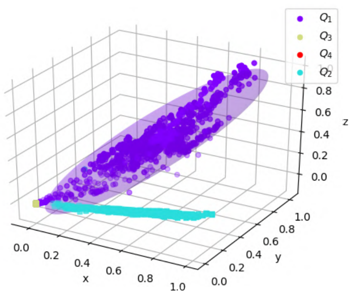
Fig. 14: Demonstrations alongside HMM states in normalized $\vec{\lambda}$ space, with ellipsoids at the 95% confidence region. The two most significant visible states Q_1 and Q_2 clearly separate the two object types.

VII. LIMITATIONS AND FUTURE WORK

While we successfully incorporated geometry-based tactile sensing to learn the demonstrated tasks, they were both simple tasks by design. The sensed geometries were either static, as with the capsules and prism, or evolving slowly, as with the curved tube. It would be compelling to investigate dynamic tasks where the contacts evolve rapidly. In addition, our method of featurizing tactile signals using PCA of the deformed membrane worked well for simple shapes, but cannot be sufficient for more complicated shapes or distributed contacts. Finally, we used just one tactile sensing palm that can only push and twist. A robot hand with multiple tactile sensing fingers would pose a more interesting learning problem that requires higher dimensional coordination.

REFERENCES

[1] B.D. Argall et al. "A Survey of Robot Learning from Demonstration". In: *Robot. Auton. Syst.* 57.5 (2009), 469–483.

[2] P. Kormushev et al. "Imitation Learning of Positional and Force Skills Demonstrated via Kinesthetic Teaching and Haptic Input". In: *Advanced Robotics* 25.5 (2011), pp. 581–603.

[3] P. Bakker and Y. Kuniyoshi. "Robot See, Robot Do: An Overview of Robot Imitation". In: *AISB96 Workshop on Learning in Robots and Animals*. 1996, pp. 3–11.

[4] D. Grollman and O. Jenkins. "Can We Learn Finite State Machine Robot Controllers from Interactive Demonstration?" In: *From Motor Learning to Interaction Learning in Robots*. Vol. 264. 2010, pp. 407–430.

[5] D. Kulić et al. "Incremental learning of full body motion primitives and their sequencing through human motion observation". In: *The International Journal of Robotics Research* 31.3 (2012), pp. 330–345.

[6] M. Edmonds et al. "Feeling the Force: Integrating Force and Pose for Fluent Discovery through Imitation Learning to Open Medicine Bottles". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017.

[7] Y. Mohammad and Y. Nishida. "Tackling the Correspondence Problem". In: *Active Media Technology*. Cham: Springer International Publishing, 2013, pp. 84–95.

[8] L. et al Rozo. "A Robot Learning from Demonstration Framework to Perform Force-based Manipulation Tasks". In: *Journal of Intelligent Service Robotics, Special Issue on AI Techniques for Robotics* 6 (2013), pp. 33–51.

[9] S. Calinon, F. Guenter, and A. Billard. "On Learning, Representing, and Generalizing a Task in a Humanoid Robot". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 37 (2007), pp. 286–298.

[10] A.J. Ijspeert et al. "Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors". In: *Neural Computation* 25.2 (2013), 328–373.

[11] S. Calinon et al. "Learning and Reproduction of Gestures by Imitation". In: *IEEE Robotics Automation Magazine* 17.2 (2010), pp. 44–54.

[12] H.G. Sung. "Gaussian Mixture Regression and Classification". PhD thesis. Rice University, 2004.