

Learning to Represent Haptic Feedback for Partially-Observable Tasks

Jaeyong Sung^{1,2}, J. Kenneth Salisbury¹ and Ashutosh Saxena³

Abstract—The sense of touch, being the earliest sensory system to develop in a human body [1], plays a critical part of our daily interaction with the environment. In order to successfully complete a task, many manipulation interactions require incorporating haptic feedback. However, manually designing a feedback mechanism can be extremely challenging. In this work, we consider manipulation tasks that need to incorporate tactile sensor feedback in order to modify a provided nominal plan. To incorporate partial observation, we present a new framework that models the task as a partially observable Markov decision process (POMDP) and learns an appropriate representation of haptic feedback which can serve as the state for a POMDP model. The model, that is parametrized by deep recurrent neural networks, utilizes variational Bayes methods to optimize the approximate posterior. Finally, we build on deep Q-learning to be able to select the optimal action in each state without access to a simulator. We test our model on a PR2 robot for multiple tasks of turning a knob until it clicks.

I. INTRODUCTION

Many tasks in human environments that we do without much effort require more than just visual observation. Very often they require incorporating the sense of touch to complete the task. For example, consider the task of turning a knob that needs to be rotated until it clicks, like the one in Figure 1. The robot could observe the consequence of its action if any visible changes occur, but such clicks can often only be directly observed through the fingers. Many of the objects that surround us are explicitly designed with feedback — one of the key interaction design principles — otherwise “one is always wondering whether anything has happened” [2].

Recently, there has been a lot of progress in making robots understand and act based on images [3], [4], [5] and point-clouds [6]. A robot can definitely gain a lot of information from visual sensors, including a nominal trajectory plan for a task [6]. However, when the robot is manipulating a small object or once the robot starts interacting with small parts of appliances, self-occlusion by its own arms and its end-effectors limits the use of the visual information.

However, building an algorithm that can examine haptic properties and incorporate such information to influence a motion is very challenging for multiple reasons. First, haptic feedback is a dynamic response that is dependent on the action the robot has taken on the object as well as internal states and properties of the object. Second, every haptic sensor produces a vastly different raw sensor signal.

Moreover, compared to the rich information that can be extracted about a current state of the task from few images

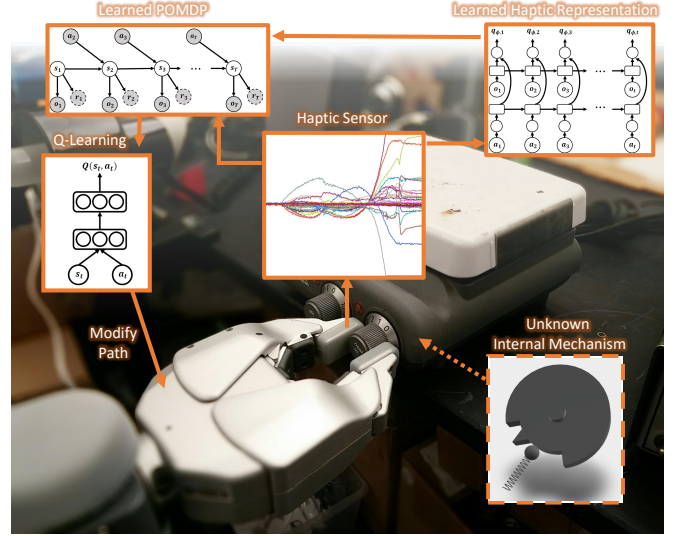


Fig. 1: **Haptic feedback** from a tactile sensor being used to modify a nominal plan of manipulation. Our framework learns an appropriate representation (embedding space) which in turn is used to learn to find optimal control.

(e.g. position and velocity information of an end-effector and an object [5], [3]), a short window of haptic sensor signal is merely a partial consequence of the interaction and of the changes in an unobservable internal mechanism. It also suffers from perceptual aliasing — *i.e.* many segments of a haptic signal at different points of interaction can produce a very similar signal. These challenges make it difficult to design an algorithm that can incorporate information from haptic modalities (in our case, tactile sensors).

In this work, we introduce a framework that can learn to represent haptic feedback for tasks requiring incorporation of a haptic signal. Since a haptic signal only provides a partial observation, we model the task using a partially observable Markov decision process (POMDP). However, since we do not know of definition of states for a POMDP, we first learn an appropriate representation from a haptic signal to be used as continuous states for a POMDP. To overcome the intractability in computing the posterior, we employ a variational Bayesian method, with a deep recurrent neural network, that maximizes lower bound of likelihood of the training data.

Using a learned representation of the interaction with feedback, we build on deep Q-learning [5] to identify an appropriate phase of the action from a provided nominal plan. Unlike most other applications of successful reinforce-

¹Department of Computer Science, Stanford University. ²Department of Computer Science, Cornell University. ³Brain Of Things, Inc. {jysung, jks, asaxena}@cs.stanford.edu

ment learning [5], [7], the biggest challenge is a lack of a robotics simulation software that can generate realistic haptic signals for a robot to safely simulate and explore various combinations of states with different actions.

To validate our approach, we collect a large number of sequences of haptic feedback along with their executed motion for the task of ‘turning a knob until it clicks’ on objects of various shapes. We empirically show on a PR2 robot that we can modify a nominal plan and successfully accomplish the task using the learned models, incorporating tactile sensor feedback on the fingertips of the robot. In summary, the key contributions of this work are:

- an algorithm which learns task relevant representation of haptic feedback
- a framework for modifying a nominal manipulation plan for interactions that involves haptic feedback
- an algorithm for learning optimal actions with limited data without simulator

II. RELATED WORK

Haptics. Haptic sensors mounted on robots enable many different interesting applications. Using force and tactile input, a food item can be classified with characteristics which map to appropriate class of motions [8]. Haptic adjectives such as ‘sticky’ and ‘bumpy’ can be learned with biomimetic tactile sensors [9]. Whole-arm tactile sensing allows fast reaching in dense clutter. We focus on tasks with a nominal plan (e.g. [6]) but requires incorporating haptic (tactile) sensors to modify execution length of each phase of actions.

For closed-loop control of robot, there is a long history of using different feedback mechanisms to correct the behavior [10]. One of the common approaches that involves contact relies on stiffness control, which uses the pose of an end-effector as the error to adjust applied force [11], [12]. The robot can even self-tune its parameters for its controllers [13]. A robot also uses the error in predicted pose for force trajectories [14] and use vision for visual servoing [15].

Haptic sensors have also been used to provide feedback. A human operator with a haptic interface device can teleoperate a robot remotely [16]. Features extracted from tactile sensors can serve as feedback to planners to slide and roll objects [17]. [18] uses tactile sensor to detect success and failure of manipulation task to improve its policy.

Partial Observability. A POMDP is a framework for a robot to plan its actions under uncertainty given that the states are often only obtained through noisy sensors [19]. The framework has been successfully used for many tasks including navigation and grasping [20], [21]. Using wrist force/torque sensors, hierarchical POMDPs help a robot localize certain points on a table [22]. While for some problems [20], states can be defined as continuous robot configuration space, it is unclear what the ideal state space representation is for many complex manipulation tasks.

When the knowledge about the environment or states is not sufficient, [23] use a fully connected DBN for learning factored representation online, while [24] employ a two step method of first learning optimal decoder then learning to

encode. While many of these work have access to a good environment model, or is able to simulate environment where it can learn online, we cannot explore or simulate to learn online. Also, the reward function is not available. For training purposes, we perform privileged learning [25] by providing an expert reward label only during the training phase.

Representation Learning. Deep learning has recently vastly improved the performance of many related fields such as compute vision (e.g. [26]) and speech recognition (e.g. [27]). In robotics, it has helped robots to better classify haptic adjectives by combining images with haptic signals [28], predict traversability from long-range vision [29], and classify terrains based on acoustics [30].

For controlling robots online, a deep auto-encoder can learn lower-dimensional embedding from images and model-predictive-control (MPC) is used for optimal control [31]. DeepMPC [14] predicts its future end-effector position with a recurrent network and computes an appropriate amount of force. Convolutional neural network can be trained to directly map images to motor torques [3], [32]. As mentioned earlier, we only take input of haptic signals, which suffers from perceptual aliasing, and contains a lot less information in a single timestep compared to RGB images.

Recently developed variational Bayesian approach [33], [34], combined with a neural network, introduces a recognition model to approximate intractable true posterior. Embed-to-Control [4] learns embedding from images and transition between latent states representing unknown dynamical system. Deep Kalman Filter [35] learns very similar temporal model based on Kalman Filter but is used for counterfactual inference on electronic health records.

Reinforcement learning (RL), also combined with a neural network, has recently learned to play computer games by looking at pixels [5], [36]. Applying standard RL to a robotic manipulation task, however, is challenging due to lack of suitable state space representation [32]. Also, most RL techniques rely on trial and error [37] with the ability to try different actions from different states and observe reward and state transition. However, for many of the robotic manipulation tasks that involve physical contact with the environment, it is too risky to let an algorithm try different actions, and reward is not trivial without instrumentation of the environment for many tasks. In this work, the robot learns to represent haptic feedback and find optimal control from limited amount of haptic sequences despite lack of good robotic simulator for haptic signal.

III. OUR APPROACH

Our goal is to build a framework that allows robots to represent and reason about haptic signals generated by its interaction with an environment.

Imagine you were asked to turn off the hot plate in Figure 1 by rotating the knob until it clicks. In order to do so, you would start by rotating the knob clockwise or counterclockwise until it clicks. If it doesn’t click and if you feel the wall, you would start to rotate it in the opposite direction. And, in order to confirm that you have successfully

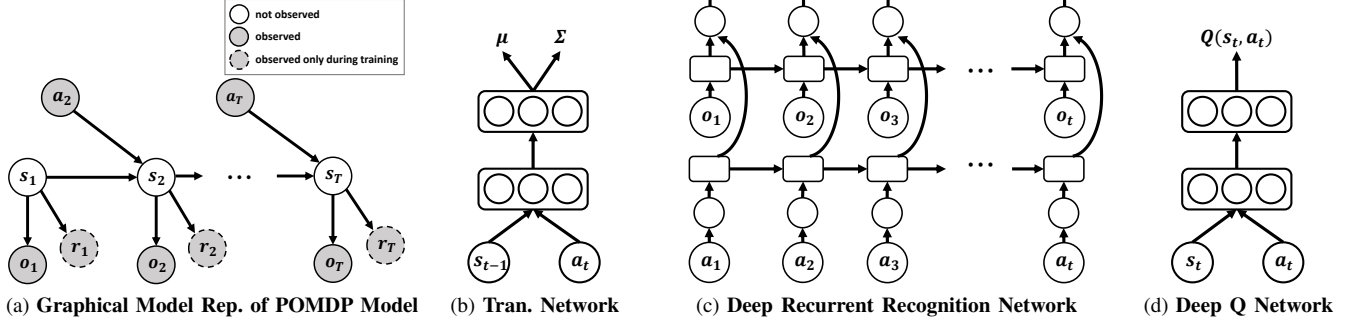


Fig. 2: **Framework Overview.** We model the task that requires incorporation of tactile feedback in a partially observable MDP (a) which its transition and emission functions are parametrized by neural networks (b). To find an appropriate representation of states for the POMDP, we approximate the posterior with a Deep Recurrent Recognition Network (c), consisting of two LSTM (square blocks) recurrent networks. Deep Q-Network (d), consisting of two fully connected layers, utilizes a learned representation from (c) and a learned transition model from (a) to train Deep Q-Network (d).

completed the task or hit the wall, you would use your sense of touch on your finger to feel a click. There could also be a sound of a click as well as other observable consequences, but you would not feel very confident about the click in the absence of haptic feedback.

However, such haptic signal itself does not contain sufficient information for a robot to directly act on. It is unclear what is the best representation for a state of the task, whether it should only be dependent on states of internal mechanisms of the object (which are unknown) or it should incorporate information about the interaction as well. The haptic signal is merely a noisy partial observation of latent states of the environment, influenced by many factors such as a type of interaction that is involved and a type of grasp by the robot.

To learn an appropriate representation of the state, we first define our manipulation task as a POMDP model. However, posterior inference on such latent state from haptic feedback is intractable. In order to approximate the posterior, we employ variational Bayes methods to jointly learn model parameters for both a POMDP and an approximate posterior model, each parametrized by a deep recurrent neural network.

Another big challenge is the limited opportunity to explore with different policies to fine-tune the model, unlike many other applications that employs POMDP or reinforcement learning. Real physical interactions involving contact are too risky for both the robot and the environment without lots of extra safety measures. Another common solution is to explore in a simulated environment; however, none of the available robot simulators, as far as we are aware, are capable of generating realistic feedback for objects of our interest.

Instead, we learn offline from previous experiments by utilizing a learned haptic representation along with its transition model to explore offline and learn Q-function.

A. Problem Formulation

Given a sequence of haptic signals ($\vec{o} = o_1, \dots, o_t$) up to current time frame t along with a sequence of actions taken ($\vec{a} = a_1, \dots, a_t$), our goal is to output a sequence of

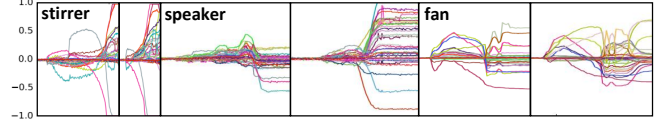


Fig. 3: **Samples of haptic signals** from three different objects with a PR2 fingertip tactile sensor. Notice a large variation in feedback produced by what humans identify as a ‘click’.

appropriate state representations ($\vec{s} = s_1, \dots, s_t$) such that we can take an optimal next action a_{t+1} inferred from the current state s_t .

B. Generative Model

We formulate the task that requires haptic feedback as a POMDP model, defined as (S, A, T, R, O) . S represents a set of states, A represents a set of actions, T represents a state transition function, R represents a reward function, and O represents an observation probability function. Fig. 2a represents a graphical model representation of a POMDP model and all notations are summarized in Table I.

Among the required definitions of a POMDP model, most importantly, state S and its representation are unknown. Thus, all functions T, R, O that rely on states S are also not available.

We assume that all transition and emission probabilities are distributed as Gaussian distributions; however, they can take any appropriate distribution for the application. Mean and variance of each distribution are defined as a function with input as parent nodes in the graphical model (Fig. 2a):

$$\begin{aligned} s_1 &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ s_t &\sim \mathcal{N}(f_{s_\mu}(s_{t-1}, a_t), f_{s_\Sigma}(s_{t-1}, a_t)^2 \mathbf{I}) \\ o_t &\sim \mathcal{N}(f_{o_\mu}(s_t), f_{o_\Sigma}(s_t)^2 \mathbf{I}) \\ r_t &\sim \mathcal{N}(f_{r_\mu}(s_t), f_{r_\Sigma}(s_t)^2 \mathbf{I}) \end{aligned}$$

We parametrize each of these functions as a neural network. Fig. 2b shows a two layer network for parametrization of the transition function, and emission networks take a similar structure. The parameters of these networks form the parameters of the generative model $\theta = \{s_\mu, s_\Sigma, o_\mu, o_\Sigma, r_\mu, r_\Sigma\}$.

TABLE I: Summary of Notations.

Notations	Descriptions
S	continuous state space (a learned representation)
O	observation probability ($S \rightarrow O$) of haptic signal
T	conditional probability between states ($S \times A \rightarrow S$)
A	a set of possible actions to be taken at each time step
R	a reward function ($S \rightarrow \mathbb{R}$)
p_θ	a generative model for O and R
θ	parameters of generative model
q_ϕ	an approximate posterior distribution (a recognition network for representing haptic signal)
ϕ	parameters of recognition network (recurrent neural network)
$Q(s, a)$	an approximate action-value function ($S \times A \rightarrow \mathbb{R}$)
γ	a discount factor

C. Deep Recurrent Recognition Network

Due to non-linearity of multi-layer neural network, computing the posterior distribution $p(\vec{s}|\vec{o}, \vec{r}, \vec{a})$ becomes intractable [35]. The variational Bayes method [33], [34] allows us to approximate the real posterior distribution with a recognition network (encoder) $q_\phi(\vec{s}|\vec{o}, \vec{r}, \vec{a})$.

Although it is possible to build a recognition network $q_\phi(\vec{s}|\vec{o}, \vec{r}, \vec{a})$ that takes the reward \vec{r} as a part of the input, such recognition network would not be useful during a test time when the reward \vec{r} is not available. Since a reward is not readily available for many of the interaction tasks, we assume that the sequence of rewards \vec{r} is available only during a training phase given by an expert. Thus, we build an encoder $q_\phi(\vec{s}|\vec{o}, \vec{a})$ without a reward vector while our goal will be to reconstruct a reward \vec{r} as well (Sec. III-D).

Among many forms and structures q_ϕ could take, through validation with our dataset, we chose to define $q_{\phi,t}(s_t|o_1, \dots, o_t, a_1, \dots, a_t)$ as a deep recurrent network with two long short-term memory (LSTM) layers as shown in Fig. 2c.

D. Maximizing Variational Lower-bound

To jointly learn parameters for the generative θ and the recognition network ϕ , our objective is to maximize likelihood of the data:

$$\max_{\theta} [\log p_{\theta}(\vec{o}, \vec{r}|\vec{a})]$$

Using a variational method, a lower bound on conditional log-likelihood is defined as:

$$\begin{aligned} \log p_{\theta}(\vec{o}, \vec{r}|\vec{a}) &= D_{KL}(q_{\phi}(\vec{s}|\vec{o}, \vec{r}, \vec{a})||p_{\theta}(\vec{s}|\vec{o})) + \mathcal{L}(\theta, \phi) \\ &\geq \mathcal{L}(\theta, \phi) \end{aligned}$$

Thus, to maximize $\max_{\theta} [\log p_{\theta}(\vec{o}, \vec{r}|\vec{a})]$, the lower bound $\mathcal{L}(\theta, \phi)$ can instead be maximized.

$$\begin{aligned} \mathcal{L}(\theta, \phi) &= -D_{KL}(q_{\phi}(\vec{s}|\vec{o}, \vec{r}, \vec{a})||p_{\theta}(\vec{s}|\vec{a})) \\ &\quad + \mathbb{E}_{q_{\phi}(\vec{s}|\vec{o}, \vec{r}, \vec{a})} [\log p_{\theta}(\vec{o}, \vec{r}|\vec{s}, \vec{a})] \end{aligned} \quad (1)$$

Using a reparameterization trick [33] twice, we arrive at following lower bound (refer to Appendix for full derivation): $\mathcal{L}(\theta, \phi) \approx -D_{KL}(q_{\phi}(s_1|\vec{o}, \vec{r}, \vec{a})||p(s_1))$

$$\begin{aligned} & - \frac{1}{L} \sum_{t=2}^T \sum_{l=1}^L [D_{KL}(q_{\phi}(s_t|s_{t-1}, \vec{o}, \vec{r}, \vec{a})||p(s_t|s_{t-1}^{(l)}, u_{t-1}))] \\ & + \frac{1}{L} \sum_{l=1}^L [\log p_{\theta}(\vec{o}|\vec{s}^{(l)}) + \log p_{\theta}(\vec{r}|\vec{s}^{(l)})] \\ & \text{where } \vec{s}^{(l)} = g_{\phi}(\epsilon^{(l)}, \vec{o}, \vec{r}, \vec{a}) \text{ and } \epsilon^{(l)} \sim p(\epsilon) \end{aligned} \quad (2)$$

Algorithm 1 Deep Q-Learning in Learned Latent State Space

```

 $D_{gt} = \{\}$  ▷ “ground-truth” transitions by  $q_{\phi}$ 
for all timestep  $t$  of  $(\vec{o}, \vec{a})$  in training data  $(i)$  do
     $s_t, s_{t+1} \leftarrow q_{\phi, \mu} + \epsilon q_{\phi, \Sigma}$  where  $\epsilon \sim p(\epsilon)$ 
     $D_{gt} \leftarrow D_{gt} \cup \langle s_t^{(i)}, a_{t+1}^{(i)}, r_{t+1}^{(i)}, s_{t+1}^{(i)} \rangle$ 
end for
loop
     $D_{explore} = \{\}$  ▷ explore with learned transition
    for all  $s_t^{(i)}$  in training data that succeeded do
         $a_{t+1} = \begin{cases} \text{rand}(a \in A) & \text{with prob. } \epsilon \\ \arg\max_{a \in A} Q(s_t^{(i)}, a) & \text{otherwise} \end{cases}$ 
         $r_{t+1} = \begin{cases} r_t^{(i)} & \text{if } a_{t+1} == a_{t+1}^{(i)} \\ -1 & \text{otherwise} \end{cases}$ 
         $s_{t+1} \leftarrow T(s_t^{(i)}, a_t)$ 
         $D_{explore} \leftarrow D_{explore} \cup \langle s_t^{(i)}, a_{t+1}, r_{t+1}, s_{t+1} \rangle$ 
    end for
     $D \leftarrow D_{gt} \cup D_{explore}$  ▷ update deep Q-network
    for all minibatch from  $D$  do
         $y_t \leftarrow r_t + \gamma \max_{a'} Q(s_{t+1}, a')$ 
        Take gradient with loss  $[y_t - Q(s_t, a_{t+1})]^2$ 
    end for
end loop

```

We jointly back-propagate on neural networks for both sets of encoder ϕ and decoder θ parameters with mini-batches to maximize the lower bound using AdaDelta [38].

E. Optimal Control in Learned Latent State Space

After learning a generative model for the POMDP and a recognition network using a variational Bayes method, we need an algorithm for making an optimal decision in learned representation of haptic feedback and action. We employ a reinforcement learning method, Q-Learning, which learns to approximate an optimal action-value function [37]. The algorithm computes a score for each state action pair:

$$Q : S \times A \rightarrow \mathbb{R}$$

The Q function is approximated by a two layer neural network as shown in Fig. 2d.

In a standard reinforcement learning setting, in each state s_t , an agent learns by exploring the selected action $\arg\max_{a \in A} Q(s_t, a)$ with a current Q function. However, doing so requires an ability to actually take or simulate the chosen action from s_t and observe r_{t+1} and s_{t+1} . However, there does not exist a good robotics simulation software that can simulate complex interactions between a robot and an object and generate different haptic signals. Thus, we cannot freely explore any states.

Instead, we first take all state transitions and rewards $\langle s_t^{(i)}, a_{t+1}^{(i)}, r_{t+1}^{(i)}, s_{t+1}^{(i)} \rangle$ from the i -th training data sequence and store in D_{gt} . Both $s_t^{(i)}$ and $s_{t+1}^{(i)}$ are computed by the recognition network q_{ϕ} with a reparameterization technique (similar to Sec. III-D).

At each iteration, we first have an exploration stage. For explorations, we start from states $s_t^{(i)}$ of training sequences that resulted in successful completion of the task and choose an action a_{t+1} with ϵ -greedy. With the learned transition function T (Sec. III-B), the selected action a_{t+1} is executed from $s_t^{(i)}$. However, since we are using a learned transition

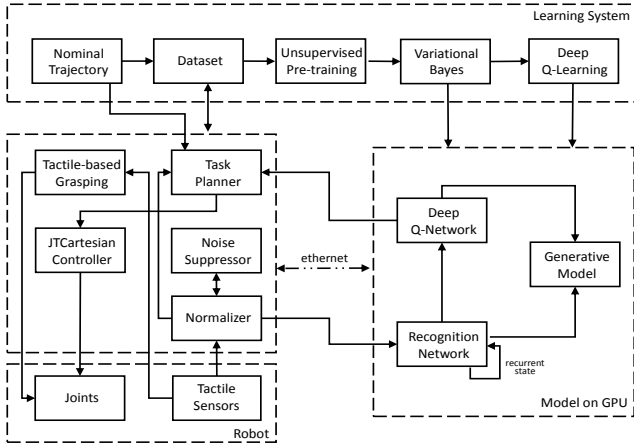


Fig. 4: **System Details** of our system for learning and robotic experiments.

function, any deviation from the distribution of training data could result in unexpected state, unlike explorations in a real or a simulated environment.

Thus, if the optimal action a_{t+1} using a current Q-function deviates from the ground-truth action $a_{t+1}^{(i)}$, the action is penalized with a negative reward to prevent deviations into unexplored states. If the optimal action is same as the ground-truth, the same reward as the original is given. For such cases, the only difference from the ground-truth would be in s_{t+1} , which is inferred by the learned transition function. All exploration steps are recorded in $D_{explore}$.

After the exploration step in each iteration, we take minibatches from $D = D_{gt} \cup D_{explore}$ and backpropagate on the deep Q-network with the loss function:

$$[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)]^2$$

The algorithm is summarized in Algorithm 1.

IV. SYSTEM DETAILS

Robotic Platform. All experiments were performed on a PR2 robot, a mobile robot with two 7 degree-of-freedom arms. Each two-fingered end-effector has an array of tactile sensors located at its tips. We used a Jacobian-transpose based JTCartesian controller [39] for controlling its arm during experiments.

For stable grasping, we take advantage of the tactile sensors to grasp an object. The gripper is slowly closed until certain thresholds are reached on both sides of the sensors, allowing the robot to easily adapt to objects of different sizes and shapes. To avoid saturating the tactile sensors, the robot does not grasp the object with maximal force.

Tactile Sensor. Each side of the fingertip of a PR2 robot is equipped with RoboTouch tactile sensor, an array of 22 tactile sensors covered by protective silicone rubber cover. The sensors are designed to detect range of 0 – 30 psi (0 – 205 kPa) with sensitivity of 0.1 psi (0.7 kPa) at the rate of 35 Hz.

We observed that each of the 44 sensors has a significant variation and noise in raw sensor readings with drifts over time. To handle such noise, values are first offset by starting values when interaction between an object and

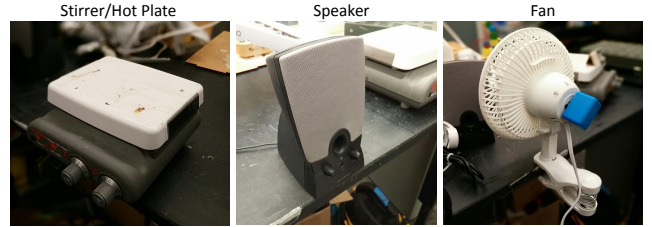


Fig. 5: A set of objects used for experiment. All three objects have different surface area and shape, which results in vastly different types of ‘clicks’ when observed via a tactile sensor.

the robot started (*i.e.* when a grasp occurred). Given the relative signals, we find a normalization value for each of 44 sensors such that none of the values goes above 0.05 when stationary and all data is clipped to the range of -1 and 1 . Normalization takes place by recording few seconds of sensor readings after grasping.

Learning Systems. For fast computation and executions, we offload all of our models onto a remote workstation with a GPU connected over a direct ethernet connection. Our models run on a graphics card using Theano [40], and our high level task planner sends a new goal location at the rate of 20 Hz. The overall system detail is shown in Figure 4.

V. EXPERIMENTS & RESULTS

In order to validate our approach, we perform a series of experiments on our dataset and on a PR2 robot.

A. Dataset

In order to test our algorithm that learns to represent haptic feedback, we collected a dataset of three different objects — a stirrer, a speaker, and a desk fan (Fig. 5) — each of which have a knob with a detent structure (an example CAD model shown in Fig. 1). Although these objects internally have some type of a detent structure that produce a feedback that humans would identify as a ‘click’, each ‘click’ from each object is very distinguishable. As shown in Fig. 3, different shapes of objects and the flat surface of the two fingers result in vastly differently tactile sensor readings.

In our model, for the haptic signals \vec{o} , we use a vector of normalized 44 tactile sensor array as described in Sec. IV. The reward \vec{r} is given as one of three classes at each time step, representing a positive, a negative and a neutral reward. For every object, action \vec{a} is an array of binary variables, each representing a phase in its nominal plan.

In more detail, the **stirrer** (hot plate) has a knob with a diameter of $22.7mm$ with a depth of $18.7mm$, which produces a haptic feedback that lasts about 30° rotations when it is turned on or off. Our robot starts from both the left (off state) and the right side (on state) of the knob. The **speaker** has a tiny cylindrical knob that decreases in its diameter from $13.1mm$ to $9.1mm$ with height of $12.8mm$ and requires 30° degree rotation. However, since PR2 fingertips are parallel plates and measure $23mm$ with silicon covers, grasping a $9.1mm$ knob results in drastically different sensor readings at every execution of the task. The **desk fan** has a rectangular knob with a width of $25.1mm$ and a large surface area. It has a two-step detent control

TABLE II: **Result of haptic signal prediction and robotic experiment.** The prediction experiment reports the average L2-norm from the haptic signal (44 signals in newtons) and the robotic experiment reports the success rate. It shows the results of more than 200 robotic experiments.

	Haptics Prediction			Robotic Experiment		
	0.05secs	0.25secs	0.50secs	Stirrer	Speaker	Desk Fan
<i>Chance</i>	6.68 (± 0.18)	6.68 (± 0.17)	6.69 (± 0.18)	31.6%	38.1%	28.5%
<i>Non-recurrent Recognition Network [4]</i>	1.39 (± 2.51)	5.03e5 ($\pm 5.27e7$)	3.23e7 ($\pm 1.07e10$)	52.9%	57.9%	62.5%
<i>Recurrent-network as Representation [14]</i>	0.33 (± 0.01)	1.01 (± 0.09)	1.76 (± 0.03)	63.2%	68.4%	70.0%
<i>Our Model without Exploration</i>	-	-	-	35.0%	33.3%	52.6%
<i>Our Model</i>	0.72 (± 0.08)	0.79 (± 0.09)	0.78 (± 0.10)	80.0%	73.3%	86.7%

with a click that lasts 45° degree rotation and has a narrow stoppable window of about $\pm 20^\circ$ degrees.

The stirrer and the speaker can both be rotated clockwise and counterclockwise and have a wall at both ends. The desk fan has three stoppable points (near 0°, 45°, and 90°) to adjust fan speed and can get stuck in-between if a rotation is not enough or exceeds a stopping point.

Each object is provided with a nominal plan with multiple phases, each defined as a sequence of smoothly interpolated waypoints consisting of end-effector position and orientation along with gripper actions of grasping similar to [6]. For each of the objects, we collected at least 25 successes and 25 failures. The success cases only includes rotations that resulted in successful transition of states of objects (e.g. from off to on state). The failures include slips, excessive rotations beyond acceptable range, rotation even after hitting a wall, and near breaking of the knob. There also exists less dramatic failures such as insufficient rotations. Especially for the desk fan, if a rotation results in two clicks beyond the first stopping point, it is considered a failure. Each data sequence consists of a sequence of trajectory (phases) as well as tactile sensor signal after an execution of each waypoint.

To label the reward for each sequence, an external camera with a microphone was placed nearby the object. By reviewing the audio and visually inspecting haptic signal afterwards, an expert labeled the timeframe that each sequence succeeded or failed. These extra recordings were only used for labeling the rewards, and such input is not made available to the robot during our experiments. For sequences that turned the knob past the successful stage but did not stop the rotation, only negative rewards were given.

Among multiple phases of a nominal plan, which includes pre-grasping and post-interaction trajectories, we focus on three phases (before-rotation/rotation/stopped). These phases occur after grasping and success is determined by ability to correctly rotate and detect when to shift to the final phase.

B. Baselines

We compare our model against several baseline methods on our dataset and in our robotic experiment. Since most of the related works are applied to problems in different domains, we take key ideas (or key structural differences) from relevant works and fit them to our problem.

- 1) *Chance*: It follows a nominal plan and makes a transition between phases by randomly selecting the amount of degree to rotate a knob without incorporating haptic feedback.
- 2) *Non-recurrent Recognition Network*: Similar to [4], we take non-recurrent deep neural network of only observations

without actions. However, it has access to a short history in a sliding window of haptic signal at every frame. For control, we apply the same Q-learning method as our full model.

3) *Recurrent Network as Representation*: Similar to [14], we directly train a recurrent network to predict future haptic signals. At each time step t , a LSTM network takes concatenated observation o_t and previous action a_t as input, and the output of the LSTM is concatenated with a_{t+1} to predict o_{t+1} . However, while [14] relies on hand-coded MPC cost function to choose an action, we apply same Q-learning that was applied to our full model. For haptic prediction experiment, transitions happen by taking the output of the next time step as input to the next observation.

4) *Our Model without Exploration*: During the final deep Q-Learning (Sec. III-E) stage, it skips the exploration step that uses a learned transition model and only uses sequences of representation from the recognition network.

C. Results and Discussion

To evaluate all models, we perform two types of experiments — haptic signal prediction and robotic experiment.

Haptic Signal Prediction. We first compare our model against baselines on a task of predicting future haptic signal. For all sequences that either eventually succeeded or failed, we take every timestep t , and predict timestep $t + 1$ (0.05secs), $t + 5$ (0.25secs) and $t + 10$ (0.5secs). The prediction is made by encoding (recognition network) a sequence up to time t and then transiting encoded states with a learned transition model to the future frames of interest. We take the L2-norm of the prediction of 44 sensor values (which are in newtons) and take the average of that result. The result is shown in the middle column of Table II.

Robotic Experiment. On a PR2 robot, we test the task of turning a knob until it clicks on three different objects: stirrer, speaker, and desk fan (Fig. 5). The right hand side of Table II shows the result of over 200 executions. Each algorithm was tested on each object at least 15 times.

Can it predict future haptic signals? When it predicts randomly (*chance*), regardless of the timestep, it has an average of 6.7. When the primary goal is to be able to perform the next haptic signal prediction, for one step prediction, recurrent-network as representation baseline performs best of 0.330 among all models, while ours performed 0.718. On the other hand, our model does not diverge and performs consistently well. After 0.5secs, when other models started to diverge to an error of 1.757 or much larger, our model still had prediction error of 0.782.

TABLE III: **Time difference** between the time the robot stopped and the time the expert indicated it ‘clicked’.

<i>Stirrer</i>	<i>Speaker</i>	<i>Desk Fan</i>
0.180 secs (± 0.616)	0.539 secs (± 1.473)	-0.405 secs (± 0.343)

What does learned representation represent? We visualize our learned embedding space of haptic feedback using t-SNE [41] in Fig. 6. Initially, both successful (blue paths) and unsuccessful (red paths) all starts from similar states but they quickly diverge into different clusters of paths much before they eventually arrive at states that were given positive or negative rewards shown as blue and red dots.

Does good representation lead to successful execution?

Our model allows robot to successfully execute on the three objects 80.0%, 73.3%, and 86.7% respectively, performing the highest compared to any other models. The next best model which uses recurrent network as representation performed at 63.2%, 68.4%, and 70.0%. However, note that this baseline still take advantage of our Q-learning method. Our model that did not take advantage of simulated exploration performed much poorly (35.0%, 33.3%, and 52.6%), showing that good representation combined with our Q-learning method leads to successful execution of the tasks.

Is recurrent network necessary for haptic signals? *Non-recurrent recognition network* quickly diverged to extremely large number of $3.2e7$ even though it successfully predicted 1.389 for a single step prediction. Note that it takes windowed haptic sequence of last 5 frames as input. Unlike images, short window of data does not hold enough information about haptic sequence which lasts much longer timeframe. For robotic experiment, non-recurrent network performed 52.9%, 57.9%, and 62.5% even with our Q learning method.

How accurately does it perform the task? When our full model was being tested on three objects, we also had one of the author observe (visually and audibly) very closely and press a button as soon as a click occurs. On successful execution of the task, we measure the time difference between the time the robot stops turning and the time the expert presses the key, and the results are shown in Table III.

The positive number represents that the model was delayed than the expert and the negative number represents that the model transitioned earlier. Our model only differed from human with an average of 0.37 seconds. All executions of tasks were performed at same translational and rotational velocity as the data collection process.

Note that just like a robot has a reaction time to act on perceived feedback, an expert has a reaction time to press the key. However, since the robot was relying on haptic feedback while the observer was using every possible human senses available including observation of the consequences without touch, some differences are expected. We also noticed that fan especially had a delay in visible consequences compared to the haptic feedback because robot was rotating these knobs slower than normal humans would turn in daily life; thus, the robot was able to react 0.4 seconds faster.

Video of robotic experiments are available at this website: http://jaeyongsung.com/haptic_feedback/

VI. CONCLUSION

In this work, we present a novel framework for learning to represent haptic feedback of an object that requires sense of touch. We model such tasks as partially observable model with its generative model parametrized by neural networks. To overcome intractability of computing posterior, variational Bayes method allows us to approximate posterior with a deep recurrent recognition network consisting of two LSTM layers. Using a learned representation of haptic feedback, we also introduce a Q-learning method that is able to learn optimal control without access to simulator in learned latent state space utilizing only prior experiences and learned generative model for transition. We evaluate our model on a task of rotating a knob until it clicks against several baseline. With more than 200 robotic experiments on the PR2 robot, we show that our model is able to successfully manipulate knobs that click while predicting future haptic signals.

APPENDIX

A. Lowerbound Derivation

To continue our derivation of the lower bound from Sec. III-D. The second term of equation 1:

$$\begin{aligned} \mathbb{E}_{q_\phi(\vec{s}|\vec{o}, \vec{r}, \vec{a})} [\log p_\theta(\vec{o}, \vec{r}|\vec{s}, \vec{a})] \\ &= \mathbb{E}_{q_\phi(\vec{s}|\vec{o}, \vec{r}, \vec{a})} [\log p_\theta(\vec{o}|\vec{s}) + \log p_\theta(\vec{r}|\vec{s})] \\ &\approx \frac{1}{L} \sum_{l=1}^L [\log p_\theta(\vec{o}|\vec{s}^{(l)}) + \log p_\theta(\vec{r}|\vec{s}^{(l)})] \\ &= \frac{1}{L} \sum_{l=1}^L \sum_{t=1}^T [\log p_\theta(o_t|s_t^{(l)}) + \log p_\theta(r_t|s_t^{(l)})] \\ &\quad \text{where } \vec{s}^{(l)} = q_{\phi, \mu} + \epsilon^{(l)} q_{\phi, \Sigma} \text{ and } \epsilon^{(l)} \sim p(\epsilon) \end{aligned}$$

Reparametrization trick ([33], [34]) at last step samples from the inferred distribution by a recognition network q_ϕ .

And, for the first term from equation 1:

$$\begin{aligned} D_{KL}(q_\phi(\vec{s}|\vec{o}, \vec{r}, \vec{a})||p_\theta(\vec{s}|\vec{a})) \\ &= \int_{s_1} \dots \int_{s_T} q_\phi(\vec{s}|\vec{o}, \vec{r}, \vec{a}) \left[\log \frac{q_\phi(\vec{s}|\vec{o}, \vec{r}, \vec{a})}{p_\theta(\vec{s}|\vec{a})} \right] \\ &= D_{KL}(q_\phi(s_1|\vec{o}, \vec{r}, \vec{a})||p(s_1)) \\ &\quad + \sum_{t=2}^T \mathbb{E}_{s_{t-1} \sim q_\phi(s_{t-1}|\vec{o}, \vec{r}, \vec{a})} [\\ &\quad \quad D_{KL}(q_\phi(s_t|s_{t-1}, \vec{o}, \vec{r}, \vec{a})||p(s_t|s_{t-1}, a_{t-1}))] \end{aligned}$$

using reparameterization trick again,

$$\begin{aligned} &= D_{KL}(q_\phi(s_1|\vec{o}, \vec{r}, \vec{a})||p(s_1)) \\ &\quad + \sum_{t=2}^T \frac{1}{L} \sum_{l=1}^L [D_{KL}(q_\phi(s_t|s_{t-1}, \vec{o}, \vec{r}, \vec{a})||p(s_t|s_{t-1}^{(l)}, a_{t-1}))] \\ &\quad \text{where } s_{t-1}^{(l)} = q_{\phi, t-1, \mu} + \epsilon^{(l)} q_{\phi, t-1, \Sigma} \text{ and } \epsilon^{(l)} \sim p(\epsilon) \end{aligned}$$

Combining these two terms, we arrive at equation 2.

We do not explain each step of the derivation at length since similar ideas behind the derivation can be found at [35] although exact definition and formulation are different.

Acknowledgment. We thank Ian Lenz for useful discussions. This work was supported by Microsoft Faculty Fellowship and NSF Career Award to Saxena.

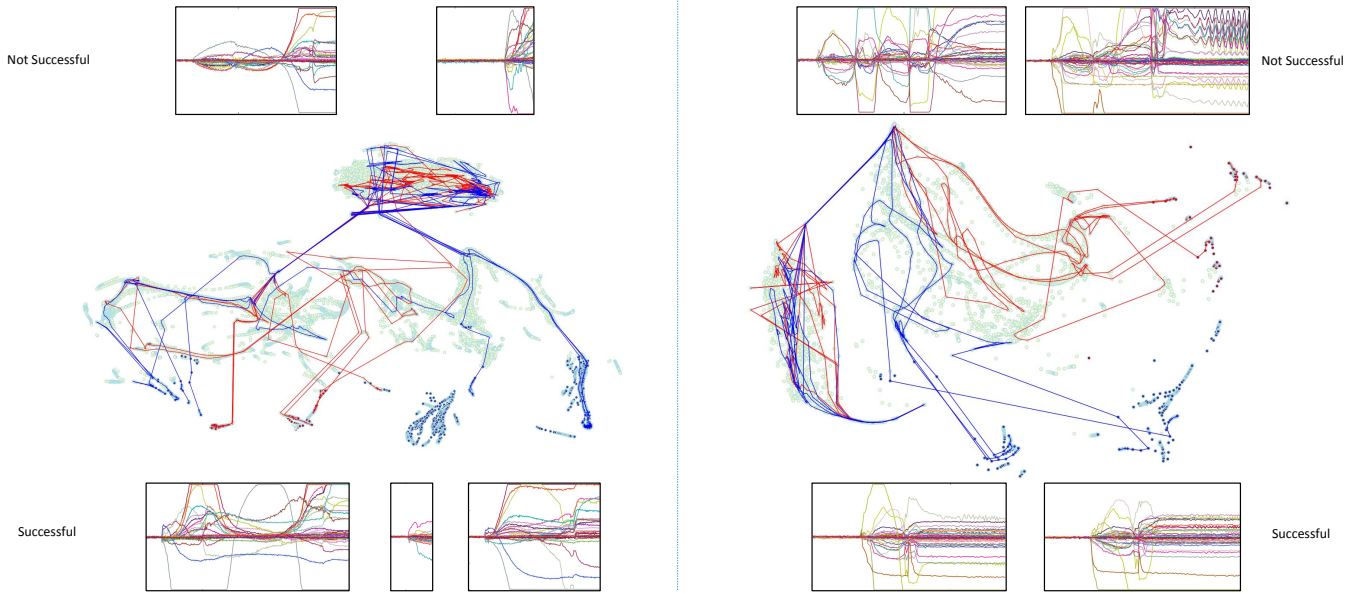


Fig. 6: **Projection of learned representation** of haptic feedback using t-SNE [41] for ‘stirrer’ and ‘fan’. Each dot represents an inferred state at each time frame, and blue and red dots represents positive and negative reward at those time frame. Here we show some of successful (blue) and unsuccessful (red) sequences. For both objects, notice both classes initially starts from similar state and then diverges, forming clusters. Several successful and unsuccessful haptic signals are shown as well.

REFERENCES

- [1] A. Montagu, “Touching: The human significance of the skin.” 1971.
- [2] D. A. Norman, *The design of everyday things*, 1988.
- [3] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” *arXiv preprint arXiv:1504.00702*, 2015.
- [4] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, “Embed to control: A locally linear latent dynamics model for control from raw images,” in *NIPS*, 2015.
- [5] V. Mnih, *et al.*, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.
- [6] J. Sung, S. H. Jin, and A. Saxena, “Robobarista: Object part-based transfer of manipulation trajectories from crowd-sourcing in 3d point-clouds,” in *ISRR*, 2015.
- [7] D. Silver, *et al.*, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [8] M. C. Gemic and A. Saxena, “Learning haptic representation for manipulating deformable food objects,” in *IROS*. IEEE, 2014.
- [9] V. Chu, *et al.*, “Using robotic exploratory procedures to learn the meaning of haptic adjectives,” in *ICRA*, 2013.
- [10] S. Bennett, “A brief history of automatic control,” *IEEE Control Systems Magazine*, vol. 16, no. 3, pp. 17–25, 1996.
- [11] J. K. Salisbury, “Active stiffness control of a manipulator in cartesian coordinates,” in *Decision and Control*. IEEE, 1980.
- [12] J. Barry, K. Hsiao, L. Kaelbling, and T. Lozano-Pérez, “Manipulation with multiple action types,” in *ISER*, no. 1122374, 2012.
- [13] S. Trimpe, A. Millane, S. Doesseger, and R. DAndrea, “A self-tuning lqr approach demonstrated on an inverted pendulum,” in *IFAC World Congress*, 2014, p. 11.
- [14] I. Lenz, R. Knepper, and A. Saxena, “Deepmpc: Learning deep latent features for model predictive control,” in *RSS*, 2015.
- [15] F. Chaumette and S. Hutchinson, “Visual servo control. i. basic approaches,” *Robotics & Automation Magazine, IEEE*, vol. 13, no. 4, pp. 82–90, 2006.
- [16] J. Park and O. Khatib, “A haptic teleoperation approach based on contact force control,” *IJRR*, vol. 25, no. 5-6, pp. 575–591, 2006.
- [17] Q. Li, C. Schürmann, R. Haschke, and H. J. Ritter, “A control framework for tactile servoing,” in *R:SS*, 2013.
- [18] P. Pastor, M. Kalakrishnan, S. Chitta, *et al.*, “Skill learning and task outcome prediction for manipulation,” in *ICRA*, 2011.
- [19] S. Thrun, W. Burgard, D. Fox, *et al.*, *Probabilistic robotics*. MIT press Cambridge, 2005.
- [20] K. Hsiao, L. P. Kaelbling, and T. Lozano-Perez, “Grasping pomdps,” in *ICRA*, 2007.
- [21] H. Kurniawati *et al.*, “Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces,” in *RSS*, 2008.
- [22] N. A. Vien and M. Toussaint, “Touch based pomdp manipulation via sequential submodular optimization,” in *Humanoids*, 2015.
- [23] B. Sallans, “Learning factored representations for partially observable markov decision processes,” in *NIPS*. Citeseer, 1999, pp. 1050–1056.
- [24] G. Contardo, L. Denoyer, T. Artieres, and P. Gallinari, “Learning states representations in pomdp,” in *ICLR*, 2014.
- [25] V. Vapnik and A. Vashist, “A new learning paradigm: Learning using privileged information,” *Neural Networks*, 2009.
- [26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012.
- [27] A. Hannun, *et al.*, “Deep speech: Scaling up end-to-end speech recognition,” *arXiv preprint arXiv:1412.5567*, 2014.
- [28] Y. Gao, L. A. Hendricks, *et al.*, “Deep learning for tactile understanding from visual and haptic data,” *arXiv preprint 1511.06065*, 2015.
- [29] R. Hadsell, *et al.*, “Deep belief net learning in a long-range vision system for autonomous off-road driving,” in *IROS*, 2008.
- [30] L. S. Abhinav Valada and W. Burgard, “Deep feature learning for acoustics-based terrain classification,” in *ISRR*, 2015.
- [31] N. Wahlström, T. B. Schön, and M. P. Deisenroth, “From pixels to torques: Policy learning with deep dynamical models,” *arXiv preprint arXiv:1502.02251*, 2015.
- [32] C. Finn, *et al.*, “Deep spatial autoencoders for visuomotor learning,” in *ICRA*, 2016.
- [33] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.
- [34] D. J. Rezende, S. Mohamed, and D. Wierstra, “Stochastic backpropagation and approximate inference in deep generative models,” *arXiv preprint arXiv:1401.4082*, 2014.
- [35] R. G. Krishnan, U. Shalit, and D. Sontag, “Deep kalman filters,” *arXiv preprint arXiv:1511.05121*, 2015.
- [36] M. Hausknecht and P. Stone, “Deep recurrent q-learning for partially observable mdps,” *arXiv preprint arXiv:1507.06527*, 2015.
- [37] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.
- [38] M. D. Zeiler, “Adadelta: An adaptive learning rate method,” *arXiv preprint arXiv:1212.5701*, 2012.
- [39] “Jt cartesian controller.” [Online]. Available: http://wiki.ros.org/robot/_mechanism/_controllers/JTCartesian/%20Controller
- [40] F. Bastien, *et al.*, “Theano: new features and speed improvements,” NIPS DLUF Workshop, 2012.
- [41] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne,” *JMLR*, vol. 9, no. 2579–2605, p. 85, 2008.