

Learning Dexterous Manipulation for a Soft Robotic Hand from Human Demonstrations

Abhishek Gupta¹

Clemens Eppner²

Sergey Levine¹

Pieter Abbeel¹

Abstract—Dexterous multi-fingered hands can accomplish fine manipulation behaviors that are infeasible with simple robotic grippers. However, sophisticated multi-fingered hands are often expensive and fragile. Low-cost soft hands offer an appealing alternative to more conventional devices, but present considerable challenges in sensing and actuation, making them difficult to apply to more complex manipulation tasks. In this paper, we describe an approach to learning from demonstration that can be used to train soft robotic hands to perform dexterous manipulation tasks. Our method uses object-centric demonstrations, where a human demonstrates the desired motion of manipulated objects with their own hands, and the robot autonomously learns to imitate these demonstrations using reinforcement learning. We propose a novel algorithm that allows us to blend and select a subset of the most feasible demonstrations, which we use with an extension of the guided policy search framework that learns generalizable neural network policies. We demonstrate our approach on the RBO Hand 2, with learned motor skills for turning a valve, manipulating an abacus, and grasping.

I. INTRODUCTION

Control of multi-fingered hands for fine manipulation skills is exceedingly difficult, due to the complex dynamics of the hand, the challenges of non-prehensile manipulation, and under-actuation. Furthermore, the mechanical design of multi-finger hands tends to be complex and delicate. Although a number of different hand designs have been proposed in the past [1], [2], many of these hands are expensive and fragile.

In this work, we address the problem of autonomously learning dexterous manipulation skills with an inexpensive and highly compliant multi-fingered hand — the RBO Hand 2 [3]. This hand (see Fig. 1) is actuated by inflating air-filled chambers. We cannot accurately control the kinematic finger motion, but can only actuate the hand by inflating or deflating the air chambers. Lack of sensing and precise actuation makes standard control methods difficult to apply directly to devices like this. Instead, we use an approach based on learning from demonstrations (LfD) and reinforcement learning (RL).

In LfD, the robot observes a human teacher solving a task and learns how to perform the demonstrated task and

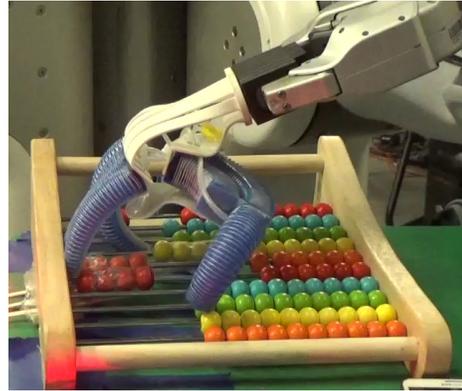


Fig. 1. The RBO Hand 2 manipulating an abacus.

apply it to new situations. Demonstrations are typically given visually, by kinesthetic teaching, or through teleoperation. However, these techniques are difficult in case of the RBO Hand 2. A demonstrator cannot manually move all of the fingers of the hand for kinesthetic teaching, and the hand lacks position sensing to store such demonstrations. Even direct teleoperation via intuitive interfaces, such as gloves or motion capture, is non-trivial, because although the RBO Hand 2 is anthropomorphic in design, its degrees of freedom do not match those of the human hand well enough to enable direct mapping of human hand motion.

However, the goal of most dexterous manipulation is to manipulate the poses of objects in the world, and a task can often be fully defined by demonstrating the motion of these objects. These demonstrations can be provided by putting trackers on the objects being manipulated and using a human demonstrator to physically move the objects along the desired trajectories. The object-centric demonstrations consist of just the trajectories of the object trackers, without any other states or actions. These kinds of “object-centric” demonstrations are intuitive and easy to provide, but because the robot does not directly control the degrees of freedom of moving objects in the world, they cannot be imitated directly. Instead, we use reinforcement learning to construct controllers that reproduce object-centric demonstrations.

One crucial challenge that we must address to utilize object-centric demonstrations is to account for the mismatch between the morphology of the human expert and the robot. Since the robot cannot always reproduce all object-centric demonstrations, we propose a novel algorithm that automatically selects and blends those demonstrations that the robot can follow most closely, while ignoring irrelevant demonstrations that cannot be reproduced.

¹Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, CA, USA. This research was funded in part by ONR through a Young Investigator Program award and by the Berkeley Vision and Learning Center (BVLC). {abhigupta, svlevine, pabbeel}@berkeley.edu

²Robotics and Biology Laboratory, Technische Universität Berlin, Germany. The author gratefully acknowledges financial support by the European Commission (SOMA, H2020-ICT-645599). clemens.eppner@tu-berlin.de

Individual controllers may start from different initial conditions, such as different relative configurations of the hand and manipulated object, and thus may only be able to realize some subset of the demonstrated object-centric trajectories. Furthermore, some demonstrations may be difficult to achieve even from the same initial conditions as the ones they were demonstrated in, due to morphological differences between the robot and the human providing demonstrations, so the only way to determine if a demonstration is achievable is to attempt to imitate it. Our algorithm automatically determines which of the demonstrations are actually feasible to achieve from each initial state by alternating between reinforcement learning to learn the controllers and correspondence assignment to choose which demonstrations to follow from each state.

Our goal is to find a unified control policy that can generalize to a variety of initial states. To achieve generalization, we train a single nonlinear neural network policy to reproduce the behavior of multiple object-centric demonstrations. This approach follows the framework of guided policy search (GPS [4]), where multiple local controllers are unified into a single high-dimensional policy. This method is chosen because of its effectiveness in learning high dimensional control policies for real robots using a very small number of samples. However, unlike standard GPS, our approach requires only a set of object-centric demonstrations from a human expert to learn a new skill, rather than hand-specified cost functions.

The contributions of this paper are:

- 1) We propose a novel algorithm for learning from object-centric demonstrations. This algorithm enables complex dexterous manipulators to learn from multiple human demonstrations, selecting the most suitable demonstration to imitate for each initial state during training. The algorithm alternates between softly assigning demonstrations to individual controllers, and optimizing those controllers with an efficient trajectory-centric reinforcement learning algorithm.
- 2) We demonstrate that a single generalizable policy can be learned from this collection of controllers by extending the guided policy search algorithm to learning from demonstrations.
- 3) We evaluate our approach by learning a variety of dexterous manipulation skills with the RBO Hand 2, showing that our method can effectively acquire complex behaviors for soft robots with limited sensing and challenging actuation mechanisms.

II. RELATED WORK

A. Dexterous Manipulation using Planning

A variety of methods for generating manipulation behaviors with multi-fingered hands are based on planning. These approaches assume that a detailed model of the hand and object is available a priori. They generate open-loop trajectories that can be executed on real hardware. There exist planners that integrate contact kinematics, non-holonomic motion

constraints, and grasp stability to come up with manipulation plans based on finger gaits [5], rolling and sliding fingertip motions [6], or nonprehensile actions involving only the palm [7]. Optimization-based techniques [8] have also been used for in-hand manipulation tasks. All of these approaches rely on detailed models, or make simplifying assumptions about the system. Modelling and simulating the behavior of a soft hand like the RBO Hand 2 is computationally expensive, since it requires finite-element method models [9] to achieve accuracy. Moreover, it is extremely hard to do accurate system identification on such systems. In order to tackle this problem, our approach does not rely on detailed a priori models but learns the task-specific consequences of actions from interactions of the real hardware with the environment, during a task.

B. Reinforcement Learning for Manipulation

In order to avoid planning with fixed handcrafted models, control policies that solve continuous manipulation problems can be found using reinforcement learning. A widely used approach is to learn the parameters of a dynamic motor primitive [10] (DMP) with relative entropy policy search [11] or PI2 [12]. This has been used to learn striking movements [13] and bi-manual transportation tasks [14]. Although DMPs are often used to describe the kinematic state of a system, they can be used to generate compliant behavior for picking up small objects or opening doors [15]. However, DMPs typically require either a model of the system or the ability to control kinematic state, neither of which is straightforward on a soft hand that lacks position sensing.

Controllers for reaching and grasping have been learned by approximating the Q-function with a multilayer perceptron [16]. Policy search methods have succeeded in training neural network controllers to solve contact-rich peg-in-hole-like tasks [17] based on positional or visual feedback [18].

Some RL methods for manipulation have been applied to in-hand manipulation. Van Hoof et al. [19] learn a policy based on tactile feedback which lets an under-actuated hand slide cylindrical objects horizontally while being rolled between two fingers. Similar to our work is the learning method for an in-hand rotation tasks by Kumar et al. [20]. In contrast, we learn global policies that aim to generalize local solutions.

C. Exploiting Human Demonstrations for Learning Manipulation Skills

Learning from demonstrations has been effective in teaching robots to perform manipulation tasks with a limited amount of human supervision. By building statistical models of human demonstrations, gestures [21] and dual-arm manipulations [22] have been reproduced on robotic systems. Pure LfD can lead to suboptimal behavior when demonstrator and imitator do not share the same embodiment. To circumvent this problem the learning objective is often extended with additional feedback. This can be provided by a human, e.g. in the case of iteratively improving grasp adaptation [23]. Alternatively, demonstrations can provide the coarse structure of a solution, while the details are iteratively refined and learned

by the imitator itself. This has been shown for dexterous manipulation [24] where an in-hand manipulation is broken down into a sequence of canonical grasps. In combination with reinforcement learning, demonstrations often serve as an initial policy rollout or they constrain the search space by providing building blocks. This has been applied to reaching motions [25] and dynamic manipulation tasks.

III. ALGORITHM OVERVIEW

To find manipulation strategies for the RBO Hand 2 that solve different manipulation tasks, we take advantage of two general concepts: imitating human demonstrations and reinforcement learning. In order to learn from human demonstrations, we exploit task-specific information offered by human demonstrators using object-centric demonstrations, i.e. we only capture the motion of the object being manipulated, not hand-specific information. We use reinforcement learning to learn a policy which imitates these object centric demonstrations. However, due to kinematic and dynamic differences between the human hand and the RBO Hand 2, following some of these demonstrations might not be possible, and hence trying to imitate them closely is undesirable. We describe a novel demonstration selection algorithm that selects *which* demonstration should be imitated, and use a reinforcement learning method to solve the problem of *how* to imitate.

We define our learning problem as optimizing a policy π_θ to perform the demonstrated task by learning from demonstrations. In order to learn this policy, we first train multiple different local controllers to imitate the most closely achievable demonstration from their respective initial states. This involves solving the joint problem of selecting the appropriate demonstration for each controller, and using reinforcement learning to train each controller to actually follow its chosen demonstration. By modeling the objective as a minimization of KL divergence between a distribution of controllers and a mixture of demonstrations modeled as Gaussians, as shown in Section IV, this joint problem reduces to an alternating optimization between computing correspondence weights assigning a demonstration to each controller, and optimizing each controller using an optimal control algorithm. This algorithm can be used within the BADMM-based guided policy search framework [18], to train a neural network policy π_θ to generalize over the learned controllers. We propose a novel learning from demonstrations algorithm based on the GPS framework, which consists of three phases described in sections IV, V, and VI:

- 1) Perform a weight assignment which computes soft correspondences between demonstrations and individual controllers (Sec. IV).
- 2) With the soft correspondences fixed, solve an optimal control problem based on the correspondences and deviations from individual demonstrations (Sec. V).
- 3) Perform supervised learning over the trajectory distributions from the optimal control phase, using the framework of BADMM-based GPS (Sec. VI).

Algorithm 1 Guided policy search with demonstration selection

- 1: **for** iteration $k = 1$ to K **do**
 - 2: Generate samples $\{\bar{\tau}_j\}$ from each controller $p_j(\bar{\tau})$ by running it on the soft hand.
 - 3: Compute soft correspondence weights a_{ij}
 - 4: Estimate system dynamics $p(x_{t+1}|x_t, u_t)$ from $\{\tau_j\}$
 - 5: **for** iteration $inner = 1$ to n **do**
 - 6: Perform optimal control to optimize objective defined in Section IV
 - 7: Perform supervised learning to match π_θ with the samples $\{\bar{\tau}_j\}$
 - 8: **return** θ ▷ the optimized policy parameters
-

IV. LEARNING CONTROLLERS FROM MULTIPLE DEMONSTRATIONS

As the first step to generalizing dexterous manipulation skills, we learn a collection of controllers starting from different initial conditions, such that each controller imitates the demonstration which is most closely achievable from its initial condition. This problem can be cast as minimizing the divergence between two distributions: one corresponding to the demonstrated trajectories, and one to the controllers.

For our given dynamical system, we define the states to be x_t , and the actions to be u_t at every time step t . The system dynamics are specified by the model $p(x_{t+1}|x_t, u_t)$. Each controller j is defined in terms of a conditional distribution $p_j(u_t|x_t)$, which along with the dynamics model $p(x_{t+1}|x_t, u_t)$ induces a distribution $p_j(\tau) = p_j(x_0) \prod p(x_{t+1}|x_t, u_t) p_j(u_t|x_t)$ over trajectories $\tau = x_1, u_1, \dots, x_T, u_T$, where T is the length of an episode. We define $p(\tau) = \sum_{j=1}^C \frac{1}{C} p_j(\tau)$ to be the uniform mixture of C controllers $p_j(\tau)$.

Our state x_t can be expressed as $x_t = [\bar{x}_t, x'_t]$, where \bar{x}_t denotes the "object-centric" parts tracking the manipulated objects and x'_t is the rest of the state. In our experimental setting, \bar{x}_t consists of positions and velocities of motion capture markers placed on manipulated objects.

As we are using object-centric demonstrations, our objective is to match our controllers with the demonstrations but only over the object centric elements (\bar{x}) of the state. For each controller $p(\tau)$, we can marginalize to obtain $p(\bar{\tau})$, which is a uniform mixture of C distributions $p_j(\bar{\tau})$, such that $p(\bar{\tau}) = \sum_{j=1}^C w_j p_j(\bar{\tau})$ where $w_j = \frac{1}{C}$ and $\bar{\tau} = \{\bar{x}_1, \bar{x}_2, \dots, \bar{x}_T\}$. This distribution is over just the object-centric trajectories $\bar{\tau}$.

The distribution of D demonstrations over the trajectories $\bar{\tau}$ is also modeled as a mixture, given by $d(\bar{\tau}) = \sum_i^D v_i d_i(\bar{\tau})$. Each $d_i(\bar{\tau})$ is defined as a multivariate Gaussian, constructed according to $d_i(\bar{\tau}) = \mathcal{N}(\mu_i, \Sigma_i)$, where $\mu_i = \{\bar{x}_1, \bar{x}_2, \dots, \bar{x}_T\}$ is the trajectory of the objects recorded in each demonstration, and the covariance Σ_i is a parameter that decides how closely the demonstration needs to be tracked by the controller. The number of demonstrations and controllers do not have to be the same.

Our goal is to match the distribution of demonstrations with the distribution of controllers, which we formalize as a KL divergence objective: $\min_{p(\bar{\tau})} D_{KL}(p(\bar{\tau}) || d(\bar{\tau}))$. Although the objective is defined with respect to the object-centric

distributions $p(\bar{\tau})$, the optimization is done with respect to the entire controller mixture $p(\tau)$ which includes other parts of the state, and actions.

Due to the mode seeking behavior of the KL divergence, this objective encourages each $p_j(\bar{\tau})$ to match the closest achievable demonstration. However, the KL divergence between mixtures cannot be evaluated analytically. Methods such as MCMC sampling can be used to estimate it, but we find a variational upper bound [26] to be the most suitable for our formulation. In order to simplify our objective, we decompose each mixture weight w_j and v_i into individual variational parameters a_{ij} and b_{ij} , such that $\sum_i a_{ij} = w_j$ and $\sum_j b_{ij} = v_i$. We can rewrite

$$\begin{aligned} D_{KL}(p(\bar{\tau})||d(\bar{\tau})) &= \int p(\bar{\tau}) \log \frac{p(\bar{\tau})}{d(\bar{\tau})} \\ &= \int -p(\bar{\tau}) \log \frac{\sum_{i,j} b_{ij} d_i(\bar{\tau})}{p(\bar{\tau})} \\ &= - \int p(\bar{\tau}) \log \sum_{i,j} \frac{b_{ij} d_i(\bar{\tau}) a_{ij} p_j(\bar{\tau})}{a_{ij} p_j(\bar{\tau}) p(\bar{\tau})}. \end{aligned}$$

From Jensen's inequality we get an upper bound as follows:

$$\begin{aligned} D_{KL}(p(\bar{\tau})||d(\bar{\tau})) &\leq - \int p(\bar{\tau}) \sum_{i,j} \frac{a_{ij} p_j(\bar{\tau})}{p(\bar{\tau})} \log \frac{b_{ij} d_i(\bar{\tau})}{a_{ij} p_j(\bar{\tau})} \\ &= - \sum_{i,j} \int p_j(\bar{\tau}) a_{ij} \log \frac{b_{ij} d_i(\bar{\tau})}{a_{ij} p_j(\bar{\tau})} \\ &= \left[\sum_{i,j} a_{ij} \int p_j(\bar{\tau}) \log \frac{p_j(\bar{\tau})}{d_i(\bar{\tau})} \right] - \left[\sum_{i,j} a_{ij} \log \frac{b_{ij}}{a_{ij}} \right] \\ &= \sum_{i,j} a_{ij} D_{KL}(p_j(\bar{\tau})||d_i(\bar{\tau})) + D_{KL}(a||b) \end{aligned}$$

Thus, our optimization problem becomes

$$\min_{p(\tau), a, b} \sum_{i,j} a_{ij} D_{KL}(p_j(\bar{\tau})||d_i(\bar{\tau})) + D_{KL}(a||b). \quad (1)$$

While the first term $\sum_{i,j} a_{ij} D_{KL}(p_j(\bar{\tau})||d_i(\bar{\tau}))$ depends on the distribution $p(\tau)$, the second term $D_{KL}(a||b)$ depends on the mixing components a_{ij} and b_{ij} but is independent of the distribution $p(\tau)$.

We can perform the optimization in two alternating steps, where we first optimize $D_{KL}(p(\bar{\tau})||d(\bar{\tau}))$ with respect to a, b , followed by an optimization of $D_{KL}(p(\bar{\tau})||d(\bar{\tau}))$ with respect to $p(\tau)$, giving us a block coordinate descent algorithm in $\{a, b\}$ and p . The convergence of the algorithm is guaranteed by the convergence of a block coordinate descent method on a quasiconvex function and the fact that KL divergence is quasiconvex. The convergence properties of the weight assignment phase is shown in [26].

Intuitively, the first optimization with respect to a, b is a weight assignment with the correspondence weight a_{ij} representing the probability of assigning demonstration i to controller j . The second optimization with respect to $p(\tau)$, keeps the correspondence parameters a, b fixed, and finds optimal controllers using an optimal control algorithm to minimize a weighted objective specified in Eq. 3.

1) *Weight assignment phase:* The objective function $D_{KL}(p(\bar{\tau})||d(\bar{\tau}))$ is convex in both a and b , so we can optimize it by keeping one variable fixed while optimizing the other, and vice versa. We refer the reader to [26] for further details on this optimization. This yields the following closed form solutions:

$$b_{ij} = \frac{v_i a_{ij}}{\sum_{j'} a_{ij'}} \quad \text{and} \quad a_{ij} = \frac{w_j b_{ij} e^{-D_{KL}(p_j(\bar{\tau})||d_i(\bar{\tau}))}}{\sum_{i'} b_{i'j} e^{-D_{KL}(p_j(\bar{\tau})||d_{i'}(\bar{\tau}))}}.$$

In order to compute the optimal a and b , we alternate between these updates for a and b until convergence.

2) *Controller optimization phase:* Once the optimal values for a and b have been computed, we fix these as correspondences between demonstrations and controllers and optimize Eq. 1 to recover the optimal $p(\tau)$. As a and b are fixed, $D_{KL}(a||b)$ is independent of p . Hence, our optimization becomes:

$$\begin{aligned} &\min_{p(\tau)} \sum_{i,j} a_{ij} D_{KL}(p_j(\bar{\tau})||d_i(\bar{\tau})) \\ &= \sum_{i,j} a_{ij} \left(-E_{p_j(\bar{\tau})} [\log d_i(\bar{\tau})] - H(p_j(\bar{\tau})) \right) \\ &= \sum_j -w_j H(p_j(\bar{\tau})) - \sum_{i,j} a_{ij} E_{p_j(\bar{\tau})} [\log d_i(\bar{\tau})]. \end{aligned}$$

Factorizing the optimization to be independently over each of the controller distributions, for each controller $p_j(\tau)$, we optimize the objective:

$$-w_j H(p_j(\bar{\tau})) - \sum_i a_{ij} E_{p_j(\bar{\tau})} [\log d_i(\bar{\tau})] \quad (2)$$

$$= -w_j \left(H(p_j(\bar{\tau})) + \sum_i \frac{a_{ij}}{w_j} E_{p_j(\bar{\tau})} [\log d_i(\bar{\tau})] \right) \quad (3)$$

In practice, the weight assignment is performed independently per time step, as the controllers we consider are time varying.

V. CONTROLLER OPTIMIZATION WITH AN LFD OBJECTIVE

While the controller optimization phase could be performed with a variety of optimal control and reinforcement learning methods, we choose a simple trajectory-centric reinforcement learning algorithm that allows us to control systems with unknown dynamics, such as soft hands. Building on prior work, we learn time-varying linear Gaussian controllers by using iteratively refitted time-varying local linear models [4]. This is predicated on the assumption that the system has Gaussian noise, which has been shown to work well in practice for robotic systems [4]. The derivation follows prior work, but is presented here for the specific case of our LfD objective. Action-conditionals for the time-varying linear-Gaussian controllers are given by

$$p_j(u_t|x_t) = \mathcal{N}(K_{jt}x_t + k_{jt}, C_{jt})$$

where K_{jt} is a feedback term and k_{jt} is an open loop term. Given this form, the maximum entropy objective (Eq. 3), can be optimized using differential dynamic programming [27],

[28]. As $d_i(\bar{\tau})$ is a multivariate Gaussian $\mathcal{N}(\mu_i, \Sigma_i)$, we can rewrite the optimization problem in Eq. 3 as

$$\min_{p_j(\bar{\tau})} \sum_{t,i} \frac{a_{ijt}}{\sum_{i'} a_{i'jt}} E_{\bar{x}_t \sim p_j(\bar{\tau})} \left[\frac{1}{2} (\bar{x}_t - \mu_{it})^T \Sigma_i^{-1} (\bar{x}_t - \mu_{it}) \right] - H(p_j(\bar{\tau}))$$

where we express the objective as a sum over individual time steps. In this maximum entropy objective, the cost function defined as the expectation of a sum of l_2 distances of trajectory samples to each demonstration, weighted by the normalized correspondence weights $\frac{a_{ij}}{\sum_{i'} a_{i'j}}$. The trajectory samples denote the trajectories of the objects which we recover through object markers, and we compute the l_2 distance of these samples to the object-centric demonstrations. Under linearized dynamics, this objective can be locally optimized using LQG [29]. However, for robots like the RBO Hand 2, the dynamics are complex and difficult to model analytically. Instead, we can fit a time-varying locally linear model of the dynamics, of the form $p(x_{t+1}|x_t, u_t) = \mathcal{N}(f_{xt}x_t + f_{ut}u_t|C_d)$, to samples obtained by running the physical system in the real world. The dynamics matrices f_{xt} and f_{ut} can then be used in place of the system linearization to optimize the controller objective using LQG (for details see [29], [4]). Important to note here is that the iLQG optimization learns K_{jt} , k_{jt} and C_{jt} for the trajectory controller, while the term $\frac{a_{ij}}{\sum_{i'} a_{i'j}}$ is learned in the weight assignment phase and kept fixed for the iLQG optimization.

One issue with optimizing a controller using fitted local dynamics models is that the model is only valid close to the previous controller. The new controller might visit very different states where the fitted dynamics are no longer valid, potentially causing the algorithm to diverge. To avoid this, we bound the maximum amount the controller changes between iterations. This can be expressed as an additional constraint on the optimization:

$$D_{KL}(p_j(\tau) || \hat{p}_j(\tau)) < \varepsilon, \quad (4)$$

where $\hat{p}_j(\tau)$ is the previous trajectory-controller and $p_j(\tau)$ the new one. As shown in [18], this constrained optimization problem can be formulated in the same maximum entropy form as Eq. 3, using Lagrange multipliers, and solved via dual gradient descent (for details and a full derivation see [4], [18]). In practice, each iteration of this controller optimization algorithm involves generating N samples on the real physical system by running the previous controller, fitting a time-varying linear dynamics model to these samples as in previous work [4], and optimizing a new controller $p_j(\tau)$ by solving the constrained optimization using dual gradient descent, with LQG used to optimize with respect to the primal variables K_{jt} , k_{jt} , and C_{jt} . This can be viewed as an instance of model-based reinforcement learning.

VI. SUPERVISED LEARNING USING GPS

The multiple controllers defined in the previous section learn to imitate the most closely imitable demonstration from their individual starting positions. However, given an unseen initial state, it is not clear which controller $p_j(\tau)$ should be



Fig. 2. The RBO Hand 2 is an anthropomorphic pneumatically actuated soft hand consisting of seven actuators. Three of them form the palm and thumb. The air chambers can be physically coupled or actuated separately.

used. For effective generalization, we need to obtain a single policy $\pi_\theta(u_t|x_t)$ that will succeed under various conditions. To do this, we extend the framework of GPS [18] to combine controllers into a single nonlinear neural network policy.

We learn the parameters θ of a neural network π_θ to match the behavior shown by the individual controllers by regressing from the state x_t to the actions u_t taken by the controllers at each of the N samples generated on the physical system. Simply using supervised learning is not in general guaranteed to produce a policy with good long-horizon performance. In fact, supervised learning is effective only when the state distribution of π_θ matches that of the controllers $p_j(\tau)$. To ensure this, we use the BADMM-based variant of GPS [18], which modifies the cost function for the controllers to include a KL-divergence term to penalize deviation of the controllers from the latest policy π_θ at each iteration. This is illustrated in Algorithm (1), by first assigning correspondences between demonstrations and controllers, then alternating between trajectory optimization and supervised learning at every iteration, eventually leading to a good neural network policy π_θ . For further details on the guided policy search algorithm, we refer the reader to prior work [18].

VII. RBO HAND 2 AND SYSTEM SETUP

The RBO Hand 2 is an inexpensive, compliant, under-actuated robotic hand which has been shown to be effective for a variety of grasping tasks [3]. The hand consists of a polyamide scaffold to which multiple pneumatic actuators are attached (see Fig. 2). Each of the four fingers is a single actuator, while the thumb consists of three independent pneumatic actuators. This makes the thumb the most dexterous part of the hand, achieving seven out of eight configurations of the Kapandji test [30]. The actuators are controlled via external air valves and a separate air supply. Control is challenging since the air valves can only be either fully closed or open and have a switching time of ~ 0.02 s. Each actuator has a pressure sensor located close to the air valve.

The hand is controlled by specifying valve opening durations to either inflate or deflate a single actuator. We turn the discrete valve actions into a continuous control signal using pulse width modulation. Given a constant frequency of 5Hz, the control signal is interpreted as the duration the inflation (if it is positive) or deflation (negative) valve is opened during a single time step. To ensure that the control signal does not exceed the duration of a single time step we apply a sigmoid function to the commands from the learning algorithm.

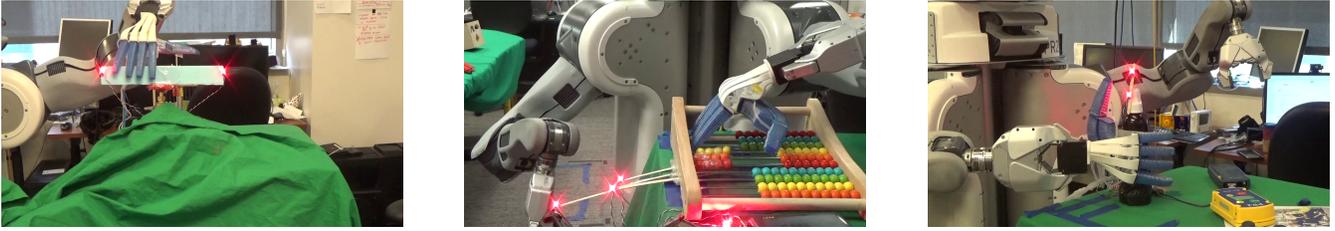


Fig. 3. The three manipulation tasks used in our experiments: Turning a valve, pushing beads on an abacus, and grasping a bottle from a table.

The positions and velocities of the manipulated objects are captured in real time with a PhaseSpace Impulse system, which relies on active LED markers. The state x_t of our system is the concatenation of the seven pressure readings of the hand, their time derivatives, the 3D positions and velocities of markers attached to the object, and joint angles of the robot arm (depending on the task). We placed no LED markers on the hand itself, only the object was equipped to record object-centric demonstrations, and the positions and velocities of these markers constitute object-centric state \bar{x}_t .

VIII. EXPERIMENTS

We evaluated our algorithm on a variety of manipulation and grasping tasks. Our experiments aim to show that

- 1) It is possible to perform fine manipulation with the RBO Hand 2.
- 2) Our algorithm can learn feedback policies from demonstrations that perform nearly as well as an oracle with the correct demonstrations (depending on the context) manually assigned to controllers.
- 3) A single neural network policy learned from demonstrations with our method is able to generalize to different initial states.

We will evaluate our learning approach on three different tasks: turning a valve, pushing beads on an abacus, and grasping a bottle (see Fig. 3). A video of the experiments can be found at <https://youtu.be/XyZFkJWu0Q0>. For the first two tasks, we compare our method to the following baselines:

Hand designed baseline: A controller with a hand-designed open loop policy. In the case of the abacus task, we evaluate the performance of two different strategies for simple hand-designed baselines.

Single demo baseline: A single controller trained to imitate a single demonstration. We use two separate baselines which are trained to follow different demonstrations.

Oracle: Depending on the context we manually assign the correct achievable demonstration to controllers. This comparison is useful to test whether the correspondence assignments are accurate.

A. Turning a valve

1) *Experimental setup:* Rotating a gas valve is a challenging task since it involves coordinating multiple fingers. Our valve consists of a blue horizontal lever that increases its range of motion (Fig. 3). Varying wrist positions along the lever require different finger motions to rotate it.

We mount the RBO Hand 2 on a PR2 robot arm, with the objective to rotate the valve away from the initial center position in either direction, using just its fingers. The arm is kept stationary for each episode, but changes positions between the training of different controllers. The joint angles are part of the state to determine the relative position of the hand with respect to the valve.

A human demonstrated three different valve rotations with their own hand, while two LED markers tracked the motion of the lever. Two demonstrations were of the valve rotating clockwise and anti-clockwise at the same position, and a third demonstration with the valve placed at a different position and rotated anticlockwise. All three demonstrations are valid for the task, but not all of them are achievable from every training position. Our algorithm trained three individual controllers and a single neural network policy to generalize over them.

2) *Results and Discussion:* During evaluation, the policy learned by each method was sampled ten times at four positions of the hand relative to the valve. The results in Fig. 4 show that our method generates the most robust policy compared with the baselines, which each fail to turn the valve significantly in at least one position. Our method does nearly as well as the oracle, for which demonstrations are assigned to controllers manually. While learning the correspondence weights and the individual controller policies, our method determines which of the demonstration it can actually perform from its initial positions, and disregards distant unachievable demonstrations.

Our method is able to learn distinctly different behavior at various test positions. At position 1, the policy pushes the lever using its last two fingers, with support given by the thumb. At position 2, the policy uses the thumb to rotate the valve by pushing the lever as the fingers are blocked. At position 3, our policy extends the thumb out of the way and pushes strongly with the index finger to rotate the valve. Simple open loop hand-designed strategies and the baselines learned from a single demonstration fail to learn this distinctly different behavior needed to generalize to different positions along the valve lever. By learning that different joint angles of the arm require different behaviors to be performed, our method is able to perform the task in various positions.

B. Pushing the beads of an abacus

1) *Experimental setup:* The RBO Hand 2 is required to push particular beads on an abacus while leaving other beads stationary. This task is challenging due to the precise

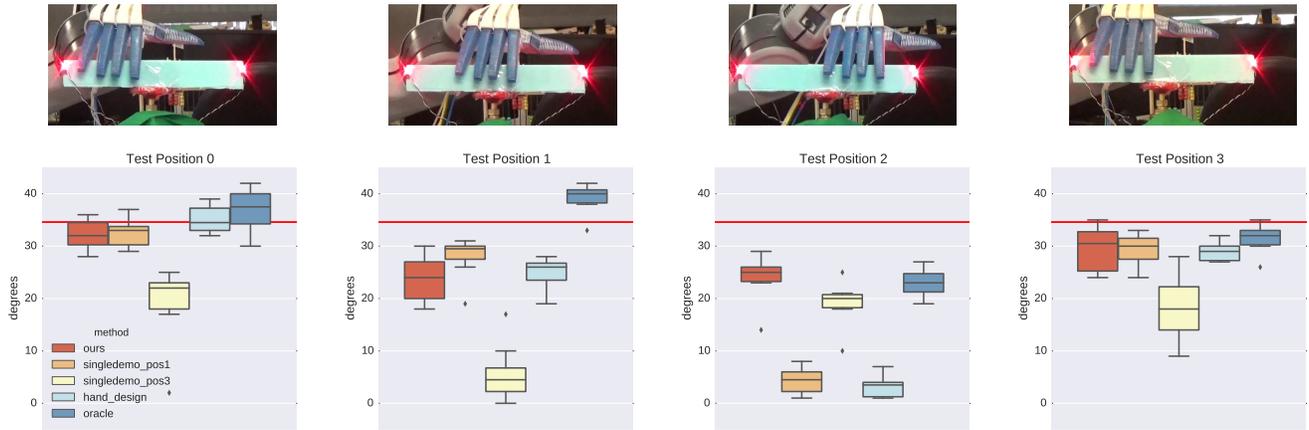


Fig. 4. Comparison of different policies for the valve task: the red line indicates the demonstrated rotation of the valve by ≈ 35 deg. On average our method learns the most general feedback strategy. The boxes in the box plot for each test position are our method, single demo baseline 1, single demo baseline 2, hand-designed baseline and oracle plotted from left to right. Although the baselines do well in some positions, the only methods which do consistently well across all positions are our method and the oracle.

	Bead	Target	Ours	SingleDemo1	SingleDemo2	Oracle	HandDesign1	HandDesign2
	1	8.4	7.49 \pm 0.47	7.02 \pm 0.50	6.33 \pm 2.15	7.66 \pm 0.23	8.38 \pm 0.04	0 \pm 0
	2	0	0.14 \pm 0.18	0.60 \pm 0.69	7.08 \pm 1.04	0.27 \pm 0.42	0 \pm 0	6.5 \pm 0
	3	0	0.89 \pm 1.00	0.28 \pm 0.18	1.23 \pm 2.20	1.08 \pm 0.72	0 \pm 0	8.43 \pm 0.29
	Bead	Target	Ours	SingleDemo1	SingleDemo2	Oracle	HandDesign1	HandDesign2
	1	8.4	7.95 \pm 0.19	1.04 \pm 2.15	7.27 \pm 0.65	7.52 \pm 0.66	0.00 \pm 0.00	8.38 \pm 0.08
	2	0	0.10 \pm 0.10	0.85 \pm 1.21	0.19 \pm 0.14	0.09 \pm 0.11	0.00 \pm 0.00	8.40 \pm 0.00
	3	0	0.00 \pm 0.00	0.00 \pm 0.00				
	Bead	Target	Ours	SingleDemo1	SingleDemo2	Oracle	HandDesign1	HandDesign2
	1	8.4	7.21 \pm 0.69	2.47 \pm 2.22	3.39 \pm 1.98	7.74 \pm 0.23	0.00 \pm 0.00	8.38 \pm 0.05
	2	0	0.00 \pm 0.00	0.00 \pm 0.00				
	3	0	0.00 \pm 0.00	0.00 \pm 0.00				

Fig. 5. Comparison of the distance moved by the various beads in cm using different policies for the abacus task, at 3 different positions, namely Positions 1, 2 and 3 going downwards. The target column in each table indicates the demonstrated movement of the three beads, and the other columns indicate the mean and standard deviation of other methods. On average our method learns the most general feedback strategy besides the oracle

individual finger motions needed to move only the desired beads. The hand is mounted on a stationary PR2 arm (Fig. 3), while the abacus is moved to several positions. The beads of relevance here are the central yellow, orange and red ones. Markers were attached to each of the three beads to capture their motion. As the position of the abacus with respect to the hand changes, different fingers need to be used. During demonstrations a human pushed only the yellow beads along their spindle at each of the three positions shown in Fig. 5.

2) *Results and Discussion:* We evaluated ten samples of each policy at each of the three positions and recorded the distances that each bead moved. The results are shown in Fig. 5. Our method moves the bead closer to the target position than the single demonstration and hand designed baselines for all the test positions. Only the oracle policy produces equally good performance. By interleaving selection of the right demonstration to imitate, with optimal control and supervised learning, our algorithm is able to learn a policy which uses discretely different fingers depending on the positions of the abacus relative to the hand. On the other hand, the hand-designed baselines being open loop can never learn different behaviors for different fingers. The controller trained at a single position fails because it has no notion of

generalization.

C. Grasping a bottle

1) *Experimental setup:* This task involves using the soft hand mounted on a moving arm, to grasp a deodorant bottle placed on a table. The arm has a scripted motion of moving up after 8 seconds, and we use reinforcement learning to learn the behavior of the fingers to go with this arm motion. The objective of the task is to grasp the bottle before the arm starts moving and keep it grasped until the end of the episode at the final arm location.

As grasping tasks for several objects largely succeed in open loop, our aim is to demonstrate that we can match the performance of a hand-designed baseline with a controller learned from a human demonstration through optimal control. This experiment is challenging for reinforcement learning algorithms due to the delayed nature of the reward signal in grasping.

We provide a demonstration of the bottle being lifted by a human, and use it to define the cost function for trajectory optimization as the l_2 distance of trajectory samples from the provided demonstration. We also apply a Gaussian filter to the noise generated in the controllers to be more temporally

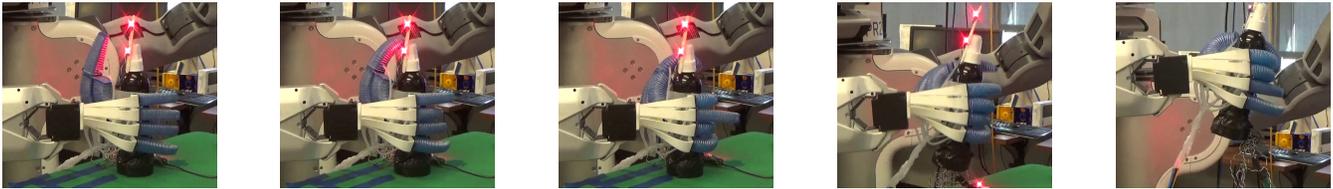


Fig. 6. Execution of a learned policy to grasp a bottle. Our method learns to grasp the bottle tightly and performs as well as the hand designed baseline.

coherent, allowing tight grasping. The resulting learned control policy is then tested on 10 sample trajectories in order to evaluate whether a successful grasp has occurred where the object is lifted and kept at the maximum arm height.

2) *Results and Discussion:* We find that on the grasping task, the control policy learned through optimal control does just as well as a hand-designed policy on ten samples of grasping the bottle. Both the hand-designed policy and the learned policy were able to grasp the bottle for all 10 test samples. This indicates that the learning has comparable results to a hand-designed baseline, despite not having prior information besides a human-provided demonstration.

D. Limitations

Although the LfD algorithm shows good performance on several tasks using the RBO Hand 2, there are many directions for future work. Instead of using a motion capture system, we hope to use better computer vision techniques such as deep convolutional nets to track trajectories of relevant feature points in future work. Extending the neural network policy to learn policies dependent on just the pressure sensors in the fingers and/or additional tactile sensors, would be an exciting future direction.

IX. CONCLUSIONS

We presented an algorithm for learning dexterous manipulation skills with a soft hand from object-centric demonstrations. Unlike standard LfD methods, our approach only requires the human expert to demonstrate the desired behaviors with their own hand. Our method automatically determines the most relevant demonstrations to track, using reinforcement learning to optimize a collection of controllers together with controller to demonstration correspondences. To generalize the demonstrations to new initial conditions, we utilize the GPS framework to train nonlinear neural network policies that combine the capabilities of all of the controllers. We evaluate our method on the RBO Hand 2, and show that it is capable of learning a variety of dexterous manipulation skills, including valve turning, moving beads on an abacus, and grasping.

REFERENCES

- [1] T. Mouri, H. Kawasaki, K. Yoshikawa, J. Takai, and S. Ito, "Anthropomorphic robot hand: Gifu hand iii," in *Proc. Int. Conf. ICCAS*, 2002.
- [2] "Shadow robot company. the shadow dexterous hand." <http://www.shadowrobot.com/products/dexterous-hand/>.
- [3] R. Deimel and O. Brock, "A novel type of compliant and underactuated robotic hand for dexterous grasping," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 161–185, March 2016.
- [4] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *NIPS*, 2014.
- [5] L. Han and J. C. Trinkle, "Dexterous manipulation by rolling and finger gaiting," in *ICRA*, vol. 1. IEEE, 1998.
- [6] M. Cherif and K. K. Gupta, "Planning quasi-static fingertip manipulations for reconfiguring objects," *Robotics and Automation, IEEE Transactions on*, vol. 15, no. 5, pp. 837–848, 1999.
- [7] Y. Bai and C. K. Liu, "Dexterous manipulation using both palm and fingers," in *ICRA 2014*. IEEE, 2014.
- [8] I. Mordatch, Z. Popović, and E. Todorov, "Contact-invariant optimization for hand manipulation," in *Proc. of the ACM SIGGRAPH*, 2012.
- [9] P. Polygerinos, Z. Wang, J. T. B. Overvelde, K. C. Galloway, R. J. Wood, K. Bertoldi, and C. J. Walsh, "Modeling of soft fiber-reinforced bending actuators," *IEEE T-RO*, vol. 31, no. 3, pp. 778–789, June 2015.
- [10] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," Tech. Rep., 2002.
- [11] J. Peters, K. Mülling, and Y. Altun, "Relative entropy policy search," in *AAAI*. Atlanta, 2010.
- [12] B. J. S. S. Theodorou, E. A., "Learning policy improvements with path integrals," in *AISTATS 2010*, 2010.
- [13] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *IJRR*, vol. 32, no. 3, pp. 263–279, 2013.
- [14] O. Kroemer, C. Daniel, G. Neumann, H. van Hoof, and J. Peters, "Towards learning hierarchical skills for multi-phase manipulation tasks," in *ICRA*. IEEE, 2015.
- [15] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal, "Learning force control policies for compliant manipulation," in *IROS*. IEEE, 2011.
- [16] T. Lampe and M. Riedmiller, "Acquiring visual servoing reaching and grasping skills using neural reinforcement learning," in *IJCNN*, 2013.
- [17] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," in *ICRA*, 2015.
- [18] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *arXiv preprint arXiv:1504.00702*, 2015.
- [19] H. van Hoof, T. Hermans, G. Neumann, and J. Peters, "Learning robot in-hand manipulation with tactile features," in *Humanoid Robots (Humanoids)*. IEEE, 2015.
- [20] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation."
- [21] S. Calinon and A. Billard, "Incremental learning of gestures by imitation in a humanoid robot," in *Proceedings of the ACM/IEEE international conference on Human-robot interaction*. ACM, 2007.
- [22] T. Asfour, P. Azad, F. Gyarfas, and R. Dillmann, "Imitation learning of dual-arm manipulation tasks in humanoid robots," *International Journal of Humanoid Robotics*, vol. 5, no. 02, pp. 183–202, 2008.
- [23] E. L. Sauser, B. D. Argall, G. Metta, and A. G. Billard, "Iterative learning of grasp adaptation through human corrections," *Robotics and Autonomous Systems*, vol. 60, no. 1, pp. 55–71, 2012.
- [24] U. Prieur, V. Perdereau, and A. Bernardino, "Modeling and planning high-level in-hand manipulation actions from human knowledge and active learning from demonstration," in *IROS*. IEEE, 2012.
- [25] F. Guenter, M. Hersch, S. Calinon, and A. Billard, "Reinforcement learning for imitating constrained reaching movements," *Advanced Robotics*, vol. 21, no. 13, pp. 1521–1544, 2007.
- [26] J. Hershey and P. Olsen, "Approximating the Kullback Leibler divergence between Gaussian mixture models," in *ICASSP*, vol. 4, April 2007, pp. IV-317–IV-320.
- [27] D. H. Jacobson and D. Q. Mayne, *Differential dynamic programming*. New York: Elsevier, 1970.
- [28] S. Levine and V. Koltun, "Guided policy search," in *Proceedings of The 30th International Conference on Machine Learning*, 2013.
- [29] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *ICINCO (1)*, 2004.
- [30] I. A. Kapanji, "Cotation clinique de l'opposition et de la contre-opposition du pouce," *Annales de Chirurgie de la Main*, vol. 5, no. 1, pp. 68–73, 1986.