2011 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 25-30, 2011. San Francisco, CA, USA

Learning Force Control Policies for Compliant Manipulation

Mrinal Kalakrishnan*, Ludovic Righetti*, Peter Pastor*, and Stefan Schaal*†

Abstract— **Developing robots capable of fine manipulation skills is of major importance in order to build truly assistive robots. These robots need to be compliant in their actuation and control in order to operate safely in human environments. Manipulation tasks imply complex contact interactions with the external world, and involve reasoning about the forces and torques to be applied. Planning under contact conditions is usually impractical due to computational complexity, and a lack of precise dynamics models of the environment. We present an approach to acquiring manipulation skills on compliant robots through reinforcement learning. The initial position control policy for manipulation is initialized through kinesthetic demonstration. We augment this policy with a force/torque profile to be controlled in combination with the position trajectories. We use the Policy Improvement with Path Integrals (PI^2) algorithm to learn these force/torque profiles by optimizing a cost function that measures task success. We demonstrate our approach on the Barrett WAM robot arm equipped with a 6-DOF force/torque sensor on two different manipulation tasks: opening a door with a lever door handle, and picking up a pen off the table. We show that the learnt force control policies allow successful, robust execution of the tasks.**

I. Introduction

Developing robots capable of fine manipulation skills is of major importance in order to build truly assistive robots. Manipulation tasks imply complex contact interactions with an unstructured environment and require a controller that is able to handle force interactions in a meaningful way. In order for robots to co-exist in an environment with humans, safety is a prime consideration. Therefore, touching and manipulating an unstructured world requires a certain level of compliance while achieving the intended tasks accurately.

Methods for planning kinematic trajectories for manipulators are well-studied and widely used. Rigid body dynamics models even allow us to plan trajectories that take the robot dynamics into account. However, once the robot comes into contact with the environment, planning algorithms would require precise dynamics models of the resulting contact interactions. These models are usually unavailable, or so imprecise that the generated plans are unusable. This seems to suggest alternate solutions that can learn these manipulation skills through trial and error.

Acquisition of manipulation skills using reinforcement learning has been previously demonstrated [1], [2], [3], [4], [5]. In most of these approaches, the policy encodes positions which are tracked by a controller. These position trajectories are adapted in order to achieve the task, thus indirectly exerting forces on the objects being manipulated. A desired position trajectory must penetrate into the object being manipulated, in order to apply a force on it. This can potentially be dangerous, for example, if the object is wrongly positioned, and the resulting forces generated are too high. In contrast, we propose to learn the forces and torques to be controlled at the end-effector in conjuction with a demonstrated kinematic trajectory. This corresponds more directly to the physical quantities that need to be controlled. Control of forces instead of positions allows the system to deal with position and state estimation errors that are unavoidable when acting in the real world. Furthermore, in tasks that involve contact with the environment, exploration in high gain position control mode could be damaging to the robot and its surroundings.

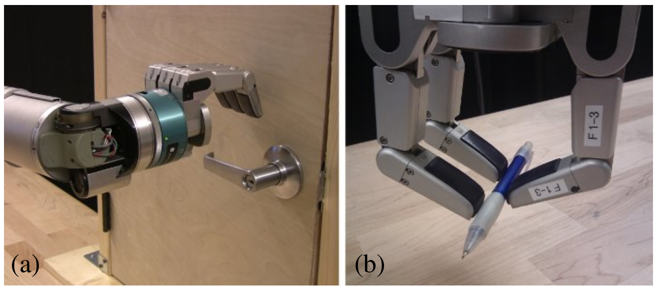


Fig. 1. Force control policies for two different manipulation tasks were learnt using our method: (a) opening a door, and (b) picking up a pen from the table.

Methods for learning force profiles from demonstrations provided via haptic input have been proposed [6]. The method does not explicitly consider task performance: it merely reproduces the demonstrated trajectories. In the context of our work, such techniques could be used to initialize a force profile for futher optimization to improve task performance.

In this paper, we present an approach to learning manipulation tasks on compliant robots through reinforcement learning. An initial kinesthetic demonstration of the task is provided. Execution of this demonstration on a robot with compliant control fails to execute the task succesfully, because the demonstration does not include information about forces to be exerted at the end-effector. We propose to learn the required end-effector forces through trial and error reinforcement learning. This allows the robot to acquire a robust strategy to perform the manipulation task, while remaining compliant in its control. We demonstrate our approach on two different manipulation tasks: opening a door with a lever door handle, and picking up a pen off the table (Fig. 1). We show that our approach can learn the force profiles required

*Computational Learning and Motor Control Lab, University of Southern California, Los Angeles, CA 90089, USA. †Max-Planck-Institute for Intelligent Systems, 72076 Tübingen, Germany. Email: {kalakris, pastorsa, sschaal}@usc.edu, ludovic.righetti@a3.epfl.ch

This research was supported in part by National Science Foundation grants ECS-0326095, IIS-0535282, IIS-1017134, CNS-0619937, IIS-0917318, CBET-0922784, EECS-0926052, CNS-0960061, the DARPA program on Advanced Robotic Manipulation, the Army Research Office, the Okawa Foundation, and the ATR Computational Neuroscience Laboratories.

to achieve both tasks successfully. The contributions of this paper are two-fold: (1) we demonstrate that learning force control policies enables compliant execution of manipulation tasks with increased robustness as opposed to stiff position control, and (2) we introduce a policy parameterization that uses finely discretized trajectories coupled with a cost function that ensures smoothness during exploration and learning. The remainder of this paper is structured as follows: in Section II, we review the **P**olicy **I**mprovement with **P**ath **I**ntegrals (**PI^2**) algorithm [7], and describe how it can used to optimize desired trajectories subject to certain smoothness constraints. We then discuss the application of PI^2 to learning force control policies in Section III. In Section IV, we show experimental results from the application of these ideas to two different manipulation tasks. Finally, we conclude and present ideas for future work in Section V.

II. Policy Improvement with Path Integrals (PI^2)

We first review the application of the path integral stochastic optimal control framework to the optimization of parameterized policies, specifically Dynamic Movement Primitives [8], as was originally developed [7]. We then apply these results to a policy which encodes discretized desired trajectories directly, subject to a smoothness cost.

A. *PI^2 with Dynamic Movement Primitives*

PI^2 is a model-free reinforcement learning method that optimizes the parameters of a policy to minimize a given cost function. We merely introduce the notation and the algorithm here; detailed derivations may be found in [7].

We consider control systems of the following form:

$$\dot{\mathbf{x}}_t = \mathbf{f}(\mathbf{x}_t, t) + \mathbf{G}(\mathbf{x}_t)(\mathbf{u}_t + \boldsymbol{\epsilon}_t), \tag{1}$$

where $\mathbf{x}_t \in \mathbb{R}^n$ denotes the state of the system at time t, $\mathbf{u}_t \in \mathbb{R}^p$ the control vector at time t, $\mathbf{f}(\mathbf{x}_t, t)$ the passive dynamics, $\mathbf{G}(\mathbf{x}_t)$ the control transition matrix, and $\boldsymbol{\epsilon}_t \in \mathbb{R}^p$ Gaussian noise with variance $\boldsymbol{\Sigma}_\epsilon$. The immediate cost function is defined as:

$$r_t = q_{\mathbf{x}}(\mathbf{x}, t) + \frac{1}{2}\mathbf{u}_t^{\mathrm{T}}\mathbf{R}\mathbf{u}_t, \tag{2}$$

where $q_{\mathbf{x}}(\mathbf{x}, t)$ is an arbitrary state-dependent cost function, and $\mathbf{R} \in \mathbb{R}^{p\times p}$ is the positive definite weight matrix of the quadratic control cost. The cost of a trajectory $\boldsymbol{\tau}$ of fixed duration T, discretized into N time-steps is then defined as:

$$S(\boldsymbol{\tau}) = \sum_{i=1}^{N-1} r_{t_i} + \phi_{t_N}, \tag{3}$$

where ϕ_{t_N} is a terminal cost assigned at time t_N. The parameterized policy to be optimized is of the form of the control system (1), with only one controlled equation and a one-dimensional controllable state:

$$\dot{x}_t = \mathbf{f}(x_t, t) + \mathbf{g}_t^{\mathrm{T}}(\boldsymbol{\theta} + \boldsymbol{\epsilon}_t), \tag{4}$$

where x_t is the one-dimensional state, $\mathbf{g}_t \in \mathbb{R}^p$ are time-dependent basis functions, and $\boldsymbol{\theta} \in \mathbb{R}^p$ is the policy parameter vector. Dynamic Movement Primitives (DMPs) [8] are a special case of such a parameterized policy which guarantee attractor properties towards the goal of the movement while remaining linear in the policy parameters $\boldsymbol{\theta}$.

The pseudocode for one iteration of the PI^2 algorithm for optimizing the policy parameters $\boldsymbol{\theta}$ is listed below:

- Create K roll-outs of the system $\boldsymbol{\tau}_1 \ldots \boldsymbol{\tau}_K$ from the same start state x_0 using stochastic parameters $\boldsymbol{\theta} + \boldsymbol{\epsilon}_t$ at every time step.
- For $k = 1 \ldots K$, and time-steps $i = 1 \ldots N$, compute:

$$\mathbf{M}_{t_i} = \frac{\mathbf{R}^{-1}\mathbf{g}_{t_i}\mathbf{g}_{t_i}^{\mathrm{T}}}{\mathbf{g}_{t_i}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{g}_{t_i}} \tag{5}$$

$$S(\boldsymbol{\tau}_{i,k}) = \phi_{t_N,k} + \sum_{j=i}^{N-1} q_{t_j,k} + \frac{1}{2}\sum_{j=i}^{N-1}(\boldsymbol{\theta} + \mathbf{M}_{t_j}\boldsymbol{\epsilon}_{t_j})^T\mathbf{R}(\boldsymbol{\theta} + \mathbf{M}_{t_j}\boldsymbol{\epsilon}_{t_j}) \tag{6}$$

$$P(\boldsymbol{\tau}_i) = \frac{e^{-\frac{1}{\lambda}S(\boldsymbol{\tau}_{i,k})}}{\sum_{j=1}^{K} e^{-\frac{1}{\lambda}S(\boldsymbol{\tau}_{i,j})}} \tag{7}$$

- For time-steps $i = 1 \ldots N$, compute:

$$\delta\boldsymbol{\theta}_{t_i} = \sum_{k=1}^{K} P(\boldsymbol{\tau}_{i,k})\,\mathbf{M}_{t_i,k}\boldsymbol{\epsilon}_{t_i,k} \tag{8}$$

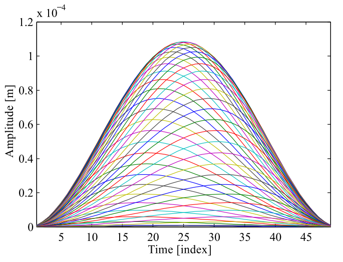
- Compute $\delta\boldsymbol{\theta} = \frac{1}{N}\sum_{i=0}^{N-1}\delta\boldsymbol{\theta}_{t_i}$
- Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta\boldsymbol{\theta}$

$\mathbf{M}_{t_i}$ is a matrix that projects the noise $\boldsymbol{\epsilon}_{t_i}$ onto the range space of the basis vector $\mathbf{g}_{t_i}$ under the metric $\mathbf{R}^{-1}$, at time t_i. $P(\boldsymbol{\tau}_i)$ is a discrete probability assigned to the sampled trajectory $\boldsymbol{\tau}_i$ based on its cost: the lower the cost, the higher the probability. $\delta\boldsymbol{\theta}_{t_i}$ is the vector of parameter updates computed at time t_i, which is a weighted combination of the noise $\boldsymbol{\epsilon}_{t_i}$ projected onto $\mathbf{M}_{t_i}$. $\delta\boldsymbol{\theta}$ is the final parameter update, computed as the average of updates for each time step. This algorithm is iterated until convergence of the noise-less trajectory cost.

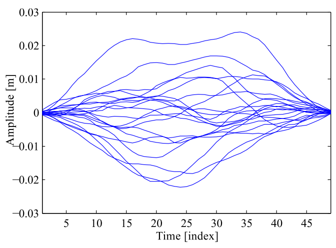
B. *PI^2 with discretized trajectories*

In this paper, we apply PI^2 to the optimization of a parameterized policy whose parameters represent the trajectory states discretized in time. Most policy representations, including DMPs involve the use of basis functions in order to reduce the parameter space, and achieve smoothness in the resulting trajectories. This requires apriori selection of the number of basis functions, which in turn influences the complexity of the trajectory that can be learnt. We propose to instead represent trajectories by finely discretizing them, and control their complexity using a smoothness cost function imposed over the entire trajectory. While this is not the primary focus of this paper, it is nevertheless a novel component, and is potentially useful for other learning applications.

The main challenge with using a finely discretized trajectory representation is that adding noise directly to this trajectory would make the resulting trajectory jerky and difficult to execute on a real robot. To address this problem, we impose a control cost on the trajectory (via the quadratic cost matrix $\mathbf{R}$) that measures its squared derivatives (velocities, accelerations and jerk). The PI^2 algorithm relies on the central assumption [7] that the noise $\boldsymbol{\epsilon}$ in the parameters is proportional to $\mathbf{R}^{-1}$. Conversely, it imposes restrictions

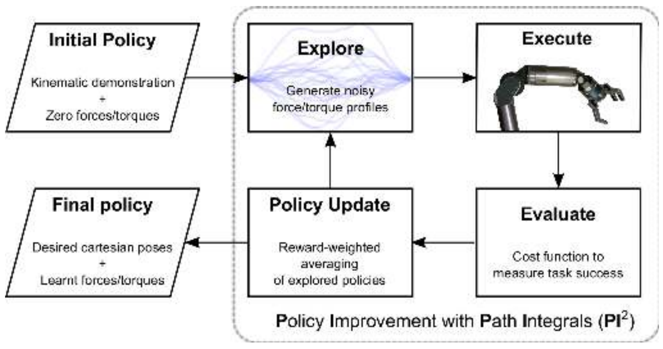


(a)



(b)

Fig. 2. (a) Each curve depicts a column/row of the symmetric matrix $\mathbf{R}^{-1}$, when $\mathbf{R}$ measures the sum of squared accelerations along the trajectory. (b) 20 random samples of $\boldsymbol{\epsilon}$, drawn from a zero mean normal distribution with covariance $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}} = \mathbf{R}^{-1}$.



Initial Policy
Kinematic demonstration
+
Zero forces/torques
Explore
Generate noisy force/torque profiles
Execute
Final policy
Desired cartesian poses
+
Learnt forces/torques
Policy Update
Reward-weighted averaging of explored policies
Evaluate
Cost function to measure task success
Policy Improvement with Path Integrals (PI²)

Fig. 3. A high-level overview of our approach to learning force control policies for manipulation.

on the generation of exploration noise, such that the noise minimally impacts the control cost. In our case, this allows generation of exploration trajectories that are smooth. The structure of the $\mathbf{R}^{-1}$ matrix ensures that the trajectory does not diverge from the start or goal. Finally, the projection step in the PI^2 update equation (Eq. (8)) maintains smoothness at every iteration.

We assume a desired trajectory of duration T, discretized into N timesteps, represented by $\boldsymbol{\theta} \in \mathbb{R}^N$. These discretized waypoints can be tracked with the following controller:

$$\dot{x}_t = K\left(\mathbf{g}_t^{\mathrm{T}}(\boldsymbol{\theta}+\boldsymbol{\epsilon}) - x_t\right), \tag{9}$$

where x_t is the desired state at time t, K is a control gain, and $\mathbf{g}_t \in \mathbb{R}^N$ is a vector that contains 1 at position t and 0 at all other indices. This system, with a sufficiently high gain K, ensures that the state x tracks the desired trajectory $\boldsymbol{\theta}$ over time. This control system is of the same form as Eq. (4), allowing PI^2 to optimize the parameter vector $\boldsymbol{\theta}$.

Next, we choose the quadratic control cost matrix $\mathbf{R}$ such that the control cost $\boldsymbol{\theta}^{\mathrm{T}}\mathbf{R}\boldsymbol{\theta}$ measures the sum of squared derivatives along the entire trajectory. Similar to previous work on optimization-based motion planning [9], [10], we define $\mathbf{R}$ using finite differencing matrices $\mathbf{A}_1 \ldots \mathbf{A}_D$:

$$\mathbf{R} = \sum_{d=1}^{D} w_d \|\mathbf{A}_d\|^2, \tag{10}$$

where d is the order of differentiation, and w_d is the weight of each term in the cost function. In our experiments, we choose to minimize squared accelerations, i.e., $w_2 = 1$, and all other weights are 0. The corresponding finite differencing matrix A_2 is of the form:

$$\mathbf{A}_2 = \begin{bmatrix} 1 & 0 & 0 & & 0 & 0 & 0 \\ -2 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & -2 & 1 & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & 1 & -2 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -2 \\ 0 & 0 & 0 & & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

Note that $\mathbf{R}$ is always both symmetric and positive definite when constructed in this fashion.

This particular choice of the $\mathbf{R}$ matrix exhibits a few interesting properties. Fig. 2(a) shows the structure of $\mathbf{R}^{-1}$. Fig. 2(b) shows samples of the noise $\boldsymbol{\epsilon}$, drawn from the distribution $\mathcal{N}(0, \mathbf{R}^{-1})$. The covariant structure of the noise makes exploration in discrete trajectory space possible, since these samples can be executed without trouble on a real system. Since our policy parameter vector $\boldsymbol{\theta}$ is constant over time, we also sample the noise $\boldsymbol{\epsilon}$ just once and keep it fixed over the entire trajectory. If the noise were resampled at every time-step, it would lose its covariant property. Additionally, the noise samples do not cause the trajectory to diverge from the start or goal, which is a property that facilitates learning of point to point movements [11]. The level of discretization used does not affect convergence rates, since exploration and learning is performed covariantly across the entire trajectory, not independently for each parameter. If adaptation of the start or goal point is required, the construction of the finite differencing matrices can be suitably adapted to treat the start or goal point as a variable instead of a constant [12]. This representation thus affords a significant amount of flexibility to adapt the learning process to the problem at hand.

The PI^2 update equations are the same as in the case of the DMPs, with the sparse nature of g_t allowing for some simplifications. These update equations are strikingly similar to those used in STOMP [10], a stochastic trajectory optimizer used for kinematic motion planning, although derived differently. There is, however, one key difference: the planning problems considered by STOMP contain no dynamics, hence it optimizes the immediate cost at every time-step. For reinforcement learning on a dynamical system, an action at time t influences all future costs, and hence requires optimization of the cost-to-go (computed in Eq. (6).)

III. Learning Force Feedback Control

Fig. 3 shows a high-level overview of our approach to learning manipulation tasks on a compliant robot. The policy is initialized from a user-provided kinematic demonstration. The PI^2 reinforcement learning algorithm is used to optimize the policy and acquire the right force/torque profiles through trial and error. We further discuss some of the steps involved:

A. Demonstration

We record the end-effector position and orientation trajectories during a kinesthetic demonstration of the task provided by the user. Since forces and torques applied by the robot on the object being manipulated cannot be observed correctly during kinesthetic demonstration, the force-torque profiles are initialized to zero. When controlling zero forces, the robot is maximally compliant, i.e., the end-effector gives in to contact forces easily. This forms a safe starting point for exploration, and ensures that only the forces required to satisfy the task are learnt. We use the discretized policy representation discussed in Sec. II-B, Eq. (9) for both position and force trajectories.

B. Cost Function

In order to acquire the correct force/torque profiles, the reinforcement learning algorithm requires a measure of task success for every trial that it executes. This feedback could be provided by the user for every trial, but it is more convenient if task success can be evaluated in an automated fashion. PI^2 has previously been used to optimize boolean cost functions [3], but learning tends to be faster if the cost function has a gradient to follow.

C. Execution

The combined position/force trajectory needs to be controlled by the robot in a suitable way. Numerous methods can be found in the literature that control forces and positions simultaneously [13]. PI^2, being a *model-free* reinforcement learning algorithm, is indeed agnostic to the type of controller used. It simply optimizes the policy parameters to improve the resulting cost function, treating the intermediate controllers and unmodeled system dynamics as a black box. However, the generalization ability of the learnt policy to different parts of the workspace will depend on the force and position tracking performance of the controller.

D. Rollout reuse

In every iteration of PI^2, K noisy rollouts are generated, evaluated, and used to update the policy. We instead prefer to preserve a few good rollouts from previous iterations, so that we continue to learn from them in future iterations, and achieve stable convergence. However, if the task itself is stochastic in nature, this procedure may be counter-productive. For example, if a noisy rollout happens to achieve a low cost during its evaluation, but is not repeatable, it nevertheless continues to contribute to future policy updates. In order to mitigate this effect, we reevaluate all the reused rollouts at every iteration. This ensures that only rollouts that consistently generate low costs are carried forward at each iteration. This feature was critical for PI^2 to converge to a robust policy in the pen grasping experiment presented in Sec. IV-B.

IV. Experiments

Our approach was verified using two different manipulation tasks: opening a door and picking up a pen lying on a table. These tasks were chosen because each one involves significant contact with the environment, and are thus appropriate test-beds for the use of force control policies. The learning process and final executions of both tasks are shown in the attached video [14].

Both tasks were performed on the 7 degree of freedom (DOF) Barrett WAM arm, equipped with a three-fingered Barrett Hand and a 6-DOF force-torque sensor at the wrist. Fig. 4 shows an overview of the controllers running on our system. Our control law for the 7-DOF arm is as follows:

$$\boldsymbol{\tau}_{\text{arm}} = \boldsymbol{\tau}_{\text{inv. dyn.}} + \boldsymbol{\tau}_{\text{joint}} + \boldsymbol{\tau}_{\text{force}}$$

Desired task-space position/orientation trajectories are converted into joint space using the Jacobian pseudo-inverse. The resulting joint velocities are integrated and differentiated, to get joint positions and accelerations respectively. $\boldsymbol{\tau}_{\text{inv.dyn.}}$ are the inverse dynamics torques obtained from a recursive Newton Euler algorithm [15]. $\boldsymbol{\tau}_{\text{joint}}$ is obtained from low-gain joint PD controllers. Finally, $\boldsymbol{\tau}_{\text{force}}$ is obtained from a PI controller in task space on the desired force/torque trajectory, converted to joint space torques using the Jacobian transpose. The fingers are position controlled using low-gain PD controllers. All our controllers run at a rate of 300Hz on a desktop computer running the Xenomai real-time operating system.

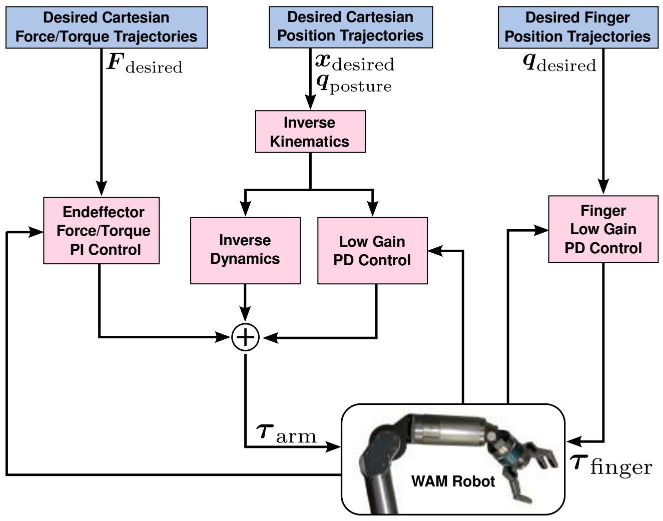

Desired Cartesian Force/Torque Trajectories
$\boldsymbol{F}_{\text{desired}}$
Desired Cartesian Position Trajectories
$\boldsymbol{x}_{\text{desired}}$
$\boldsymbol{q}_{\text{posture}}$
Desired Finger Position Trajectories
$\boldsymbol{q}_{\text{desired}}$
Inverse Kinematics
Endeffector Force/Torque PI Control
Inverse Dynamics
Low Gain PD Control
Finger Low Gain PD Control
$\boldsymbol{\tau}_{\text{arm}}$
$\boldsymbol{\tau}_{\text{finger}}$
WAM Robot

Fig. 4. Overview of the controllers used in our experiments.

A. Opening a door

The aim of this experiment is to learn a control policy to successfully operate a lever door handle and open the door shown in Fig. 1(a). A kinesthetic demonstration provides the desired cartesian positions and orientations, while the desired force/torque trajectories are initialized to 0. The trajectory was 10 seconds long, discretized into 100 time-steps. Direct playback of the demonstration fails to achieve the task due to the compliance of the robot and failure to apply the required forces and torques on the door handle.

In order to measure task success, we attached a MicroStrain 3DM-GX3-25 Inertial Measurement Unit (IMU) to the door handle. The resulting orientation measurement gives us the angle of the handle and of the door itself. We recorded the desired trajectories of the door angle and handle angle during the demonstration: tracking these trajectories forms the primary task success criterion. The immediate cost function at time t is: $r_t = 300q_{door} + 100q_{handle} + 100q_{pos} + 10q_{orient} + 0.1q_{fmag} + 0.02q_{tmag} + 0.02q_{ttrack} + 0.01q_{ftrack} + 0.0001\boldsymbol{\theta}^{\mathrm{T}}\mathbf{R}\boldsymbol{\theta}$, where q_{door} and q_{handle} are the squared tracking errors of the door and handle angles respectively, q_{pos} and q_{orient} are the squared tracking errors of the position and orientation of the hand, q_{fmag} and q_{tmag} are the squared magnitudes of the desired forces and torques, q_{ftrack} and q_{ttrack} are the squared force and torque tracking errors, and $\boldsymbol{\theta}^{\mathrm{T}}\mathbf{R}\boldsymbol{\theta}$ is the control cost.

We use PI^2 to learn policies for all 6 force/torque dimensions. A relatively small exploration noise is also simulta-

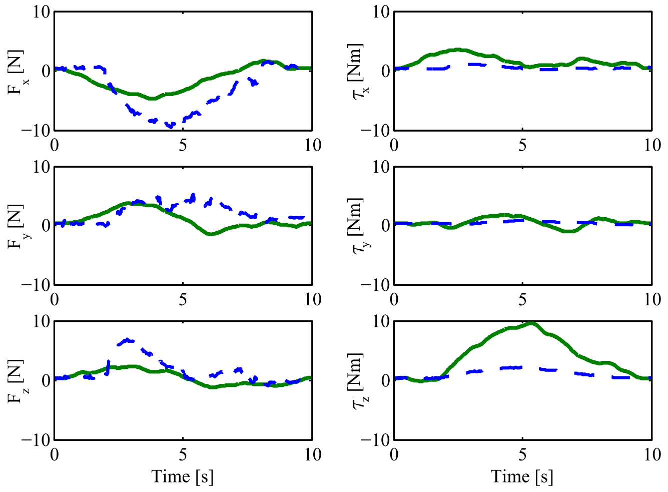


Fig. 5. Learnt force/torque control policy and tracking errors for the door opening task. The green solid lines show the learnt desired forces and torques respectively. The blue dashed lines indicate the corresponding measured force/torque values.

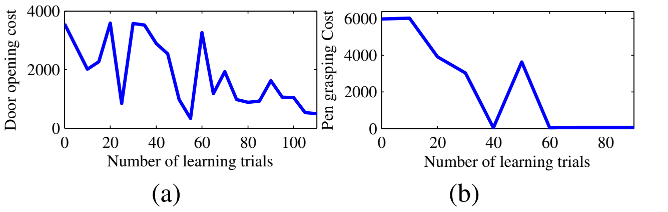


Fig. 6. Evolution of cost functions during learning for the two manipulation tasks: (a) door opening, and (b) pen grasping.

neously introduced in the cartesian position and orientation, to correct for imperfections in the demonstrated trajectory. Fig. 6(a) shows the improvement in the cost with the number of trials executed. After 110 trials, we obtained a policy which achieved the task successfully 100% of the time, with a cost of 498.1 $\pm$ 5.3, averaged over 23 trials. Learning was terminated when all the noisy rollouts in a particular learning iteration successfully achieved the task, indicating that the learnt policy was robust to the exploration noise. Fig. 5 shows the learnt force/torque profiles and their tracking performance. Fig. 9 shows snapshots from an execution of the final policy.

For this experiment, the proportional gains for the force controller were set to 1, while integral gains are set to 0. In theory, a proportional gain of 1 should compensate for the difference in desired and sensed forces. However, due to the addition of the joint-space position controller, these forces cannot be perfectly realized. Despite the force tracking inaccuracies seen in Fig. 5, the algorithm learns to achieve the task successfully. This is possible because the PI^2 learning algorithm treats the controllers as a black box; it simply learns the required sequence of control inputs that minimize the task cost function.

B. Grasping a pen

The next task is to pick up a pen from the table, as shown in Fig. 1(b). This task is considered difficult, because the size of the pen is comparable to the size of the finger-tips, making the chance of slippage very high. In addition, this task would be quite difficult to achieve using a pure position-control

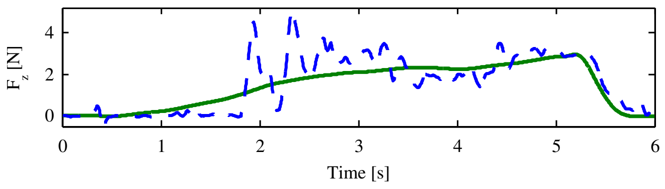
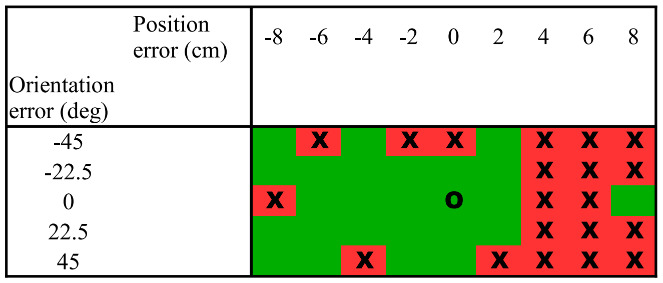


Fig. 7. Learnt force/torque control policy and tracking errors for the pen grasping task. The green solid line shows the learnt desired force in the z axis. The blue dashed line indicates the corresponding measured force.



Orientation error (deg) \ Position error (cm)	-8	-6	-4	-2	0	2	4	6	8
-45	🟩	🟥 X	🟩	🟥 X	🟥 X	🟩	🟥 X	🟥 X	🟥 X
-22.5	🟩	🟩	🟩	🟩	🟩	🟩	🟥 X	🟥 X	🟥 X
0	🟥 X	🟩	🟩	🟩	🟩 O	🟩	🟥 X	🟥 X	🟩
22.5	🟩	🟩	🟩	🟩	🟩	🟩	🟥 X	🟥 X	🟥 X
45	🟩	🟩	🟥 X	🟩	🟩	🟥 X	🟥 X	🟥 X	🟥 X

Fig. 8. This table shows the extent of position and orientation errors that the final pen grasping force control policy was able to handle. Position error was introduced along the direction of closing of the fingers. The 'O' in the center marks the original position of the pen during learning. The red 'X' squares indicate failure, while the green empty squares indicate success. Negative position errors were tolerated much better than positive errors, because the hand has two fingers on this side, versus only one on the other.

strategy, since this would require very precise knowledge of the pen position and the table height.

We use the following initial policy: we keep the desired position and orientation of the hand fixed over the pen, while closing the fingers to perform the grasp. The kinematics of the fingers are such that the fingers dig into the table during the motion (see Fig. 9 for a visual depiction of the grasping motion). Servoing zero forces at the wrist allows the hand to move up when the fingers come into contact with the table. While this simple strategy works for larger objects, it was not successful at grasping the pen. Hence, we would like to learn a profile of downward forces to be applied into the table that can help in robustly grasping the pen. We use both proportional and integral gains in the force controller in order to achieve good force tracking. We use PI^2 to learn the desired force in the z axis, while fixing desired x and y torques to 0. Force control gains for the remaining 3 axes are set to zero.

We detect whether the pen has been grasped or not based on the finger joint angles crossing a threshold value. We assign a cost for every time-step that the joint angles are above this threshold. This provides a gradient to the learning algorithm, i.e., the longer the pen remains grasped without slipping, the lower the cost. Costs are accumulated during the 6-second grasping phase as well as a subsequent 6-second lifting phase, which verifies the stability of the grasp. As with the previous experiment, trajectories were discretized into 100 time-steps. The immediate cost function at time t is: $r_t = 100q_{pen} + 1.0q_{ftrack} + 0.5q_{fingertrack} + 0.1q_{fmag} + 0.0001\boldsymbol{\theta}^{\mathrm{T}}\mathbf{R}\boldsymbol{\theta}$, where q_{pen} is an indicator cost which is 1 if the pen has slipped out of the hand (as described above), q_{ftrack} is the squared force tracking error, $q_{fingertrack}$ is the squared finger position tracking error, q_{fmag} is the squared force magnitude, and $\boldsymbol{\theta}^{\mathrm{T}}\mathbf{R}\boldsymbol{\theta}$ is the control cost. After 90 trials, we

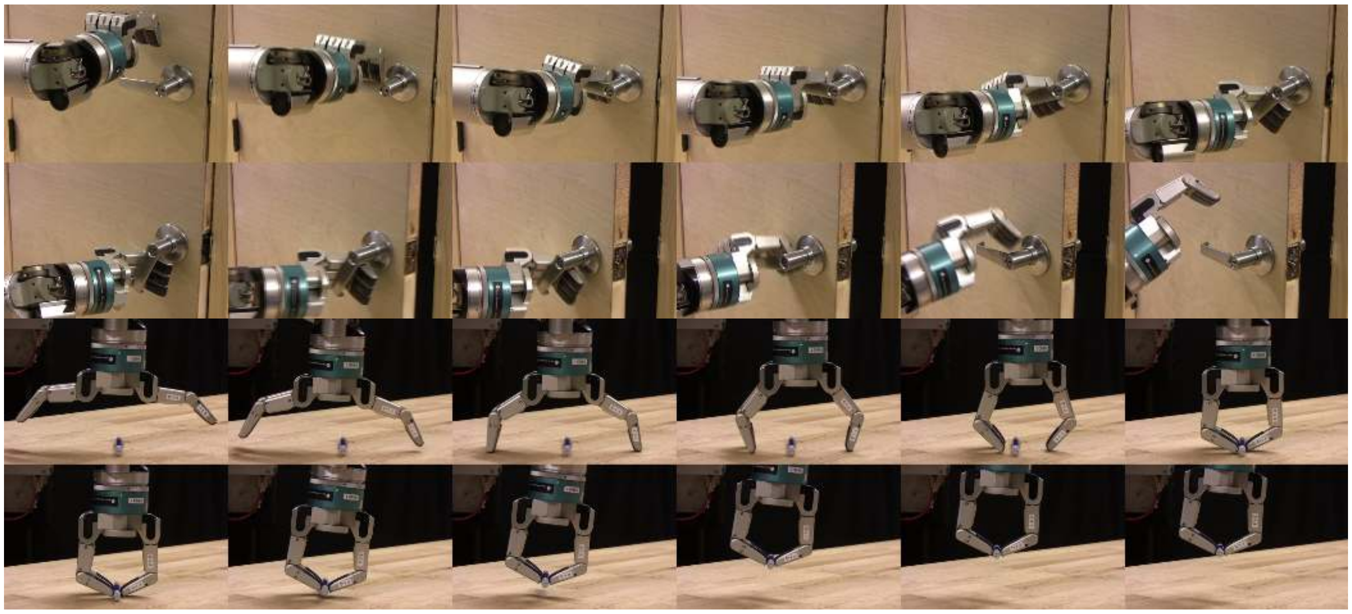


Fig. 9. Sequence of images from executions of the learnt policies for the two manipulation tasks: (top) opening a door with a lever handle, and (bottom) picking up a pen from the table. The attached video shows the learning process and final executions of each of these tasks.

converged to a policy for which all noisy rollouts succeeded. This final policy achieved a success rate of 100%, with a cost of 47.1 $\pm$ 0.8, averaged over 21 trials. Fig. 6(b) shows the evolution of costs during the learning process. For this task, we set the proportional gains of the force controller to 1, but also use an integral gain of 0.1. This allows better force tracking when the fingers are in contact with the table. The learnt force profile and its tracking performance are shown in Fig. 7.

During learning, we keep the position and orientation of the pen fixed, i.e. robustness to position and orientation uncertainty was not incorporated in the learning process. We found, however, that the final learnt policy was robust to position and orientation uncertainty. Fig. 8 shows the extent of position and orientation errors that could be tolerated by our policy. We believe these robustness results could further be improved upon if uncertainty were to be incorporated in the learning process [3].

Finally, we recorded the cartesian position and orientation trajectories from a successful grasp, and replayed these trajectories without the force controller. This policy was unsuccessful at grasping the pen, which shows that the learnt force control policy plays a significant role in achieving the task.

V. CONCLUSION

The need for compliant actuation in robotics is well understood. In this paper, we have presented a learning approach for compliant manipulation. The approach relies on initialization of a desired position trajectory through kinesthetic demonstration, followed by learning of a desired force profile through reinforcement learning. We have successfully demonstrated the application of this approach to two different manipulation tasks. We have also shown that the use of force control policies can potentially allow robots to increase their robustness towards localization errors. Explicit training of these policies for robustness is a promising direction for future work. Another potential improvement can arise from limiting the generation of noise in force/torque space to directions constrained by contact.

REFERENCES

[1] J. Buchli, E. Evangelos Theodorou, F. Stulp, and S. Schaal, "Variable impedance control - a reinforcement learning approach," in *Robotics Science and Systems*, 2010.

[2] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," in *IEEE Intl. Conf. on Robotics and Automation*, 2011.

[3] F. Stulp, E. Theodorou, J. Buchli, and S. Schaal, "Learning to grasp under uncertainty," in *IEEE Intl. Conf. on Robotics and Automation*, 2011.

[4] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with em-based reinforcement learning," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, October 2010, pp. 3232–3237.

[5] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning*, pp. 1–33, 2009.

[6] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Advanced Robotics*, vol. 25, no. 5, pp. 581–603, 2011.

[7] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *Journal of Machine Learning Research*, pp. 3137–3181, 2010.

[8] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems 15*. MIT Press, 2002, pp. 1547–1554.

[9] N. Ratliff, M. Zucker, J. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *IEEE Intl. Conf. on Robotics and Automation*, 2009, pp. 12–17.

[10] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *IEEE Intl. Conf. on Robotics and Automation*, 2011.

[11] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682 – 697, 2008.

[12] A. Dragan, N. Ratliff, and S. Srinivasa, "Manipulation planning with goal sets using constrained trajectory optimization," in *IEEE Intl. Conf. on Robotics and Automation*, May 2011.

[13] B. Siciliano, L. Sciavicco, and L. Villani, *Robotics: modelling, planning and control*. Springer Verlag, 2009.

[14] "Video," http://www.youtube.com/watch?v=LkwQJ9_i6vQ.

[15] R. Featherstone, *Rigid body dynamics algorithms*. Springer-Verlag New York Inc, 2008.