Learning Motion Primitive Goals for Robust Manipulation

Freek Stulp[1], Evangelos Theodorou[1], Mrinal Kalakrishnan[1], Peter Pastor[1], Ludovic Righetti[1], Stefan Schaal[1,2]
[1]Computational Learning and Motor Control Lab, University of Southern California, Los Angeles, CA 90089, USA
[2]Max-Planck-Institute for Intelligent Systems, 72076 Tübingen, Germany

Abstract— **Applying model-free reinforcement learning to manipulation remains challenging for several reasons. First, manipulation involves physical contact, which causes discontinuous cost functions. Second, in manipulation, the end-point of the movement must be chosen carefully, as it represents a grasp which must be adapted to the pose and shape of the object. Finally, there is uncertainty in the object pose, and even the most carefully planned movement may fail if the object is not at the expected position.**

To address these challenges we 1) present a simplified, computationally more efficient version of our model-free reinforcement learning algorithm PI^2; 2) extend PI^2 so that it simultaneously learns shape parameters *and* goal parameters of motion primitives; 3) use shape and goal learning to acquire motion primitives that are robust to object pose uncertainty. We evaluate these contributions on a manipulation platform consisting of a 7-DOF arm with a 4-DOF hand.

This paper is accompanied by a video, which can also be downloaded at: `http://www-clmc.usc.edu/movies/iros2011/`

I. Introduction

In recent years, model-free reinforcement learning algorithms that search directly in policy parameter space have been able to learn a variety of motor skills, such as baseball batting [13], dart throwing [6], table tennis [6], pancake flipping [7], and archery [8]. These are quite intricate tasks, which even most humans find difficult to master. We believe however, that the main potential of direct model-free reinforcement learning lies in applying it to learning everyday object manipulation skills, as required in for instance pick-and-place tasks. Reinforcement learning promises to provide robots with a new level of autonomy and flexibility for acquiring skills for everyday manipulation tasks.

Applying model-free direct reinforcement learning (RL) to manipulation remains challenging for several reasons. First, manipulation involves physical contact, and the transition from non-contact to contact can cause discontinuities in the cost function. Computing a gradient on discontinuous, noisy cost functions is problematic; hence the recent trend from using gradient-based methods [13] to algorithms based on reward-weighted averaging [6], [18]. We recently introduced the Policy Improvement with Path Integrals (PI^2) algorithm, which can outperform existing direct RL algorithms by an order of magnitude both in terms of learning speed and final cost of the learned movement [18].

The second challenge is that in manipulation, the end-point of the movement must be chosen carefully, as it represents a grasp which must be adapted to the pose and shape of the object. However, direct reinforcement learning has so far only been applied to learning the shape of the movement, not the end-point, i.e. the goal. In this paper we therefore apply several simplifications to PI^2 which enable us to derive a new update rule for goal learning with PI^2.

Finally, the pose and/or shape of the object to be manipulated is hardly ever known with perfect accuracy. A previously successful grasp might fail because the object is not at the expected position, which leads to stochastic cost functions [17]. Rather than learning a movement that successfully grasps the object at the expected position, the movement should maximize the expectation of successfully grasping the object over *all* possible positions that the object might have due to uncertainty, as depicted in Fig. 1. Therefore, we propose to use reinforcement learning of shape and goal to learn to maximize this expectation, to yield motion primitives that are robust to object position uncertainty.

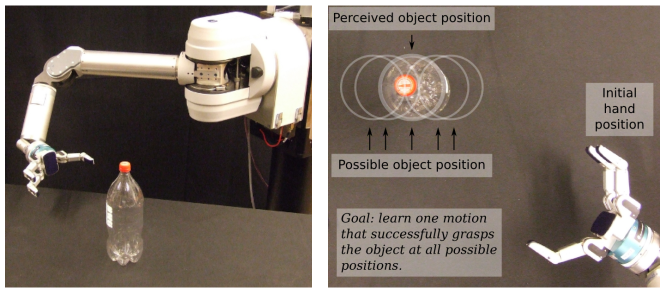
Perceived object position
Initial hand position
Possible object position
Goal: learn one motion that successfully grasps the object at all possible positions.

Fig. 1. Left: The manipulation platform used in this paper. Right: The aim is to learn one motion that successfully grasps the object at all of the positions shown. To do so, we simultaneously learn the goal and shape of a motion primitive with the PI^2 algorithm.

Summarizing, the main contributions of this paper are: • Presenting a simplified, computationally more efficient version of PI^2 (Section IV-A). • Adapting simplified PI^2 to simultaneously learning shape parameters *and* goal parameters of motion primitives (Section IV). • Using these methods to learn reaching trajectories and grasps for manipulation, which are robust towards uncertainty in the object position (Section V). • Empirically evaluating each of these contributions

This research was supported in part by National Science Foundation grants ECS-0325383, IIS-0312802, IIS-0082995, IIS-9988642, ECS-0326095, ANI-0224419, DARPA program on Advanced Robotic Manipulation, and the ATR Computational Neuroscience Laboratories. F.S. was supported by a Research Fellowship from the German Research Foundation (DFG). E.T. was supported by a Myronis Fellowship.

on a manipulation platform consisting of a 7-DOF arm with a 4-DOF hand, depicted in Fig. 1 (Section V-D)[1]

Before presenting these contributions, we first describe related work (Section II), and the existing algorithms on which this work is based (Section III).

II. Related work

Goal Learning. Nemec et al. [11] optimize the goal parameters of a DMP through reinforcement learning, by iteratively approximating a continuous value function in state space. Learning a manipulation task (pouring a liquid) takes place in a 2D space, as 4 of the 6 degrees of freedom at the end-effector pose are kept fixed. Learning directly in policy parameter space without a value function allows us to scale to higher dimensional problems, enabling us to learn goal and shape parameters simultaneously in the full 11D action space (3D end-effector position, 4D end-effector quaternion orientation, and 4D hand posture).

Kober et al. [6] present a variant of goal learning, where the robot learns a function $\gamma(s)$, which maps the current situation s to a set of meta-parameters, which are the goal and duration of the movement. The aim is to learn the mapping $\gamma(s)$ that minimizes a cost. Krömer et al. [9] also determine low-dimensional meta-parameters that automatically adapt the shape of the movement to a pre-specified goal, and apply it to grasping. Our specific aim in this paper is to rather learn the goal and shape parameters simultaneously in the full dimensionality of the motion primitive.

There are several other methods that could potentially be used for learning optimal motion primitive goals, such as cross-entropy methods [14] or reward regression [8]. However, learning the goal of the movement is tightly coupled to learning the shape of the movement, and these methods do not readily apply to learning shape parameters, as shape parameters have a temporally extended effect. Therefore, we propose a method that simultaneously learns shape and goal using the same cost function and update rule.

Grasping Under Uncertainty. Uncertainty in an object's position can make even the most carefully planned motion fail, as the object might simply not be at the location where the planner expects it to be [10], [1]. There are several strategies to dealing with object pose uncertainty in grasping. One strategy is based using a sampling-based motion planner to generate a *robust motion plan* that is consistent with all object pose hypotheses [1]. This approach has high computational cost, and requires an accurate model of the robot and the environment. The second strategy uses exploratory actions to acquire more information to *actively reduce the uncertainty*. An example is using Partially Observable Markov Decision Processes to determine when and which exploratory actions are required to reduce the object pose uncertainty [3]. A third strategy uses *reactive control* during execution, based on feedback from tactile or force/torque sensors, enabling the robot to adapt on-line to cases where the object is not at the expected location [4], [12]. In Section V, we shall see that humans use a fourth strategy, which forms the inspiration for the work described in this paper.

[1]We previously considered the task of learning to grasp under uncertainty in [17]. In contrast to the work in [17], here we • use a simplified version of PI^2 • apply simultaneous learning of goal *and* shape parameters, rather than learning shape alone. As the grasp is the end-point of a reaching movement, this is a more natural representation of the problem than in [17] • we perform experiments with a real robot.

III. Direct Reinforcement Learning

In this section, we briefly introduce Dynamic Movement Primitives and the reinforcement learning algorithm Policy Improvement with Path Integrals (PI^2), thus laying the groundwork for the contributions in Sections IV and V.

A. Dynamic Movement Primitives

Dynamic Movement Primitives (DMPs) are a flexible representation for motion primitives [5], which consist of a set of dynamic system equations:

Dynamic Movement Primitives

$$\frac{1}{\tau}\ddot{x}_t = \alpha(\beta(g - x_t) - \dot{x}_t) + \mathbf{g}_t^T\boldsymbol{\theta} \quad \text{Transform. system} \quad (1)$$

$$[\mathbf{g}_t]_j = \frac{w_j(s_t)\cdot s_t}{\sum_{k=1}^{p} w_k(s_t)}(g - x_0) \quad \text{Basis functions} \quad (2)$$

$$w_j = \exp\left(-0.5h_j(s_t - c_j)^2\right) \quad \text{Gaussian kernel} \quad (3)$$

$$\frac{1}{\tau}\dot{s}_t = -\alpha s_t \quad \text{Canonical. system} \quad (4)$$

The core idea behind DMPs is to perturb a simple linear dynamical system (the first part of Eq. 1) with a non-linear component ($\mathbf{g}_t^T\boldsymbol{\theta}$) to acquire smooth movements of arbitrary shape. The non-linear component consists of basis functions $\mathbf{g}_t$, multiplied with a parameter vector $\boldsymbol{\theta}$.

We leave the details of DMPs to [5], [18]. For this paper, the important features of DMPs are: • When integrated over time, DMPs generate trajectories $[x_{d,t}\ \dot{x}_{d,t}\ \ddot{x}_{d,t}]$, which are used as for instance desired joint angles or desired end-effector positions. • DMPs converge from the initial value x_0 towards the goal parameter g. So at the end of the movement, $x_t = g$. • The general shape of the trajectory (i.e. the values of x_t between x_0 and g) is determined by the shape parameters $\boldsymbol{\theta}$. The effects of changing $\boldsymbol{\theta}$ and g are visualized in Fig. 2.

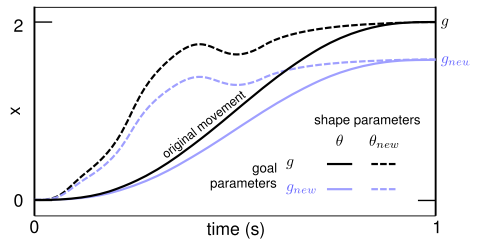

Fig. 2. Effects of changing shape parameters $\boldsymbol{\theta}$ and goal parameters g.

Eq. 1 describes a 1-dimensional system. Multi-dimensional DMP are represented by coupling several dynamical systems equations as in Eq. 1 with one shared phase variable s. For an n-DOF arm for instance, an n-dimensional DMP can be used to generate desired joint angle trajectories. In multi-dimensional DMPs, each dimension has its own goal (g) and shape ($\boldsymbol{\theta}$) parameters.

B. Policy Improvement for Reinforcement Learning

The shape parameters $\boldsymbol{\theta}$ are commonly acquired through imitation learning, i.e. a DMP is trained with an observed trajectory through supervised learning [5]. The aim of policy improvement methods is to tune the policy parameters $\boldsymbol{\theta}$ such that they minimize a cost function. The imitated trajectory is thus not the end result, but rather an initialization for further improvement through learning. In this paper, we consider the generic cost function

$$J(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} (r_t + \frac{1}{2}\boldsymbol{\theta}_t^T \mathbf{R} \boldsymbol{\theta}_t)\ dt \quad \text{Traj. cost} \tag{5}$$

where J is the finite horizon cost over a trajectory $\boldsymbol{\tau}_i$ starting at time t_i and ending at time t_N. This cost consists of a terminal cost ϕ_{t_N}, an immediate cost r_t, and an immediate control cost $\frac{1}{2}\boldsymbol{\theta}_t^T \mathbf{R} \boldsymbol{\theta}_t$. This formulation of the cost function is very generic indeed, as ϕ_{t_N} and r_t may be chosen freely, and must not be continuous or differentiable. The cost function J is task-dependent, and provided by the user. Example cost functions are given in the experiments in Section IV through V.

Policy improvement methods minimize cost functions through an iterative process of exploration and parameter updating, which we explain using Fig. 3. Exploration is done by executing a DMP K times, each time with slightly different parameters $\boldsymbol{\theta} + \boldsymbol{\epsilon^\theta}_{t,k}$, where $\boldsymbol{\epsilon^\theta}_{t,k}$ is noise which is added to explore the parameter space. This noise is sampled from a Gaussian distribution with variance $\boldsymbol{\Sigma^\theta}$.

$$\frac{1}{\tau}\ddot{x}_t = \alpha(\beta(g - x_t) - \dot{x}_t) + \mathbf{g}_t^T (\underbrace{\boldsymbol{\theta} + \boldsymbol{\epsilon^\theta}_{t,k}}_{\text{Shape exploration}}) \tag{6}$$

These 'noisy' DMP parameters generate slightly different movements $[\ddot{x}_{t,k}, \dot{x}_{t,k}, x_{t,k}]$, which each lead to different costs. Given the costs and noisy parameters of the K DMP executions, called *trials*, policy improvement methods then update the parameter vector $\boldsymbol{\theta}$ such that it is expected to generate movements that lead to lower costs in the future. The process then continues with the new $\boldsymbol{\theta}$ as the basis for exploration.

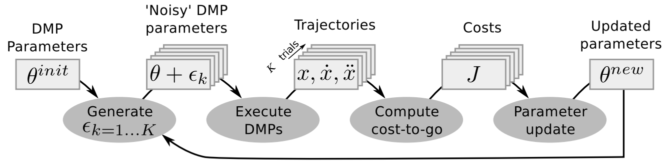

Fig. 3. Generic loop of policy improvement algorithms.

C. PI² Algorithm

The most crucial part of the policy improvement loop in Fig. 3 is the parameter update; it is here that the key differences between PI² and other policy improvement methods lie. Rather than focussing on its derivation from first principles of stochastic optimal control, which is presented extensively in [18], we provide a post-hoc interpretation of the resulting update rule. For an in-depth comparison with related approaches, and a theoretical and empirical analysis of the advantages of PI², we also refer to [18].

PI² Shape Parameter Update Rule

$$S(\boldsymbol{\tau}_{i,k}) = \phi_{t_N,k} + \sum_{j=i}^{N-1} r_{t_j,k} + \frac{1}{2}\sum_{j=i+1}^{N-1} (\boldsymbol{\theta} + \mathbf{M}_{t_j,k}\boldsymbol{\epsilon^\theta}_{t_j,k})^T \mathbf{R} (\boldsymbol{\theta} + \mathbf{M}_{t_j,k}\boldsymbol{\epsilon^\theta}_{t_j,k}) \tag{7}$$

$$\mathbf{M}_{t_j,k} = \frac{\mathbf{R}^{-1}\mathbf{g}_{t_j}\ \mathbf{g}_{t_j}^T}{\mathbf{g}_{t_j}^T \mathbf{R}^{-1}\mathbf{g}_{t_j}} \tag{8}$$

$$P(\boldsymbol{\tau}_{i,k}) = \frac{e^{-\frac{1}{\lambda}S(\boldsymbol{\tau}_{i,k})}}{\sum_{l=1}^{K}[e^{-\frac{1}{\lambda}S(\boldsymbol{\tau}_{i,l})}]} \tag{9}$$

$$\delta\boldsymbol{\theta}_{t_i} = \sum_{k=1}^{K} \left[P(\boldsymbol{\tau}_{i,k})\ \mathbf{M}_{t_i,k}\ \boldsymbol{\epsilon^\theta}_{t_i,k}\right] \tag{10}$$

$$[\delta\boldsymbol{\theta}]_j = \frac{\sum_{i=0}^{N-1}(N-i)\ w_{j,t_i}\ [\delta\boldsymbol{\theta}_{t_i}]_j}{\sum_{i=0}^{N-1} w_{j,t_i}(N-i)} \tag{11}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta\boldsymbol{\theta} \tag{12}$$

Fig. 4. PI² update rule:
Eq. 7 – Determine cost-to-go of each trial. Compute the cost-to-go $S(\boldsymbol{\tau}_{i,k})$ at each time step i and for each trial k. This is an evaluation of the cost function $J(\boldsymbol{\tau}_i)$ in Equation 5 , which is task-dependent and provided by the user. The matrix $\mathbf{M}_{t_j,k}$ (Eq. 8) is needed to project the exploration noise onto the parameter space.
Eq. 9 – Compute probability of each trial. Compute the probability $P(\boldsymbol{\tau}_{i,k})$ of each trial k at each time step i by exponentiating the cost-to-go. The intuition behind this step is that trajectories of lower cost should have higher probabilities. This intuition also has a rigorous mathematical representation through the exponentiation of the value function [18].
Eq. 10 – Average over trials. Compute the parameter update $\delta\boldsymbol{\theta}$ for each time step i through probability weighted averaging over the exploration $\boldsymbol{\epsilon^\theta}$ of all K trials. Trajectories with higher probability, and thus lower cost, therefore contribute more to the parameter update. Again, $\mathbf{M}_{t_j,k}$ is needed to project the exploration noise onto the parameter space.
Eq. 11 – Average over time-steps. In the final step, we average the parameter update $\delta\boldsymbol{\theta}_{t_i}$ per time step i over all time steps. Each parameter update is weighted according to the number of steps left in the trajectory. This is to give earlier points in the trajectory higher weights, as they influence a larger part of the trajectory. They are also weighted with the activation of the corresponding basis function w_j at time t_i, as the influence of parameter θ_j is highest when w_j is highest. Finally, the actual parameter update is performed with Eq. 12.

IV. Goal Learning with PI²

In this section, we explain how a simplified version of PI² is used to not only update the shape parameters θ, but also the goal parameters g. Learning the end-point of the movement is a key aspect of learning motion primitives for grasping.

A. Simplification of PI^2

In Eq. 8, we see that $\mathbf{M}_{t_i,k}$ depends only on $\mathbf{R}$, which is constant, and $\mathbf{g}_{t_i}$. When looking carefully at the basis vector $\mathbf{g}_{t_i}$ in Eq. 2 one realizes that $\mathbf{g}_{t_i}$ depends only on the phase variable s_t, which is common for all sampled trajectories. Thus the projection matrix can be pushed out of the summation and Eq. 10 takes the form:

$$\delta\boldsymbol{\theta}_{t_i} = \mathbf{M}_{t_i} \sum_{k=1}^{K} \left[P\left(\boldsymbol{\tau}_{i,k}\right)\ \boldsymbol{\epsilon}^{\boldsymbol{\theta}}{}_{t_i,k} \right] \tag{13}$$

When the new parameters $(\boldsymbol{\theta}_{t_i} + \delta\boldsymbol{\theta}_{t_i})$ are used in the policy (Eq. 1), they are multiplied by $\mathbf{g}_{t_i}^T$. Therefore we have

$$\mathbf{g}_{t_i}^T \left(\boldsymbol{\theta}_{t_i} + \delta\boldsymbol{\theta}_{t_i}\right) = \tag{14}$$
$$= \mathbf{g}_{t_i}^T\boldsymbol{\theta}_{t_i} + \mathbf{g}_{t_i}^T\delta\boldsymbol{\theta}_{t_i} \tag{15}$$
$$= \mathbf{g}_{t_i}^T\boldsymbol{\theta}_{t_i} + \mathbf{g}_{t_i}^T\mathbf{M}_{t_i,k} \sum_{k=1}^{K} \left[P\left(\boldsymbol{\tau}_{i,k}\right) \boldsymbol{\epsilon}^{\boldsymbol{\theta}}{}_{t_i,k} \right] \tag{16}$$
$$= \mathbf{g}_{t_i}^T\boldsymbol{\theta}_{t_i} + \mathbf{g}_{t_i}^T \frac{\mathbf{R}^{-1}\mathbf{g}_{t_i}\ \mathbf{g}_{t_i}^T}{\mathbf{g}_{t_i}^T\mathbf{R}^{-1}\mathbf{g}_{t_i}} \sum_{k=1}^{K} \left[P\left(\boldsymbol{\tau}_{i,k}\right) \boldsymbol{\epsilon}^{\boldsymbol{\theta}}{}_{t_i,k} \right] \tag{17}$$
$$= \mathbf{g}_{t_i}^T\boldsymbol{\theta}_{t_i} + \mathbf{g}_{t_i}^T\delta\boldsymbol{\theta}_{t_i}^{new} \tag{18}$$

The term $\mathbf{M}_{t_i}$ thus drops, and the new update of the policy is $\delta\boldsymbol{\theta}_{t_i}^{new} = \sum_{k=1}^{K} \left[P\left(\boldsymbol{\tau}_{i,k}\right) \boldsymbol{\epsilon}^{\boldsymbol{\theta}}{}_{t_i,k} \right]$. This is equivalent to substituting $\mathbf{M}_{t_i} = \mathbf{I}$ in the previous version PI^2. It should be noted, however, that this simplification cannot be applied in general, but rather it depends on how the parameter noise $\boldsymbol{\epsilon}^{\boldsymbol{\theta}}{}_{t,k}$ is generated, and which space is spanned by the basis function $\mathbf{g}_{t_i}$.

B. Applying Simplified PI^2 Update Rule to Goal Learning

PI^2 can optimize multiple sets of parameters at the same time [18]. To include goal learning, we add another differential equation to the DMP equations in form of:

$$\frac{1}{\tau}\dot{g}_t = \alpha_g(\underbrace{g + \epsilon^g{}_k}_{\text{Goal exploration}} - g_t) \tag{19}$$

$$\frac{1}{\tau}\ddot{x}_t = \alpha(\beta(g_t - x_t) - \dot{x}_t) + \mathbf{g}_t^T(\underbrace{\boldsymbol{\theta} + \boldsymbol{\epsilon}^{\boldsymbol{\theta}}{}_{t,k}}_{\text{Shape exploration}}) \tag{20}$$

The goal exploration noise ϵ^g is drawn from a Gaussian with variance $\boldsymbol{\Sigma}^g$ at the beginning of a trajectory and kept constant throughout the roll-out. We assume that α_g is chosen so large that, effectively, the goal state g_t converges immediately to $g + \epsilon^g{}_k$. In the goal parameter update rule in Eq. 21–23, the cost-to-go at $t = 0$ is used to compute the probability $P(\boldsymbol{\tau}_{0,k})$. This means that we are using the total cost of the trajectory. The motivation behind this is that as the effect of g remains constant during execution, there is no temporal dependency of g on the cost. Note that $P(\boldsymbol{\tau}_{0,k})$ in Eq. 21 is equivalent to Eq. 9, with $t = 0$. Thus if shape parameters $\boldsymbol{\theta}$ are updated first, $P(\boldsymbol{\tau}_{0,k})$ is shared with the shape parameter update, and this probability need not be computed again. Probability weighted averaging (Eq. 22) and goal updating (Eq. 23) are equivalent to the update rule for $\boldsymbol{\theta}$.

PI^2 Goal Parameter Update Rule

$$P\left(\boldsymbol{\tau}_{0,k}\right) = \frac{e^{-\frac{1}{\lambda}J(\boldsymbol{\tau}_{0,k})}}{\sum_{l=1}^{K}[e^{-\frac{1}{\lambda}J(\boldsymbol{\tau}_{0,l})}]} \quad \text{Probability} \tag{21}$$

$$\delta g = \sum_{k=1}^{K} \left[P\left(\boldsymbol{\tau}_{0,k}\right)\ \epsilon^g{}_k \right] \quad \text{Weighted averaging} \tag{22}$$

$$g \leftarrow g + \delta g \quad \text{Update} \tag{23}$$

By updating g in a similar fashion to updating $\boldsymbol{\theta}$, several important advantages are inherited from the PI^2 shape update rule: • Discontinuous cost functions are not a problem, as probability weighted averaging does not rely on computing a gradient. • Due to the averaging, $g + \delta g$ always lies within the convex hull of $g = g + \epsilon_k$. Thus, if the exploration is safe (joint limits are respected, the robot does not collide with itself or the environment), the new g after updating will also be safe. • Since g and $\boldsymbol{\theta}$ are updated simultaneously using the exact same costs and thus the same probability weights, there is no negative interference between learning g and $\boldsymbol{\theta}$ [2].

C. Example Via-point Task

In the next section, we will consider the main application task – learning to grasp under uncertainty. In this section, we first evaluate shape and goal learning in the context of a simple 3D via-point task for illustration purposes. In this task, the output $[x_{d,t}\ \dot{x}_{d,t}\ \ddot{x}_{d,t}]$ of the 3-dimensional DMP represents the desired end-effector trajectory in Cartesian space. The initial DMP is trained with a straight minimum-jerk trajectory of 2 seconds, from the initial position x_0 to the goal position g. The aim of this task is to learn to move the end-effector through a via-point, whilst minimizing the acceleration at the end-effector. We express this aim by implementing the cost function in Eq. 5 as follows:

$$J_{wp}(\boldsymbol{\tau}_i) = \int_{t_i}^{t_N} (C(t) + 10^{-4}(\ddot{\mathbf{x}}_t)^2 + \frac{1}{2}\boldsymbol{\theta}_t^T\mathbf{R}\boldsymbol{\theta}_t)\ dt \tag{24}$$

$$C(t) = \delta(t - 1.0)|\ \mathbf{x} - [0.55\ 0.90\ 1.20]^T\ | \tag{25}$$

Here, $C(t)$ is the distance to the via-point $[0.55\ 0.90\ 1.20]$ at $t = 1.0$. Furthermore, end-effector accelerations are penalized at each time step with $10^{-4}(\ddot{\mathbf{x}}_t)^2$ to avoid large accelerations. The control cost matrix is $\mathbf{R} = 10^{-6}\mathbf{I}$.

[2]Note that shape parameters $\boldsymbol{\theta}$ are usually kept fixed after having been optimized with policy improvement methods, and the goal parameter g is used to adapt the end-point of the motion to specific scenarios, for instance to reach to different locations [5]. Therefore, it might seem counter-intuitive to optimize g to a specific task context. But g is only specific to that context if it is learned in a global frame of reference. In the case of object grasping, g should be learned relative to the object, thus representing an offset to the object (e.g. grasp 5cm to the right of the object). Therefore, if the object moves 20cm to the left for instance, so does g.

With this set-up, we perform three experiments: 1) learning only shape $\boldsymbol{\theta}$, 2) learning only the goal g, 3) learning both. For all these experiments, the number of trials per update is $K = 5$. The exploration noise for shape is $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}^{\boldsymbol{\theta}}} = 20.0\gamma^u$, and for goals is $\boldsymbol{\Sigma}_g = 0.05\gamma^u$, where u is the number of updates so far, and $\gamma = 0.7$. Learning of θ or g is easily switched off by setting the respective exploration noise to 0. The results are depicted in Fig. 5.

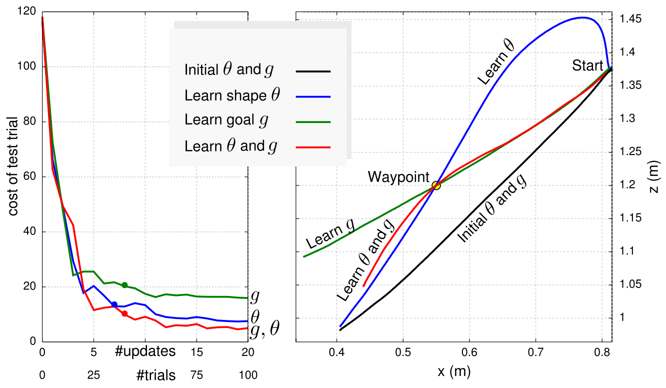

Fig. 5. Results of the via-point experiment on the real robot. The left graph depicts the learning curves, and the right graph initial and optimized end-effector trajectories, projected onto the x, z-plane for clarity.

From these results, we draw the following conclusions: • In all experiments, the robot quickly learns to go through the via-point. The update after which the distance to the via-point drops below 5mm is indicated by a dot in the learning curves. After these points, optimization is mainly concerned with fine-tuning $\boldsymbol{\theta}$ to minimize the acceleration. • Learning only $\boldsymbol{\theta}$ (blue) – as is to be expected, the robot adapts the shape of the motion so that the trajectory passes through the via-point. As g is not learned, the DMP converges to the same goal position as before learning. • Learning only g (green) – as the shape parameters of the straight trajectory are not changed, the robot places the goal behind the via-point, such that the trajectory passes through it. Since the robot cannot change $\boldsymbol{\theta}$, it can also do little to minimize the cost due to acceleration, and the final cost is much higher than when learning $\boldsymbol{\theta}$. • Learning both $\boldsymbol{\theta}$ and g (red) – here the robot has the best of both worlds. The goal and shape of the movement are adapted such that the final cost is lower than when optimizing them individually. Since learning is as quick as in the first two experiments, there is apparently no interference between learning $\boldsymbol{\theta}$ and g.

In summary, learning shape $\boldsymbol{\theta}$ and goal g simultaneously with PI^2 is as quick as learning shape $\boldsymbol{\theta}$ alone, but leads to lower costs in this task.

V. Application Domain – Grasping under Uncertainty

In Section II, we discussed three strategies to dealing with object position uncertainty: 1) generate a robust motion plan; 2) actively reduce the uncertainty; 3) use reactive control. In recent bio-physical experiments, Christopoulos and Schrater [2] demonstrate that humans use a fourth strategy to deal with position uncertainty. In their experimental set-up, subjects perceived an object to be at a certain position, but the actual position (which was hidden from the subject) when grasping the object was sampled from a Gaussian distribution. This *environmentally induced position uncertainty* artificially increases the uncertainty in the object's position. It was shown that over time, humans adapt their reaching motion and grasp to the shape of the uncertainty distribution, and that these adaptations lead to significantly better force-closure performance. Thus, rather than optimizing force-closure for the expected object position, humans learn one motion that optimizes the average force-closure over *all* positions in the distribution.

In this paper, we apply this experimental paradigm to robotics, by formulating it as a reinforcement learning problem. We do so by inducing position uncertainty during learning, and penalizing failed grasps with a cost. Since the resulting motion primitive has been trained off-line to deal with uncertainty that we have induced, it is more robust to state estimation uncertainty on-line during task execution.

Our approach of learning robust motion primitives may well be combined with methods that actively reduce uncertainty [3], or use reactive control based on sensor feedback [4], [12]. In fact, we believe that learning motions that have an intrinsic capability to deal with uncertainty is one of many approaches, albeit an important one, which are necessary to achieve truly robust robotic manipulation.

A. Initialization of the Motion Primitive for Grasping

The task is to grasp a plastic bottle with a radius of 5cm, as depicted in Fig. 1. The Dynamic Movement Primitive we use for this task has 11 dimensions; 3 to represent the position of the end-effector, 4 for its quaternion orientation, and 4 for the joint angles of the hand. The initial grasp is demonstrated to the robot, which constitutes the goal g which is optimized with PI^2.

Then, the shape of the DMP is initialized with two minimum jerk trajectories. One moves the hand from the initial pose to the final demonstrated pose in 3s. The second sub-motion closes the gripper in 1s. These trajectories are represented with $\boldsymbol{\theta}$, which is also optimized with PI^2. Fig. 6 illustrates these trajectories. The video attachment also shows the initial movement before learning.

B. Formulation as a Reinforcement Learning Problem

The cost function for this task is

$$J_{gr}(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} (10^{-7}(\ddot{\mathbf{x}}_t)^2 + \frac{1}{2}\boldsymbol{\theta}_t^T \mathbf{R} \boldsymbol{\theta}_t)\, dt \tag{26}$$

$$\phi_{t_N} = 100(1 - \text{success of lifting}) \tag{27}$$

where $\mathbf{R} = 10^{-5}\mathbf{I}$. The terminal cost ϕ_{t_N} reflects if the robot successfully grasped and lifted the object, and is determined during the 1s lifting motion after executing the DMP. The 'success of lifting' is determined by measuring the time (0s-1s) each finger was in contact with the object, and averaging over the 3 fingers. For instance, if all three fingers were in

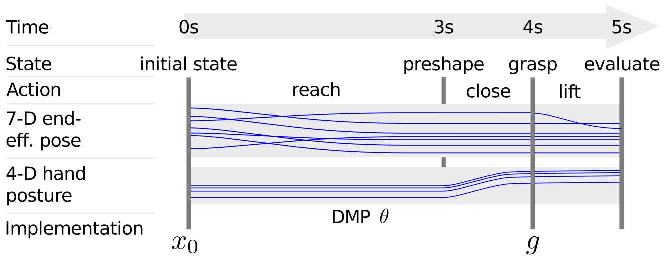

Fig. 6. Grasp procedure used in this section. The DMP generates desired trajectories for the 7-D end-effector pose (3-D position and 4-D quaternion orientation) and 4 joint angles of the hand, from the initial state x_0 to the grasp posture g. The shape of this reaching movement is determined by $\boldsymbol{\theta}$. The whole movement takes 4 seconds. After grasping, the robot lifts the object, after which the grasp is evaluated.

contact with the objects throughout the lifting motion, the cost is $\phi_{t_N} = 100(1-1) = 0$, if the object was knocked over and not lifted at all $\phi_{t_N} = 100(1-0) = 100$, and if only two fingers were in contact and the object dropped out after $0.4s$, $\phi_{t_N} = 100(1 - avg(0.4 + 0.4 + 0.0)) = 73.3$. A finger is deemed to be in contact with the object if the value of the strain gauge of the finger exceeds a certain threshold.

The exploration noise for shape is $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}^{\boldsymbol{\theta}}} = 3.0$, 15.0 and 30.0 for the position, quaternion orientation and finger joint angles respectively. For the goals $\boldsymbol{\Sigma}_g = 0.03$, 0.07, 0.03 respectively. Again, all exploration decays with γ^u, where u is the number of updates so far, and $\gamma = 0.85$.

C. Environmentally Induced Position Uncertainty

In this paper, we use two uncertainty distributions for the object position. The object is placed {-6,-4,0,4,6} centimeters from the perceived object position along either the x or y-axis. The 5 positions along the x-axis are depicted in Fig. 1. These distributions are intended to represent the $\mu \pm 2\sigma$ intervals for $\sigma = \{2\text{cm}, 3\text{cm}\}$; typical values for our state estimation module. The 'perceived' object position is made known to the robot, but not the actual possibly displaced position, allowing us to control the amount of uncertainty as in [2]. The robot must thus learn a motion that not only successfully grasps the object at the expected position, but also at the other positions in the distribution.

During learning, the robot executes the same exploratory motion for each of the 5 object positions. The cost for this exploratory motion is than the average of these 5 trials. We thus use the *expected* cost of an exploratory motion. For instance, if the robot successfully grasped 3 out of 5 objects, the cost for failed grasping for this exploratory motion is (0+0+0+100+100)/5=40. For each PI^2 update, we perform 10 new trials to compute the expected cost for 2 exploratory motions. To accelerate learning, we also reuse the 10 best trials from the previous update, as described in [16].

D. Results

For each of the two uncertainty distributions (5 positions aligned with either the x or y-axis), 3 learning sessions were performed with 10 updates per session. Fig. 7 depicts the learning curves for these 6 learning sessions. The variation between learning session is rather large in the beginning. But after 7 updates (80 trials), all motions lead to successful grasping and lifting of objects at all the positions, although one session ends with the robot lifting the object rather precariously with only two fingers. The command costs and costs due to acceleration increase, to enable the movement that more successfully grasps the objects.

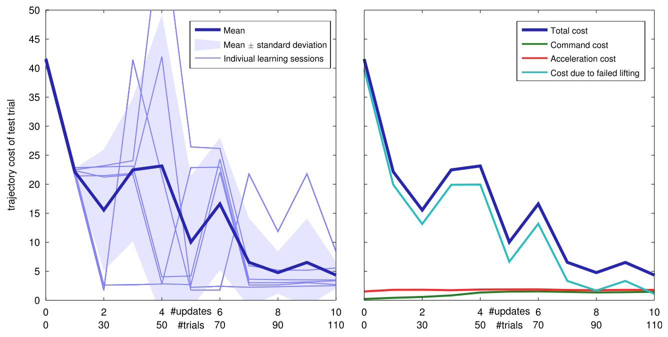

Fig. 7. Learning curves of the grasping under uncertainty experiments. Left: Learning curves of the individual learning sessions and their mean. Right: Mean learning curves, split into the different cost components that constitute the total cost.

Initially before learning, the average cost over both uncertainty distributions is 42, and after learning it is 4 (averaged over the 6 learned movements). To analyze the importance of learning shape *and* goal, we executed the learned shape with the initial goal and vice versa. When the goal g is set to the initial value, but the learned $\boldsymbol{\theta}$s are used, the average cost is 32. When the learned goals g are used, but the initial $\boldsymbol{\theta}$, the average cost is 56. Learning both goal and shape adaptation is thus essential in achieving the low cost of 4, which corresponds to always successfully lifting the object.

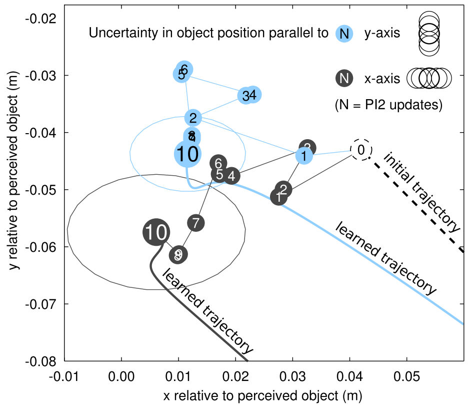

Fig. 8. The location of the goals g during learning after 0...10 updates, averaged over 3 learning sessions per uncertainty distribution. The trajectories towards the goal before (update=0) and after learning (update=10) are depicted as well. The standard deviation in the goal location after learning (update=10) over the 3 learning session is depicted as a circle around the final location.

To analyze the learned movements, Fig. 8 depicts how the location of the goals change during learning. Here, the mode of the object distribution is at (0,0). The goals are adapted in different directions, depending on the orientation of the uncertainty in object position. In both cases, they move closer to the objects to reach far enough to grasp the furthest ones in the distribution.

Because not all adaptations are consistent across all movements, we highlight some features of the specific motion shown in the video attachment in Fig. 9. It depicts the desired postures of the hand before and after a learning session where the object uncertainty distribution is aligned along the y axis.

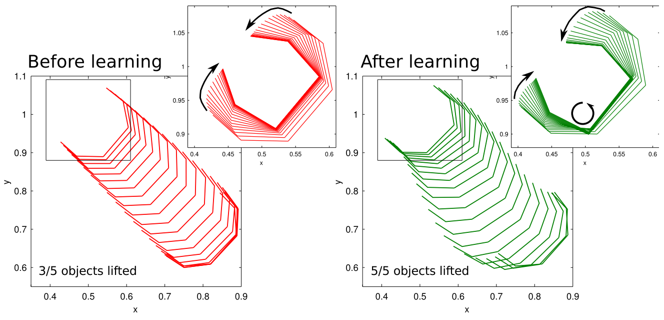

Fig. 9. Hand posture during the motion of one of the learning sessions, before and after learning. Larger graphs depict the reaching motion for [0s-2.8s] and smaller insets the closing of the gripper [2.8s-4s]. Before and after learning, the fingers of the hand close symmetrically. But after learning, the tip of the left finger moves less than the right finger (see arrows). This is because the hand is pivoting around a point near the left finger (see circle with arrow). In essence, the hand is wrapping around the object, thus avoiding collisions. The S-curve motion during reaching is seemingly a preparation for this pivoting. As this leads to a higher success rate, and thus is an adaptation to the uncertainty in the object position.

In summary, PI^2 is able to adapt the shape and goal of motion primitives so that they are robust towards state estimation uncertainty in the position of the object to be grasped.

VI. Conclusion

In this paper we have derived a simplified version of the PI^2 algorithm, and adapted it to learn goal parameters of motion primitives. Learning goal parameters is especially useful for manipulation tasks, where the goal represents a grasp, which must be carefully adapted to the (uncertainty in the position of the) object.

In our previous work, we have demonstrated that learned movements generalize well to different object positions on the table [17]. For instance, a motion primitive optimized to grasp an object at one perceived position was able to successfully grasp all perceived objects (with the same position uncertainty distribution) within a 40x30cm area. We also demonstrated that robust motion primitives can be learned for more complex non-convex objects. We expect these generalization properties to also hold when goal parameters are learned, and we are continuing our empirical evaluations to verify this.

Appendix

The robotic platform used in this paper is depicted in Fig. 1, and consists of a 7-DOF Barret WAM arm with a three-fingered 4-DOF Barret BH280 hand. Low-level control and physical simulations of the robot are done with the SL software package [15], and high-level communications with the Robot Operating System `www.ros.org`. 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. Feed-forward inverse dynamics torques for the arm are obtained from a recursive Newton Euler algorithm. Feed-back joint torques are obtained from low-gain joint PD controllers. All our controllers run at a rate of 300Hz on a host computer running the Xenomai real-time operating system.

References

[1] Dmitry Berenson, Siddhartha S. Srinivasa, and James J. Kuffner. Addressing pose uncertainty in manipulation planning using task space regions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS*, 2009.

[2] V.N. Christopoulos and Paul R. Schrater. Grasping objects with environmentally induced position uncertainty. *PLOS Computational Biology*, 5(10), 2009.

[3] K. Hsiao, L. Kaelbling, and T. Lozano-Perez. Task-driven tactile exploration. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

[4] Kaijen Hsiao, Sachin Chitta, Matei Ciocarlie, and E. Gil Jones. Contact-reactive grasping of objects with partial shape information. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[5] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[6] J. Kober, E. Oztop, and J. Peters. Reinforcement learning to adjust robot movements to new situations. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

[7] P. Kormushev, S. Calinon, and D. G. Caldwell. Robot motor skill coordination with EM-based reinforcement learning. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 2010.

[8] P. Kormushev, S. Calinon, R. Saegusa, and G. Metta. Learning the skill of archery by a humanoid robot icub. In *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, Nashville, TN, USA, December 2010.

[9] O. Kroemer, R. Detry, J. Piater, and J. Peters. Combining active learning and reactive control for robot grasping. *Robotics and Autonomous Systems*, 58(9):1105–1116, 2010.

[10] A. Morales, E. Chinellato, A. H. Fagg, and A. P. del Pobil. Using experience for assessing grasp reliability. *International Journal of Humanoid Robotics*, 1(4):671–691, 2004.

[11] B. Nemec, M. Tamosiunaite, F. Wörgöter, and A. Ude. Task adaptation thorough exploration and action sequencing. In *9th IEEE-RAS International Conference on Humanoid Robots*, 2009.

[12] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal. Online movement adaptation based on previous sensor experiences. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, USA, 2011.

[13] J. Peters. *Machine learning of motor skills for robotics.* PhD thesis, Department of Computer Science, 2007.

[14] R.Y. Rubinstein and D.P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning.* Springer-Verlag, 2004.

[15] Stefan Schaal. The SL simulation and real-time control software package. Technical report, University of Southern California, 2009.

[16] Freek Stulp, Jonas Buchli, Evangelos Theodorou, and Stefan Schaal. Reinforcement learning of full-body humanoid motor skills. In *10th IEEE-RAS International Conference on Humanoid Robots*, 2010. *Best paper finalist.*

[17] Freek Stulp, Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. Learning to grasp under uncertainty. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[18] E. Theodorou, J. Buchli, and S. Schaal. A generalized path integral approach to reinforcement learning. *Journal of Machine Learning Research*, 11(Nov):3137–3181, 2010.