Skill Learning and Task Outcome Prediction for Manipulation

Peter Pastor[1]   Mrinal Kalakrishnan[1]   Sachin Chitta[2]   Evangelos Theodorou[1]   Stefan Schaal[1]

Abstract— **Learning complex motor skills for real world tasks is a hard problem in robotic manipulation that often requires painstaking manual tuning and design by a human expert. In this work, we present a Reinforcement Learning based approach to acquiring new motor skills from demonstration. Our approach allows the robot to learn fine manipulation skills and significantly improve its success rate and skill level starting from a possibly coarse demonstration. Our approach aims to incorporate task domain knowledge, where appropriate, by working in a space consistent with the constraints of a specific task. In addition, we also present an approach to using sensor feedback to learn a predictive model of the task outcome. This allows our system to learn the proprioceptive sensor feedback needed to monitor subsequent executions of the task online and abort execution in the event of predicted failure. We illustrate our approach using two example tasks executed with the PR2 dual-arm robot: a straight and accurate pool stroke and a box flipping task using two chopsticks as tools.**

I. INTRODUCTION

As robots move into the real world, they will have to acquire complex perceptuo-motor skills. Skill learning is, in general, a hard task for robots. A common approach to this problem has been to use learning from demonstration to teach the robot a new skill, e.g. as in teaching an industrial manipulator to perform a particular movement sequence. This approach often involves a human expert who can guide the robot to do the task correctly but must also manually tune the parameters of the task carefully to achieve accurate performance. This manual process is often laborious and significantly reduces the flexibility of the robot in learning a variety of tasks. Moreover, such teaching approaches often neglect the interplay of movement and perception. Conversely, learning entire tasks from scratch by exploring the complete state space is intractable due to the high dimensionality of modern mobile manipulation systems.

In this paper, we present an approach to learning skills which bootstraps the skill from imitation learning, refines it with reinforcement learning, and finally learns to anticipate the quality of performance from the statistics of sensory data during the movement. An initial solution (or initial policy) is most often demonstrated by a human expert using an appropriate teleoperation framework or motion capture setup. This initial policy is often feasible but not very robust, i.e. its success rate on a series of repeated experiments will often be

[1] Peter Pastor, Mrinal Kalakrishnan, Evangelos Theodorou, and Stefan Schaal are with the CLMC Laboratory, University of Southern California, Los Angeles, USA {pastorsa, kalakris, etheodor, sschaal}@usc.edu
[2] Sachin Chitta is with Willow Garage Inc., Menlo Park, CA 94025, USA sachinc@willowgarage.com

very low. Our approach uses a Reinforcement Learning (RL) approach to explore a region around this policy to obtain an optimal solution (final policy). Our approach is based on the PI^2 algorithm [1], which, due to its minimal number of manual tuning parameters, has been shown to work robustly and efficiently in contrast to previous RL approaches, even in very high dimensional motor systems.

Fig. 1: The PR2 robot learning a pool stroke and manipulating a box using chopsticks.

While learning a robust policy in the proposed manner will allow the robot to execute the desired skills, it is important to note that due to the uncertainty inherent in real-world tasks, the robot may still fail from time to time. Thus, the ability to predict the outcome of a task based on previous experience is very valuable, and in particular, to predict a negative outcome *before* the failure actually occurred. Humans seem to learn to predict sensory information for most of the tasks they perform on a daily basis. For example during walking, humans continuously predict the ground reaction forces that they expect to feel at the soles of their feet, enabling them to instantly detect and react to deviations that might arise due to changes in ground terrain (like stepping into a pothole). For robots, such prediction of performance and sensory events is either largely missing or, if it exists, is the result of the design of a human expert who may have to spend a lot of time manually tuning thresholds.

In this paper, we introduce a method of using sensory statistics from multiple trials for a specific skill to predict the outcome of new executions of the same skill. We demonstrate the generality of our approach through an application to learning skills for two particular tasks using the PR2 mobile manipulation system. The first task involves learning a straight, accurate and fast pool shot. This behavior had been previously demonstrated on the PR2 [2] using a hand-tuned behavior that required additional hardware to be added to the robot. In contrast, starting from an initial demonstration, we learn a fast and accurate stroke motion for the robot gripping a pool stick in a more natural manner. The second task involves the use of additional tools in the form of a chopstick grasped in each hand to flip a box placed on a table. This task is a fine manipulation task requiring more dexterity than the pool task. We demonstrate how proprioceptive sensor information can be used to achieve a more gentle execution of this task while also providing a predicted outcome of the task during execution.

The paper is structured as follows. After a brief literature review in the next section, Section III presents (in brief) background information on DMPs and the PI^2 algorithm. In Section IV, we describe in detail our approach to manipulation problems. We describe the two parts of the approach and illustrate them with two example tasks in Section V. We conclude with a discussion of our experimental results and future avenues for research.

II. RELATED WORK

Encoding new motor skills in a robot is a challenging task. Although it is often easy to intuitively describe how a robot should act in a particular task, it is often difficult (due to the high dimensionality of the search space) to find the necessary policy parameters that embody a desired behavior. Therefore, recent work has turned towards using human demonstrations to extract feasible policies and focus the search around these policies [3]. Trajectory-based Reinforcement Learning (RL) algorithms [4], [5] that have been applied to continuous domains and real robotic systems most often follow this approach. Parameterized control policies have been employed to cope with the high dimensionality of real world tasks and reduce the search space. In particular, recent trends towards probabilistic reinforcement learning methods have made major progress towards algorithms that can work in high-dimensional continuous state-action spaces [6], [1], [7]

Dynamic Movement Primitives (DMPs) [8] have been used as one such compact policy representation and have been successfully applied in many RL experiments including biped locomotion [9], ball-in-cup manipulation [10], table tennis, dart throwing [11], and pan cake flipping [12]. The initial set of policy parameters for the motor primitives were obtained from human demonstration in all these cases. A large number of related project have been published in recent years [13], [14].

We would also like to point out the approach by Calinon et. al [15], where Gaussian Mixture models were used to form motor primitives, and the statistics of multiple demonstrations during imitation learning was used to find important sensory features in a movement. This approach has similarities to the way we do data mining in sensory trajectories for performance prediction.

III. LEARNING MOTOR SKILLS

Our approach to learning motor skills involves two parts: a compact representation for the policies using DMPs and a RL based approach to learning the skill itself using the PI^2 algorithm. We will now describe these in more detail.

A. Dynamic Movement Primitives

Dynamic Movement Primitives (DMPs) can be used as a compact representation of high-dimensional planning policies. DMPs have the following favorable features:

- Any movement can be efficiently learned and generated.
- The representation is translation and time invariant.
- Convergence to the goal state g is guaranteed.
- Temporal/spatial coupling terms can be incorporated.

Furthermore, DMPs serve as compact policy representation: attractor landscapes can be adapted by only changing a few parameters, making the DMP representation particularly suitable for RL. DMPs consist of three components: a transformation system, a nonlinear function approximator, and a canonical system. The transformation system is an analytically well understood dynamical system with convenient stability properties that, when integrated, generates the desired behavior. The nonlinear function approximator ensures that arbitrary movements can be encoded, and the canonical system avoids the explicit time dependency of the primitive and provides temporal coupling across multiple degrees of freedom (DOFs).

In this paper, the realization of the transformation system presented in [16], [17] has been chosen – this is a special variant of the original DMP formulation that added robustness towards some numerical characteristics of the original DMPs, but has only reduced (localized) generalization as global translational invariance cannot be guaranteed. A discrete movement is generated by integrating the following transformation system[1]:

$$\tau \dot{v} = K(g - x) - D\,v - K(g - x_0)s + Kf(s) \quad (1)$$
$$\tau \dot{x} = v \ ,$$

where x and v are position and velocity of the system; x_0 and g are the start and goal position; τ is a temporal scaling factor; K is a spring constant, and D a damping term. The non-linear function f is defined as

$$f(s) = \frac{\sum_i \psi_i(s)\,\theta_i\,s}{\sum_i \psi_i(s)} \ , \quad (2)$$

where $\psi_i(s) = \exp(-h_i(s - c_i)^2)$ are Gaussian basis functions, with center c_i and width h_i, and θ_i are adjustable parameters. The function f does not directly depend on time; instead, it depends on a phase variable s, which monotonically changes from 1 towards 0 during a movement and is generated by the canonical system given by:

$$\tau \dot{s} = -\alpha s \ , \quad (3)$$

where α is a pre-defined time constant. Multiple DOFs are realized by maintaining a separate set of transformation systems (Eq 1) as well as separate forcing terms (Eq 2) for each DOF and synchronize them via the canonical system which becomes the central clock.

To encode a recorded trajectory $x(t)$ into a DMP, the parameters θ_i in (Eq 2) are adapted such that the nonlinear function forces the transformation system to follow this recorded trajectory. For this adaptation, the following four steps are carried out: first, using the recorded trajectory $x(t)$, the derivatives $v(t)$ and $\dot{v}(t)$ are computed for each time step $t = 0, ..., T$. Second, the canonical system (Eq 3) is integrated and $s(t)$ evaluated. Third, using these arrays, $f_{\text{target}}(s)$ is computed based on (Eq 1), where x_0 and g are set to $x(0)$ and $x(T)$, respectively. Finally, finding the parameters θ_i that minimize the error criterion $J = \sum_s (f_{\text{target}} - f(s))^2$ is a linear regression problem, which can be solved efficiently.

B. Policy Improvement using Path Integrals: PI^2

We briefly outline the **P**olicy **I**mprovement with **P**ath **I**ntegrals (**PI**^{2}) RL algorithm and show how it can be applied to optimize DMPs. A more detailed derivation of the PI^2 algorithm as well as comparison with related work in the areas of stochastic optimal control and reinforcement learning can be found in [1], [7]. DMPs (Eq 1) are part of the general class of stochastic differential equations of the form

$$\dot{\boldsymbol{x}}_t = \boldsymbol{p}(\boldsymbol{x}_t, t) + \boldsymbol{g}^\mathsf{T}(\boldsymbol{x}_t)(\boldsymbol{\theta}_t + \boldsymbol{\epsilon}_t) \ , \quad (4)$$

---

[1]A different notation is used as in [18], [8] to highlight the spring-like character of these equations.

where $\boldsymbol{x}_t \in \mathbb{R}^n$ denotes the state of the system at time t, $\boldsymbol{\theta}_t \in \mathbb{R}^m$ is the parameter vector of our DMP function approximator in (Eq 2) at time t, $\boldsymbol{p}(\boldsymbol{x}_t, t)$ the passive dynamics, $\boldsymbol{g}(\boldsymbol{x}_t) \in \mathbb{R}^m$ the control transition matrix, and $\boldsymbol{\epsilon}_t \in \mathbb{R}^m$ Gaussian noise with variance $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}$. The immediate cost function is defined as

$$r(t) = \phi_t + \frac{1}{2}\,\boldsymbol{\theta}_t^\mathsf{T}\,\mathbf{R}\,\boldsymbol{\theta}_t \ , \qquad (5)$$

where ϕ_t is an arbitrary state-dependent cost function and $\mathbf{R} \in \mathbb{R}^{m \times m}$ is the positive semi-definite weight matrix of the quadratic control cost. The total cost of a behavior is defined as

$$R(\tau) = \phi_T + \int_0^T r(t)\,dt \ , \qquad (6)$$

where ϕ_T is a special terminal cost. The cost $J = \mathbf{E}_\tau\{R(\tau)\}$ to be optimized is the average over all experienced trajectories τ, such that the optimal cost is the value function $V = \min_{\boldsymbol{\theta}_{0:T}} E\{R(\tau)\}$. The resulting **P**olicy **I**mprovement with **P**ath **I**ntegrals (**PI**^{2}) is summarized in Table I. Essentially, in step 2.2, the term $P(\boldsymbol{\tau}_{k,i})$ is the discrete probability at time t_i of each trajectory roll-out k that is computed with the help of the cost $S(\boldsymbol{\tau}_{k,i})$ step 2.1. For every time step of the trajectory, a parameter update $\delta\boldsymbol{\theta}_{t_i}$ is computed in step 3.1 based on a probability weighted average cost over the sampled trajectories. The parameter updates at every time step are finally averaged in step 4 by giving every $m - th$ coefficient of the parameter vector an update weight according to the time steps left in the trajectory and the activation of the Gaussian kernel ψ_t^m, and the final parameter update $\boldsymbol{\theta}^{(\text{new})} = \boldsymbol{\theta}^{(\text{old})} + \delta\boldsymbol{\theta}$ takes place in step 5. The entire formulation allows an iterative updating of $\boldsymbol{\theta}$. The parameter λ regulates the sensitivity of the exponentiated cost and can automatically be optimized for every time step i to maximally discriminate between the experienced trajectories.

TABLE I: Pseudocode of the PI^2 algorithm for a 1D DMP.

Given:
- immediate cost function $r_t = \phi_t + \boldsymbol{\theta}_t^\mathsf{T}\mathbf{R}\boldsymbol{\theta}_t$
- terminal cost term $\phi_T = \phi_{t_N}$
- DMP policy $\dot{x}_t = p_t + \boldsymbol{g}_t^\mathsf{T}(\boldsymbol{\theta} + \boldsymbol{\epsilon})$ with initial parameter vector $\boldsymbol{\theta}$
- basis function $\boldsymbol{g}_{t_i}$ from the system dynamics
- variance $\Sigma_{\boldsymbol{\epsilon}}$ of the mean-zero noise $\boldsymbol{\epsilon}$

Repeat until convergence of the trajectory cost R:
- **step 1:** Create K rollouts $\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_K$ of the system from the same start state x_0 using stochastic parameters $\boldsymbol{\theta} + \boldsymbol{\epsilon}_k$
- **step 2:** For $k = 1..K$, for $i = 1..N - 1$, do:

    step 2.1: Compute the cost for each sampled trajectory
    $S(\boldsymbol{\tau}_{k,i}) = \phi_{t_N,k} + \sum_{j=i}^{N-1}\phi_{t_j,k}$
    $\qquad + \frac{1}{2}\sum_{j=i}^{N-1}(\boldsymbol{\theta} + \mathbf{M}_{t_j,k}\boldsymbol{\epsilon}_k)^\mathsf{T}\mathbf{R}(\boldsymbol{\theta} + \mathbf{M}_{t_j,k}\boldsymbol{\epsilon}_k)$,
    where $\mathbf{M}_{t_j,k} = \dfrac{\mathbf{R}^{-1}\boldsymbol{g}_{t_j,k}\boldsymbol{g}_{t_j,k}^\mathsf{T}}{\boldsymbol{g}_{t_j,k}^\mathsf{T}\mathbf{R}^{-1}\boldsymbol{g}_{t_j,k}}$

    step 2.2: $P(\boldsymbol{\tau}_{k,i}) = \dfrac{e^{-\frac{1}{\lambda}S(\boldsymbol{\tau}_{k,i})}}{\sum_{l=1}^K e^{-\frac{1}{\lambda}S(\boldsymbol{\tau}_{l,i})}}$
- **step 3:** For $i = 1..N - 1$ compute:
    step 3.1: $\delta\boldsymbol{\theta}_{t_i} = \sum_{k=1}^K\left[P(\boldsymbol{\tau}_{k,i})\,\mathbf{M}_{t_i,k}\boldsymbol{\epsilon}_k\right]$
- **step 4:** Compute $\delta\theta^m = \dfrac{\sum_{i=0}^{N-1}(N-i)\,\psi_{t_i}^m\,\delta\theta_{t_i}^m}{\sum_{i=0}^{N-1}(N-i)\psi_{t_i}^m}$
- **step 5:** Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta\boldsymbol{\theta}$
- **step 6:** Create one noiseless rollout to compute the trajectory cost
    $R = \phi_{t_N} + \sum_{i=0}^{N-1}r_{t_i}$

IV. REINFORCEMENT LEARNING FOR MANIPULATION

A common approach to manipulation involves the design of specialized controllers or behaviors targeted towards specific tasks. These specialized controllers require manual tuning, often a painstaking process performed by a human expert. The expert also has to manually specify the manner in which sensor feedback will be used in determining anomalous events or other failure conditions. Reinforcement learning is a good approach to address these issues. The system we describe here greatly reduces the manual effort required for the robot to learn a specific skill. Human input is still required, though, for two inputs to the system: in defining a task specific representation and in defining a task specific cost function that encapsulates the failure or success of the task. We will now expand upon each of these components in this section before illustrating them further with examples in the next section.

A. Task Space Representation

Manipulation tasks often impose constraints on the degrees of freedom of a robotic system. Most RL approaches to manipulation have used either the joint or end-effector space to represent the task variables. However, task variables can often be chosen appropriately so as to naturally conform to the constraints while *not* eliminating possible solutions. Further, task domain knowledge can also be used to define the most meaningful space for exploration, e.g. the noise level $\Sigma_{\boldsymbol{\epsilon}}$ can be set according to the task.

B. Cost Function

The specification of a task must include a definition of a cost function to be minimized. Unfortunately, the cost function requires careful tuning. It has to be designed to reward task success appropriately while penalizing task failure.

C. Learning from Sensor Data

After the manipulation task has reached satisfying performance, we propose to collect from successful trials statistics of sensor signals to build up a predictive model capable of detecting online failure conditions of future trials. Monitoring manipulation task progress requires considering multiple sensor modalities to both increase robustness against noisy sensor readings and avoid perceptual aliasing. For example, the sensor readings experienced during successful trials of a door opening task will differ from trials in which the door is locked. These differences will be apparent in several modalities, e.g. pressure sensor and joint effort readings as well as door pose estimation. Perceptual aliasing may occur when monitoring the task of picking up objects. Failure to pick up a specific object is not observable by pressure sensors; conversely perceived object pose will not detect grasp failure. While most state-of-the-art robots employ multiple sensors, they are rarely used together. Our proposed approach is a first step towards integrating all available sensor modalities to monitor task performance online and predict task outcome.

D. System Architecture

An overview of the proposed system is sketched in Fig. 2. It consists of four parts: the specification of the initial policy, which in our case happens through demonstration; the task specification, i.e. a cost function and task space representation; the reinforcement learning system that optimizes the initial policy; and a sensor based system that collects statistics of successful trials to predict the outcome of future tasks during execution.

Imitation learning is used to initialize the learning process and RL is applied to optimize the behavior for the desired
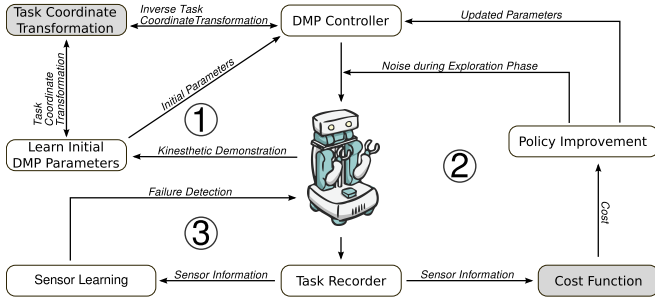
Fig. 2: Proposed system architecture: First, imitation learning is applied to learn an initial DMP policy (1). Then, the policy improvement algorithm PI^2 optimizes this policy using feedback from a task specific cost function (2). Finally, sensor information is collected to make failure predictions to trigger recoveries (3).

outcome. After learning the final policy, statistics over several trials are collected from all available sensors allowing us to make predictions of failure to trigger recovery behaviors. The only modules not shared across applications and that need to be engineered for each particular task are the task specific coordinate transformation (Sec. IV-A) and the cost function that specifies the desired task outcome (Sec. IV-B).

V. MANIPULATION TASKS

The concepts presented in the previous Section are best illustrated with two example tasks. In the first task, a two armed mobile manipulation robot was used to learn a pool stroke from demonstration. In the second task, the robot learned the finer manipulation skill of using a pair of chopsticks to flip a horizontal box upright. Both tasks serve to illustrate the use of DMPs and RL to learn and improve a skill from demonstration. Using the second task, we illustrate how proprioceptive sensor data gathered over multiple trials can be used to learn a predictive model that allows us to predict the outcome of the task online.

A. Learning a Pool Stroke

The goal of the pool task is to learn a pool stroke motion that maximizes the resulting speed of the cue ball while precisely hitting a target location on the pool table.

1) Experimental Setup: The setup consists of the PR2 robot holding the cue stick in its right gripper and the custom-made bridge (see Fig. 3) in its left gripper.
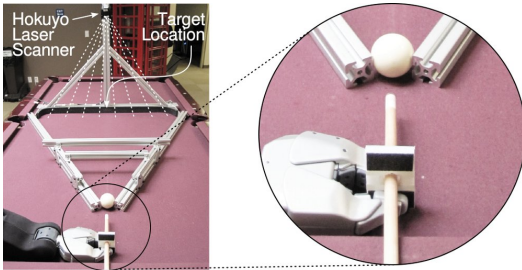

Fig. 3: The Hokuyo laser scanner used to measure the offset of the cue ball from the target location (left), and a closeup of the left gripper holding the bridge (right). The apparatus on the tilted pool table is used to guide the backwards rolling ball after each pool shot to its original position.

The demonstration for initialization with imitation learning was obtained through kinesthetic demonstration. During the teaching phase, the controllers of the robot's right arm were switched off, allowing a human demonstrator to easily guide the cue stick to hit the cue ball. The duration of the demonstration was set to 1 second. The cue stick, which is held
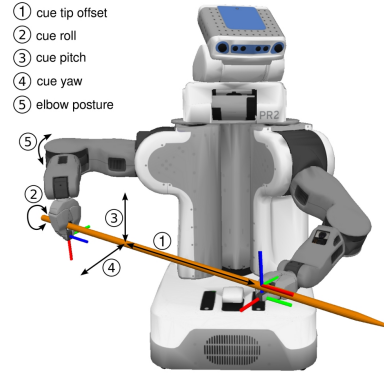

Fig. 4: Illustration of the pool task frame. The stroke motion was transformed in order to allow constraint-satisfying exploration.

in the right gripper and constraint by the bridge held in the left gripper, closes a kinematic chain and therefore imposes a constraint on the right gripper orientation during movement execution. Encoding the demonstrated movement in joint or end-effector space would violate these task constraints during exploration and create unnecessary internal torques. Instead, by choosing an appropriate task frame, exploration can be restricted to the constraint-satisfying manifold of the particular task. In case of the pool stroke, the appropriate task variables are the translational offset from the right gripper to the bridge, the roll, pitch, and yaw angles of the cue around the bridge, and the redundant degree-of-freedom in the arm (Fig. 4).

The recorded joint trajectories were first transformed into the described task space and then encoded in a single DMP with 5 transformation systems as described in Sec. III-A. To enable autonomous learning, the pool table was tilted such that the cue ball rolls back after each pool shot and a special apparatus (see Fig. 3) was installed which guides the downwards rolling ball to its original position. Furthermore, a Hokuyo laser scanner was mounted on the opposing side of the pool table to measure both where and when the cue ball crossed the scan line. A high-bandwidth Bosch DMA150 digital accelerometer embedded in the palm of robot's gripper and sampled at rate of 3 kHz was utilized to detect the point in time at which the cue stick impacts with the ball. Therefore, the accelerometer signals were band-pass filtered to contain data between 40-1000 Hz. The variance of the mean-zero noise Σ_ϵ for the 5 task space dimensions (see Fig. 4) was set to $\epsilon_{\text{cue tip offset}} = 0.14$, $\epsilon_{\text{cue roll}} = \epsilon_{\text{cue pitch}} = \epsilon_{\text{cue yaw}} = 0.01$, and $\epsilon_{\text{elbow posture}} = 0.1$.

2) Cost Function: The task objective of the pool stroke is to maximize the cue ball speed, while attempting to have the ball cross the center of the scan line (see Fig. 3). Equivalent to maximizing the speed of the cue ball is minimizing the ball travel duration t_b which is computed from the difference of the point in time at which the cue stick impacts the ball to the point in time at which the ball crosses the scan line. The target offset error x_e is defined to be the offset from the target location. In order to avoid the cue stick possibly slipping out of the bridge during exploration, a high cost was assigned whenever the backwards displacement of the cue tip $x_{\text{cue tip offset}}$ exceeds a certain threshold. The terminal cost ϕ_T is computed as weighted according to

$$\phi_T = \omega_1 \, t_b^2 + \omega_2 \, x_e^2 + \omega_3 \, f(x_{\text{cue tip offset}}) \ , \qquad (7)$$

where $f(x_{\text{cue tip offset}})$ is equal to 1 if the backwards displacement boundary was violated during the rollout, or 0
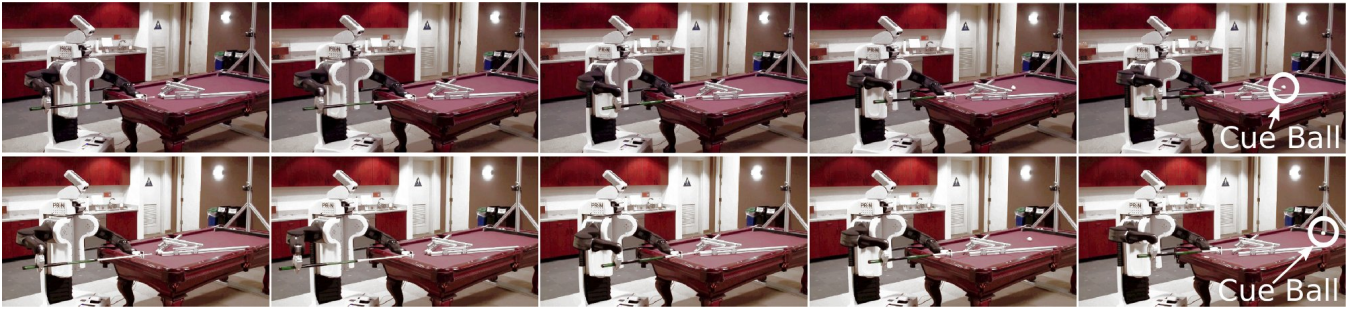
Fig. 5: The PR2 robot learning a pool stroke: The apparatus on the tilted pool table automatically positions the cue ball after each trial allowing for autonomous learning. The initial policy fails to hit the target on the tilted table (top row). After learning (bottom row) the robot learned to retract the pool cue first allowing for a more powerful shot such that the cue ball hits the target location, shown in Fig 3.

otherwise. For the learning experiment, the following weights were used: $\omega_1 = 100$, $\omega_2 = 10000$, and $\omega_3 = 10000$.

3) Results: The robot was able to improve the initial policy completely autonomously within 20 minutes, as shown in Fig. 5 and the video supplement. Fig. 6 confirms this observation since the cost of the noiseless rollouts decrease over time.
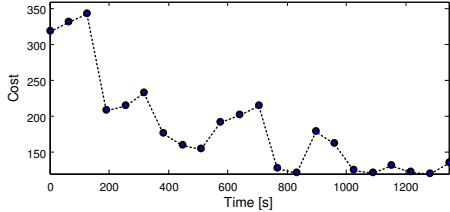

Fig. 6: The learning curve showing the cost of the 22 noiseless rollouts over the time span of roughly 20 minutes. Note: The robot improved its task execution completely autonomously.

B. Box Flipping using Chopsticks

The goal of the second task is to learn the finer manipulation skill of gently flipping a horizontal box upright using a pair of chopsticks (see Fig 1).

1) Experimental Setup: The experimental setup consists of the PR2 robot positioned in front of a table on which the box is placed, as shown in Fig 9. The two fingertips on each of the parallel jaw grippers are equipped with a pressure sensor array consisting of 22 individual cells covered by a protective layer of silicone rubber. The arrangements of the individual cells is shown in Fig. 7 (left). These capacitive sensors measure the perpendicular compressive force applied in each sensed region and are sampled simultaneously at a rate of 24.4 Hz.

Fig. 7: Arrangement of the pressure sensors on the finger pads (left) and closeup of the placement of the chopsticks in the gripper (right).

In order to overcome the sensor's limitation on detecting shear forces, a washer was attached to the chopstick as shown in Fig. 7 (right). The chopsticks are placed between the two finger pads and the sensor readings are used to servo

a force of approximately 10 N. The reasons for not holding the chopsticks with maximum force are two fold: to enable the pressures sensors to sense contact of the chopsticks with the box or table and to impose an interesting aspect on the manipulation task and making it more difficult. In order to measure the box orientation and its linear accelerations during each trial, a MicroStrain Inertia-Link was attached inside the box. The measurements are obtained at 100 Hz.

2) Cost Function: The primary task objective is to flip the box upright. Therefore, the terminal cost ϕ_T provides feedback of the maximum roll of the box $\gamma_{\max}$ over all time steps $t = 1, \ldots, T$ as well as its final roll γ_T and is computed as a weighted sum according to

$$\phi_T = \omega_1\, \gamma_{\max}^{10} + \omega_2\, f(\gamma_T) \ , \tag{8}$$

where $f(\gamma_T)$ is equal to 1 if $|\gamma_T - \frac{\pi}{2}| < 0.1$, or 0 otherwise. The secondary task objective is to flip the box upright as gently as possible. Thus, the state cost $\phi(t)$ includes the square of the box linear accelerations $a_b(t) = a_{b_x}(t)^2 + a_{b_y}(t)^2 + a_{b_z}(t)^2$, the amount of force sensed on each of the 4 pressure sensors $p(t) = \sum_{i=0}^{88} p_i(t)$, and the sum of the sensed square accelerations in both grippers $a_g(t) = a_{g_x}(t)^2 + a_{g_y}(t)^2 + a_{g_z}(t)^2$. The gripper acceleration signals are band-pass filtered to contain data between 5-1000 Hz so that low frequency signals due to gravity and slow motions are filtered out while impacts of the chopstick with the table and/or box are still observable. The state cost of a rollout for each time step $\phi(t)$ is also calculated as a weighted sum:

$$\phi(t) = \omega_3\, a_b + \omega_4\, p(t) + \omega_5\, a_g(t) \ . \tag{9}$$

The total cost is computed by adding the terminal cost ϕ_T to the last time step of the state cost $\phi(t = T)$. For the learning experiment, the following weights were used: $\omega_1 = 50.0$, $\omega_2 = 200.0$, $\omega_3 = 0.1$, $\omega_4 = 1.0$, and $\omega_5 = 0.1$. Note: these parameters also account for normalization across different sensor modalities.

To illustrate the difficulty of the task, we conducted an informal user study: 10 subjects were asked to each perform 20 attempts to flip the box by guiding the robot's gripper

Fig. 8: An informal user study was conducted: 10 subjects were asked to flip the box by guiding the robot's gripper and only an average of 3 out of 20 attempts were successful.
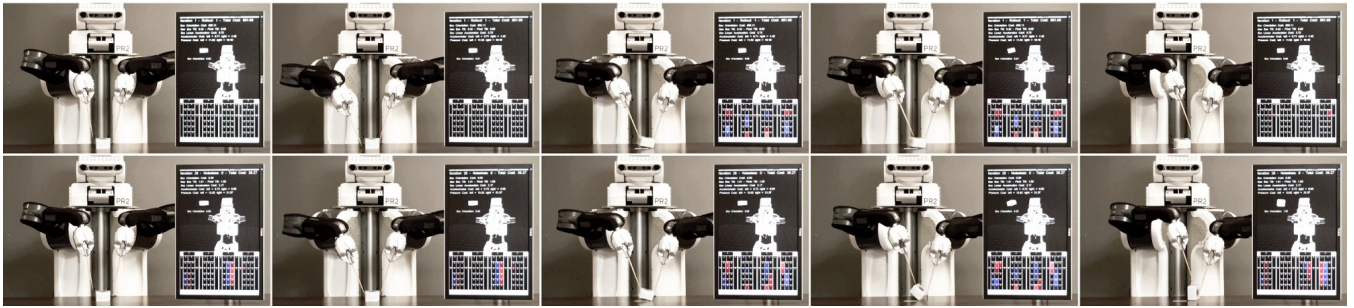
Fig. 9: The PR2 robot learning a finer manipulation skill of gently flipping a horizontal box upright: The initial policy learned from demonstration fails (top row), whereas the learned policy succeeds (bottom row). The color display on the monitor visualize the amount of sensed pressure of the tactile sensors.

as pictured in Fig. 8. Although the subjects were using visual and tactile feedback to correct for errors during the execution, the success rate was rather low. Each subject achieved an average of 3 successful trials out of 20 attempts. As in the previous task, the initial policy is learned from a human demonstrator by directly guiding the robot's gripper. The task variables for this particular task were naturally chosen to be the two tips of the chopsticks. Thus, the joint trajectory, recorded during teaching, is first transformed into the task space and then encoded in a single DMP with 12 transformation systems (position $\mathrm{tcp}_x, \mathrm{tcp}_y, \mathrm{tcp}_z$ as well as orientation $\mathrm{tcp}_{\mathrm{roll}}, \mathrm{tcp}_{\mathrm{pitch}}, \mathrm{tcp}_{\mathrm{yaw}}$ of the tips of each chopstick). The duration of the demonstration was 4 seconds. In the learning phase, the exploration noise ϵ was set to $\epsilon_{\mathrm{x}} = 0.01$, $\epsilon_{\mathrm{y}} = 0.04$, $\epsilon_{\mathrm{z}} = 0.06$, $\epsilon_{\mathrm{roll}} = 0.2$, $\epsilon_{\mathrm{pitch}} = 0.5$, and $\epsilon_{\mathrm{yaw}} = 0.2$ for both of the end-effectors task spaces.
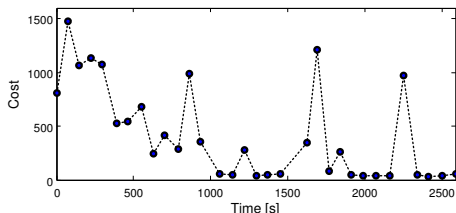

Fig. 10: Learning curve of the box flipping task showing the cost of 32 noiseless rollouts over the time span of roughly 41 minutes. The cost is minimized over time, however, occasional failures remain due to the stochasticity of the task.

3) Results: Although a successful demonstration was used to initialize the learning experiment, simply replaying the initial policy only succeeded 3 times out of 100 runs. This observation suggests that not all task relevant aspects had been captured from imitation. However, after 26 iterations (roughly 35 minutes) the robot learned to achieve the task (see Fig 10). For performance evaluation the learned policy was executed (without the addition of noise) a total of 200 trials out of which 172 succeeded.

C. Predicting Task Outcome from Sensor Information

The idea of the proposed approach is to use sensor information collected over a series of trials in the training phase to compute the average and variance of the expected sensor feedback to predict the outcome of future trials. To demonstrate the feasibility of our approach and to evaluate the task performance (see Section V-B.3), we recorded a total of 132 task relevant sensor signals along with each of the 200 noiseless rollouts. These signals are: the joint positions, joint velocities, and joint efforts of both 7 DOFs arms, the linear accelerations of both grippers, and each individual pressure sensor cell of all 4 finger pads. The

recorded signals were truncated using the start and end time of the DMP and are re-sampled such that each signal only contains 100 samples. To demonstrate the ability of our approach to online detect failure conditions, we divided the data recorded from 200 trials into a training and test set of equal size. A subset of these 132 signals that are extracted from the training set are shown in Fig 11. Our aim is to detect failure conditions during future task executions the moment unexpected sensor information is perceived. As a first step, we propose to use this knowledge about task performance to detect failure conditions during task execution the moment several perceived sensor signals deviate from the predicted ones. Table II summarizes our approach.

TABLE II: Online detection of failure conditions from sensor information.

Given:
- The critical value for the cost ϕ_c that determines task success.

Iterative Training:
- **step 1:** Execute rollout and record sensor information along with the achieved cost ϕ_t.
- **step 2:** Use ϕ_c to compute whether task was successful, i.e. $\phi_t < \phi_c$.
- **step 3:** For successful rollouts: compute a weight according to $w = (\phi_c - \phi_t)/(\phi_c)$.
- **step 4:** Update the weighted mean and weighted standard deviation for each time step over all rollouts for all signals.

Online Testing:
- **step 1:** During task execution, monitor all sensor signals and compute at each time step for each signal a z-test using the weighted mean and weighted standard deviation from the training phase under the null-hypotheses with confidence c.
- **step 2:** If the z-test fails, then the perceived sensor value is with confidence c not correlated with the statistics computed from successful trials. If this occurs for n consecutive time steps in more than m signals, a failure condition is predicted.

1) Experimental Results: In the training phase, the 73 successful trials contained in the training data set were used to compute the weighted mean and weighted $\pm$ one standard deviation for each time step, see Fig 11. In the testing phase, 100 test trials (containing 89 successful and 11 unsuccessful trials) were presented to the proposed method in a sequential manner. Even though the testing phase was carried out offline, the data was input into the system in the same temporal order as perceived by the robot during execution. The method sketched in Table II, setup with the parameters $c = 10e - 9$, $n = 3$, and $m = 5$, managed to detect all 11 failures in the test set without triggering any false positives. It is important to note the system was able to detect these 11 failure conditions the moment more than 5 signals were outside the confidence interval for more than 3 consecutive time steps. These 11 failure conditions were detected on average after 2.4 seconds, which roughly corresponds to the instant the chopsticks lost contact with
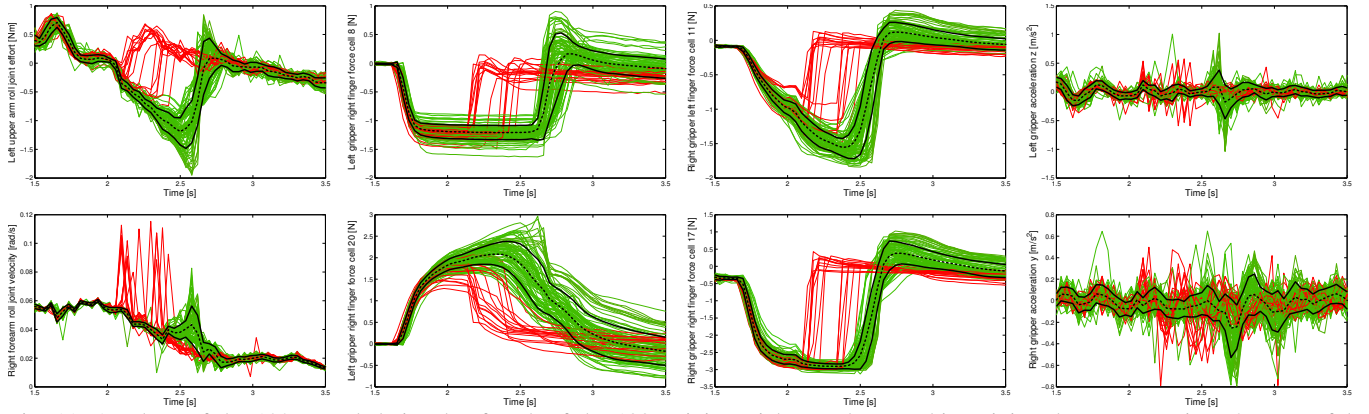
Fig. 11: A subset of the 132 recorded signals of each of the 100 training trials are shown. This training data set contains 73 successful (green) and 17 unsuccessful (red) trials. The weighted mean (dashed line) and the weighted $\pm$ one standard deviation (solid line) are computed from the successful trials. The subset shows examples of signals that can be used to detect failure conditions, however, it also shows two examples (rightmost) of signals that may not be useful.

the box causing it to fall back, see Fig. 12. The parameters for n and m were chosen specifically for the task at hand and will need to be adjusted in future tasks. One possible solution might involve inferring the optimal set of parameters from the training data, but this remains future work.
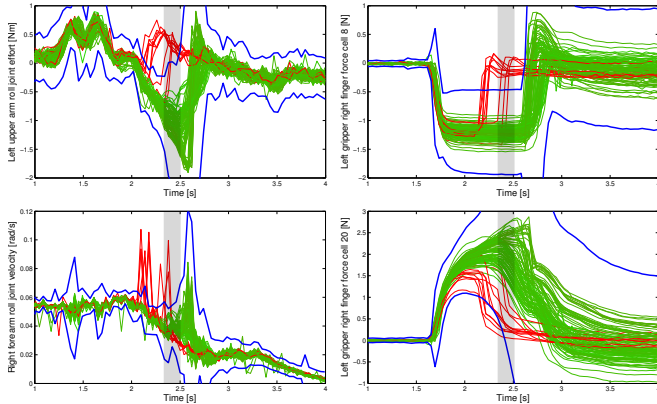

Fig. 12: The plots show the test data set. The $6 \pm$ weighted standard deviation (blue line) is computed from the training set (shown in Fig11).The failure condition for all 11 unsuccessful trials were detected correctly after an average of 2.4 seconds (shaded area).

VI. CONCLUSIONS AND FUTURE WORK

This paper demonstrated an approach to learning motor skills including imitation learning, reinforcement learning, and finally learning to predict performance of the skill. We used the established framework of Dynamic Movement Primitives for imitation, a novel "black-box" probabilistic reinforcement learning method called PI^2, and performance prediction based on statistical hypothesis testing in sensory data of a robot. We would like to highlight the ease of reinforcement learning in the given setup with the PI^2 algorithm, and that it was straightforward to build a successful and rather simple performance predictor from simple sensory statistics. We believe that our work contributes to an increasingly complete approach to learning and performing motor skills in dynamic and stochastic environments. In particular the prediction of sensory events is ubiquitous in humans, but hardly realized in robotic systems. Obviously, predicting failure before it happens allows much safer interaction with the environment, which is among the most critical properties that robots will need if they are going to share the same workspace as humans.

Future work will expand the ideas of this paper towards methods of automatically determining task space variables, and towards more sophisticated machine learning methods to predict task performance without manual tuning parameters.

REFERENCES

[1] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: a path integral approach," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2010.
[2] J. Faust, J. Hsu, P. Mihelich, A. Oyama, S. Glaser, T. Field, T. Foote, and B. Gerkey, "PR2 Robot Plays Pool," in *Willow Garage*, 2010. http://www.youtube.com/watch?v=HMx1xW2E4Gg.
[3] S. Schaal, "Is imitation learning the route to humanoid robots?" in *Trends in Cognitive Sciences*, no. 6, 1999, pp. 233–242.
[4] S. Schaal and C. G. Atkeson, "Learning control in robotics – trajectory-based opitimal control techniques," *Robotics and Automation Magazine*, vol. 17, no. 2, pp. 20–29, 2010.
[5] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Network*, vol. 21, no. 4, pp. 682–97, 2008.
[6] J. Kober and J. Peters, "Learning Motor Primitives for Robotics," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 12-19, 2009.
[7] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral approach to reinforcement learning," *Journal of Machine Learning Research*, 2010, accepted for publication.
[8] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning Attractor Landscapes for Learning Motor Primitives," in *Advances in Neural Information Processing Systems (NIPS)*, 2003, pp. 1547–1554.
[9] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato, "Learning from demonstration and adaptation of biped locomotion," *Robotics and Autonomous Systems*, no. 2-3, 2004.
[10] J. Kober and J. Peters, "Policy search for motor primitives in robotics," in *Advances in Neural Information Processing Systems (NIPS)*, 2008.
[11] J. Kober, E. Oztop, and J. Peters, "Reinforcement Learning to adjust Robot Movements to New Situations," in *Proceedings of Robotics: Science and Systems (R:SS)*, 2010.
[12] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill co-ordination with em-based reinforcement learning," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2010.
[13] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous Helicopter Aerobatics through Apprenticeship Learning," in *International Journal of Robotics Research (IJRR)*, June 2010.
[14] J. van den Berg, S. Miller, D. Duckworth, H. Hu, X. Fu, K. Goldberg, and P. Abbeel, "Superhuman Performance of Surgical Tasks by Robots using Iterative Learning from Human-Guided Demonstrations," in *International Conference on Robotics and Automation*, 2010.
[15] S. Calinon and A. Billard, "A probabilistic programming by demonstration framework handling skill constraints in joint space and task space," in *Intelligent Robots and Systems (IROS)*, 2008, pp. 367–372.
[16] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and Generalization of Motor Skills by Learning from Demonstration," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2009.
[17] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, Kobe, Japan, May 12-19, 2009.
[18] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2002.