2010 IEEE-RAS International Conference on Humanoid Robots
Nashville, TN, USA, December 6-8, 2010

Reinforcement Learning of Full-body Humanoid Motor Skills

Freek Stulp, Jonas Buchli, Evangelos Theodorou, Stefan Schaal

Abstract— **Applying reinforcement learning to humanoid robots is challenging because humanoids have a large number of degrees of freedom and state and action spaces are continuous. Thus, most reinforcement learning algorithms would become computationally infeasible and require a prohibitive amount of trials to explore such high-dimensional spaces. In this paper, we present a probabilistic reinforcement learning approach, which is derived from the framework of stochastic optimal control and path integrals. The algorithm, called Policy Improvement with Path Integrals (PI^2), has a surprisingly simple form, has no open tuning parameters besides the exploration noise, is model-free, and performs numerically robustly in high dimensional learning problems. We demonstrate how PI^2 is able to learn full-body motor skills on a 34-DOF humanoid robot. To demonstrate the generality of our approach, we also apply PI^2 in the context of variable impedance control, where both planned trajectories and gain schedules for each joint are optimized simultaneously.**

This paper is accompanied by a movie:
`http://www-clmc.usc.edu/movies/humanoids2010/`

I. Introduction

Reinforcement learning is a general approach that, in principle, enables humanoid robots to autonomously learn motor skills from interaction with the environment and given only a relatively unspecific feedback on the quality of completing the task. As such, it is an attractive alternative to hand-coding behaviors, which is tedious and error-prone. In practice however, applying reinforcement learning to humanoid robots poses several challenges: 1) state and action spaces are continuous; 2) humanoid robots have a large number of degrees of freedom, and thus lead to high-dimensional learning problems; 3) exploration in high-dimensional spaces is costly and time consuming, so learning should be efficient and converge quickly; 4) it is difficult to acquire an accurate model of the robot and its interaction with the environment to enable more efficient model-based reinforcement learning.

In this paper, we present a probabilistic reinforcement learning approach, which is derived from the framework of stochastic optimal control and path integrals, based on the original work of [14]. As will be detailed in the sections below, this approach makes an appealing theoretical connection between value function approximation using the stochastic HJB equations and direct policy learning by approximating a path integral, i.e., by solving a statistical inference problem from sample roll-outs. The resulting algorithm, called **P**olicy **I**mprovement with **P**ath **I**ntegrals (**PI^2**), takes on a surprisingly simple form, has no open tuning parameters besides the exploration noise, and performs numerically robustly in high dimensional learning problems.

Computational Learning and Motor Control Lab, University of Southern California, Los Angeles, CA 90089
`stulp@clmc.usc.edu, jonas@buchli.org, etheodor@usc.edu, sschaal@usc.edu`

This research was supported in part by National Science Foundation grants ECS-0326095, IIS-0535282,CNS-0619937, IIS-0917318, CBET-0922784, EECS-0926052, the DARPA program on Learning Locomotion, the Army Research Office, the Okawa Foundation, and the ATR Computational Neuroscience Laboratories. F.S. was supported by a Research Fellowship from the German Research Foundation (DFG). J.B. was supported by a prospective researcher fellowship from the Swiss National Science Foundation. E.T. was supported by a Myronis Fellowship.

The main contribution of this paper is to show how PI^2 is able to learn full-body motor skills on a 34-DOF humanoid robot. To demonstrate the generality of our approach, we also apply PI^2 in the context of variable impedance control, where both planned trajectories and gain schedules for each joint are optimized simultaneously.

The rest of this paper is structured as follows. In the next section, we discuss related work. The PI^2 algorithm is presented in Section III, along with an evaluation of applying it to the acquisition of a full-body motor skill. In Section IV, we describe how PI^2 is applied to learning variable impedance control, and demonstrate its use in the context of a humanoid manipulation task. We conclude with Section V.

II. Related Work

Classical value-function based methods with function approximation offer one possible approach to reinforcement learning [11], but function approximation under the non-stationary iterative learning process of the value-function remains difficult when the learning problem exceeds about 5-10 dimensions.

Similar humanoid tasks and and cost functions are considered in [13], [6], [9]. However, [9] is a pure control approach, and does not perform planning as PI^2 does. As we shall see, the tasks considered in this paper require a complex plan, which cannot be achieved through control alone. In contrast, [13], [6] are planning approaches, which take certain dynamical constraints into account. However, the planning is performed with simplified models (e.g. an inverted pendulum for the ZMP), and require post-processing steps (trajectory smoothing and interpolation with minimum-jerk trajectories) to be executed on a robot. There are no guarantees that the post-processed plan will still achieve the task. Also, dynamic aspects of the task are also mostly eliminated by slowing down the movement, which leads to quasi-static motion. But the main difference of all these methods, to PI^2 is that they all require accurate models, which are hard to acquire in high-dimensional systems which interact with the environment.

As we show in this paper, our approach covers and therefore unifies both the control (as in [9]) and planning (as in [6], [13]) aspects of full-body humanoid motor skills. This unification becomes especially clear in Section IV, where joint trajectories (planning) and gain schedules (control) are learned simultaneously. Since our approach is model-free, it also enables us to learn skills that involve complex interactions between the robot and the environment which are difficult to model, e.g. physical contact during manipulation.

Direct model-free policy learning from trajectory roll-outs has recently made significant progress [5], but can still become numerically brittle and full of open tuning parameters in complex learning problems. In new developments, RL researchers have started to combine the well-developed methods from statistical learning and empirical inference with classical RL approaches in order to minimize tuning parameters and numerical problems, such that ultimately more efficient algorithms can be developed that scale to significantly more complex learning systems [5]. An advantage of PI^2 is that it has only one open parameter, the exploration noise, which has a clear physical interpretation. The need for further open parameters is eliminated from the resulting algorithm, as PI^2 is derived from first principles of stochastic optimal control and reinforcement learning, with minimal auxiliary assumptions. Also, PI^2 does not compute a gradient, which is usually sensitive to noise and large derivatives in the value function. To the best of our knowledge, these model-free methods have not yet been applied to full-body 34-DOF humanoid motor skills, or simultaneous learning of planned trajectories and control gain schedules.

III. REINFORCEMENT LEARNING IN HIGH DIMENSIONS – THE PI^2 ALGORITHM

Reinforcement learning algorithms can be derived from different frameworks, e.g., dynamic programming, optimal control, policy gradients, or probabilistic approaches. Recently, an interesting connection between stochastic optimal control and Monte Carlo evaluations of path integrals was made [14]. In [12] this approach is generalized, and used in the context of model-free reinforcement learning with parameterized policies, which resulted in the PI^2 algorithm. In the following, we provide a short outline of the prerequisites and the development of the PI^2 algorithm as needed in this paper. For more details refer to [12].

The foundation of PI^2 comes from (model-based) stochastic optimal control for continuous time and continuous state-action systems. We assume that the dynamics of the control system is of the form

$$\dot{\mathbf{x}}_t = \mathbf{f}(\mathbf{x}_t) + \mathbf{G}(\mathbf{x}_t)\left(\mathbf{u}_t + \epsilon_t\right) = \mathbf{f}_t + \mathbf{G}_t\left(\mathbf{u}_t + \epsilon_t\right) \tag{1}$$

with $\mathbf{x}_t \in \Re^{n\times 1}$ denoting the state of the system, $\mathbf{G}_t = \mathbf{G}(\mathbf{x}_t) \in \Re^{n\times p}$ the control matrix, $\mathbf{f}_t = \mathbf{f}(\mathbf{x}_t) \in \Re^{n\times 1}$ the passive dynamics, $\mathbf{u}_t \in \Re^{p\times 1}$ the control vector and $\epsilon_t \in \Re^{p\times 1}$ Gaussian noise with covariance $\mathbf{\Sigma}_\epsilon$. Many robotic systems fall into this class of control systems. For the finite horizon problem $t_i : t_N$, we want to find control inputs $\mathbf{u}_{t_i:t_N}$ which minimize the value function

$$V(\mathbf{x}_{t_i}) = V_{t_i} = \min_{\mathbf{u}_{t_i:t_N}} E_{\boldsymbol{\tau}_i}\left[R(\boldsymbol{\tau}_i)\right] \tag{2}$$

where R is the finite horizon cost over a trajectory starting at time t_i in state $\mathbf{x}_{t_i}$ and ending at time t_N

$$R(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} r_t \; dt \tag{3}$$

and where $\phi_{t_N} = \phi(x_{t_N})$ is a terminal reward at time t_N. r_t denotes the immediate reward at time t. $\boldsymbol{\tau}_i$ are trajectory pieces starting at $\mathbf{x}_{t_i}$ and ending at time t_N.

As immediate reward we consider

$$r_t = r(\mathbf{x}_t, \mathbf{u}_t, t) = q_t + \frac{1}{2}\mathbf{u}_t^T \mathbf{R} \mathbf{u}_t \tag{4}$$

where $q_t = q(\mathbf{x}_t, t)$ is an arbitrary state-dependent reward function, and $\mathbf{R}$ is the positive semi-definite weight matrix of the quadratic control cost. From stochastic optimal control [10], it is known that the associated Hamilton Jacobi Bellman (HJB) equation is

$$\begin{aligned}\partial_t V_t = q_t + (\nabla_{\mathbf{x}} V_t)^T \mathbf{f}_t - \frac{1}{2}(\nabla_{\mathbf{x}} V_t)^T \mathbf{G}_t \mathbf{R}^{-1} \mathbf{G}_t^T (\nabla_{\mathbf{x}} V_t) \\ + \frac{1}{2} trace\left((\nabla_{\mathbf{xx}} V_t)\mathbf{G}_t \mathbf{\Sigma}_\epsilon \mathbf{G}_t^T\right)\end{aligned} \tag{5}$$

The corresponding optimal control is a function of the state and it is given by the equation:

$$\mathbf{u}(\mathbf{x}_{t_i}) = \mathbf{u}_{t_i} = -\mathbf{R}^{-1}\mathbf{G}_{t_i}^T (\nabla_{x_{t_i}} V_{t_i}) \tag{6}$$

We are leaving the standard development of this optimal control problem by transforming the HJB equations with the substitution $V_t = -\lambda \log \Psi_t$ and by introducing a simplification $\lambda \mathbf{R}^{-1} = \mathbf{\Sigma}_\epsilon$. In this way, the transformed HJB equation becomes a linear 2^{nd} order partial differential equation. Due to the Feynman-Kac theorem, the solution for the exponentially transformed value function becomes

$$\Psi_{t_i} = \lim_{dt \to 0} \int p\left(\boldsymbol{\tau}_i | \mathbf{x}_i\right) \exp\left[-\frac{1}{\lambda}\left(\phi_{t_N} + \sum_{j=0}^{N-1} q_{t_j} dt\right)\right] d\boldsymbol{\tau}_i \tag{7}$$

Thus, we have transformed our stochastic optimal control problem into an approximation problem of a path integral. As detailed in [12], it is not necessary to compute the value function explicitly, but rather it is possible to derive the optimal controls directly:

$$\begin{aligned}\mathbf{u}_{t_i} &= \int P\left(\boldsymbol{\tau}_i\right) \mathbf{u}\left(\boldsymbol{\tau}_i\right) d\boldsymbol{\tau}_i \\ \mathbf{u}(\boldsymbol{\tau}_i) &= \mathbf{R}^{-1}\mathbf{G}_{t_i}{}^T \left(\mathbf{G}_{t_i}\mathbf{R}^{-1}\mathbf{G}_{t_i}{}^T\right)^{-1} \left(\mathbf{G}_{t_i}\boldsymbol{\epsilon}_{t_i} - \mathbf{b}_{t_i}\right)\end{aligned} \tag{8}$$

where $P\left(\boldsymbol{\tau}_i\right)$ is the probability of a trajectory $\boldsymbol{\tau}_i$, and $\mathbf{b}_{t_i}$ is a more complex expression, beyond the space constraints of this paper. The important conclusion is that it is possible to evaluate Eq. (8) from Monte Carlo roll-outs of the control system, i.e., our optimal control problem can be solved as an estimation problem.

A. The PI^2 Algorithm

The PI^2 algorithm is just a special case of the optimal control solution in Eq. (8), applied to control systems with parameterized control policy:

$$\mathbf{a}_t = \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_t) \tag{9}$$

i.e., the control command is generated from the inner product of a parameter vector $\boldsymbol{\theta}$ with a vector of basis function $\mathbf{g}_t$ – the noise $\boldsymbol{\epsilon}_t$ is interpreted as user controlled exploration noise.

A particular case of a system with a parameterized control policy is the Dynamic Movement Primitives (DMP) approach introduced by [4]:

$$\frac{1}{\tau}\dot{v}_t = f_t + \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_t) \tag{10}$$

$$\frac{1}{\tau}\dot{q}_{d,t} = v_t$$

$$f_t = \alpha(\beta(g - q_{d,t}) - v_t)$$

$$\frac{1}{\tau}\dot{s}_t = -\alpha s_t \tag{11}$$

$$[\mathbf{g}_t]_j = \frac{w_j s_t}{\sum_{k=1}^{p} w_k}(g - q_0) \tag{12}$$

$$w_j = \exp\left(-0.5 h_j (s_t - c_j)^2\right) \tag{13}$$

The intuition of this approach is to create desired trajectories $q_{d,t}, \dot{q}_{d,t}, \ddot{q}_{d,t}{=}\tau\dot{v}_t$ for a motor task out of the time evolution of a nonlinear attractor system, where the goal g is a point attractor and q_0 the start state. The parameters $\boldsymbol{\theta}$ determine the shape of the attractor landscape within a nonlinear function approximator, which allows to represent almost arbitrary smooth trajectories, e.g., a tennis swing, a reaching movement, or a complex dance movement. While leaving the details of the DMP approach to [4], for this paper the important ingredients of DMPs are that i) the attractor system Eq. (10) has the same form as Eq. (1), and that ii) the p-dimensional parameter vector can be interpreted as motor commands as used in the path integral approach to optimal control. Learning the optimal values for $\boldsymbol{\theta}$ will thus create an optimal reference trajectory for a given motor task. The PI^2 learning algorithm applied to this scenario is summarized in Table I. As illustrated in [12], PI^2 outperforms previous RL algorithms for parameterized policy learning by at least one order of magnitude in learning speed and also lower final cost performance. As an additional benefit, PI^2 has no open algorithmic parameters, except for the magnitude of the exploration noise $\boldsymbol{\epsilon}_t$ (the parameter λ is set automatically, cf. [12]). We would like to emphasize one more time that PI^2 *does not* require knowledge of the model of the control system or the environment.

Key Innovations in PI^2: In summary we list the key innovations in PI^2 that we believe lead to its superior performance. These innovations make applications like the the learning of gain schedules for high dimensional tasks possible.

- The basis of the derivation of the PI^2 algorithm is the transformation of the optimal control problem to a path integral. This transformation is very critical since there is no need to calculate a gradient that is usually sensitive to noise and large derivatives in the value function.
- With PI^2 the optimal control problem is solved with the forward propagation of dynamics. Thus no backward propagation of approximations of the value function is required. This is a very important characteristic of PI^2 that allows for sampling (i.e. roll-out) based estimation of the path-integral.
- For high dimensional problems, it is not possible to sample the whole state space and that is the reason for applying path integral control in an iterative fashion in PI^2 to update the parameters of the DMPs.
- The derivation of an RL algorithm from first principles largely eliminates the need for open parameters in the final algorithm.

TABLE I
PI^2 ALGORITHM PSEUDO-CODE FOR A 1D PARAMETERIZED POLICY.

```
• Given:
    – An immediate cost function $r_t = q_t + \boldsymbol{\theta}_t^T \mathbf{R} \boldsymbol{\theta}_t$ (cf. Eq. (3))
    – A terminal cost term $\phi_{t_N}$ (cf. 3)
    – A stochastic parameterized policy $\mathbf{a}_t = \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_t)$ (cf. Eqs. (9) and (10))
    – The basis function $\mathbf{g}_{t_i}$ from the system dynamics (cf. 12)
    – The variance $\Sigma_\epsilon$ of the mean-zero noise $\boldsymbol{\epsilon}_t$
    – The initial parameter vector $\boldsymbol{\theta}$
• Repeat until convergence of the trajectory cost $R$:
    – Create $K$ roll-outs of the system from the same start state $\mathbf{x}_0$ using stochastic parameters $\boldsymbol{\theta} + \boldsymbol{\epsilon}_t$ at every time step
    – For all $K$ roll-outs, compute:
        * $P\left(\boldsymbol{\tau}_{i,k}\right) = \frac{e^{-\frac{1}{\lambda}S(\boldsymbol{\tau}_{i,k})}}{\sum_{k=1}^{K}[e^{-\frac{1}{\lambda}S(\boldsymbol{\tau}_{i,k})}]}$
        * $S(\boldsymbol{\tau}_{i,k}) = \phi_{t_N,k} + \sum_{j=i}^{N-1} q_{t_j,k} + \frac{1}{2}\sum_{j=i+1}^{N-1}(\boldsymbol{\theta} + \mathbf{M}_{t_j,k}\boldsymbol{\epsilon}_{t_j,k})^T \mathbf{R}(\boldsymbol{\theta} + \mathbf{M}_{t_j,k}\boldsymbol{\epsilon}_{t_j,k})$
        * $\mathbf{M}_{t_j,k} = \frac{\mathbf{R}^{-1}\mathbf{g}_{t_j,k}\ \mathbf{g}_{t_j,k}^T}{\mathbf{g}_{t_j,k}^T \mathbf{R}^{-1}\mathbf{g}_{t_j,k}}$
    – For all $i$ time steps, compute:
        * $\delta\boldsymbol{\theta}_{t_i} = \sum_{k=1}^{K}\left[P\left(\boldsymbol{\tau}_{i,k}\right)\mathbf{M}_{t_i,k}\ \boldsymbol{\epsilon}_{t_i,k}\right]$
    – Compute $[\delta\boldsymbol{\theta}]_j = \frac{\sum_{i=0}^{N-1}(N-i)\ w_{j,t_i}\ [\delta\boldsymbol{\theta}_{t_i}]_j}{\sum_{i=0}^{N-1} w_{j,t_i}(N-i)}$
    – Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta\boldsymbol{\theta}$
    – Create one noiseless roll-out to check the trajectory cost $R = \phi_{t_N} + \sum_{i=0}^{N-1} r_{t_i}$. In case the noise cannot be turned off, i.e., a stochastic system, multiple roll-outs need to be averaged.
```

B. Task 1: Passing through a way-point whilst balancing

In this section, we evaluate PI^2 in the context of learning the joint trajectories for a way-point task executed on a 34-DOF humanoid robot. The task is for a standing humanoid robot to pass through a way-point with its right hand, and return to the initial standing position without falling, all within 3s. The experiment was conducted with the 34-DOF robot 'CBi' [2], simulated with the Simulation Lab software [7]. The simulated robot and the position of the way-point are depicted in Fig. 1. This task is difficult because the way-point lies just within the workspace of the robot, and simply reaching for the way-point with the right arm will cause the robot to fall, even if it done very slowly. Therefore, passing through the way-point requires a full-body motion that exploits the momentum of certain links to remain standing.

The joint trajectories are represented by a 34-dimensional DMP. The duration of the movement is 3.0s. The initial 'movement' of the robot before learning is that it stands still at its default position, with constant joint angles.

The cost function for PI^2 consists of three parts. At

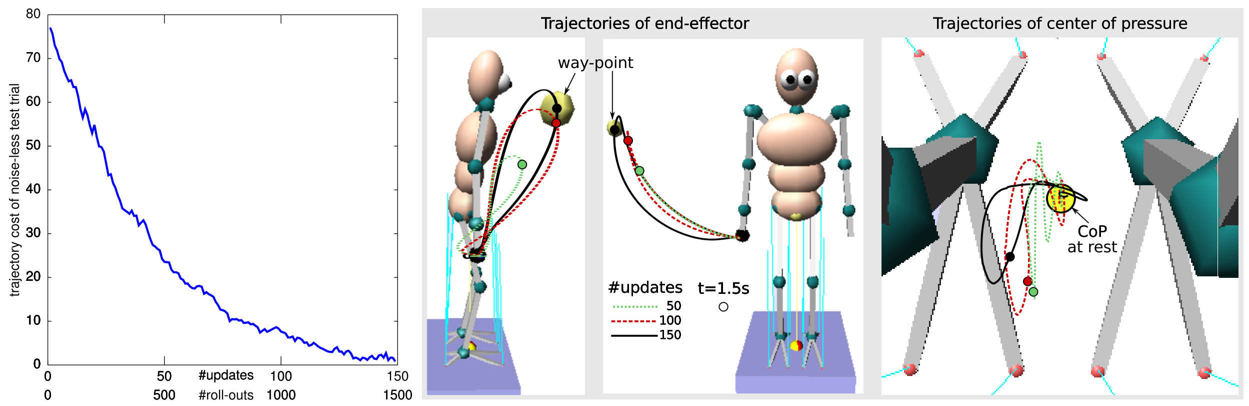

Fig. 1. Left: Learning curve for the way-point task. Right: Trajectories of the end-effector and the center of pressure after 50, 100, 150 updates.

1.5s, there is an intermediate cost which is equal to the distance from the position of the right hand $\mathbf{x}_{t=1.5s}$ to the waypoint: $r^{waypoint}_{t=1.5s} = |\mathbf{x}_{t=1.5s} - \mathbf{x}^{waypoint}|$. There is also an immediate cost for the distance of the center of pressure $\mathbf{c}$ to its default position when standing still. If the center of pressure is within the support polygon, the cost is $r^{CoP}_t = \frac{1}{N}|\mathbf{c}_t - \mathbf{c}^{default}|$. This cost is divided by the number of time steps of the trajectory N, to be independent of trajectory duration. If the center of pressure is outside of the support polygon, the robot is falling, which is severely punished with a value of $r^{falling}_t = 1$ for each simulation tick, at 500Hz. Although the actual movement takes 3s, the simulation is continued and costs are recorded for 5s more, as the robot might fall after the movement is finished.

Finally, the parameters for PI^2 are as follows. The exploration noise for each joint is $\epsilon = 2\lambda^n$, with the decay parameter $\lambda = 0.98$ and n the number of learning updates of PI^2 . Since the arms can be moved more vigorously without causing the robot to fall, the exploration noise of the joints belonging to the arms are set higher, to $\epsilon = 10\lambda^n$. Before each update, 10 roll-outs are executed on the robot, and the 5 roll-outs with lowest cost from the previous parameter update are kept, so each update is computed over $K = 15$ trajectories. This elitarianist reuse of roll-outs make sure the robot always has some 'good examples' amongst the trajectories, without having to actually execute extra roll-outs.

a) Results: The learning curve of this task is depicted in Fig. 1. The trajectories of the end-effector and center of pressure after 50, 100 and 150 updates are depicted in Fig. 1. As learning progresses, the end-effector comes closer to the way-point, and the trajectory of the center of pressure becomes smoother. After 1500 roll-outs and 150 updates, the distance of the end-effector to the waypoint is 0.45cm, and the mean distance of the center of pressure to the default position is 0.35cm. This leads to a final cost of 0.80. A movie of the resulting movement can be downloaded from: `http://www-clmc.usc.edu/movies/humanoids2010/`

To investigate the role of each joint in the learned overall movement, we compare two measures. The first is the range of movement of a joint, i.e. subtracting the minimum from the maximum joint angle along the trajectory. The second is the cost of trajectory when performing the final learned movement, but keeping one of the joints *fixed* at the default position. The new trajectory cost of a trial with the n^{th} joint 'knocked out' is an indication of the importance of joint n to the overall movement. These two measures are plotted against each other in Fig. 2. Note the logarithmic scale on the x-axis.

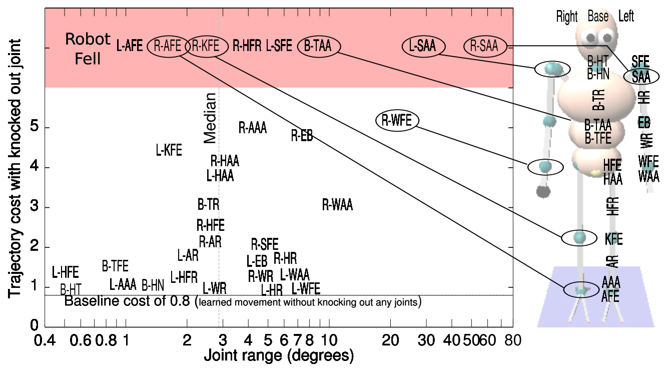

Fig. 2. The effect of 'knocking out' each joint (by fixing it to the default angle) on the trajectory cost. The shaded (red) area at the top indicates that the robot fell when knocking out a joint.

The joints with the largest range are the right shoulder abductor (R-SAA, 70°, to make the reaching movement), the left shoulder abductor (L-SAA, 28°, for counter-balance), as well as the right wrist extensor (R-WFE, 20°) and torso abductor (B-TAA, 9°). Although most joints have a range of less than 3° (the median is 2.8°), they also play an important role in achieving the task. For instance, the flexor-extensor joint of the right knee (R-KFE) and ankle (R-AFE) have a maximum rotation of only 2.3° and 1.7° respectively, but not moving them causes the robot to fall (indicated by the red shaded area at the top). These joints are responsible for leaning forward. Most joint trajectories do not make the robot fall when knocked out, but still lead to a much higher cost (> 0.8) than when performing the movement with all learned joints. This figure demonstrates that achieving the task requires a full-body movement.

We perform two further experiments to analyze the role of dynamics during the movement. First, the robot slowly (100s) reaches for the waypoint using differential inverse kinematics based on the pseudo-inverse of the Jacobian. In the second experiment we execute the learned movement 2, 10 and 100 times slower (i.e. the duration is 6s, 30s, 300s respectively), so that dynamics are negligible. In all these experiments, the robot falls. This demonstrates that a quasi-static movement (as in [6], [13]) does not suffice to achieve this task, but that the robot must exploit the momentum of different body links to remain balanced.

To summarize, the robot learns to achieve this dynamic task using model-free reinforcement learning in a 34-dimensional continuous space, using only 1500 roll-outs. More generally, it shows that PI^2 is able to exploit the dynamics of the system being controlled.

IV. PI^2 for Variable Impedance Control

In this section, we demonstrate how PI^2 is used to learn task-relevant gain schedule for a manipulation task. PI^2 thus not only learns to optimize a planned trajectory, but also time-dependent control parameters. The resulting *variable impedance control* [3] has advantages over high proportional-derivative (PD) error feedback control because 1) it is known from optimal control theory that a finite time optimal controller involves time varying gains, 2) fixed task space (e.g. end-effector) impedance requires varying joint space gains, furthermore it 3) requires less energy, 4) reduces wear-and-tear on the robot, 5) leads to more compliant robots, and therefore safer interactions with the environment and humans.

In the following we will consider robots with torque controlled joints. The motor commands $\mathbf{u}$ are calculated via a PD control law with feed-forward control term $\mathbf{u}_{ff}$:

$$\mathbf{u} = -\mathbf{K}_P(\mathbf{q} - \mathbf{q}_d) - \mathbf{K}_D(\dot{\mathbf{q}} - \dot{\mathbf{q}}_d) + \mathbf{u}_{ff} \qquad (14)$$

where $\mathbf{K}_P$, $\mathbf{K}_D$ are the positive definite position and velocity gain matrices, $\mathbf{q}$, $\dot{\mathbf{q}}$ are the joint positions and velocities, and $\mathbf{q}_d$, $\dot{\mathbf{q}}_d$ are the desired joint positions and velocities. The feed-forward control term may come, for instance, from an inverse dynamics control component, or a computed torque control component [8]. Thus, the impedance of a joint is parameterized by the choice of the gains $\mathbf{K}_P$ ("stiffness") and $\mathbf{K}_D$ ("damping").

The PI^2 algorithm as introduced above seems to be solely suited for optimizing a trajectory plan, and not directly the controller. However, the term $\mathbf{g}(\boldsymbol{\theta} + \boldsymbol{\epsilon})$ must not necessarily be used as the non-linear component of a DMP, but, as demonstrated in [1], can also be used to encode the proportional gain of a joint over time as follows:

$$\begin{aligned} K_{P,i} &= \mathbf{g}_{t,K}^{i,T}(\boldsymbol{\theta}_K^i + \boldsymbol{\epsilon}_{K,t}^i) && (15) \\ [\mathbf{g}_t]_j &= \frac{w_j}{\sum_{k=1}^{p} w_k} && (16) \\ w_j &= \exp\left(-0.5 h_j (s_t - c_j)^2\right) && (17) \end{aligned}$$

These gains are coupled to the DMP through the canonical system in Eqn. 11, and s_t is used as a phase variable in $\mathbf{g}$ for both the joint accelerations (Eqn. 10) and gains (Eqn. 15). Thus, $K_{P,i}$ is represented by a basis function representation linear with respect to the learning parameter $\boldsymbol{\theta}_K^i$, and these parameter can be learned with the PI^2 algorithm.

To compute the damping gain K_D^i, we use the common practice that it can be written as the square root of the proportional gain K_P^i with a user determined multiplier ξ^i. This multiplier is computed from the initial default gains of the robot, which were manually tuned.

Summarizing, for an N degree-of-freedom robot, we have N differential equations as in Eqn. 10 that represent the joint accelerations $\ddot{\mathbf{q}}$ over time, and N functions as in Eqn. 15 that represent the gain schedules $K_{P,D}$ for each joint. All these systems are coupled with the canonical system in Eqn. 11. This leads to $2N$ parameter vectors $\boldsymbol{\theta}$, which PI^2 optimizes w.r.t. a cost function.

A. Task 2: Pushing open a door

We apply simultaneous learning of trajectories and gain schedules with PI^2 to the simulated CBi humanoid robot, where the task is to open a door. Learning such variable impedance control enables the robot to keep its gains as low as possible (with resulting energy efficiency, reduced wear-and-tear, and compliance), switching to high-gain control only when the task requires it (i.e. when force is required to open the door).

In this task, we fix the base of the robot, and consider only the 7 degrees of freedom in the left arm. The initial trajectory before learning is a minimum jerk trajectory in joint space. In the initial state, the upper arm is kept parallel to the body, and the lower arm is pointing forward. The target state is depicted in Fig. 3. The intention of this task is not to learn the movement required to open the door from scratch, but rather to learn the gain schedules and fine-tuning of the trajectories. The gains of the 7 joints are initialized to $1/10th$ of their default values. This leads to extremely compliant behavior, whereby the robot is not able to exert enough force to overcome the static friction of the door, and thus cannot move it. Optimizing both joint trajectories and gains leads to a 14-dimensional learning problem.

The terminal cost is the degree to which the door was opened, i.e. $\phi_{t_N} = 10^4 \cdot (\psi_{max} - \psi_N)$, where the maximum door opening angle ψ_{max} is $0.3rad$ (it is out of reach otherwise). The immediate cost for the gains is $r_t = \frac{1}{N}\sum_{i=1}^{3} K_P^i$. The sum of the gains of all joints is divided by the number of time steps of the trajectory N, to be independent of trajectory duration. The cost for the gains expresses our preference for low gain control.

Finally, the parameters for PI^2 are as follows. The exploration noise for each joint is $\epsilon = 10\lambda^n$ as before, and for the gains is $\epsilon = 10^{-4}\lambda^n$, both with decay parameter $\lambda = 0.99$ and n the number of updates[1]. The number of executed and reused 'elite' roll-outs is both 5, so the number of roll-outs on which the update is performed is $K = 10$.

[1]The relatively low exploration noise for the gains does not express less exploration per se, but is rather due to numerical differences in using the function approximator to model the gains directly (Equation 15) rather than as the non-linear component of a DMP (Equation 10).

b) Results: Fig. 3 (right) depicts the total cost of the noise-less test trial after each update. The costs for the gains are plotted separately. Coincidence of two graphs implies that all costs are attributed to the gains, and thus the task of opening the door to ψ_{max} is achieved.

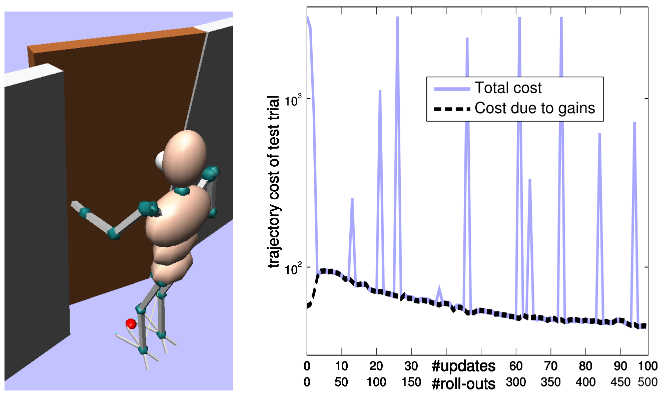

Fig. 3. Left: Task scenario. Right: Learning curve for the door task. The costs specific to the gains are plotted separately.

The joint trajectories and gain schedules after 0, 6 and 100 updates are depicted in Fig. 4.

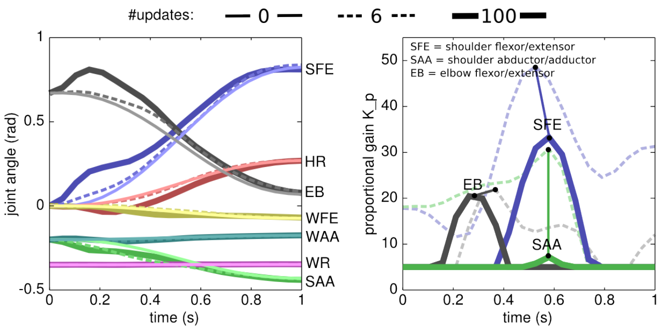

Fig. 4. Learned joint angle trajectories (center) and gain schedules (right) of the CBi arm after 0/6/100 updates. The gain schedules of only three joints have been depicted for sake of clarity.

The learning goes through two phases. In the first 6 updates, the gains are increased in order to open the door. This leads to a strong decrease in the penalty of not fully opening the door, at the cost of a higher penalty for higher gains, as depicted in Fig. 3 (right). Essentially, the robot is learning that it is able to solve the task with high-gain control. This is also apparent when inspecting the (dashed) gain schedules after 6 updates in Fig. 4.

In the second phase which starts after 6 updates, the exact timing and magnitudes of the joint trajectories and gains required to open the door are determined. During this process, the robot sometimes lowers the gains too much, and is not able to open the door, as indicated by the spikes in the learning curve. That the robot is always able to open the door one update after a spike is because of the elitarianism, which always leads to at least some roll-outs with successful door opening to be among the pool of 10 roll-outs on which the parameter update is performed.

Finally, after 100 updates, the gains are actually 25% lower than at initialization, but by timing and tuning them appropriately as depicted in Fig. 4, the robot is now able to open the door.

V. Conclusion

In this paper we demonstrate that PI^2 is able to efficiently learn humanoid motor skills which require full-body motion and variable impedance control, and involve direct contact with the environment. The main reasons for this is that PI^2 learns in *continuous* state and action spaces, and it is thus straight-forward to apply directly to planned joint trajectories. Furthermore, PI^2 scales to *high-dimensional* spaces, which are typical for humanoid robot tasks. It is also general enough to be applied not only to *planned trajectories*, but also to learning variable *gain schedules*. Because PI^2 is *model-free*, it is possible to use it for manipulation tasks which involve physical contact with the environment, for which it is difficult to acquire a model. Finally, the *one open parameter* in PI^2 is the exploration noise, which has a clear physical meaning, and does not require precise tuning for successful and fast learning.

Future work includes the evaluation of PI^2 on the physical CBi robot, and the use of imitation to determine initial values for the policy parameter vector $\boldsymbol{\theta}$. Determining $\boldsymbol{\theta}$ from observed (human) motion is straight-forward within the Dynamic Movement Primitive framework, and we expect that seeding the learning process with example trajectories will lead to even faster learning of humanoid motor skills.

References

[1] J. Buchli, E. Theodorou, F. Stulp, and S. Schaal. Variable impedance control - a reinforcement learning approach. In *Robotics: Science and Systems Conference (RSS)*, 2010.

[2] G. Cheng, S. Hyon, J. Morimoto, A. Ude, J.G. Hale, G. Colvin, W. Scroggin, and S. C. Jacobsen. Cb: A humanoid research platform for exploring neuroscience. *Journal of Advanced Robotics*, 21(10):1097–1114, 2007.

[3] N. Hogan. Impedance control - an approach to manipulation. *ASME, Transactions, Journal of Dynamic Systems, Measurement, and Control*, 107:1–24, 1985.

[4] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *ICRA*, 2002.

[5] J. Kober and J. Peters. Learning motor primitives in robotics. In *Advances in Neural Information Processing Systems 21 (NIPS)*, 2009.

[6] J.J. Kuffner, S.Kagami, K. Nishiwaki, M. Inaba, and H. Inoue. Dynamically-stable motion planning for humanoid robots. *Autonomous Robots*, 12(1):105–118, 2002.

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

[8] L. Sciavicco and B. Siciliano. *Modelling and Control of Robot Manipulators*. Springer, London, New York, 2000.

[9] L. Sentis. *Synthesis and control of whole-body behaviors in humanoid systems*. PhD thesis, Stanford University, 2007.

[10] R.F. Stengel. *Optimal Control and Estimation*. Dover Publications, New York, 1994.

[11] R. S. Sutton and A. G. Barto. *Reinforcement learning : An introduction*. MIT Press, 1998.

[12] E. Theodorou, J. Buchli, and S. Schaal. Reinforcement learning in high dimensional state spaces: A path integral approach. *Journal of Machine Learning Research*, 2010. To appear.

[13] M. Toussaint. Robot trajectory optimization using approximate inference. In *Proc. 25th Int'l Conf. on Machine Learning*, 2009.

[14] B. van den Broek, W. Wiegerinck, and B. Kappen. Graphical model inference in optimal control of stochastic multi-agent systems. *Journal of Artificial Intelligence Research*, 32:95–122, 2008.