Learning Variable Impedance Control

Jonas Buchli*, Freek Stulp, Evangelos Theodorou, Stefan Schaal †

November 16, 2011

Abstract

One of the hallmarks of the performance, versatility, and robustness of biological motor control is the ability to adapt the impedance of the overall biomechanical system to different task requirements and stochastic disturbances. A transfer of this principle to robotics is desirable, for instance to enable robots to work robustly and safely in everyday human environments. It is, however, not trivial to derive variable impedance controllers for practical high degree-of-freedom (DOF) robotic tasks.

In this contribution, we accomplish such variable impedance control with the reinforcement learning (RL) algorithm PI^2 (**P**olicy **I**mprovement with **P**ath **I**ntegrals). PI^2 is a model-free, sampling based learning method derived from first principles of stochastic optimal control. The PI^2 algorithm requires no tuning of algorithmic parameters besides the exploration noise. The designer can thus fully focus on cost function design to specify the task. From the viewpoint of robotics, a particular useful property of PI^2 is that it can scale to problems of many DOFs, so that reinforcement learning on real robotic systems becomes feasible. We sketch the PI^2 algorithm and its theoretical properties, and how it is applied to gain scheduling for variable impedance control.

We evaluate our approach by presenting results on several simulated and real robots. We consider tasks involving accurate tracking through via-points, and manipulation tasks requiring physical contact with the environment. In these tasks, the optimal strategy requires both tuning of a reference trajectory *and* the impedance of the end-effector.

The results show that we can use path integral based reinforcement learning not only for planning but also to derive variable gain feedback controllers in realistic scenarios. Thus, the power of variable impedance control is made available to a wide variety of robotic systems and practical applications.

1 Introduction

Biological motor systems excel in terms of versatility, performance, and robustness in environments that are highly dynamic, often unpredictable, and partially stochastic. Whereas classical robotics is mostly characterized by high

*Current Address: Dept. of Advanced Robotics, Italian Institute of Technology, Via Morego 30, 16163 Genova, Italy

†The authors are at the Computational Learning and Motor Control Lab, University of Southern California, Los Angeles, CA 90089, USA jonas@buchli.org, stulp@clmc.usc.edu, {theodor,sschaal}@usc.edu

To appear in International Journal of Robotics Research – (c) SAGE Publications Inc.

gain negative error feedback control, biological systems derive some of their superiority from low gain compliant control with variable and task dependent impedance [26]. If we adapt this concept of adaptive impedance for PD negative error feedback control, this translates into time varying proportional and derivative gains, also known as gain scheduling. Finding the appropriate gain schedule for a given task is, however, a hard problem [8, 27].

One possible solution to this problem is Reinforcement Learning (RL) [31]. The idea of RL is that, given only a reward function, the learning algorithm finds strategies that yield high reward through trial and error. As a special and important feature, RL can accomplish such optimal performance *without* knowledge of the models of the motor system and/or the environment. This property is especially appealing for learning how to interact with objects and the environment, as good contact models are notoriously difficult to obtain. However, so far, RL does not scale well to high-dimensional continuous state-action control problems.

Closely related to RL is optimal control theory [29], where gain scheduling is a natural outcome of many optimal control algorithms. However, optimal control requires *model-based* derivations, such that it is frequently not applicable to complex robotic systems and environments, where models are unknown or not sufficiently known.

In this paper, we present PI^2 (**P**olicy **I**mprovement with **P**ath **I**ntegrals) [33], an RL algorithm which is derived from first principles of stochastic optimal control, and which *does* scale to complex robotic system [30].

The PI^2 algorithm is tailored to optimize Dynamic Movement Primitives (DMPs), a specific implementation of a parameterized policy, based on a set of dynamical system equations [11]. The system overview in Figure 1 illustrates how PI^2 simultaneously optimizes planned trajectories and gain schedules in a DMP. First, a DMP is initialized with an initial trajectory and constant gains. Here, the planned trajectory and gain schedule of each joint are represented as separate dimensions in the DMP. For a robot with n joints, the DMP therefore has $2n$ dimensions. PI^2 then iteratively updates the DMP parameters by executing it to accomplish the desired behavior, and computing the cost for each resulting trajectory with the task-specific cost function. During execution, exploration is ensured by adding exploration noise to the DMP parameters. Learning continues until the cost converges, or a certain number of iterations is reached. With this approach, PI^2 is able to simultaneously optimize both the trajectories and gain schedules, as both are homogeneously represented in the same DMP.

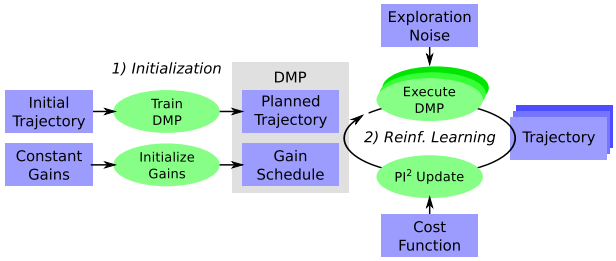

Figure 1: System overview. After initialization, the planned trajectory and gain schedules of a Dynamic Movement Primitive are optimized with respect to a cost function with the Reinforcement Learning algorithm PI^2 .

By penalizing high gains in the cost function, the robot learns to be compliant when it can be, and stiffens up only when the task requires it. In comparison to high-gain control approaches, this

To appear in International Journal of Robotics Research – (c) SAGE Publications Inc.

leads to lower energy consumption, less wear-and-tear for the robot, and safer human-robot interaction.

We evaluate our approach on three different simulated robotic systems, a 3-DOF Phantom Premium Robot, a 6-DOF Kuka Lightweight Robot, and the humanoid SARCOS CBi. We also validate the results on a real Phantom Premium Robot. The tasks require accurate tracking through a via-point or physical manipulation of objects in the environment.

The main contributions of this article are: 1) the use of parametrized control policies to represent the parameters not only of a reference trajectory, but also of a feedback controller. As the reinforcement algorithm PI2 is able to optimize the parameters of all policies simultaneously, we thus present a novel formulation of learning both the feedback controller *and* the reference trajectory on a multi-dimensional robotic system in a model-free reinforcement learning setting. 2) applying this approach to learning variable gain schedules for a PD controller; 3) implementing and evaluating the proposed method on both simulated and real robotic systems; 4) demonstrating how the learned gain schedules enable the robot to be as compliant as possible, stiffening up only when the task requires it.

In this paper we are building on initial work presented in [3], and show further results by applying the proposed approach to manipulation tasks and a real robot. We also provide a more extensive discussion of related work.

The rest of this paper is structured as follows. We first motivate variable impedance control in Section 2. In Section 3, we present our RL algorithm PI^2 , and explain how it is applied to variable gain scheduling in Section 4. An empirical evaluation of our methods on simulated and real robots is presented in Section 5. In Section 6, we discuss related concepts and work. Finally, we conclude with Section 7.

2 Variable impedance control

The classical approach to robot control is negative feedback control with high proportional-derivative (PD) gains. This type of control is straightforward to implement, robust towards modeling uncertainties, and computationally cheap. Unfortunately, high gain control is not ideal for many tasks involving interaction with the environment, e.g. force control tasks or locomotion. In contrast, impedance control [8] seeks to realize a specific impedance of the robot, either in end-effector or joint space. The issue of specifying the target impedance, however, is not completely addressed as of yet. While for simple factory tasks, where the properties of the task and environment are known a priori, suitable impedance characteristics may be derivable, it is usually not easy to understand how impedance control is applied to more complex tasks such as a walking robot over difficult terrain or the manipulation of objects in daily life (e.g. pillows, hammers, cans, etc.). An additional benefit of variable impedance behavior in a robot comes from the added active safety due to soft "giving in", both for the robot and its environment.

In the following we consider robots with torque controlled joints. The motor torques $\mathbf{T}$ are calculated via a PD control law with feedforward control term $\mathbf{T}_{ff}$:

$$\mathbf{T} = -\mathbf{K}_P(\mathbf{q} - \mathbf{q}_d) - \mathbf{K}_D(\dot{\mathbf{q}} - \dot{\mathbf{q}}_d) + \mathbf{T}_{ff} \quad (1)$$

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 [24]. Thus, the impedance of a joint is parameterized by the choice of the gains $\mathbf{K}_P$ ("stiffness") and $\mathbf{K}_D$ ("damping").

For many applications, the joint space impedance is, however, of secondary interest. Most often, regulating impedance matters the most at certain points that contact with the environment, e.g., the end-effectors of the robot. We therefore need to assess the impedance at these points of contacts rather than the joints. Joint space impedance is computed from the desired task space impedance $\mathbf{K}_{P,\mathbf{x}}, \mathbf{K}_{D,\mathbf{x}}$ by help of the Jacobian $\mathbf{J}$ of the forward kinematics of the robot as follows [24]:

$$\mathbf{K}_{P,\mathbf{q}} = \mathbf{J}^T \mathbf{K}_{P,\mathbf{x}} \, \mathbf{J} \quad \text{and} \quad \mathbf{K}_{D,\mathbf{q}} = \mathbf{J}^T \mathbf{K}_{D,\mathbf{x}} \, \mathbf{J} \tag{2}$$

Here we assume that the geometric stiffness due to the change of the Jacobian is negligible in comparison to the terms in Eq.(2). Regulating the task space impedance thus implies regulating the joint space impedance. Furthermore, this fundamental mathematical relationship between joint and task space also implies that a constant task stiffness in general means varying gains at the joint level.

In the next section we will sketch a reinforcement learning algorithm that is then applied to learning the time dependent gain matrices.

3 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 [33] 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 most important points in the development of the PI^2 algorithm as needed in this paper. The development of the algorithm in its entirety can be found in [33].

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 controlled system is of the form

$$\dot{\mathbf{x}}_t = \mathbf{f}(\mathbf{x}_t, t) + \mathbf{G}(\mathbf{x}_t) \left(\mathbf{u}_t + \epsilon_t\right) = \mathbf{f}_t + \mathbf{G}_t \left(\mathbf{u}_t + \epsilon_t\right) \tag{3}$$

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 variance $\mathbf{\Sigma}_\epsilon$. 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[1]

$$V(\mathbf{x}_{t_i}) = V_{t_i} = \min_{\mathbf{u}_{t_i:t_N}} E_{\boldsymbol{\tau}_i} \left[R(\boldsymbol{\tau}_i)\right] \tag{4}$$

[1] Eq. 4 is using a shorthand notation $E_{\boldsymbol{\tau}_i}$ for the expectation of the finite horizon cost $R(\boldsymbol{\tau}_i)$. The finite horizon cost is a random variable with probability distribution $p(\boldsymbol{\tau}_i | \mathbf{u}_{t_i:t_N})$ and therefore $E_{\boldsymbol{\tau}_i}\left[R(\boldsymbol{\tau}_i)\right] = \int p(\boldsymbol{\tau}_i | \mathbf{u}_{t_i:t_N}) R(\boldsymbol{\tau}_i) d\boldsymbol{\tau}_i$. $\mathbf{u}_{t_i:t_N}$ is a 'parameter' influencing the probability distribution and ultimately the value of the expectation, used in defining the value function. And therefore minimization has to happen with respect to this parameter.

To appear in International Journal of Robotics Research – (c) SAGE Publications Inc.

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

$$R(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} r_t \, dt \tag{5}$$

and where $\phi_{t_N} = \phi(x_{t_N})$ is a terminal cost at time t_N. r_t denotes the immediate cost at time t. As immediate cost we consider:

$$r_t = r(\mathbf{x}_t, \mathbf{u}_t, t) = q_t + \frac{1}{2}\mathbf{u}_t^T \mathbf{R} \mathbf{u}_t \tag{6}$$

where $q_t = q(\mathbf{x}_t, t)$ is an arbitrary state-dependent cost function, and $\mathbf{R}$ is the positive semi-definite weight matrix of the quadratic control cost.

Based on the principles of stochastic optimal control [29] and as detailed in [33] by minimizing the Hamilton-Jacobi-Bellman (HJB) equation of our problem we can derive a 2^{nd} order partial differential equation for the time derivative of the value function:

$$\begin{aligned} -\partial_t V_t &= q_t + (\nabla_{\mathbf{x}} V_t)^T \mathbf{f}_t \\ &\quad -\frac{1}{2}(\nabla_{\mathbf{x}} V_t)^T \mathbf{G}_t \mathbf{R}^{-1} \mathbf{G}_t^T (\nabla_{\mathbf{x}} V_t) \\ &\quad +\frac{1}{2} trace\left((\nabla_{\mathbf{xx}} V_t) \mathbf{G}_t \boldsymbol{\Sigma}_\epsilon \mathbf{G}_t^T\right) \end{aligned} \tag{7}$$

The same principles also provide us with a result for the corresponding optimal control input, which is a function of the state and is given by the equation:

$$\mathbf{u}(\mathbf{x}_{t_i}^*) = \mathbf{u}_{t_i}^* = -\mathbf{R}^{-1} \mathbf{G}_{t_i}^T (\nabla_{x_{t_i}} V_{t_i}) \tag{8}$$

We are leaving the standard development of this optimal control problem by transforming the HJB equation with the substitution $V_t = -\lambda \log \Psi_t$ and by introducing the assumption[2] that $\lambda \mathbf{R}^{-1} = \boldsymbol{\Sigma}_\epsilon$. This assumption allows us to simplify the mathematical treatment of the HJB equation. As shown in detail in [33] this way, the transformed HJB equation becomes a linear 2^{nd} order partial differential equation:

$$\begin{aligned} -\partial_t \Psi_t &= -\frac{1}{\lambda} q_t \Psi_t + \mathbf{f}_t^T (\nabla_{\mathbf{x}} \Psi_t) \\ &\quad + \frac{1}{2} tr\left((\nabla_{\mathbf{xx}} \Psi_t) \mathbf{G}_t \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}} \mathbf{G}_t^T\right) \end{aligned} \tag{9}$$

with boundary condition $\Psi_{t_N} = \exp\left(-\frac{1}{\lambda}\phi_{t_N}\right)$. Using the Feynman-Kac theorem [21, 41, 33], the solution for the exponentially transformed value function becomes:

$$\Psi_{t_i} = \lim_{dt \to 0} \int p\left(\boldsymbol{\tau}_i | \mathbf{x}_i\right) \exp\left[-\frac{1}{\lambda}\left(\phi_{t_N} + \sum_{j=0}^{N-1} q_{t_j} dt\right)\right] d\boldsymbol{\tau}_i \tag{10}$$

Thus, we have transformed our stochastic optimal control problem into an approximation problem of a path integral. As detailed in [33], it is not necessary to compute the value function explicitly, but rather it is possible to derive the optimal controls directly. The optimal controls take again the form of an expectation:

$$\mathbf{u}_{t_i} = \int P\left(\boldsymbol{\tau}_i\right) \mathbf{u}\left(\boldsymbol{\tau}_i\right) d\boldsymbol{\tau}_i \tag{11}$$

$$\begin{aligned} \mathbf{u}(\boldsymbol{\tau}_i) &= \mathbf{R}^{-1} \mathbf{G}_{t_i}{}^T \left(\mathbf{G}_{t_i} \mathbf{R}^{-1} \mathbf{G}_{t_i}{}^T\right)^{-1} \\ &\quad \left(\mathbf{G}_{t_i} \boldsymbol{\epsilon}_{t_i} - \mathbf{b}_{t_i}\right) \end{aligned} \tag{12}$$

where $P\left(\boldsymbol{\tau}_i\right)$ is the probability of a trajectory $\boldsymbol{\tau}_i$,

$$P\left(\boldsymbol{\tau}_i\right) = \frac{e^{-\frac{1}{\lambda} S(\boldsymbol{\tau}_i)}}{\int e^{-\frac{1}{\lambda} S(\boldsymbol{\tau}_i)} d\boldsymbol{\tau}_i},$$

$S(\boldsymbol{\tau}_i)$ is the generalized cost (cf. Table 1), and $\mathbf{b}_{t_i}$ is a more complex expression, beyond the space constraints of this paper.

[2] In the final algorithm the parameter λ is set automatically, cf. [33].

A post-hoc interpretation of this result, which confirms the intuition about how optimal controllers should be chosen, is: The probability $P(\boldsymbol{\tau}_i)$ is the weighting of a local control with the cost function. The optimal control is thus the expectation of the local controls when they are associated with a probability that decreases for controllers yielding a high cost. The important conclusion is that it is possible to evaluate Eq. (11) using Monte Carlo sampling [5] of the control system, i.e., our optimal control problem can be solved as an estimation problem. Eq. (11) can therefore be approximated by drawing random samples of the noise vector $\boldsymbol{\epsilon}_{t_i}$ and calculating the associated probability $P(\boldsymbol{\tau}_i)$ by forward-integrating the system dynamics and calculating the costs. The P-weighted sum of the local controls $\mathbf{u}(\boldsymbol{\tau}_i)$ of these samples then approximates the value of the integral. In application to robot learning, the forward integration of the system dynamics is replaced by drawing local controls from a probability distribution and running the randomized controllers on the real system. The cost statistics are then collected from these experiments. Each of these experiments is called a *roll-out*.

3.1 The PI^2 Algorithm

The PI^2 algorithm is just a special case of the optimal control solution in Eq. (11), applied to control systems with parameterized control policy:

$$\mathbf{a}_t = \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_t) \tag{13}$$

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 control system with parameterized policy is the Dynamic Movement Primitives (DMP) approach introduced by [11]:

$$\begin{aligned}
\frac{1}{\tau}\dot{v}_t &= f_t + \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_t) && (14)\\
\frac{1}{\tau}\dot{q}_{d,t} &= v_t \\
f_t &= \alpha(\beta(g - q_{d,t}) - v_t) \\
\frac{1}{\tau}\dot{s}_t &= -\alpha s_t && (15)\\
[\mathbf{g}_t]_j &= \frac{w_{j,t}s_t}{\sum_{k=1}^{p} w_{k,t}}(g - q_0) && (16)\\
w_j &= \exp\left(-0.5h_j(s_t - c_j)^2\right) && (17)
\end{aligned}$$

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, 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 [11], for this paper the important ingredients of DMPs are that i) the overall system formed by the attractor system Eq. (14) coupled with a nonlinear robot dynamics via the control law Eq. (1) has the same form as Eq. (3), and that ii) the p-dimensional parameter vector $\boldsymbol{\theta}$ can be interpreted as motor commands as used in the path integral approach to optimal control (i.e. $\mathbf{u} = \boldsymbol{\theta}$). Learning the optimal values for $\boldsymbol{\theta}$ will thus create a optimal reference trajectory for a given motor task. A further key step in the application of the path integral theory to robot learning problems is its formulation as an iterative algorithm. Instead of evaluating the path integral Eq. (11) with a large number of samples and calculating the optimal vector directly, the optimal value is approached iteratively. By not sampling

large irrelevant parts of the state space, solving high DOF learning problems thus becomes feasible. The PI^2 learning algorithm applied to this scenario is summarized in Table 1. As illustrated in [33, 34], 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. [33]). 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 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 into a probabilistic estimation problem which can then be solved by sampling techniques. This transformation is achieved with the assumption $\lambda \mathbf{R}^{-1} = \boldsymbol{\Sigma}_\epsilon$ to transform a nonlinear partial differential equation (PDE) into a linear one, and the use the Feynman-Kac lemma [21, 33] to approximate its solution.

- Paths with higher cost have lower probability. A clear intuition that has also rigorous mathematical representation through the exponentiation of the value function. This transformation is necessary for the linearization of HJB into a linear 2^{nd} order partial differential equation.

input : $r_t = q_t + \boldsymbol{\theta}_t^T \mathbf{R} \boldsymbol{\theta}_t$; *immediate cost function*
ϕ_{t_N} ; *terminal cost term*
$\mathbf{a}_t = \mathbf{g}_t^T (\boldsymbol{\theta} + \boldsymbol{\epsilon}_t)$; *parameterized policy*
$\mathbf{g}_{t_i}$; *basis function from the system dynamics*
Σ_ϵ ; *variance of the mean-zero noise* $\boldsymbol{\epsilon}_t$
$\boldsymbol{\theta}_{init}$; *initial parameter vector*
K ; *number of roll-outs per update*
output : $\boldsymbol{\theta}$; *final parameter vector*

while *trajectory cost R not converged* **do**
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 ($\boldsymbol{\epsilon}_{t,k} \sim N(0, \gamma^{(\#\text{updates so far})} \cdot \sigma^2)$.
foreach k **in** K ; *for all roll-outs*
do
$\mathbf{M}_{t_j,k} = \frac{\mathbf{R}^{-1} \mathbf{g}_{t_j,k} \mathbf{g}_{t_j,k}^T}{\mathbf{g}_{t_j,k}^T \mathbf{R}^{-1} \mathbf{g}_{t_j,k}}$
$S(\boldsymbol{\tau}_{i,k}) = \phi_{t_N,k} + \sum_{j=i}^{N-1} q_{t_j,k} +$
$\frac{1}{2} \sum_{j=i+1}^{N-1} (\boldsymbol{\theta} + \mathbf{M}_{t_j,k} \boldsymbol{\epsilon}_{t_j,k})^T \mathbf{R} (\boldsymbol{\theta} + \mathbf{M}_{t_j,k} \boldsymbol{\epsilon}_{t_j,k})$
$P(\boldsymbol{\tau}_{i,k}) = \frac{e^{-\frac{1}{\lambda} S(\boldsymbol{\tau}_{i,k})}}{\sum_{k=1}^{K} [e^{-\frac{1}{\lambda} S(\boldsymbol{\tau}_{i,k})}]}$
end
foreach i **in** N ; *for all time steps*
do
$\delta \boldsymbol{\theta}_{t_i} = \sum_{k=1}^{K} [P(\boldsymbol{\tau}_{i,k}) \mathbf{M}_{t_i,k} \boldsymbol{\epsilon}_{t_i,k}]$
$[\delta \boldsymbol{\theta}]_j = \frac{\sum_{i=0}^{N-1} (N-i) \; w_{j,t_i} \; [\delta \boldsymbol{\theta}_{t_i}]_j}{\sum_{i=0}^{N-1} w_{j,t_i} (N-i)}$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta \boldsymbol{\theta}$; *parameter update*
end
Create one noiseless roll-out to evaluate the trajectory cost $R = \phi_{t_N} + \sum_{i=0}^{N-1} r_{t_i}$ of the current parameters $\boldsymbol{\theta}$.
end

Table 1: Pseudocode of the PI^2 algorithm for a 1D Parameterized Policy.

- 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 to update the parameters of the DMPs.

- The number of roll-outs per iteration step is a parameter that can be chosen by the user. The number of roll-outs is not very critical and can be chosen to be very small, as shown in the examples. This is very important to render the learning approach feasible on real systems.
- The derivation of an RL algorithm from first principles largely eliminates the need for open parameters in the final algorithm.

4 Variable Impedance Control with PI^2

The PI^2 algorithm as introduced above seems to be solely suited for optimizing a trajectory plan, and not directly the controller. Here we will demonstrate that this is not the case, and how PI^2 can be used to optimize a gain schedule simultaneously to optimizing the reference trajectory. For this purpose, it is important to realize how Eq. (3) relates to a complete robotics system. We assume a d-DOF robot that obeys rigid body dynamics. $\mathbf{q}^v$ denotes the joint velocities, and $\mathbf{q}^p$ the joint angle positions. Every DOF has its own reference trajectory from a DMP, which means that Eqs. (14) are duplicated for every DOF, while Eqs. (15), (16), and (17) are shared across all DOFs – see [11] for explanations on how to create multi-dimensional DMPs. Thus, Eq. (3) applied to this context, i.e. using rigid body dynamics equations, with $\mathbf{M}, \mathbf{C}, \mathbf{G}$ the Inertia matrix, Coriolis/centripedal and gravity forces respectively, and combining them with the reference trajectory generating DMP becomes:

$$\begin{aligned} \dot{\mathbf{q}}^v &= \mathbf{M}(\mathbf{q}^p)^{-1}\left(-\mathbf{C}(\mathbf{q}^p, \mathbf{q}^v) - \mathbf{G}(\mathbf{q}^p) + \mathbf{T}\right) \\ \dot{\mathbf{q}}^p &= \mathbf{q}^v \\ \frac{1}{\tau}\dot{s}_t &= -\alpha s_t \end{aligned} \tag{18}$$

$$\begin{aligned} \frac{1}{\tau}\dot{q}^v_{d,i} &= \alpha(\beta(g_i - q^p_{d,i}) - q^v_{d,i}) + \mathbf{g}_t^{i,T}(\boldsymbol{\theta}^i_{ref} + \boldsymbol{\epsilon}^i_t) \\ \frac{1}{\tau}\dot{q}^p_{d,i} &= q^v_{d,i} \end{aligned} \tag{19}$$

where each element T_i of the torque vector $\mathbf{T}$:

$$\begin{aligned} T_i &= -K_{P,i}\left(q^p_i - q^p_{d,i}\right) - \xi_i\sqrt{K_{P,i}}\left(q^v_i - q^v_{d,i}\right) \\ &+ T_{ff,i} \end{aligned} \tag{20}$$

The terms $q^v_{d,i}, q^p_{d,i}$ are the reference joint angle position and velocity of the ith DOF as computed by the DMP Eq. 19. The control vector to this system is $\mathbf{u}_t = \boldsymbol{\theta}^i_{ref}$.

Note that in the control law in (20), we used Eq. (1) applied to every DOF individually using a time varying gain, and we inserted the common practice that the damping gain K^i_D is written as the square root of the proportional gain K^i_P with a user determined multiplier ξ^i. A critically important result of [33] is that for the application of PI^2 only those differential equations in Eq. (18) matter that have learnable parameter $\boldsymbol{\theta}^i$. Moreover, the optimization of these parameters is accomplished by optimizing the parameter vector of each differential equation independently (as shown in Table 1), despite that the DOFs are coupled through the cost function. For this reason, PI^2 operates in a model free mode, as only one of the DMP differential equations per DOF is required, and all other equations, including the rigid body dynamics model, drop out.

For variable stiffness control, we exploit these insights and add one more differential equation per DOF in Eq. (18):

$$\dot{K}_{P,i} = \alpha_K \left(\mathbf{g}_{t,K}^{i,T} (\boldsymbol{\theta}_K^i + \boldsymbol{\epsilon}_{K,t}^i) - K_{P,i} \right) \quad (21)$$

$$[\mathbf{g}_{t,K}]_j = \frac{w_j}{\sum_{k=1}^p w_k} \quad (22)$$

This equation models the time course of the position gains, coupled to Eq. (15) of the DMP. 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 are learned with the PI^2 algorithm following Table 1. We will assume that the time constant $\frac{1}{\alpha_K}$ is so small, that for all practical purposes we can assume that $K_{P,i} = \mathbf{g}_{t,K}^{i,T} (\boldsymbol{\theta}_K^i + \boldsymbol{\epsilon}_{K,t}^i)$ holds at all times.

Essentially Eqs. (18),(19) and (21) are incorporated in one stochastic dynamical system of the form of Eq. (3). In conclusion, we achieved a novel formulation of learning both the reference trajectory and the gain schedule for a multi-dimensional robotic system with model-free reinforcement learning, using the PI^2 algorithm and its theoretical properties as foundation of our derivations.

5 Empirical Evaluation

We now present results of applying the outlined algorithms to three robots with 3, 6 and 7-DOF respectively. We show four experiments using these robots, three in simulation, and one on a real robot.

The first two experiments serve to illustrate the idea and effects of variable gain schedule learning with two via-point experiments with robotic arms. The other two experiments are manipulation tasks, where a real robot learns to flip a light switch, and the humanoid robot CBi [6] learns to open a door.

5.1 Via-point experiments

In the first two experiments the robot's primary task is to pass through an intermediate goal, either in joint space or end-effector space – such scenarios occur in tasks such as playing tennis.

For these two experiments, we express the goal of the task with the following immediate cost function, which is a task-specific implementation of the generic cost function in Eq. 6:

$$r_t = W_{gain} \sum_i K_{P,t}^i + W_{acc} ||\ddot{\mathbf{x}}|| + W_{via-point} C(t) \quad (23)$$

Here, $\sum_i K_{P,t}^i$ is the sum over the proportional gains over all joints. The reasoning behind penalizing the gains is that low gains lead to several desirable properties of the system such as compliant behavior (safety and/or robustness [2]), lowered energy consumption, and less wear and tear[3]. The term $||\ddot{\mathbf{x}}||$ is the magnitude of the accelerations of the end-effector. This quantity is penalized to avoid high-jerk end-effector motion. This penalty is low in comparison to the gain penalty.

The component of the cost function $C(t)$ that represents this primary task will be described individually for each robot in the next sections. Gains and accelerations are penalized at each time step t, but $C(t)$ only leads to a cost at specific time steps along the trajectory.

[3]During learning, we bound the gains between prespecified maximum and minimum values. Too high gains would generate oscillations and can lead to instabilities of the robot, and too low gains lead to poor tracking such that the robot frequently runs into the joint limits. This also keeps the exploration algorithm from generating negative gains.

For both via-point experiments, the cost weights are $W_{via-point} = 2000$, $W_{gain} = 1/N$, $W_{acc} = 1/N$. Dividing the weights by the number of time steps N is convenient, as it makes the weights independent of the duration of a movement.

5.1.1 Experiment 1: Phantom robot, passing through via-point in joint space

The Phantom Premium 1.5 Robot is a 3-DOF, two link arm. It has two rotational degrees of freedom at the base and one in the arm. We use a physically realistic simulation of this robot generated in SL [23], as depicted in Figure 2.

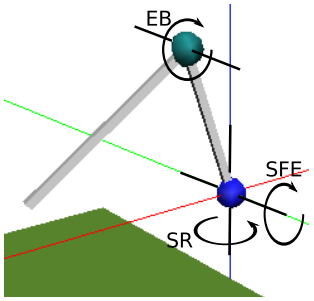

Figure 2: 3-DOF Phantom simulation in SL.

The task for this robot is intentionally simple and aimed at demonstrating the ability to tune task relevant gains in joint space with straightforward and easy to interpret data.

The duration of the movement is $2.0s$, which corresponds to 1000 time steps at 500Hz servo rate. The intermediate goals for this robot are set as follows:

$$\begin{aligned} C(t) = \quad & \delta(t-0.4) \cdot \mid q_{SR}(t) + 0.2 \mid + \\ & \delta(t-0.8) \cdot \mid q_{SFE}(t) - 0.4 \mid + \\ & \delta(t-1.2) \cdot \mid q_{EB}(t) - 1.5 \mid \end{aligned} \tag{24}$$

This penalizes joint SR for not having an angle $q_{SR} = -0.2$ at time $t = 0.4s$. Joints SFE and EB are also required to go through (different) intermediate angles at times $0.8s$ and $1.2s$ respectively.

The initial parameters $\boldsymbol{\theta}^i$ for the reference trajectory are determined by training the DMPs with a minimum jerk trajectory [42] in joint space from $\mathbf{q}_{t=0.0} = [0.0\ 0.3\ 2.0]^T$ to $\mathbf{q}_{t=2.0} = [-0.6\ 0.8\ 1.4]^T$. The function approximator for the proportional gains of the 3 joints is initialized to return a constant gain of $6.0Nm/rad$. The initial trajectories are depicted as thin black lines in Figure 4, where the angles and gains of the three joints are plotted against time. Since the task of PI^2 is to optimize both trajectories and gains with respect to the cost function, this leads to a 6-D RL problem. The robot executes 100 parameter updates, with 4 noisy roll-outs per update. After each update, we perform one noise-less test trial for evaluation purposes.

Figure 3 depicts the learning curve for the phantom robot, which is the overall cost of the noise-less test trial after each parameter update. The joint space trajectory and gain scheduling after 100 updates are depicted as thick solid lines in Figure 4.

5.1.2 Experiment 2: Kuka robot, passing through a via-point in task space

Next we show a similar task on a simulated 6-DOF Kuka Light-Weight Arm, depicted in the middle of Figure 7. This example illustrates that our approach scales well to higher-dimensional systems, and also that appropriate gains schedules are learned when intermediate targets are chosen in end-effector space instead of joint space.

The duration of the movement is $1.0s$, which

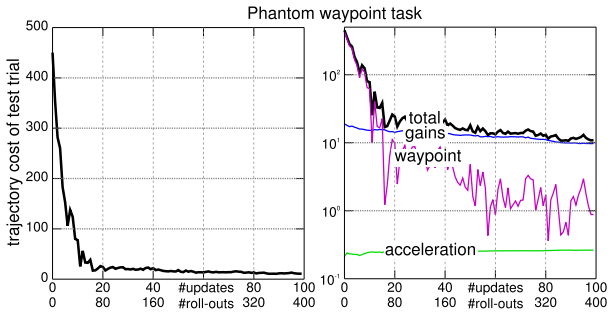

Figure 3: Learning curve for the phantom robot. Left: linear y-axis. Right: logarithmic y-axis, with the total cost broken down into the separate cost components of Equation 23.

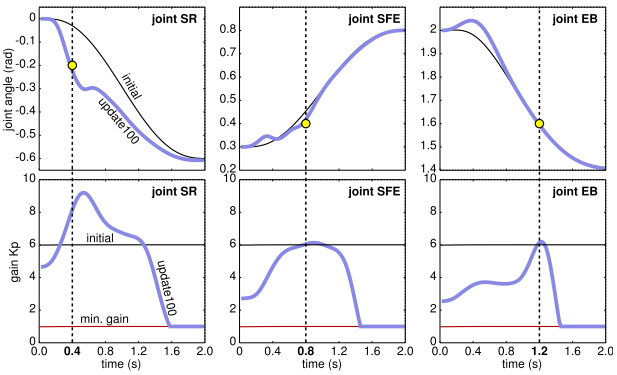

Figure 4: Initial (thin) and final (thick) joint trajectories and gain scheduling for each of the three joints of the phantom robot. Yellow circles indicate intermediate via-points in joint space at different times.

corresponds to 500 time steps. This time, the intermediate goal is for the end-effector $\mathbf{x}$ to pass through $[0.7\ 0.3\ 0.1]^T$ at time $t = 0.5s$:

$$C(t) = \quad \delta(t - 0.5)|\ \mathbf{x} - [\ 0.7\ 0.3\ 0.1]^T\ |\quad (25)$$

The six joint trajectories are again initialized as minimum jerk trajectories. As before, the resulting initial trajectory is plotted as a thin black line in Figure 6. The initial gains are set to a constant $[60, 60, 60, 60, 25, 6]^T$. Given these initial conditions, finding the parameter vectors for DMPs and gains that minimizes the cost function leads to a 12-D RL problem. We again perform 100 parameter updates, with 4 exploration roll-outs per update.

The learning curve for this problem is depicted in Figure 5. The trajectory of the end-effector before learning and after 30 and 100 updates is depicted in Figure 6. The intermediate goal at $t = 0.5$ is visualized by circles. Finally, Figure 7 shows the gain schedules before learning and after 30 and 100 updates for the 6 joints of the Kuka robot.

5.1.3 Discussion of via-point experiments

For both experiments PI^2 has adapted the initial minimum jerk trajectories such that they fulfill the task and pass through the desired joint angles or the end-effector task-space goal at the specified times with only small error (Figures 4, and 6). These intermediate goals are represented by the circles on the graphs. The remaining error is a result of the trade-off between the different factors of the cost function (i.e. penalty for distance to goal vs. penalty for high gains). It learns to do so after only 30 updates for the task space goal on the 6-DOF Kuka Arm (Figure 5) and less than 20 for the joint space goal in the 3-DOF Phantom robot (Figure 3).

Because the magnitude of gains is penalized in general, they are low when the task allows it, as illustrated in Figure 4: After $t = 1.6s$, all gains drop to their pre-specified minimum values, because accurate tracking is no longer required to fulfill the goal. Once the task is com-

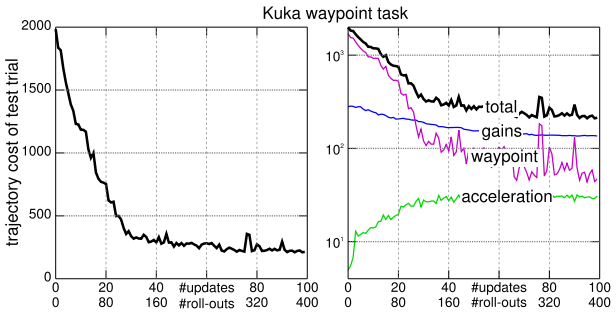

Figure 5: Learning curve for the Kuka robot. Left: linear y-axis. Right: logarithmic y-axis, with the total cost broken down into the separate cost components of Equation 23.

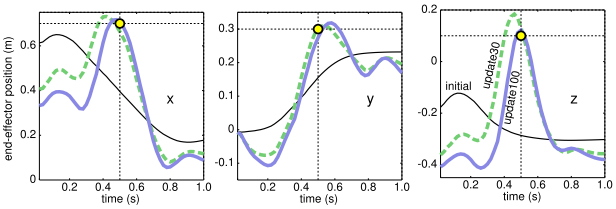

Figure 6: Initial (thin), intermediate (dashed), and final (thick) end-effector trajectories of the Kuka robot.

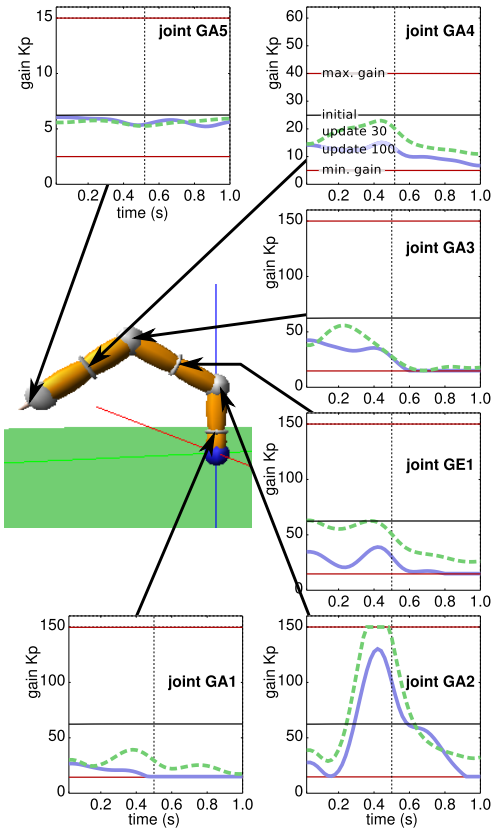

Figure 7: Initial (thin), intermediate (dashed), and final (thick) joint gain schedules for each of the six joints of the Kuka robot.

pleted, the robot becomes maximally compliant, as one would wish it to be.

The same effect is visible in the results for the Kuka Arm, where after 100 updates the peaks of most gains occur just before the end-effector passes through the intermediate goal (Figure 7), and in many cases decrease to the minimum gain directly afterwards. As with the phantom robot we observe high impedance when the task requires accuracy, and more compliance when the task is relatively unconstrained.

The second joint (GA2) has the most work to perform, as it must support the weight of all the more distal links. Its gains are by far the highest, especially at the intermediate goal, as any error in this DOF will lead to a large end-effector error.

When the robot is required to pass through the intermediate targets, it needs better tracking, and therefore higher gains. Therefore, the peaks of the gains correspond roughly to the times where the joint is required to pass through an intermediate point. Due to nonlinear effects, e.g., Coriolis and centripedal forces, the gain schedule shows more complex temporal behavior as one would initially assume from specifying

To appear in International Journal of Robotics Research – (c) SAGE Publications Inc.

three different joint space targets at three different times.

For the Kuka arm, the learning has two distinct phases. In the first phase (plotted as a dashed graph), the robot is learning to make the end-effector pass through the intermediate goal. At this point, the basic shape of the gain scheduling has been determined. In the second phase, PI^2 fine tunes the gains, and lowers them as much as the task permits.

In summary, these two experiments illustrate that we have achieved the objective of variable impedance control: the robot is compliant when possible, but has a higher impedance when the task demands it.

5.2 Manipulation experiments

The next two experiments show how the proposed method can find trajectories and gain schedules for more complex behaviors that involve contact with the environment.

5.2.1 Experiment 3: Phantom robot, flipping a light switch

The goal of this task is for the 3-DOF Phantom robot to flip a light switch. The experiment was conducted with both a real robot, depicted in Figure 8, and the simulated robot as in Experiment 1.

The initial trajectory was acquired through kinesthetic teaching. To demonstrate the trajectory, the gains of the robot were simply set to 0, as the robot is light enough to maneuver by hand without gravity compensation. The 3 joint angles were recorded over time, and a low-pass Butterworth filter is applied to these trajectories to suppress the large accelerations that arise due to the noisy recordings. The resulting trajectory in end-effector space is depicted in Figure 8. A DMP with 20 basis functions is trained for each degree of freedom.

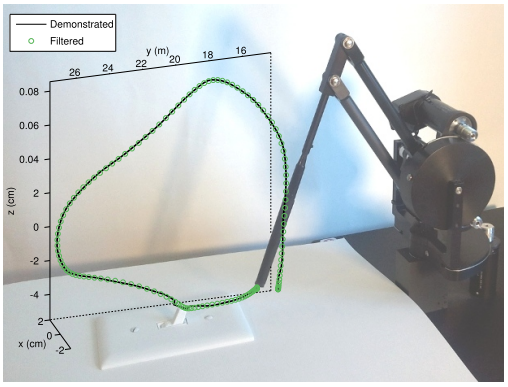

Figure 8: Experimental set-up for the Phantom experiment, including the demonstrated trajectory.

The initial gains for each joint are set to a constant value over time of 0.2, 0.5 and 0.3 for the SR, SFE and EB joint respectively. This corresponds to a quarter of the default gains for this robot. With these gains, the robot is very compliant, and hardly exerts forces, even when pushed far off the desired trajectory. These gains are also the minimum gains allowed during learning. Lower gains lead to such bad tracking of desired trajectories, that the robot frequently reaches its joint limits, which is undesirable.

The cost function for PI^2 consists of two parts. The terminal cost ϕ_{t_N} is 0 if the switch was flipped, or 500 if it was not. On the real robot, the user provides yes/no feedback (keyboard input) whether the switch was flipped or not. The intermediate costs for the gains are the same as in the via-point experiments, i.e. $r_t = \frac{1}{N}\sum_{i=1}^{3} K_{P,t}^i$, again dividing by the length of the trajectory N to be independent of trajectory duration.

The variance of the exploration noise for the gain schedule of each joint is $10^{-4}\gamma^n$, with decay parameter $\gamma = 0.98$ and n the number of updates. Before each update, 4 roll-outs are executed on the robot, and the 4 roll-outs with lowest cost from the previous parameter update are kept, so each update is computed over $K = 8$ trajectories. This elitarianist reuse of roll-outs makes sure that 'good examples' are not forgotten, as chance might have it that all newly generated roll-outs are worse than the previous mean, and reward weighted averaging would lead to

Results. Figure 9 depicts the cost of the noise-less test trial after each update for both the real and simulated robot. The gain schedules after 0/2/18 and 0/4/30 updates are depicted for the real and simulated robot respectively in Figure 10. These result will be discussed in Section 5.2.3.

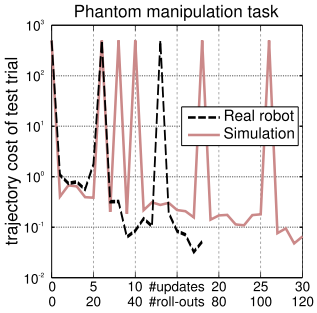

Figure 9: Learning curves of the real and simulated Phantom on the light switch task.

5.2.2 Experiment 4: CBi humanoid robot, pushing open a door

In this task, the simulated CBi humanoid robot [6] is required to open a door. This robot is accurately simulated with the SL software [23]. For this task, we not only learn the gain schedules, but also improve the planned joint trajectories with PI^2 simultaneously.

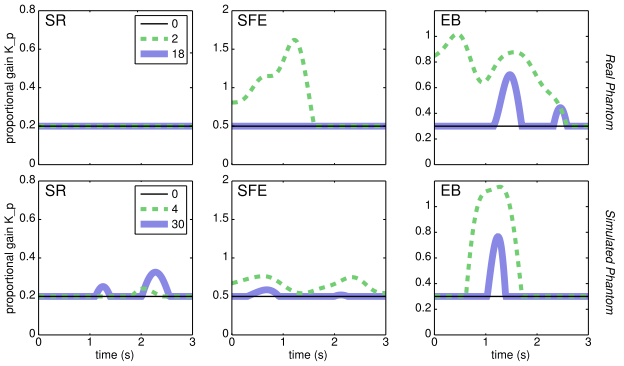

Figure 10: Learned gain schedules of the real (upper row) and simulated (lower row) of the three Phantom joints after 0/2/18 and 0/4/30 updates respectively

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 Figure 11.

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. The minimum gain for all joints was set to 5. 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 again $r_t = \frac{1}{N}\sum_{i=1}^{3} K_P^i$.

To appear in International Journal of Robotics Research – (c) SAGE Publications Inc.

The variance of the exploration noise for the gains is again $10^{-4}\gamma^n$, and for the joint trajectories $10\gamma^n$, both with decay parameter $\lambda = 0.99$ and n the number of updates[4]. 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 $K = 10$.

Results. Figure 11 (right) depicts the total cost of the noise-less test trial after each update. The costs for the gains are plotted separately. When all of the costs are due to gains, i.e. the door is opened completely to ψ_{max} and the task is achieved, the graphs of the total cost and that of the gains coincide. The joint trajectories and gain schedules after 0, 6 and 100 updates are depicted in Figure 12.

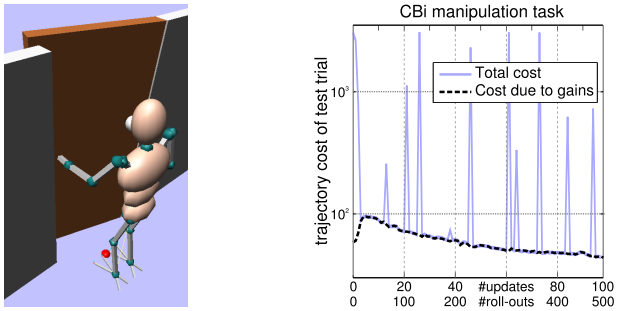

Figure 11: Left: Task scenario. Right: Learning curve for the door task. The costs specific to the gains are plotted separately.

[4]The relatively high exploration noise for the joint trajectories 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 21) rather than as the non-linear component of a DMP (Equation 14).

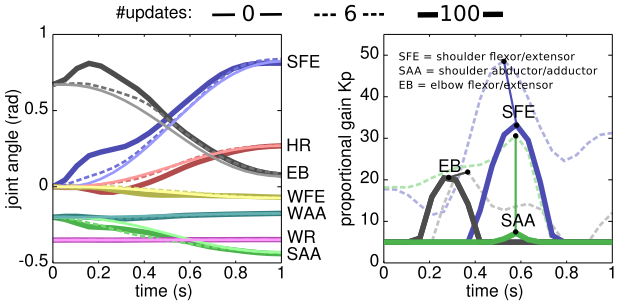

Figure 12: Learned joint angle trajectories (left) and gain schedules (right) of the CBi arm after 0/6/100 updates. The gain schedules of only three joints have been depicted for sake of clarity (EB: elbow, SAA: shoulder adduction-abduction, SFE: shoulder flexion-extension)

5.2.3 Discussion of manipulation experiments

We now discuss the results of the last two experiments: flipping the light switch, and opening the door.

In all the manipulation results above, there are two distinct phases during learning. In the first few updates, the gains are increased and a suitable trajectory is found in order to achieve the task, i.e. flip the light switch, or open the door. This leads to a strong decrease in the cost for not achieving the task, which is traded off against a higher cost for higher gains. This is clearly seen in Figure 11, where the cost due to the gains increases dramatically in the first few updates (note the logarithmic scale), whereas the overall cost decreases. 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 a few updates (2/4/6) in Figure 10 and Figure 12: the gains are much higher than their low values with which

they are initialized.

In the second phase, gains are lowered overall to reduce the immediate costs r_t, and the exact timing and magnitudes of the gains required to achieve the task are determined. On the Phantom robot, this leads to a peak in the gains of the elbow joint (EB) when the robot's end-effector comes into contact with the switch. This joint needs to stiffen up in order to exert the force necessary to flip the switch. On the CBi robot, there is a peak in the elbow joint before contact, as the elbow must be lifted to reach the door. During door opening, the gains of the shoulder flexor-extensor joint (SFE) increase, again to exert the force necessary to open the door. Too much compliance during this time will not allow the robot to achieve its task. It is interesting to see that after 100 updates, the sum of the gains (i.e. the 'cost due to gains' in Figure 11) for the CBi robot is actually 25% lower than at initialization, when it could not open the door. But by timing and tuning the gains appropriately as depicted in Figure 12, the robot is now able to open the door.

Note that during this second phase, the robot sometimes lowers the gains too much, and is not able to flip the switch/open the door anymore, as indicated by the spikes in the learning curves. That the robot is always able to open the door/flip the switch one update after a spike is because of the elitarianism, which always leads to at least some roll-outs with successful task achievement to be among the pool of K roll-outs on which the parameter update is performed.

In Figure 9, it is surprising to see that learning is faster on the real robot that in simulation. We believe this is due to the strong discontinuity in the cost function, caused by the binary nature of achieving the task or not. Due to the reproducibility of movement and interactions in simulation, this will indeed be a very sharp discontinuity. On the real robot this discontinuity is 'smoothed' by imperfect tracking, inaccurate sensors, and slight displacements of the light switch between trials. We assume that this leads to a smoother cost function, which facilitates learning.

In summary, learning such variable gain schedules 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).

6 Related work

Biological Motor control It has been shown that humans are able to control the impedance characteristics of their hand in task space [4, 26]. The mathematical treatment of the human motor system is usually developed along the same lines as in robotics, i.e. the basic kinematics and dynamics equations are borrowed from rigid body dynamics. However, actuation is done by antagonistic muscle systems to generate joint torques, which adds another space, the muscle actuation space [10]. For the discussion as it applies to the presented work this added space and complexity is of minor importance. The central characteristics is (a) relevance of impedance in task space (b) realization of impedance control in another space i.e. in robots in joint space, in humans in 'muscle space'. We currently present results only on tuning stiffness in joint space, and not in 'muscle' space (e.g. no biarticular or antagonistic actuation, no nonlinearities). However, our method generalizes to more complex parameterized impedance control

laws, and could also parameterize muscle based models. Therefore, this work opens a large field of comparative studies in biological motor control with PI^2 . It is also worth noting that in the biological motor control literature, very often only variation of stiffness is discussed, last but not least also due to experimental difficulties of measuring a more general mechanical impedance in human subjects.

Impedance control One of the motivations behind our work is the same motivation that is behind impedance control as presented in [8]: "[. . .] the controller should be capable of modulating the impedance of the manipulator as appropriate for a particular phase of a task." While, for long time, the robotics control community has realized how important it is to control the interaction dynamics of a robot properly, to this day "[T]he selection of good impedance parameters [. . .] is not an easy task" [27]. Deriving useful impedance controllers usually involves models and knowledge of both the environment and the robot and deep knowledge about designing and parameterizing such controllers. Here we show how appropriate interaction behavior can be learned purely from experience and *without the need of knowledge of the robot and the environment* and only simple reward on the quality of task achievement in form of a cost function (unspecific to the controller).

In [9] a variety of impedance control schemes are discussed. One of these schemes is 'feedback' impedance control where a desired impedance is chosen and tracked via state-feedback and force-feedback. Today, this scheme is commonly referred to as impedance control (e.g. in [27]), even though the idea of impedance control is more general. To illustrate this fact, other schemes that do not use feedback to online control the impedance are also presented in [9] along with the feedback impedance control scheme.

The approach presented in this paper is not to be confused with 'feedback' impedance control. Our controllers generate *variable impedance* at the end-effector by manipulating the joint space impedance parameters and the reference trajectory. As joint space impedance parameters we use variable gains of a PD controller, which does not mean that the most general form of impedance can be realized, but, in principle, other more general parameterizations could be used to find more general impedance realizations. The dimensionality of the learning problem will increase. Studying the gain/benefit tradeoff of these more general schemes remains future work.

Note that despite restricting the algorithm to tuning stiffness and damping in joint space, it can still tune the apparent inertia properties and other aspects of the dynamic behavior at the end-effector (in certain limits) via the reference trajectory. The reference trajectory influences the dynamic behavior at the end-effector in two ways. First by determining the configuration at a given point in the task, the apparent inertia at the end-effector is co-determined [15, 9, 25], and second by possibly purposefully using a reference trajectory which is far from the actually followed trajectory (e.g. as in indirect force control [28] where the desired force is achieved by putting the end-effector reference trajectory within the object).

Optimal control In optimal control and model based RL, Differential Dynamic Programming (DDP) [12] has been one of the most established and most commonly used frameworks

for finite horizon optimal control problems. In DDP, both state space dynamics and cost function are approximated up to the second order. The assumption of stabilizability and detectability for the local approximation of the dynamics are necessary for the convergence of DDP. The resulting state space trajectory is locally optimal, while the corresponding control policy consists of open loop feedforward commands and closed loop gains relative to a nominal and optimal reference trajectory. This characteristic allows the use of DDP for both planning and gain scheduling problems. In [7, 40] DDP was extended to incorporate constraints in state and controls. In [18] the authors suggest computational improvements to constrained DDP and apply the proposed algorithm to a low dimensional planning problem.

An example of a DDP application to robotics is presented in [20]. In this work, a min-max or Differential Game Theory approach to optimal control is proposed. There is a strong link between robust control frequency design analysis such as H_∞ control and the framework of Differential Game Theory [1]. Essentially the min-max DDP results in robust feedback control policies with respect to model uncertainty and unknown dynamics. Although, in theory, min-max DDP should resolve the issue of model uncertainty, it can lead to overly conservative control policies. The conservatism results from the need to guarantee that the game theoretic approach will be always stabilizable, i.e. making sure that the stabilizing controller wins. For linear and time invariant systems, such guarantee is feasible through γ-iteration [39]. However, for nonlinear systems providing this guarantee is not trivial.

The work on Receding Horizon DDP [32] provided an alternative and rather efficient way of computing local optimal feedback controls. Nevertheless, all the computations of optimal trajectories and control take place off-line and the model predictive component is only due to the fact that the final target state of the optimal control problem varies. Recent work on LQR-trees uses a simpler variation of DDP, the iterative Linear Quadratic Regulator (iLQR) [19], which is based on linear approximations of the state space dynamics, in combination with tools from Nonlinear Robust Control theory for region of attraction analysis. Given the local optimal feedback control policies, the sums of squares optimization scheme is used to quantify the size of of the basin of attraction, and provides so-called control funnels. These funnels improve sampling since they quantize the state space into attractor regions placed along the trajectories towards the target state. This is a model based approach and thus suffers from many of problems of model based approaches to optimal control. In addition, even though sampling is improved, it is still an issue how LQR trees scale in high dimensional dynamical systems.

The path integral formalism for optimal control was introduced in [13, 14]. In this work, the role of noise in symmetry breaking phenomena was investigated in the context of stochastic optimal control. In [38], the path integral formalism is extended for stochastic optimal control of multi-agent systems, which is not unlike our multi DOF control systems.

Recent work on stochastic optimal control by [36, 35, 37] shows that for a class of discrete stochastic optimal control problems, the Bellman equation can be written as the Kullback-Leibler (KL) divergence between the probability distribution of the controlled and uncontrolled dynamics. Furthermore, it is shown that the class of discrete KL divergence control problem is equivalent to the continuous stochastic opti-

mal control formalism with quadratic cost control function and under the presence of Gaussian noise. In all this aforementioned work, both in the path integral formalism as well as in KL divergence control, the class of stochastic dynamical systems under consideration is rather restrictive since the control transition matrix is state independent. Moreover, the connection to direct policy learning in RL and model-free learning was not made in any of the previous projects. In [37], the stochastic optimal control problem is investigated for discrete state-action spaces, and therefore it is treated as Markov Decision Process (MDP).

As was demonstrated, to apply our PI^2 algorithm, we do not discretize the state space and we do not treat the problem as an MDP. Instead we work in continuous state-action spaces which are suitable for performing RL in high dimensional robotic systems. To the best of our knowledge, our results present RL in one of the most high dimensional continuous state action spaces.

In our derivations, the probabilistic interpretation of control comes directly from the Feynman-Kac Lemma. Thus we do not have to impose any artificial pseudo-probability treatment of the cost as in [37]. In addition, for continuous state-action spaces, we do not have to learn the value function as it is suggested in [37] via Z-learning. Instead we directly obtain the controls based on our generalization of optimal controls. In the previous work, the problem of how to sample trajectories is not addressed. Sampling is performed with the hope to cover all the relevant state space. We follow a rather different approach by incremental updating, which allows us to address robotic learning problems of the complexity and dimensionality of complete humanoid robots.

Reinforcement Learning In contrast to policy gradient methods [22], in PI^2 there is no need to explicitly calculate a gradient, which is usually sensitive to noise and large derivatives in the value function. Essentially the gradient of the exponentiated value function $\nabla_{\mathbf{x}}\Psi$ is implicitly calculated by a weighted average of the exploration parameter ϵ weighted by the exponentiated cost of every sampled trajectory (last step in 1). This computation is robust to non-smooth dynamics and cost functions. As is shown in the results session, PI^2 performs RL under boolean cost functions which are introduced either to incorporate contact with objects or to encode success and failure modes in motor tasks. Thus PI^2 is robust to non-smooth cost functions since it does rely on quadratic approximations of them as model based RL methods [12].

With respect to previous work on path integral control [14, 13], in PI^2 the exploration of the state space is done with the propagation of DMPs, rather than sampling the whole state space. For high dimensional problems, it is simply not possible to sample the whole state space.

The differences discussed above enable PI^2 to outperform previous RL algorithms for parameterized policy learning by at least one order of magnitude in learning speed and also lower final cost performance, as demonstrated in [33, 34]. It also scales up to high-dimensional spaces, which enables PI^2 to learn full-body humanoid motor skills with over 30-DOFs [30]. 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. [33]). It is a model free reinforcement learning algorithm in the sense that it does not require knowledge of the model of the control system or the environment for the learning of an optimal control policy.

In [17], a robot learns the coupling between the different DOFs of a robot, to learn synergies across the different dimensions. These couplings are learned with the PoWER algorithm [16], on a representation very similar to DMPs. The initial DMP is acquired through kinesthetic teaching. The output of the motion primitive, desired accelerations, is converted into torque commands using inverse dynamics and PD control. Rather than learning couplings between joints, we learn variable gain schedules for each joint. Also, PI^2 enables us to use much simpler cost functions for specifying whether the task is achieved. In the case of the light switch flipping task, it is a simple boolean function that specifies whether the flip was switched or not. This makes it easier for non-expert users to provide feedback.

7 Conclusion

We presented a model-free reinforcement learning approach that can learn variable impedance control for robotic systems. Our approach is derived from stochastic optimal control with path integrals, a relatively new development that transforms optimal control problems into estimation problems. In particular, PI^2 goes beyond the original ideas of optimal control with path integrals by realizing the applicability to optimal control with parameterized policies and model-free scenarios.

The mathematical structure of the PI^2 algorithm makes it suitable to optimize simultaneously both reference trajectories and gain schedules. This is similar to classical differential dynamic programming (DDP) methods, but completely removes the requirements of DDP that the model of the controlled system must be known, that the cost function has to be twice differentiable in both state and command cost, and that the dynamics of the control system have to be twice differentiable. The latter constraints make it hard to apply DDP to tasks with discrete events, as is typical in force control and locomotion.

We evaluated our approach on three simulated robot systems and one real robot, which posed up to 14 dimensional learning problems in continuous state-action spaces. The goal was to learn compliant control while fulfilling kinematic task constraints, like passing through an intermediate target. The evaluations demonstrated that the algorithm behaves as expected: it increases gains when needed, but tries to maintain low gain control otherwise. The optimal reference trajectory always fulfilled the task goal. Learning speed was rather fast, i.e., within at most a few hundred roll-outs, the task objective was accomplished. From a machine learning point of view, this performance of a reinforcement learning algorithm is very fast.

The PI^2 algorithms inherits the properties of all trajectory-based learning algorithms in that it only finds locally optimal solutions. For high dimensional robotic system, this is unfortunately all one can hope for, as exploring the entire state-action space in search for a globally optimal solution is impossible.

Future work aims at applying these methods to actual robots for mobile manipulation and locomotion controllers. We believe that our methods are a major step towards realizing compliant autonomous robots that operate robustly in dynamic, stochastic environments, without harming other beings or themselves.

8 Acknowledgments

This research was supported in part by National Science Foundation grants ECS-0326095, IIS-0535282, IIS-1017134, CNS-0619937, IIS-0917318, CBET-0922784, EECS-0926052, CNS-0960061, the DARPA programs on Learning Locomotion and Advanced Robotic Manipulation, the Multidisciplinary Research Program of the Department of Defense (MURI N00014-00-1-0637), the Army Research Office, the Okawa Foundation, and the ATR Computational Neuroscience Laboratories. J.B. was supported by prospective and advanced researcher fellowships from the Swiss National Science Foundation. F.S. was supported by a Research Fellowship from the German Research Foundation (DFG). E.T. was supported by a Myronis Fellowship.

References

[1] T. Basar and P. Bernhard. *H-infinity Optimal Control and Related Minimax Design Problems: A Dynamic Game Approach*. Birkhäuser, Boston, 1995.

[2] J. Buchli, M. Kalakrishnan, M. Mistry, P. Pastor, and S. Schaal. Compliant quadruped locomotion over rough terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 814–820, 2009.

[3] J. Buchli, E. Theodorou, F. Stulp, and S. Schaal. Variable impdeance control – a reinforcement learning approach. In *Robotics: Science & Systems*, 2010.

[4] E. Burdet, K.P. Tee, I. Mareels, T.E. Milner, C.M. Chew, D.W. Franklin, R. Osu, and M. Kawato. Stability and motor adaptation in human arm movements. *Biological Cybernetics*, 94(1):20–32, 2006.

[5] R.E. Caflisch. Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica*, 7:1–49, 1998.

[6] 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.

[7] R. Fletcher. *Practical Methods of Optimization*, volume 2. Wiley, 1981.

[8] N. Hogan. Impedance control: An approach to manipulation: Part I – Theory. *ASME, Transactions, Journal of Dynamic Systems, Measurement, and Control*, 107:1–7, 1985.

[9] N. Hogan. Impedance control: An approach to manipulation: Part II – Implementation. *ASME, Transactions, Journal of Dynamic Systems, Measurement, and Control*, 107:8–16, 1985.

[10] N. Hogan. *Advances in Robot Control*, chapter Force Control with A Muscle-Activated Endoskeleton. Springer Verlag, 2006.

[11] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proc. of the IEEE International Conf. on Robotics and Automation (ICRA)*, volume 2, page 1398, 2002.

[12] D. H. Jacobson and D. Q. Mayne. *Differential Dynamic Programming*. Optimal Control. Elsevier Publishing Company, New York, 1970.

[13] H. J. Kappen. Linear theory for control of nonlinear stochastic systems. *Phys Rev Lett*, 95(20):200201, 2005.

[14] H.J. Kappen. Path integrals and symmetry breaking for optimal control theory. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(11):P11011, 2005.

[15] O. Khatib. Inertial properties in robotic manipulation: An object-level framework. *Int J Robotics Research*, 14(1):19–36, 1995.

[16] J. Kober and J. Peters. Learning motor primitives in robotics. In D. Schuurmans, J. Benigio, and D. Koller, editors, *Advances in Neural Information Processing Systems 21 (NIPS 2008)*, Vancouver, BC, Dec. 8-11, 2009. Cambridge, MA: MIT Press. clmc.

[17] 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)*, pages 3232 – 3237, 2010.

[18] G. Lantoine and R.P. Russell. A hybrid differential dynamic programming algorithm for robust low-thrust optimization. In *AAS/AIAA Astrodynamics Specialist Conference and Exhibit*, 2008.

[19] W. Li and E. Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 1, pages 222–229, 2004.

[20] J. Morimoto and C. Atkeson. Minimax differential dynamic programming: An application to robust biped walking. In *In Advances in Neural Information Processing Systems 15*, pages 1563–1570. MIT Press, Cambridge, MA, 2002.

[21] B. K. Øksendal. *Stochastic differential equations: an introduction with applications*. Springer, Berlin, New York, 6th edition, 2003.

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

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

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

[25] L. Sciavicco and B. Siciliano. *Modelling and Control of Robot Manipulators*. Springer, 2000.

[26] L.P.J. Selen, D.W. Franklin, and D.M. Wolpert. Impedance control reduces instability that arises from motor noise. *J Neuroscience*, 29(40):12606–12616, 2009.

[27] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics – Modelling, Planning and Control*. Springer, London, 2009.

[28] B. Siciliano and L. Villani. *Robot Force Control*. Springer, 2000.

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

[30] F. Stulp, J. Buchli, E. Theodorou, and S. Schaal. Reinforcement learning of full-body humanoid motor skills. In *10th IEEE-RAS International Conference on Humanoid Robots*, pages 405–410, 2010.

[31] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. Adaptive computation and machine learning. MIT Press, Cambridge, 1998.

[32] Y. Tassa, T. Erez, and W. Smart. Receding horizon differential dynamic programming. In *Advances in Neural Information Processing Systems 20*, pages 1465–1472. MIT Press, Cambridge, MA, 2008.

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

[34] E. Theodorou, J. Buchli, and S. Schaal. Reinforcement learning of motor skills in high dimensions: A path integral approach. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2397 – 2403, 2010.

[35] E. Todorov. Linearly-solvable markov decision problems. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19 (NIPS 2007)*, Vancouver, BC, 2007. Cambridge, MA: MIT Press.

[36] E. Todorov. General duality between optimal control and estimation. In *Proceedings of the 47th IEEE Conf. on Decision and Control*, 2008.

[37] E. Todorov. Efficient computation of optimal actions. *Proc Natl Acad Sci USA*, 106(28):11478–83, 2009.

[38] 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.

[39] T. Vincent and W. Grantham. *Non Linear And Optimal Control Systems*. John Wiley & Sons Inc., 1997.

[40] S. Yakowitz. The stagewise Kuhn-Tucker condition and differential dynamic programming. *IEEE Transactions on Automatic Control*, 31(1):25–30, 1986.

[41] J. Yong. Relations among ODEs, PDEs, FSDEs, BSDEs, and FBSDEs. In *Proceedings of the 36th IEEE Conference on Decision and Control, 1997*, volume 3, pages 2779–2784, Dec 1997.

[42] M. Zefran, V. Kumar, and C.B. Croke. On the generation of smooth three-dimensional rigid body motions. *IEEE Transactions on Robotics and Automation*, 14(4):576–589, 1998.