Learning to Control in Operational Space

Jan Peters
Max Planck Institute for Biological Cybernetics,
Spemannstrasse 38, 72076 Tübingen, Germany
and
University of Southern California,
3641 Watt Way, Los Angeles, CA 90089, USA
mail@jan-peters.net

Stefan Schaal
University of Southern California,
3641 Watt Way, Los Angeles, CA 90089, USA
and
ATR Computational Neuroscience Laboratory,
2-2-2 Hikaridai, Seika-cho, Soraku-gun,
Kyoto 619-0288, Japan

Abstract

One of the most general frameworks for phrasing control problems for complex, redundant robots is operational-space control. However, while this framework is of essential importance for robotics and well understood from an analytical point of view, it can be prohibitively hard to achieve accurate control in the face of modeling errors, which are inevitable in complex robots (e.g. humanoid robots). In this paper, we suggest a learning approach for operational-space control as a direct inverse model learning problem. A first important insight for this paper is that a physically correct solution to the inverse problem with redundant degrees of freedom does exist when learning of the inverse map is performed in a suitable piecewise linear way. The second crucial component of our work is based on the insight that many operational-space controllers can be understood in terms of a constrained optimal control problem. The cost function associated with this optimal control problem allows us to formulate a learning algorithm that automatically synthesizes a globally consistent desired resolution of redundancy while learning the operational-space controller. From the machine learning point of view, this learning problem corresponds to a reinforcement learning problem that maximizes an immediate reward. We employ an expectation-maximization policy search algorithm in order to solve this problem. Evaluations on a three degrees-of-freedom robot arm are used to illustrate the suggested approach. The application to a physically realistic simulator of the anthropomorphic SARCOS Master arm demonstrates feasibility for complex high degree-of-freedom robots. We also show that the proposed method works in the setting of learning resolved motion rate control on a real, physical Mitsubishi PA-10 medical robotics arm.

KEY WORDS—operational space control, robot learning, reinforcement learning, reward-weighted regression

1. Introduction

Operational-space control is one of the most elegant approaches to task control owing to its potential for dynamically consistent control, compliant control, force control, hierarchical control, and many other favorable properties, with applications from end-effector control of manipulators (Khatib 1987; Hsu et al. 1989) up to balancing and gait execution for humanoid robots (Sentis and Khatib 2005). If the robot model is accurately known, operational-space control is well-understood yielding a variety of different solution alternatives, including resolved-motion rate control, resolved-acceleration control, and force-based control (Nakanishi et al. 2005). However, particularly if compliant, low-gain control is desirable as in many new robotic systems that are supposed to operate safely in human environments, operational-space control becomes increasingly difficult in the presence of unmodeled nonlinearities, leading to reduced accuracy or even unpredictable and unstable null-space behavior in the robot system. As a potential solution to this problem, learning control methods seem to be promising. However, learning methods do not easily provide the highly structured knowledge required in traditional

The International Journal of Robotics Research
Vol. 27, No. 2, February 2008, pp. 197–212
DOI: 10.1177/0278364907087548

Figures 1, 2, 4–8 appear in color online: http://ijr.sagepub.com

operational-space control laws, i.e. Jacobians, inertia matrices, and Coriolis/centripetal and gravity forces, as all of these terms are not always instantly observable and are therefore not suitable for formulating supervised learning as traditionally used in learning control approaches (Nakanishi et al. 2004).

In this paper, we suggest a novel approach to learning operational-space control that avoids extracting such structured knowledge and rather aims at learning the operational-space control law directly. To develop our approach, we proceed as follows: first, we review operational-space control and discuss where learning can be beneficial. Second, we pose operational-space control as a learning problem and discuss why standard learning techniques cannot be applied straightforwardly. Using the alternative understanding of operational-space control as an optimal control technique, we reformulate it as an immediate reward reinforcement learning or policy search problem and suggest novel algorithms for learning some of the most standard types of operational-space control laws. These new techniques are evaluated on a simulated three degree-of-freedom (DoF) robot arm and a simulated anthropomorphic 7 DoF SARCOS robot arm.

1.1. Notation and Remarks

Throughout this paper, we assume the standard rigid body model for the description of the robot, i.e.

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \varepsilon(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{u}, \tag{1}$$

where $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}} \in \mathbb{R}^n$ denote the joint coordinates, velocities and accelerations of the robot, respectively. The torques generated by the motors of the robot, also referred to as motor commands, are given by $\mathbf{u} \in \mathbb{R}^n$. Furthermore, $\mathbf{M}(\mathbf{q})$ denotes the inertia tensor or mass matrix, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ the Coriolis and centripetal forces, $\mathbf{G}(\mathbf{q})$ is gravity, and $\varepsilon(\mathbf{q},\dot{\mathbf{q}})$ denotes unmodeled non-linearities. The unmodeled non-linearities are often time-variant and can depend on non-stationary influences such as temperature. In this paper, we only address fully actuated robotic systems: extensions to underactuated or overactuated systems are addressed in Section 5.

In operational-space control, we intend to execute trajectories or forces[1] given in the coordinate system of the actual task. A well-studied example is a robot arm where position and orientation of the end-effector are controlled Khatib 1987; Hsu et al. 1989; however, a variety of further applications exist, such as the control of the center of gravity for balancing legged robots, which can also be thought of as operational-space control (Sentis and Khatib 2005). Position and orientation $\mathbf{x} \in \mathbb{R}^m$ of the controlled element of the robot in task space, e.g. the end-effector, is given by the forward kinematics $\mathbf{x} = \mathbf{f}_{\text{Kinematics}}(\mathbf{q})$. The derivatives yield both velocity and acceleration in task space, i.e.

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}, \quad \ddot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}}, \tag{2}$$

where $\mathbf{J}(\mathbf{q}) = \partial\mathbf{f}_{\text{Kinematics}}(\mathbf{q})/\partial\mathbf{q}$ denotes the Jacobian. We assume that the robot is in general redundant, i.e. it has more DoF than required for the task or, equivalently, $n > m$.

1.2. Operational-space Control as an Optimal Control Problem

Using the framework of trajectory tracking as an example, the general problem in operational-space control can be described as follows: generate a control law $\mathbf{u} = \mathbf{f}_{\text{Control}}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)$ that controls the robot along a joint-space trajectory $\mathbf{q}(t)$, $\dot{\mathbf{q}}(t)$, and $\ddot{\mathbf{q}}(t)$, such that the controlled element (e.g. the end-effector) follows a desired trajectory in task space $\mathbf{x}_d(t)$, $\dot{\mathbf{x}}_d(t)$, and $\ddot{\mathbf{x}}_d(t)$. This problem has been thoroughly discussed since the late 1980s (see, e.g., Khatib (1987) and Hsu et al. (1989)) and, among other results, has resulted in a class of well-known control laws (Nakanishi etal. 2005). As an important insight into operational-space control it was discovered Doty et al. 1993; Israeland Greville 2003; Peters et al. 2005 that many of the suggested controllers in the literature can be derived as the solution of a constrained optimization problem given by

$$\min_{\mathbf{u}} C(\mathbf{u}) = \mathbf{u}^{\mathrm{T}}\mathbf{N}\mathbf{u} \quad \text{subject to } \mathbf{J}\ddot{\mathbf{q}} = \ddot{\mathbf{x}}_{\text{ref}} - \dot{\mathbf{J}}\dot{\mathbf{q}}, \tag{3}$$

where $\mathbf{N}$ denotes a positive definite metric that weights the contribution of the motor commands to the cost function, and $\ddot{\mathbf{x}}_{\text{ref}} = \ddot{\mathbf{x}}_d(t) + \mathbf{K}_d(\dot{\mathbf{x}}_d(t) - \dot{\mathbf{x}}(t)) + \mathbf{K}_p(\mathbf{x}_d(t) - \mathbf{x}(t))$ denotes a reference attractor in task space with gain matrices $\mathbf{K}_d$ and $\mathbf{K}_p$. The resulting control laws or solution of this optimization problem obey the general form (Peters et al. 2005)

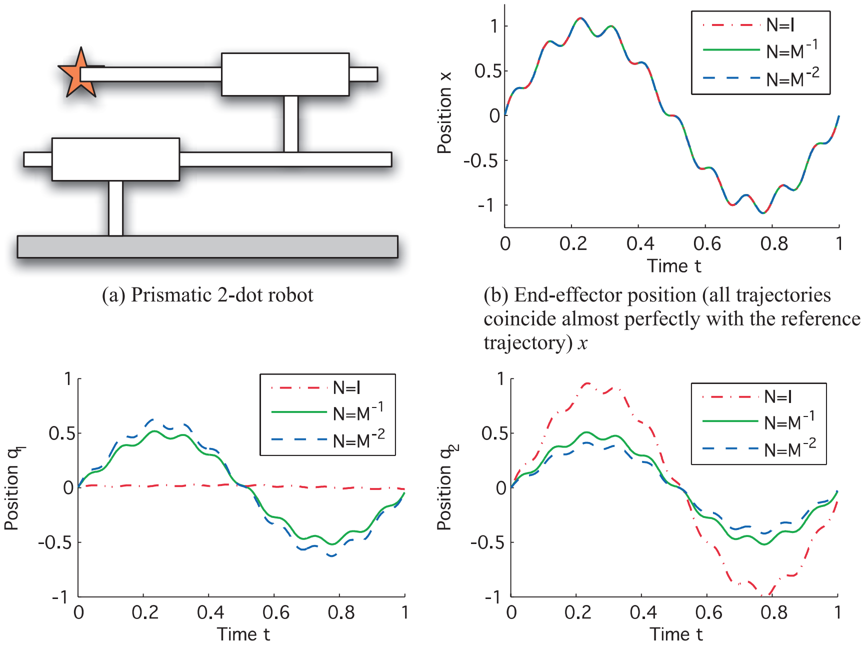
$$\mathbf{u} = \mathbf{N}^{-1/2}\left(\mathbf{J}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+}\left(\ddot{\mathbf{x}}_{\text{ref}} - \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}\mathbf{M}^{-1}\mathbf{F}\right) \tag{4}$$

with $\mathbf{F}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \varepsilon(\mathbf{q},\dot{\mathbf{q}})$, and the notation $\mathbf{D}^{+}$ defining the pseudo-inverse of a matrix such that either $\mathbf{D}^{+}\mathbf{D} = \mathbf{I}$ or $\mathbf{D}\mathbf{D}^{+} = \mathbf{I}$ (see Doty et al. (1993) and Israel and Greville (2003)), and with the matrix root $\mathbf{D}^{1/2}$ defined as $\mathbf{D}^{1/2}\mathbf{D}^{1/2} = \mathbf{D}$.

For example, the resolved-acceleration controller of Hsu et al. (1989) (without null-space optimization) is the result of using the metric $\mathbf{N} = \mathbf{M}^{-2}$, which yields $\mathbf{u} = \mathbf{M}\mathbf{J}^{\mathrm{T}}(\ddot{\mathbf{x}}_{\text{ref}} - \dot{\mathbf{J}}\dot{\mathbf{q}}) + \mathbf{F}$, and corresponds to a cascade of an inverse dynamics and an inverse kinematics control law. Another example is Khatib's formulation of operational-space control (Khatib 1987), determined by the metric $\mathbf{N} = \mathbf{M}^{-1}$ and given by

$$\mathbf{u} = \mathbf{J}^{\mathrm{T}}\left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{\mathrm{T}}\right)^{-1}\left(\ddot{\mathbf{x}}_{\text{ref}} - \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}\mathbf{M}^{-1}\mathbf{F}\right). \tag{5}$$

[1] In the more general case, the hybrid creation of forces in task space while following a desired trajectory needs to be included. For simplicity, we omit such tasks in this paper.



(a) Prismatic 2-dot robot

(b) End-effector position (all trajectories coincide almost perfectly with the reference trajectory) x

(c) Joint position q_1

(d) Joint position q_2

Fig. 1. (a) Prismatic 2 DoF robot. (b) The three control laws for the metrics $\mathbf{N} = \mathbf{I}$ (dashed-dot curves), $\mathbf{N} = \mathbf{M}^{-1}$ (solid curves), $\mathbf{N} = \mathbf{M}^{-2}$ (dashed curves) when applied to the prismatic robot from Example 1 shown in (a). This results in the same, perfect task-space tracking of a reference trajectory (consisting out of two superimposed sinusoids) but very different joint-space behavior: (c) joint position q_1; (d) joint position q_2. See Example 1 for more information.

Khatib's solution is special since the metric $\mathbf{N} = \mathbf{M}^{-1}$ is the only metric that generates torques which correspond to those created by a physical constraint pulling the robot along the trajectory (Udwadia and Kalaba 1996; Peters et al. 2005), i.e. it is the metric used by nature according to Gauss' principle (Udwadia and Kalaba 1996; Bruyninckx and Khatib 2000) and it is invariant under a change of joint coordinates (Doty et al. 1993). Other metrics such as $\mathbf{N} =$ constand can be used to distribute the required forces differently, e.g., such that stronger motors get a higher portion of the generated forces (Peters et al. 2005).

Even when completing the task perfectly, the joint-space trajectories can result in unfavorable postures or even joint-space instability (see Example 1). To handle such cases, additional controls that do not affect the task's performance but ensure a favorable joint-space behavior need to be included. From the point of view of the optimization framework, we would select a nominal control law $\mathbf{u}_0$ (e.g. a force pulling the robot towards a rest posture $\mathbf{u}_0 = -\mathbf{K}_D\dot{\mathbf{q}} - \mathbf{K}_D(\mathbf{q} - \mathbf{q}_{\text{rest}})$), and then solve the constrained optimization problem

$$\begin{aligned} \min_{\mathbf{u}} C_1(\mathbf{u}) &= (\mathbf{u} - \mathbf{u}_0)^{\mathrm{T}} \mathbf{N} (\mathbf{u} - \mathbf{u}_0) \\ \text{subject to } \mathbf{J}\ddot{\mathbf{q}} &= \ddot{\mathbf{x}}_{\text{ref}} - \dot{\mathbf{J}}\dot{\mathbf{q}}, \end{aligned} \tag{6}$$

where $\mathbf{u}_1 = \mathbf{u} - \mathbf{u}_0$ is the task-space control component. The general solution is given by

$$\begin{aligned} \mathbf{u} &= \mathbf{N}^{-1/2} \left(\mathbf{J}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+} \left(\ddot{\mathbf{x}}_{\text{ref}} - \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}\mathbf{M}^{-1}\mathbf{F}\right) \\ &\quad + \mathbf{N}^{-1/2}\left(\mathbf{I} - \left(\mathbf{J}\mathbf{M}^{-1}\mathbf{N}^{-1/2}\right)^{+}\left(\mathbf{N}^{-1/2}\mathbf{M}^{-1}\mathbf{J}\right)\right)\mathbf{N}^{1/2}\mathbf{u}_0, \end{aligned} \tag{7}$$

where the second summand fulfills the nominal control law $\mathbf{u}_0$ in the null space of the first term. When having more than two tasks, these can be nested in a similar fashion leading to a general framework of hierarchical task control (Peters et al. 2005; Sentis and Khatib 2005).

Example 1. An illustrative example of operational-space control is tracking the end-effector position $x = q_1 + q_2$ of a prismatic robot with two parallel links with joint positions q_1, q_2; see Figure 1. The goal is to track a desired trajectory $x$ref which consists of two super-imposed sinusoids. Here, the mass matrix is given by $\mathbf{M} = \operatorname{diag}(m_1, 0) + m_2\mathbf{1}$ with masses $m_1 = m_2 = 1$, and $\mathbf{1}$ denotes a matrix with all coefficients equal to one. The internal forces are $\mathbf{F} = 0$, the Jacobian is $\mathbf{J} = [1, 1]^{\mathrm{T}}$, and its derivative $\dot{\mathbf{J}} = \mathbf{0}$. The resulting operational-space control laws can be determined using Equation (7). If no null-space control law is selected, i.e. $\mathbf{u}_0 = 0$, the control law in the form of Equation (8) for executing the task $\ddot{x}_{\text{ref}} = \ddot{x}_d + K_d(\dot{x}_d - \dot{x}) + K_p(x_d - x)$ would result in unstable behavior for most metrics $\mathbf{N}$. Owing to the linearity of the prismatic robot, the control law always result in lines such as

$$\mathbf{u} = \mathbf{g}\ddot{x}_{\text{ref}} \tag{8}$$

for the case without a null-space control component. Here, $\mathbf{g}$ denotes a vector distributing the reference acceleration $\ddot{x}_{\text{ref}}$ over both joints. When adding a $\mathbf{u}_0 = -\mathbf{K}_D\dot{\mathbf{q}} - \mathbf{K}_D\mathbf{q}$ pulling the robot towards $\mathbf{q}_{\text{rest}} = 0$, we obtain stable tracking of the task-space trajectory with very different properties for the joint-space trajectory as can be observed in Figure 1: (i) metric $\mathbf{N} = \mathbf{I}$ will result in the second link tracking the end-effector and the null-space component stabilizing the first link; (ii) metric $\mathbf{N} = \mathbf{M}^{-1}$ will distribute the task on both links evenly and have the null-space component decouple the two links; (iii) metric $\mathbf{N} = \mathbf{M}^{-2}$ simply minimizes the squared acceleration.

We use this simple robot example (Example 1) to illustrate various other issues below as it allows easy analytical understanding and graphical visualizations.

Similarly to torque-based operational-space control, we can also consider learning of resolved motion rate control in a similar manner. From the point of view of the optimization framework, we would select a nominal velocity $\dot{\mathbf{q}}_0$ (e.g. a velocity $\dot{\mathbf{q}}_0 = -\mathbf{K}_P(\mathbf{q} - \mathbf{q}_{\text{rest}})$ which pulls the robot towards a rest posture $\mathbf{q}_{\text{rest}}$), and then solve the constrained optimization problem (according to Nakamura (1991))

$$\begin{aligned} \min_{\dot{\mathbf{q}}} C_1(\dot{\mathbf{q}}) &= (\dot{\mathbf{q}} - \dot{\mathbf{q}}_0)^{\mathrm{T}} \mathbf{N} (\dot{\mathbf{q}} - \dot{\mathbf{q}}_0), \\ \text{subject to } \mathbf{J}\dot{\mathbf{q}} &= \dot{\mathbf{p}}_{\text{ref}}, \\ \dot{\mathbf{q}}_0 &= -\mathbf{K}_P(\mathbf{q} - \mathbf{q}_{\text{rest}}). \end{aligned} \tag{9}$$

Here, the nominal component $\dot{\mathbf{q}}_0$ will be canceled out if it conflicts with the task performance $\dot{\mathbf{p}}_{\text{ref}}$, but otherwise will regularize the solution towards more favorable trajectory generation. This formulation results in the general solution given by

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{N}^{-1/2}\left(\mathbf{J}\mathbf{N}^{-1/2}\right)^{+}\dot{\mathbf{p}}_{\text{ref}} \\ &+ \mathbf{N}^{-1/2}\left(\mathbf{I} - \left(\mathbf{J}\mathbf{N}^{-1/2}\right)^{+}\left(\mathbf{N}^{-1/2}\mathbf{J}\right)\right)\mathbf{N}^{1/2}\dot{\mathbf{q}}_0, \end{aligned} \tag{10}$$

where the first term achieves the desired task, while the second term is responsible for appropriate null-space behavior. The conversion of this reference joint-space velocity to control inputs is a standard topic of inverse kinematics-based control, and is discussed by, e.g., Nakanishi et al. (2007) and Sciavicco and Siciliano (2007).

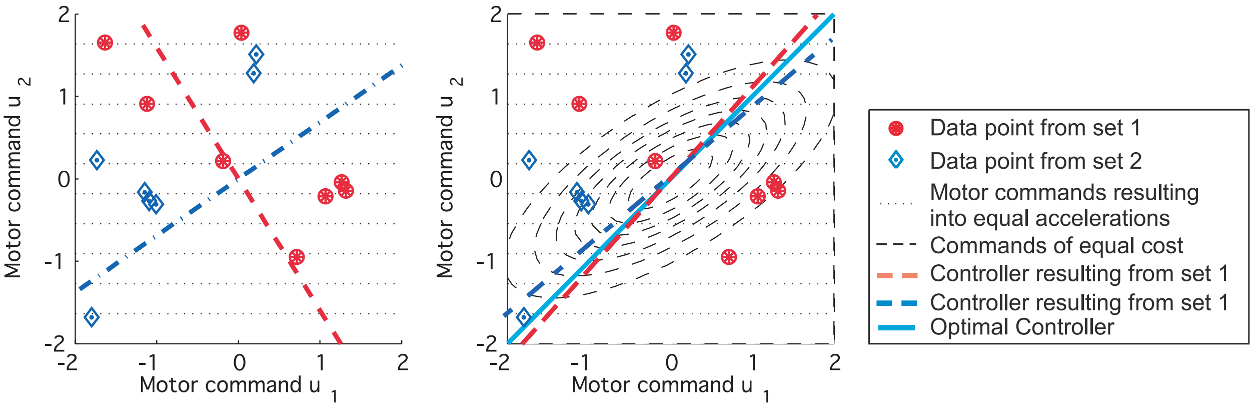
1.3. Why Should We Learn Operational-space Control?

When an accurate analytical model of the robot is available and its parameters can be well estimated, operational-space control laws can be highly successful (Khatib 1987; Park et al. 1999; Nakanishi et al. 2005; Sentis and Khatib 2005). However, in many new complex robotic systems, e.g., humanoid robots or space robots, accurate analytical models of the robot dynamics are not available owing to significant departures from idealized theoretical models such as rigid body dynamics. For instance, in our experience with anthropomorphic robots, unmodeled non-linear effects were caused by complex non-linearities in the actuation, actuator dynamics[2], hydraulic hoses and cable bundles routed along the lightweight structure of the robot as well as complex friction effects. Trying to model such non-linearities is of little use owing to the lack of generality of such an approach and the daunting task of deriving useful models for the unknown effects.

Example 2. In the prismatic robot from Example 1, already small unmodeled non-linearities can have a drastic effect. If the estimated mass matrix of the robot $\tilde{\mathbf{M}} = \text{diag}(m_1, 0) + m_2\mathbf{1}$ (where $\mathbf{1}$ is a matrix with only ones as entries) only differs from the true $\mathbf{M}$ by $M_{12} - \tilde{M}_{12} = M_{21} - \tilde{M}_{21} = 0.5\sin(q_1 + q_2)$, e.g. through unmodeled properties of the cables, then the resulting control law will result in unstable and unpredictable null-space behavior despite the fact that accurate task-space tracking is theoretically still possible. On a real physical system, excessive null-space behavior saturates the motors of the robot, such that task-space tracking also degrades, and the entire control system becomes unstable.

Example 2 demonstrates how a small modeling error decreases the performance of the operational control law and can result in joint-space instability even for simple robots. For lightweight robot arms or full-body humanoid robots, such problems become even more frequent and difficult to cope with. Traditionally, this problem is fixed by manually improving the approximation of the plant, i.e. after estimating the parameters in a structured rigid body model (using either CAD data or linear regression on sampled trajectories), the control engineer uses the model error in order to identify the unknown non-linearities. While statistical learning techniques can help both for the structured model estimation (e.g. Bayesian regression techniques can improve the accuracy, see Ting et al. (2006)), as well for approximating the unmodeled non-linearities (see, e.g., Schaal et al. (2002) and Farrell and Polycarpou (2006)), the direct learning of operational-space

[2] Note that actuator dynamics can introduce a hidden state problem and these might need to be taken into account. However, in many cases, this extension is straight forward for our learning approach (e.g., by using motor side encoders and joint side encoders as inputs to the linear model) while it would be highly complicated to create the model.



(a) Unweighted datasets

(b) Reward weighted datasets

Fig. 2. The plane of all possible motor commands with the resulting contours of task-space acceleration (for the prismatic robot in Example 1) indicated by the horizontal faintly dotted lines. Any valid control law here is a line $\mathbf{u} = \mathbf{g}\ddot{x}_{\text{ref}}$ mapping one task-space acceleration to two motor commands and will automatically pass through the origin as there is no bias term. We illustrate how (a) different randomly sampled data sets result in different least squares solutions if each data point is treated with equal importance (the dash-dotted line corresponds to the diamonds and the dashed line to the circles). If these data points are (b) weighted down using the reward function $r = \exp(-\mathbf{u}^T\mathbf{N}\mathbf{u})$ (here indicated as solid black line) the solutions of different data sets will consistently approximate the same solutions shown in the solid gray line. While for the linear prismatic robot one could live with any solution in (a), a combination of different local solutions for non-linear robots needs to have a consistent global solution over all local regions.

control is a promising novel alternative for low-gain controlled lightweight robots that are hard to model. Details are discussed in Section 1.

2. Learning Methods for Operational-space Control

Learning operational-space control with redundant manipulators is largely an unexplored problem and the literature has only few related examples. Among those, learning approaches to task-level control focused mostly on an inverse kinematics end-effector control (Guezand Ahmad 1988; Jordan and Rumelhart 1992; Bullock et al. 1993; Tevatia and Schaal 2000; D'Souza et al. 2001), i.e learning an inverse kinematics mapping, in order to create appropriate reference trajectories in joint space, which were to be executed by a given joint-space control law or were simply optimizing a certain trajectory (DeLuca and Mataloni 1991). The combination of a learned inverse kinematics and a learned inverse dynamics controller Guezand Ahmad 1988; Tevatia and Schaal 2000; D'Souza et al. 2001 can only be found occasionally in the literature. To the best of our knowledge, full operational-space control laws with redundancy have not been addressed by general learning approaches to date.

2.1. Can Operational-space Control be learned?

Learning operational-space control is equivalent to obtaining a mapping

$$(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\text{ref}}) \mapsto \mathbf{u} \tag{11}$$

from sampled data using a function approximator. However, as the dimensionality of the task-space reference trajectory $\ddot{\mathbf{x}}_{\text{ref}}$ is lower than that of the motor command $\mathbf{u}$, there are infinitely many solutions for $\mathbf{u}$ for most joint positions $\mathbf{q}$ and joint veloccities $\dot{\mathbf{q}}$. For the illustrative prismatic case in Example 1, which has no null-space component, this results in linear mapping without offset corresponding to a line in the plane of possible control laws as shown in Equation (8) and illustrated by the two lines in Figure 2(a). The same holds true for resolved motion rate control, where the simpler mapping $(\mathbf{q}, \dot{\mathbf{x}}_{\text{ref}}) \mapsto \dot{\mathbf{q}}$ is being learned; all discussions in this section transfer straightforwardly.

A major problem arises for the case of a robot with rotary joints where the motor commands $\mathbf{u}$ that achieve the same reference acceleration $\ddot{\mathbf{x}}_{\text{ref}}$ no longer form a convex set, a problem first described in the context of learning inverse kinematics (Jordan and Rumelhart 1992; D'Souza et al. 2001). Thus, when learning the inverse mapping $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\text{ref}}) \mapsto \mathbf{u}$, the learning algorithm will potentially average[3] over unconnected

[3] Note that most regression algorithms can be seen as averages over presented solutions.

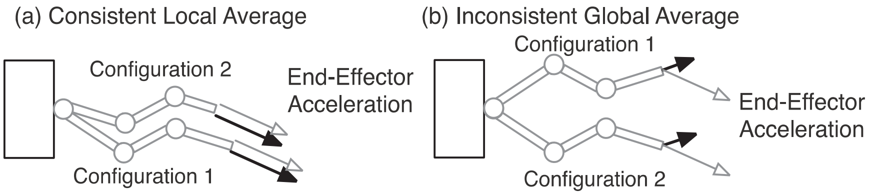

(a) Consistent Local Average
Configuration 2
End-Effector Acceleration
Configuration 1
(b) Inconsistent Global Average
Configuration 1
End-Effector Acceleration
Configuration 2

Fig. 3. Four configurations of a 3 DoF robot arm where the employed torques result in the identical end-effector accelerations, indicated by the gray arrows (with white arrowhead filling). For simplicity, we assume that the arm is at rest (zero velocities). (a) If the joint angle configurations 1 and 2 are similar, the average of the two joint torque vectors does not change the end-effector acceleration. (b) However, if the two joint angle configurations are quite different, the average of the joint torque vectors can results in a drastically different end-effector acceleration, which, from the view point of control, could push the robot into undesirable and even unstable control situations.

and/or non-convex sets of the solutions, which can result in invalid solutions to the control problem. Therefore, straightforward learning from samples with supervised learning techniques is not suitable.

Nevertheless, the convexity issues can be resolved by employing a spatially localized supervised learning system. In our case, spatial localization based on both joint-space position and velocity is required. Such an approach was first introduced in the context of inverse kinematics learning (Bullock et al. 1993; D'Souza et al. 2001).. The key idea is that the local combination of sampled data points will always result in a convex set of solutions for the inverse problem. Here, the key point is that if we have n motor commands $\mathbf{u}^1, \ldots, \mathbf{u}^n$ which result in the same end-effector acceleration $\ddot{\mathbf{x}}$, their convex combination should also result in this acceleration. Globally, this cannot be achieved as shown in Figure 3(b). However, in a small region in the vicinity of the same $\mathbf{q}$, $\dot{\mathbf{q}}$, we can show that the averages will be consistent as shown in Figure 3(a).

This can be derived by showing that we can average over samples in a local region without creating invalid solutions. Using the combination of Equations (1) and (2), and assuming a constant spatial position $\mathbf{q}$ and velocity $\dot{\mathbf{q}}$, the average end-effector acceleration can be written as

$$\begin{aligned} \bar{\ddot{\mathbf{x}}} &= \langle \ddot{\mathbf{x}} \rangle = \left\langle \mathbf{J}\mathbf{M}^{-1}(\mathbf{u}+\mathbf{F}) + \dot{\mathbf{J}}\dot{\mathbf{q}} \right\rangle \\ &= \mathbf{J}\mathbf{M}^{-1}\langle \mathbf{u}+\mathbf{F} \rangle + \dot{\mathbf{J}}\dot{\mathbf{q}} = \mathbf{J}\mathbf{M}^{-1}(\bar{\mathbf{u}}+\mathbf{F}) + \dot{\mathbf{J}}\dot{\mathbf{q}}\,. \end{aligned} \tag{12}$$

Here, $\langle\cdot\rangle$ denotes the average over all points in a given data set. Thus, we have demonstrated that the average $\bar{\mathbf{u}}$ over all motor commands resulting in a given $\ddot{\mathbf{x}}$ still achieves the same $\ddot{\mathbf{x}}$.

Thus, in the vicinity of same $\mathbf{q}$, $\dot{\mathbf{q}}$, any data set can introduce a locally valid linear control law transforming $\bar{\ddot{\mathbf{x}}}$ into[4] $\bar{\mathbf{u}}$. Locally linear controllers

$$\mathbf{u}^i = \mathbf{c}^i_\beta(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\mathrm{ref}}) = \left[\ddot{\mathbf{x}}^{\mathrm{T}}_{\mathrm{ref}}, \dot{\mathbf{q}}^{\mathrm{T}}, 1\right]\beta^i, \tag{13}$$

with parameters β^i can be used if they are only active in a region around $\mathbf{q}$, $\dot{\mathbf{q}}$ (note that we added constant input in (13) to account for the intercept of a linear function; this intercept is a result of locally constant gravity and can also absorb some of the spring-like properties of the hydraulic tubes between the joints). From a control engineering point of view, this argument corresponds to the insight that when we can linearize the plant in a certain region, we can find a local control law in that region by treating the plant as linear. In general, linear systems do not have the problem of non-convexity of the solution space when learning an inverse function.

Next we need to address how to find an appropriate piecewise linearization for the locally linear controllers. For this purpose, we learn a locally linear forward or predictor model

$$\ddot{\mathbf{x}}^i = \mathbf{p}^i_\beta(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}) = \left[\dot{\mathbf{q}}^{\mathrm{T}}, \mathbf{u}^{\mathrm{T}}, 1\right]\hat{\beta}^i \tag{14}$$

where $\hat{\beta}^i$ denotes the parameters of the local forward model. Learning this forward model is a standard supervised learning problem since the mapping is guaranteed to be a proper function[5].

A method of learning such a forward model that automatically also learns a local linearization is locally weighted projection regression (LWPR) (Schaal et al. 2002). In essence,

[4] Note that the localization in velocity $\dot{\mathbf{q}}$ can be dropped for a pure rigid body formulation as it is linear in the $\dot{q}_i\dot{q}_j$ for all DoF i, j; this, however, is not necessarily desirable as it will add new inputs to the local regression problem which grows quadratically with the number of DoF.

[5] While learning a forward model is relatively straightforward, it only serves for identifying local linear regions. While, in theory, it is possible to extract all terms from the locally linear forward models that are needed in computing the analytical control law in Equation (7), one encounters the same problems as in operational-space control with inaccurate system identification. The locally linear forward models are good predictors for the forward dynamics, but this does not mean that the regression coefficients are close to the analytically correct coefficients, particularly for high-dimensional systems. These inaccuracies create large errors in the analytical control law (7), and render such a forward-model learning approach for operational space control largely infeasible.

LWPR performs piecewise linear function approximation by means of locally weighted regression (Atkeson et al. 1997a,b) while automatically detecting the appropriate local linear region of each linear model. This fast online learning method scales into high dimensions, has been used for inverse dynamics control of humanoid robots, and can automatically determine the number of local models that are needed to represent the function (Vijayakumar et al. 2002, 2005). The membership of a local model is determined by a weight generated from a Gaussian kernel

$$w^i\left(\mathbf{q}, \dot{\mathbf{q}}\right) = \exp\left(\frac{1}{2}\left(\begin{bmatrix}\mathbf{q}\\ \dot{\mathbf{q}}\end{bmatrix} - \mathbf{c}_i\right)^{\mathrm{T}} \mathbf{D}^i \left(\begin{bmatrix}\mathbf{q}\\ \dot{\mathbf{q}}\end{bmatrix} - \mathbf{c}_i\right)\right) \quad (15)$$

centered at $\mathbf{c}_i$ in $(\mathbf{q}, \dot{\mathbf{q}})$-space and shaped by a distance metric $\mathbf{D}_i$. For a closer description of this statistical learning algorithm, see Schaal et al. (2002) and Vijayakumar et al. (2002, 2005).

For each local forward model created by LWPR, we create a local controller that uses the same localization in $(\mathbf{q}, \dot{\mathbf{q}})$ as determined by the forward model. This model is learned in parallel using the same data as used in the forward model. It will automatically result into a locally viable control law (global consistency will be treated in Section 2.2). This approach of pairwise combining predictors and controllers is related to the MOSAIC architecture (Haruno et al. 1999) where the quality of predicting a task is used to select which local controller should be used for the task.

It is straightforward to make a similar case for resolved motion rate control where a localization on regions in the joint-space suffice, i.e. instead of learning in the vicinity of a $(\mathbf{q}, \dot{\mathbf{q}})$, we only need to stay in the vicinity if a $\mathbf{q}$ in order to obtain a convex mapping and the proof transfers straightfowardly (see, e.g., D'Souza et al. (2001) and Peters and Nguyen-Tuong (2008)). In this case, we only require simpler controller models $\dot{\mathbf{q}}^i = \left[\dot{\mathbf{x}}_{\mathrm{ref}}^{\mathrm{T}}, \mathbf{q}^{\mathrm{T}}, 1\right] \Lambda^i$, simpler predictor models $\dot{\mathbf{x}}^i = \left[\dot{\mathbf{q}}^{\mathrm{T}}, \mathbf{q}^{\mathrm{T}}, 1\right] \hat{\Lambda}^i$ and a weighting $w^i\left(\mathbf{q}\right) = \exp -0.5\left(\mathbf{q} - \tilde{\mathbf{c}}_i\right)^{\mathrm{T}} \tilde{\mathbf{D}}^i \left(\mathbf{q} - \tilde{\mathbf{c}}_i\right)$. Except for these small changes, the line of thought remains the same.

2.2. Combining the Local Controllers and Ensuring Consistent Resolution of Redundancy

In order to control a robot with these local control laws, they need to be combined into a consistent global control law. The combination is given by a weighted average (Schaal et al. 2002)

$$\mathbf{u} = \frac{\sum_{i=1}^{n} w^i\left(\mathbf{q}, \dot{\mathbf{q}}\right)\left[\ddot{\mathbf{x}}_{\mathrm{ref}}^{\mathrm{T}}, \dot{\mathbf{q}}^{\mathrm{T}}, 1\right]\beta^i}{\sum_{i=1}^{n} w^i\left(\mathbf{q}, \dot{\mathbf{q}}\right)}, \quad (16)$$

where each control law $\mathbf{c}_{\beta}^{i}\left(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\mathrm{ref}}\right)$ is only valid in its local region computed by $w^i\left(\mathbf{q}, \dot{\mathbf{q}}\right)$, and β^i are the parameters of each local operational-space control law[6].

However, while the mappings $\left(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\mathrm{ref}}\right) \mapsto \mathbf{u}$ can properly be learned locally in the neighborhood of some $\mathbf{q}$, $\dot{\mathbf{q}}$, owing to the redundancy in the robotic system, there is no guarantee that across the local mappings the same type of solution is acquired. This problem is a result of the dependence of the inverse solution on the training data distribution in each local model, i.e. different distributions will pick different solutions for the inverse mapping from the infinite number of possible inverses. In Figure 2(a), we demonstrate this effect. While this problem is not devastating for the prismatic robot from Example 1, it results in severe problems for any non-linear robot requiring multiple, consistent linear models. There are two different approaches to tackling such problems: (1) by biasing the system using a pre-processed data set such that it can only produce one particular inverse solution (D'Souza et al. 2001); and (2) by incorporating a cost/reward function in order to favor a certain kind of solution (an example that is discussed later and is shown in Figure 2(b)). The first approach lacks generality and can bias the learning system such that the task is no long accomplished properly. The major shortcoming of the second approach is that the choice of the cost/reward function is, in general, non-trivial and determines the learning algorithm as well as the learned solution.

The crucial component to finding a principled approach to this inconsistency problem is based on the discussion in Section 1.2 and previous work (Peters et al. 2005). Operational-space control can be seen as a constrained optimization problem with the cost function given in Equation (3). Thus, the cost function based approach for the creation of a consistent set of local controllers for operational-space control can be based on this insight. The cost function can be turned into an immediate reward $r\left(\mathbf{u}\right)$ by running it through an exponential function, i.e.

$$\begin{aligned} r\left(\mathbf{u}\right) &= \sigma \exp\left(-0.5\sigma^{-2}C_1\left(\mathbf{u}\right)\right) \\ &= \sigma \exp\left(-\sigma^{-2}\mathbf{u}_1^{\mathrm{T}}\mathbf{N}\mathbf{u}_1\right), \end{aligned} \quad (17)$$

where σ is a scaling factor chosen for notational convenience. The task-space command $\mathbf{u}_1 = \mathbf{u} - \mathbf{u}_0$ can be computed using a desired null-space behavior $\mathbf{u}_0$ (e.g. pulling towards a rest posture as discussed in Section 1.2). The scaling factor σ does not affect the optimality of a solution $\mathbf{u}$ as it acts as a monotonic transformation in this cost function. However, this transformation can increase the efficiency of the learning algorithm

[6] As discussed before, simply inverting the weighted combination of the forward models will not suffice: the forward model does not provide sufficient information to apply (4), and inverting such forward models is numerically very fragile, particularly as actual robot data is usually highly rank deficient.

significantly when only sparse data is available for learning (i.e. as for most interesting robots the high-dimensional action spaces of complex robots will hardly ever be filled densely with data)[7]. These local rewards allow the reformulation of our learning problem as an *immediate reward reinforcement learning problem* (Dayan and Hinton 1997), as discussed in Section 3.

We are now in the position to formulate a supervised learning algorithm for the local operational-space controllers. The task constraint in (3) as well as the rigid body dynamics in (1) are automatically fulfilled by all data sampled from the real robot similar to a self-supervised learning problem. Therefore, for learning the local operational-space controllers, we have obtained a local linear regression problem where we attempt to learn primarily from the observed motor commands $\mathbf{u}^k$ which also have a high reward $r(\mathbf{u}^k)$ within each active local model $\mathbf{c}^i_{\beta}(\mathbf{q}^k, \dot{\mathbf{q}}^k, \ddot{\mathbf{x}}^k_{\text{ref}})$. An intuitive solution is to use reward-weighted regression, i.e. to find the solution which minimizes

$$\sum_{k=1}^{N} r\left(\mathbf{u}^k\right) w^i\left(\mathbf{q}^k, \dot{\mathbf{q}}^k\right)\left(\mathbf{u}^k - \left[\ddot{\mathbf{x}}^{k,T}_{\text{ref}}, \dot{\mathbf{q}}^{k,T}, 1\right]\beta^i\right)^2 \to \min \tag{18}$$

for each controller i. The solution to this problem is the well-known weighted regression formula

$$\beta = \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{W}^i\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{W}^i\mathbf{U} \tag{19}$$

with rows in the matrices $\mathbf{\Phi}$ and $\mathbf{U}$: $\mathbf{\Phi}_k = \left[(\ddot{\mathbf{x}}^k_{\text{ref}})^T, (\dot{\mathbf{q}}^k)^T, 1\right]$, $\mathbf{U}_k = (\mathbf{u}^k)^T$, and $\mathbf{W}^i_k = r\left(\mathbf{u}^k\right) w_i\left(\mathbf{q}^k, \dot{\mathbf{q}}^k\right)$. When employing this reward-weighted regression solution, we converge to a globally consistent solution across all local controllers. The learning algorithm is shown in Algorithm 1 together with an additional component derived in Section 3. Note that this step was only possible as a result of the essential cost function in Equation (6) from our previous work.

This framework transfers quite straightforwardly to resolved motion rate control by replacing the local control laws, local predictors, and weighting kernels by those described previously.

Algorithm 1 The complete learning algorithm for operational-space control. See the text for detailed explanations:

```
for each new data point $[\ddot{\mathbf{x}}^k_{\text{ref}}, \mathbf{q}, \dot{\mathbf{q}}^k, \mathbf{u}^k]$
    Add $(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}) \to \ddot{\mathbf{x}}$ to the forward model regression.
    Determine the current number of models $n$ and
    localizations of the forward models $w^i(\mathbf{q}, \dot{\mathbf{q}})$.
    Compute desired null-space behavior
    $\mathbf{u}^k_0 = f(\mathbf{q}^k, \dot{\mathbf{q}}^k)$.
    Compute costs $C^k_1 = (\mathbf{u}^k_1)^{\mathrm{T}}\mathbf{N}(\mathbf{q}^k)\mathbf{u}^k_1$ with
    $\mathbf{u}^k_1 = \mathbf{u}^k - \mathbf{u}^k_0$.
    For each model $i = 1, 2, \ldots, n$
      Update mean cost:
        $\sigma^2_i = \sum_{h=1}^{n} w^i(\mathbf{q}^h, \dot{\mathbf{q}}^h)C^h_1 / \sum_{h=1}^{n} w^i(\mathbf{q}^h, \dot{\mathbf{q}}^h)$,
      Compute reward:
        $r(\mathbf{u}^k) = \sigma_i \exp(-0.5\sigma^2_i C^k_1)$
      Add data point to weighted regression so that:
        $\mathbf{\Phi}_k = [\mathbf{q}^k, \dot{\mathbf{q}}^k, \ddot{\mathbf{x}}^k_{\text{ref}}]$
        $\mathbf{U}_i = \mathbf{u}^k$
        $\mathbf{W}^i = \mathrm{diag}(r(\mathbf{u}^1)w^i_1, \ldots, r(\mathbf{u}^n)w^i_n)$
      Perform policy update by regression
        $\beta^i_{k+1} = (\mathbf{\Phi}^{\mathrm{T}}\mathbf{W}^i\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{W}^i\mathbf{U}$,
    end
end
```

3. Reformulation as a Reinforcement Learning Problem

Another way of looking at operational-space control is to view it as an immediate reward reinforcement learning problem (Kaebling et al. 1996) with high-dimensional, continuous states $\mathbf{s} = \left[\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\text{ref}}, \mathbf{u}_0\right] \in \mathbb{R}^n$ and actions $\mathbf{u} \in \mathbb{R}^m$ for operational-space control (similarly, we have $\mathbf{s} = \left[\mathbf{q}, \dot{\mathbf{x}}_{\text{ref}}, \mathbf{u}_0\right] \in \mathbb{R}^n$ and actions $\dot{\mathbf{q}}_{\text{ref}} \in \mathbb{R}^m$ if we intend to learn resolved motion rate control). The goal of learning is to obtain an optimal policy

$$\mathbf{u} = \mu\left(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\text{ref}}, \mathbf{u}_0\right) = \mu\left(\mathbf{s}\right) \tag{20}$$

such that the system follows the reference acceleration $\ddot{\mathbf{x}}_{\text{ref}}$ while maximizing the immediate reward $r\left(\mathbf{u}\right) = -\left(\mathbf{u} - \mathbf{u}_0\right)^{\mathrm{T}}\mathbf{N}\left(\mathbf{u} - \mathbf{u}_0\right)$ for any given nominal behavior $\mathbf{u}_0$. In order to incorporate exploration during learning, we need a stochastic control policy $\mathbf{u} = \mu_{\boldsymbol{\theta}}\left(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\text{ref}}\right) + \varepsilon$, modeled as a probability distribution $\pi_{\boldsymbol{\theta}}\left(\mathbf{u}|\mathbf{s}\right) = p(\mathbf{u}|\mathbf{s}, \boldsymbol{\theta})$ with parameter vector $\boldsymbol{\theta}$. The goal of the learning system is thus to find the policy parameters $\boldsymbol{\theta}$ that maximize

[7] The reward has to be seen in the light of the relationship between the Gaussian distribution and Gauss' principle for constrained motion as suggested already by Carl-Friedrich Gauss in his original work (Udwadia 2005).

$$J_r(\boldsymbol{\theta}) = \int p(\mathbf{s}) \int \pi_{\boldsymbol{\theta}}(\mathbf{u}|\mathbf{s})\, r(\mathbf{s}, \mathbf{u})\, d\mathbf{u}\, d\mathbf{s}. \qquad (21)$$

Here $p(\mathbf{s})$ denotes the distribution of states, which is treated as fixed in immediate reward reinforcement learning problems (Kaebling et al. 1996; Peters and Schaal 2007).

Originally, we derived this algorithm from a weighted regression point of view. However, this point of view is not completely satisfying since it still has the open parameter σ^2, which determines the speed of convergence of the learning controllers. An alternative view point, i.e. in the framework of immediate reward reinforcement learning, allows the previous algorithm to be derived together with a computation rule for σ^2. Previous work in the literature suggested a variety of optimizing methods which can be applied to immediate reward reinforcement learning problems, for example, gradient-based methods (e.g. REINFORCE, Covariant REINFORCE, finite difference gradients, the Kiefer–Wolfowitz procedure, A_{RP} algorithms, CRBP, etc.) and random search algorithms (e.g. simulated annealing or genetic algorithms) (Kaebling et al. 1996; Dayan and Hinton 1997; Spall 2003). However, gradient-based methods tend to be too slow for the online learning that we desire in our problem, while randomized search algorithms can create too arbitrary solutions, and are often unsuitable for execution on a robotic system. For learning operational-space control, we require a method that is computationally sufficiently efficient to deal with high-dimensional robot systems and large amounts of data, that has a low sample complexity, that comes with convergence guarantees, and that is suitable for smooth online improvement. For instance, linear regression techniques and/or methods employing EM-style algorithms are highly desirable.

A good starting point for our work is the probabilistic reinforcement learning framework by Dayan and Hinton (1997). As we show in the following, this approach allows us to derive an EM algorithm that essentially reduces the immediate reward learning problem to a reward-weighted regression problem (Peters and Schaal 2007).

3.1. Reward Transformation

In order to maximize the expected return given by (21) using samples, we approximate

$$J_r(\boldsymbol{\theta}) \approx \sum_{i=1}^{n} \pi_{\boldsymbol{\theta}}(\mathbf{u}_i|\mathbf{s}_i)\, r_i, \qquad (22)$$

where $r_i = r(\mathbf{s}_i, \mathbf{u}_i)$. For application of the probabilistic reinforcement learning framework by Dayan and Hinton (1997), the reward needs to be strictly positive such that it resembles an (improper) probability distribution. While this can be achieved by a linear rescaling for problems with bounded rewards, for unbounded rewards as discussed in this paper this is no longer the case. Instead, a non-linear transformation of the reward $U_\tau(r)$ is required, with the constraint that the optimal solution to the underlying problem remains unchanged. Thus, we require that $U_\tau(r)$ is strictly monotonic with respect to r, and additionally that $U_\tau(r) \geq 0$ and $\int_0^\infty U_\tau(r)\, dr =$ constant, resulting in the transformed optimization problem

$$J_u(\boldsymbol{\theta}) = \sum_{i=1}^{n} \pi_{\boldsymbol{\theta}}(\mathbf{u}_i|\mathbf{s}_i)\, U_\tau(r_i). \qquad (23)$$

The reward transformation plays a more important role than initially meets the eye: as already pointed out by Dayan and Hinton (1997), this transformation can have a huge effect on convergence speed. Making $U_\tau(r)$ an adaptive part of the learning algorithm by means of some internal parameters $\boldsymbol{\tau}$ can greatly accelerate the learning speed and help avoid local minima during learning. Figure 4 demonstrates this issue with a one-dimensional continuous state and one-dimensional continuous action example where the goal is to learn an optimal linear policy $\pi(u|s) = \mathcal{N}(u|\theta_1 s + \theta_2, \sigma^2)$ for a prismatic robot with a transformed reward $u(r(s, u)) = \exp(-\tau(q_1 u^2 + q_2 us + sq_3^2))$. Using the algorithm that we introduce below, an adaptive reward transformation accelerated the convergence by a factor of four, and actually significantly helped avoid local minima during learning.

3.2. EM Reinforcement Learning with Reward Transformation

To derive our learning algorithm, similar to the algorithm of Dayan and Hinton (1997), we start by establishing the lower bound

$$\log J_u(\boldsymbol{\theta}) = \log \sum_{i=1}^{n} q(i) \frac{\pi_{\boldsymbol{\theta}}(\mathbf{u}_i|\mathbf{s}_i)\, U_\tau(r_i)}{q(i)} \qquad (24)$$

$$\geq \sum_{i=1}^{n} q(i) \log \frac{\pi_{\boldsymbol{\theta}}(\mathbf{u}_i|\mathbf{s}_i)\, U_\tau(r_i)}{q(i)} \qquad (25)$$

$$= \sum_{i=1}^{n} q(i) \left[\log \pi_{\boldsymbol{\theta}}(\mathbf{u}_i|\mathbf{s}_i) + \log U_\tau(r_i) - \log q(i)\right] \qquad (26)$$

$$= \mathcal{F}(q, \boldsymbol{\theta}, \boldsymbol{\tau}), \qquad (27)$$

owing to Jensen's inequality (MacKay 2003). The reweighting distribution $q(i)$ obeys the constraint

$$\sum_{i=1}^{n} q(i) - 1 = 0. \qquad (28)$$

The resulting EM algorithm is given in Algorithm 2.

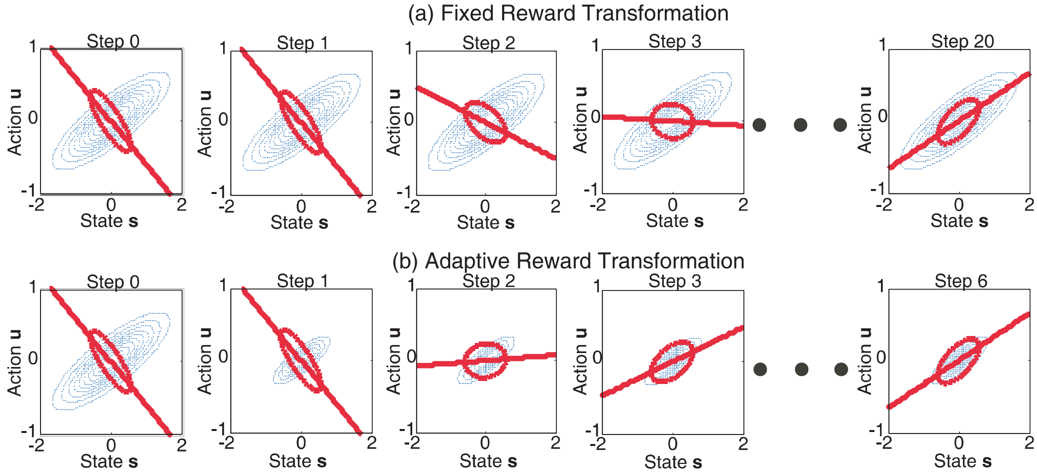


Fig. 4. A comparison of fixed and adaptive reward transformation for learning a linear policy $\pi(u|s) = \mathcal{N}(u|\theta_1 s + \theta_2, \sigma^2)$ under the transformed reward $u(r(s, u)) = \exp(-\tau(q_1 u^2 + q_2 us + sq_3^2))$. The transformed reward is indicated by the dotted ellipses, the variance of the action distribution is indicated by the thick ellipse, and the mean of the linear policy is shown by the thick line. With τ being adaptive, significantly faster learning of the optimal policy is achieved. Step 0 shows the initial policy and initial transformed reward, Step 1 shows the initial policy with adapted transformed reward. Quite clearly, the adaptation of the reward transformation in (b) speeds up the learning process in comparison with (a) while converging to the same solution.

Algorithm 2 An EM algorithm for optimizing both the expected reward as well as the reward-transformation is given by an **E-Step**

$$q_{k+1}(j) = \frac{\pi_{\boldsymbol{\theta}_k}(\mathbf{u}_j|\mathbf{s}_j)U_{\tau_k}(r_j)}{\sum_{i=1}^{n} \pi_{\boldsymbol{\theta}}(\mathbf{u}_i|\mathbf{s}_i)U_{\tau_k}(r_i)}, \tag{29}$$

an **M-Step** for the policy parameter update given

$$\boldsymbol{\theta}_{k+1} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} q_{k+1}(i) \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_i|\mathbf{s}_i), \tag{30}$$

and an **M-Step** for the adaptive reward transformation given by

$$\boldsymbol{\tau}_{k+1} = \arg\max_{\boldsymbol{\tau}} \sum_{i=1}^{n} q_{k+1}(i) \log U_{\tau}(r_i). \tag{31}$$

Proof. The E-Step is given by

$$q_{k+1} = \arg\max_{q} \mathcal{F}(q, \boldsymbol{\theta}, \boldsymbol{\tau}) \tag{32}$$

while fulfilling the constraint

$$0 = \sum_{i=1}^{n} q(i) - 1. \tag{33}$$

Thus, we obtain a constrained optimization problem with Lagrangian function of

$$\begin{aligned} L(\lambda, q) \quad &= \quad \sum_{i=1}^{n} q(i) \left[\log \pi_{\boldsymbol{\theta}}(\mathbf{u}_i|\mathbf{s}_i) + \log U_\tau(r_i)\right. \\ &- \quad \left.\log q(i) + \lambda\right] - \lambda. \end{aligned} \tag{34}$$

Optimizing $L(\lambda, q)$ with respect to q and λ results in Equation (29). Optimizing $\mathcal{F}(q_{k+1}, \boldsymbol{\theta}, \boldsymbol{\tau})$ with respect to θ and τ yields (30) and (31). □

3.3. Reinforcement Learning by Reward-weighted Regression

Let us assume the specific class of normally distributed policies (as in Peters and Schaal (2007)):

$$\pi_{\boldsymbol{\theta}}(\mathbf{u}|\mathbf{s}) = \mathcal{N}\left(\mathbf{u}|\mu_\theta(\mathbf{s}), \sigma^2\mathbf{I}\right) \tag{35}$$

with a nominal or mean behavior $\mu_{\boldsymbol{\theta}}(\mathbf{s}) = \phi(\mathbf{s})^{\mathrm{T}}\boldsymbol{\theta}$ where $\phi(\mathbf{s})$ denotes some fixed preprocessing of the state by basis functions (which are determined using locally linear forward model approximations as discussed previously) and $\sigma^2\mathbf{I}$ determines the exploration[8]. Furthermore, we choose the reward transformation

$$U_\tau(r) = \tau \exp(-\tau r), \tag{36}$$

[8] Note that $\sigma^2\mathbf{I}$ could be replaced by a full variance matrix with small changes to the algorithm. However, this would result in a quadratic growth of parameters with the dimensionality of the state and is therefore less desirable.

which, for $r > 0$, fulfills all of our requirements on a reward transformation (cited from Section 3.1). Algorithm 2 thus becomes the following.

Algorithm 3 The update equations for the policy $\pi_{\boldsymbol{\theta}}(\mathbf{u}|\mathbf{s}) = \mathcal{N}(\vec{u}|\mu_\theta(\mathbf{s}), \sigma^2\mathbf{I})$ are:

$$\boldsymbol{\theta}_{k+1} = (\Phi^{\mathrm{T}}\mathbf{W}\Phi)^{-1}\Phi^{\mathrm{T}}\mathbf{WY}, \tag{37}$$

$$\sigma^2_{k+1} = \|\mathbf{Y} - \boldsymbol{\theta}^{\mathrm{T}}_{k+1}\Phi\|^2_{\mathbf{W}}, \tag{38}$$

with a diagonal matrix with transformed rewards is denoted by

$$\mathbf{W} = \mathrm{diag}(U_\tau(r_1), U_\tau(r_2), \ldots, U_\tau(r_n))/U_\Sigma \tag{39}$$

with $U_\Sigma = \left(\sum_{i=1}^{n} U_\tau(r_i)\right)$. Furthermore, we denote the policy features by

$$\Phi = [\phi(\mathbf{s}_1), \phi(\mathbf{s}_2), \ldots, \phi(\mathbf{s}_n)]^{\mathrm{T}}, \tag{40}$$

and

$$\mathbf{Y} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n]^{\mathrm{T}} \tag{41}$$

the motor commands. The reward transformation $U_\tau(r) = \tau \exp(-\tau r)$ is updated using

$$\tau_{k+1} = \frac{\sum_{i=1}^{n} U_{\tau_k}(r_i)}{\sum_{i=1}^{n} U_{\tau_k}(r_i) r_i}. \tag{42}$$

Proof. When computing $q_{k+1}(j)$ from samples in (29), we have

$$q_{k+1}(j) = \frac{U_{\tau_k}(r_j)}{\sum_{i=1}^{n} U_{\tau_k}(r_i)} \tag{43}$$

since the probabilities are replaced by relative frequencies. We insert the policy

$$\pi_{\boldsymbol{\theta}}(\mathbf{u}|\mathbf{s}) = \mathbf{Z}_0 \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{u} - \phi(\mathbf{s})^{\mathrm{T}}\boldsymbol{\theta}\right\|^2\right) \tag{44}$$

with $\mathbf{Z}_0 = (2\pi\sigma^2)^{-\frac{d}{2}}$ into Equation (30). By differentiating with respect to θ and equating the result to zero, we obtain Equation as

$$\Phi^{\mathrm{T}}\mathbf{W}\Phi = \sum_{i=1}^{n} q_{k+1}(i)\phi(\mathbf{s}_i)\phi(\mathbf{s}_i)^{\mathrm{T}}$$

$$\Phi^{\mathrm{T}}\mathbf{WY} = \sum_{i=1}^{n} q_{k+1}(i)\phi(\mathbf{s}_i)\mathbf{u}_i^{\mathrm{T}}. \tag{45}$$

Analogously, the reward transformation is obtained from differentiation with respect to τ as

$$\sum_{i=1}^{n} q_{k+1}(i)\frac{\partial}{\partial\tau}\log U_\tau(r_i) = \sum_{i=1}^{n} q_{k+1}(i)\left(\tau^{-1} - r_i\right) = \mathbf{0}, \tag{46}$$

which results in (42). □

It is straightforward to see that the resulting algorithm is equivalent to Algorithm 1.

The derivation of this reward-weighted regression algorithm for this immediate reward reinforcement learning problem allows us now to understand the problem from the weighted regression point of view; see the review papers of Atkeson et al. (1997a,b) for more information on this topic. It allows the highly data-efficient computation of the regression problem solutions with the supervised learning method LWPR (Schaal et al. 2002; Vijayakumar et al. 2002, 2005).

4. Evaluations

We have evaluated our approach both for full, torque-based operational-space control and for resolved motion rate control. The operational-space control framework was evaluated both on two different simulated, physically realistic robots: (i) a 3 DoF planar robot arm shown in Figure 5(a); (ii) a 7 DoF simulated SARCOS Master robot arm (Figure 6(a)); see Section 4.1. The resolved motion rate control was evaluated on an actual Mitsubishi PA-10 robot, see Section 4.2.

4.1. Evaluation for Learning Torque-based Operational-space Control

Both experiments were conducted as follows: first, learning the forward models and an initial control policy in each local model was obtained from random point-to-point movements in joint space using a simple proportional derivative (PD) control law. This "motor babbling" exploration was necessary for bootstrap learning with some initial data. During the first phase, we generated a sequence of 60 arbitrary joint-space positions and connected these positions in joint-space using fifth-order polynomials in order to create desired trajectories in joint space of duration 1 s. A purposely badly tuned joint-space PD control law, which could not track the trajectories accurately, was used to generate the data. Otherwise, we would experience rather slow learning, as typically observed in similar direct-inverse learning approaches (Jordan and Rumelhart 1992). The measured end-effector accelerations served as desired acceleration in Equation (13), and all other variables for learning the local controllers were measurable as well. Subsequently, the learning controller was used on-policy with the normally distributed actuator noise serving as exploration.

Both robots learned to track desired trajectories with high accuracy as shown in Figures 5(b) and 6(b). For the 3 DoF arm, we verified the quality of the learned control commands in

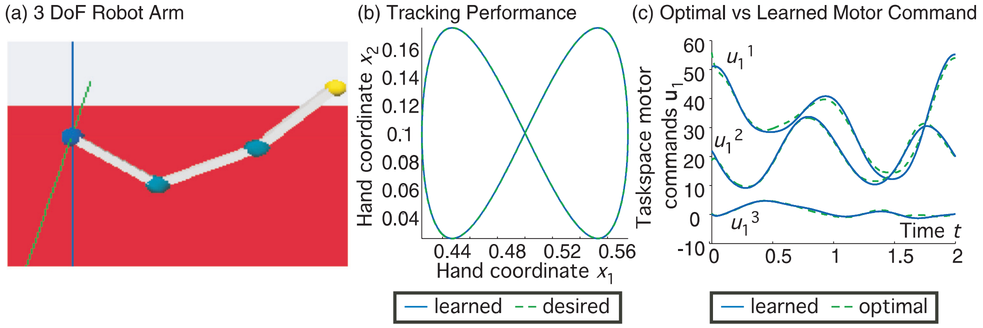


Fig. 5. (a) Screenshot of the 3 DoF arm simulator; (b) near-ideal tracking performance for a planar figure-of-eight pattern for the 3 DoF arm; and (c) a comparison between the analytically obtained optimal control commands (by solving (3) as traditionally done) and the learned commands for one figure-of-eight cycle of the 3 DoF arm exhibits that a near-optimal policy is obtained.

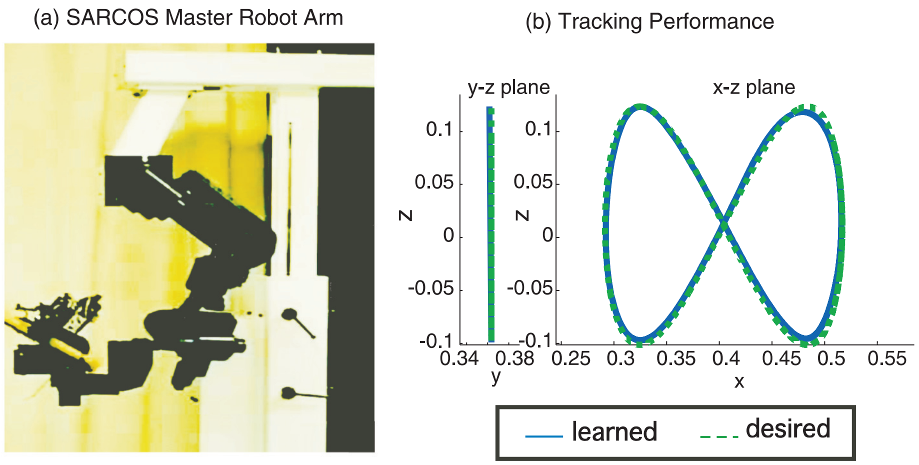
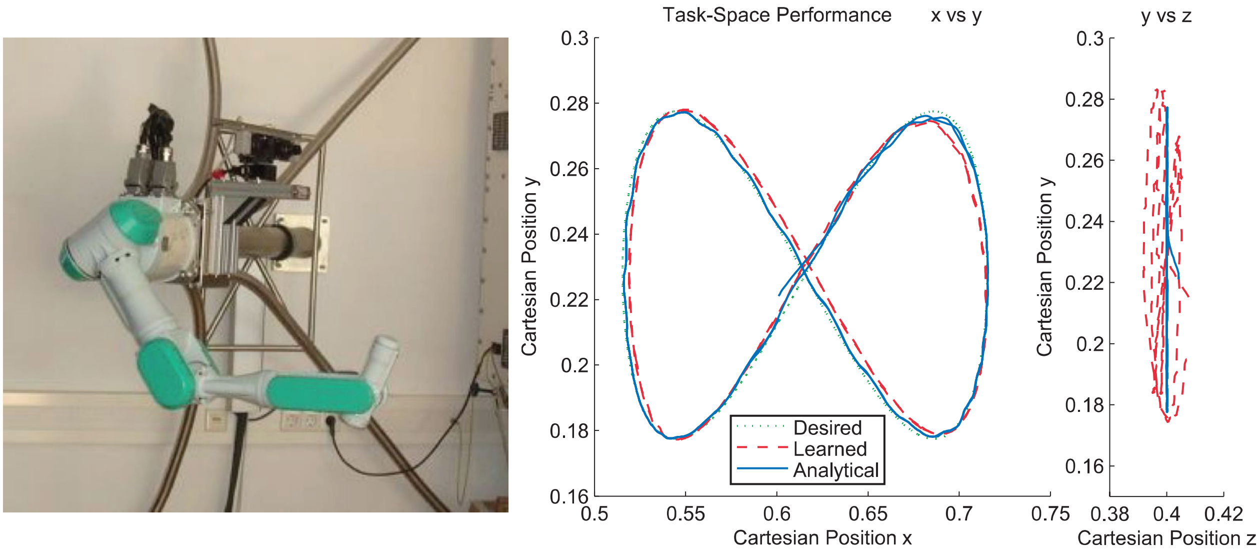


Fig. 6. (a) Anthropomorphic SARCOS Master arm used as simulated system and in progress of actual robot evaluations. (b) Tracking performance for a planar figure-of-eight pattern for the simulated SARCOS Master arm.

comparison to the analytical solution, given in (7): Figure 5(c) demonstrates that the motor commands of the learned and analytically optimal case are almost identical. Learning results of the simulated 7 DoF SARCOS robot achieved almost the same end-effector tracking quality and is shown in Figure 6. It exhibits only slightly increased errors. However, the joint commands were not quite as close to the optimal commands as for the 3 DoF arm: the rather high-dimensional learning space of the 7 DoF arm most likely requires more extensive training and more careful tuning of the LWPR learning algorithm to achieve local linearizations with very high accuracy and with enough data to find the optimal solution. Tuning of LWPR involved the proper normalization of regression inputs and outputs, the initialization of the distance metric such that the number of generated models stays low while achieving accuracy, and the proper setting of the gradient descent learning rates in order to achieve a good adaptation of the local weighting kernels. For more information on the proper application of LWPR, see (Jordan and Rumelhart 1992). The 3 DoF arm required about 2 h of real-time training. The setup, however, was optimized for the 7 DoF arm where a 60 min run of real-time training was sufficient to achieve the quality exhibited on the test trajectory in Figure 6(b).

4.2. Evaluation for Learning Resolved Velocity Control

In order to demonstrate the feasibility of our learning approach on an actual robot, we evaluated our learning approach to resolved motion rate control on a Mitsubishi PA-10 arm shown in Figure 7(a). We compare our results with the analytical solution from the robotics literature (Spong et al. 2006; Sciav-icco



(a) Mitsubishi PA-10

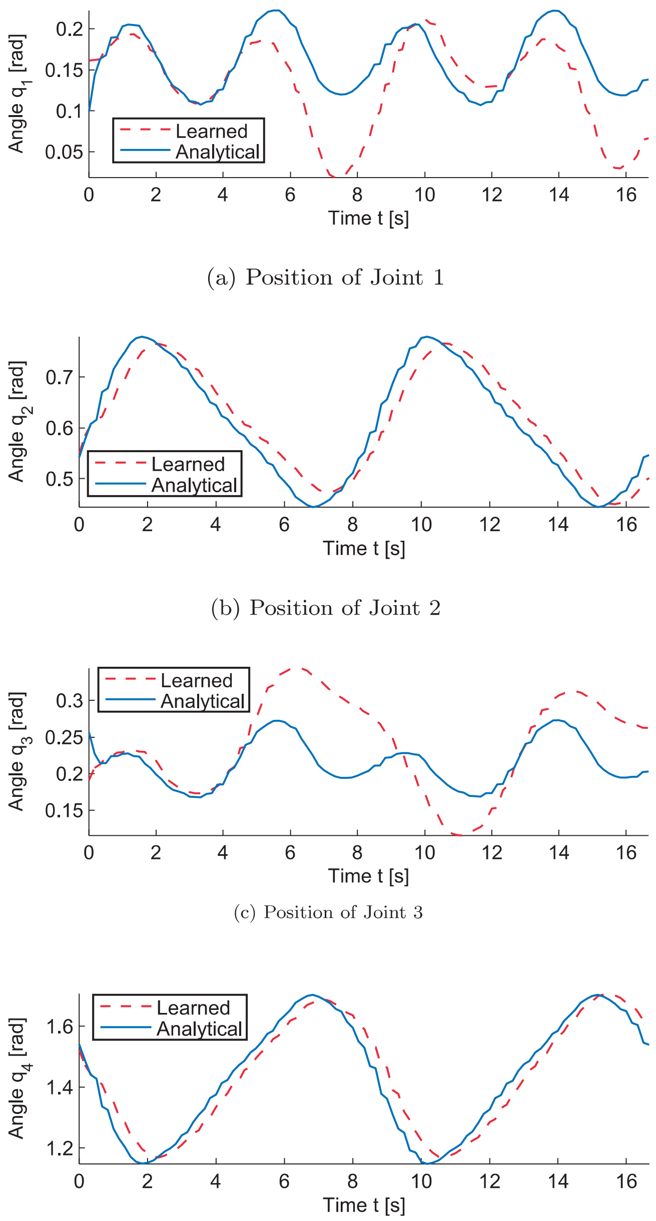
(b) Task-Space Tracking

Fig. 7. (a) The Mitsubishi PA-10 robot arm with 7 DoF used in the experiments in this paper. (b) The task-space tracking performance of both the analytical and the learned resolved motion rate control laws. Here, the dotted line shows the desired trajectory which the robot should follow, the dashed line is the performance of the real-time learning control law, while the solid line shows the performance of the resolved motion rate control law. Note, that while the online learning solution is as good as the analytical solution, it still yields comparable performance without any pre-training of the local control laws before the online learning (nevertheless, the predictors were pre-trained).

and Siciliano 2007). The Mitsubishi PA-10 is general purpose 7 DoF robot arm for low-velocity applications such as surgical tool placement or teleoperated soft-tissue manipulation (Kennedy and Desai 2005). This robot includes a Mitsubishi-supplied control box which allows the velocity to be set in an asynchronous mode at a sampling frequency of 200 Hz. The setup includes a stereo camera system with two Basler cameras mounted on a pan–tilt unit (one in color at 5 Hz, one in black and white at 30 Hz). The goal of the experiment is to show that we can learn consistent resolved motion rate control laws without observing the task beforehand. For doing so, we choose the standard task of a figure-of-eight in task space (Nakanishi et al. 2005, 2007). The rest posture is given by $\mathbf{q}_{\text{rest}} = [0.1627, 0.6076, 0.2127, 1.4407, 0.2626, 1.6965, -0.0138]^{\text{T}}$, and was selected such that it lies in the middle of the visual field of the stereo camera pan–tilt unit. For the joint-space attractor towards the rest posture we chose $\dot{\mathbf{q}}_0 = -\mathbf{K}_P(\mathbf{q} - \mathbf{q}_{\text{rest}})$ with $\mathbf{K}_P = 0.1\mathbf{I}$. The gain $\mathbf{K}_p^R$ of the reference attractor is set to $\mathbf{K}_p^R = 10\mathbf{I}$. We assume an identity metric $\mathbf{N} = \mathbf{I}$ for both the analytical control law which serves as the benchmark control law as well as for the cost function of the reward-weighted regression.

The experiment consists of two phases. In the first phase, we pre-train the predictor models by moving in a small region in joint-space around the rest posture. This initialization allows us to generate some initial predictor models; however, the controller models are not learned in this first part. In the second phase, we start the resolved motion rate control law on the desired trajectory and perturb its output with a very small amount of exploration $\varepsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ with $\sigma = 0.001$, i.e. $\dot{\mathbf{q}} = \pi(\mathbf{q}, \dot{\mathbf{p}}_{\text{ref}}) + \varepsilon$. This perturbation is necessary for fast learning of the resolved motion rate control law as the robot would otherwise never have a sufficiently rich set of observations. While this motor babbling is so small in magnitude that it cannot be observed in Figures 7 and 8, it nevertheless has an impact as it causes the robot to slightly drift in the null space of the task during execution, as can be observed in Figure 8.

The resulting online learning is achieved sufficiently fast so that the robot is capable of learning to track on the same trajectory that it is executing. Owing to the prediction accuracy, the learning system has already determined seven different local regions and will only learn five additional different regions during the execution of the trajectory. Altogether this trajectory requires 12 locally linear regions for accurate tracking. All of these models determine the activity of the locally linear control laws which are learned online during the execution of the trajectory. The gain of the reference attractor compensates



(a) Position of Joint 1

(b) Position of Joint 2

(c) Position of Joint 3

(d) Position of Joint 4

Fig. 8. The resulting joint-space trajectories for several DoF for both the analytical and the learned solution of the resolved motion rate control law executed on our actual Mitsubishi PA-10 robot shown in Figure 7(a). Please note that these differ slightly as the learning solution cannot oversample the state space and, thus, it does not converge to the optimal solution very quickly. In addition, the model error accumulates along the trajectory and the task-space control law needs to compensate for it. Note that the task-space trajectory performance of the learned approach is comparable to the analytical approach as presented in Figure 7. The solid curve shows the joint-space trajectory of the analytical approach while the dashed curve shows the real-time learning solution.

for initial model errors and could be reduced once the control law has been learned sufficiently well.

The resulting task-space performance can be observed in Figure 7. We can see that the resulting task-space tracking performance is quite close to that of the analytical resolved motion rate control law. In Figure 8, we can see a comparison of the joint-space trajectories of both the analytical resolved motion rate control law and the learned control law one. Both are similar throughout the trajectory, they differ due to the exploration and model errors.

5. Discussion and Conclusion

In this paper, a general learning framework for operational-space control of redundant robots has been presented, which is among the first successful attempts at learning such control laws to date. We overcome the difficulties of having non-convex solution spaces by only learning in the vicinity of a local model anchored both in joint velocity and joint position. The local regions are obtained by learning forward models, which predict the movement of the end-effector. The global consistency of the redundancy resolution of the local model controllers is ensured through minimizing the cost function of operational-space control. This cost function, derived in our previous work, is crucial to the success of this framework and its absence has most likely been the reason for the absence of learning operational-space controllers to date. The resulting learning algorithm for the local models can be understood from two perspectives, i.e. as a weighted regression problem where we intend to match the reward-weighted motor commands (after transforming the cost into a reward) or as a reinforcement learning problem where we attempt to maximize an immediate reward criterion. Throughout this paper, we have illustrated the problems and advantages of learning operational-space control using a prismatic 2 DoF robot arm as an example. As evaluations, we implemented our learning approach on a simulated 3 DoF robot arm, a simulated SARCOS Master arm with 7 DoF, and an actual Mitsubishi PA-10 robot with 6 DoF. In all cases, the robots learned to perform the desired task nearly perfectly, and demonstrated close to optimal null-space performance.

While this paper has presented a novel algorithm for learning operational space control for fully actuated robots, several open issues remain for future work in order to bring this approach to complex robots such as walking humanoid robots. First, we need to be able to deal with underactuation, which is possible in this framework as long as the number of actuated DoF exceeds the number of the DoF required for the task. While this condition can be fulfilled for many interesting tasks, its violation will result in the requirement of considering multiple-step planning solutions. Second, in this paper, we have focused on tracking control in operational space: the necessary and important issue of force control in task space has been neglected. Thus, this part will be an important topic for future research in learning operational-space control. Third, for implementations on high-DoF robots, computational requirements for implementing real-time learning grow significantly and require state-of-the-art high-speed and multiprocessor computers that are rarely used in real-time robotic setups so far owing to high-energy consumption and relatively large space requirements on an autonomous robot. Creating efficient hardware platforms for this purpose still requires major efforts, although it is technologically feasible. Finally, while the use of the function approximator LWPR Schaal et al. 2002; Vijayakumar et al. 2002, 2005) is currently the standard method for real-time function approximation and the natural choice for applying the reward-weighted regression in this framework, the usage of potentially more accurate or easier to train regression techniques such as support vector regression (Schölkopf and Smola 2002) or Gaussian process regression (Rasmussen and Williams 2006) could help significantly. Currently, these methods cannot be used for online, real-time training for an interesting robot system; however, specialized approaches such as local Gaussian process regression (Nguyen-Tuong 2007; Snelson and Ghahramani 2007) can be used in order to bring these function approximators into this new domain.

We anticipate that the true power of our suggested learning approach will become apparent when robotics increasingly moves away from the structured domains of industrial robotics towards complex robotic systems, which both are increasingly high dimensional and increasingly hard to model, such as humanoid robots. While real-time learning systems are more complex to implement, the techniques and theory developed in this paper will become crucial in the transition towards truly autonomous and self-tuning robotic systems.

References

Atkeson, C. G., Moore, A. W. and Schaal, S. (1997a). Locally weighted learning. *Artificial Intelligence Review*, **1**(5): 11–73.

Atkeson, C. G., Moore, A. W. and Schaal, S. (1997b). Locally weighted learning for control. *Artificial Intelligence Review*, **1**(5): 75–113.

Bruyninckx, H. and Khatib, O. (2000). Gauss' principle and the dynamics of redundant and constrained manipulators. *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, pp. 2563–2569.

Bullock, D., Grossberg, S. and Guenther, F. H. (1993). A self-organizing neural model of motor equivalent reaching and tool use by a multijoint arm. *Journal of Cognitive Neuroscience*, **5**(4): 408–435.

Dayan, P. and Hinton, G. E. (1997). Using expectation-maximization for reinforcement learning. *Neural Computation*, **9**(2): 271–278.

De Luca, A. and Mataloni, F. (1991). Learning control for redundant manipulators. *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Sacramento, CA.

Doty, K., Melchiorri, C., and Bonivento, C. (1993). A theory of generalized inverses applied to robotics. *International Journal of Robotics Research*, **12**: 1–19.

D'Souza, A., Vijayakumar, S. and Schaal, S. (2001). Learning inverse kinematics. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hawaii, USA.

Farrell, J. A. and Polycarpou, M. M. (2006). *Adaptive Approximation Based Control*. Hoboken, NJ, Wiley.

Guez, A. and Ahmad, Z. (1988). Solution to the inverse kinematics problem in robotics by neural networks. *Proceedings of IEEE International Conference on Neural Networks*, San Diego, CA, pp. 102–108.

Haruno, M., Wolpert, D. M. and Kawato, M. (1999). Multiple paired forward-inverse models for human motor learning and control. *Advances in Neural Information Processing Systems*. Cambridge, MA, MIT Press.

Hsu, P., Hauser, J. and Sastry, S. (1989). Dynamic control of redundant manipulators. *Journal of Robotic Systems*, **6**(2): 133–148.

Israel, A. B. and Greville, T. (2003). *Genralized Inverses: Theory and Applications*, 2nd edn. New York, Springer.

Jordan, I. M. and Rumelhart (1992). Supervised learning with a distal teacher. *Cognitive Science*, **16**: 307–354.

Kaebling, L. P., Littman, M. L. and Moore, A. W. (1996). Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, **4**: 237–285.

Kennedy, C. W. and Desai, J. P. (2005). Modeling and control of the Mitsubishi PA-10 robot arm harmonic drive system. *IEEE/ASME Transations on Mechatronics*, **10**(3): 263.

Khatib, O. (1987). A unified approach for motion and force control of robot manipulators: the operational space formulation. *IEEE Journal of Robotics and Automation*, **3**(1): 43–53.

MacKay, D. (2003). *InformationTheory, Inference, and Learning Algorithms*. Cambridge, Cambridge University Press.

Nakamura, Y. (1991). *Advanced Robotics: Redundancy and Optimization*. Boston, MA, Addison-Wesley.

Nakanishi, J., Cory, R., Mistry, M., Peters, J. and Schaal, S. (2005). Comparative experiments on task space control with redundancy resolution. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, AB.

Nakanishi, J., Farrell, J. A. and Schaal, S. (2004). Learning composite adaptive control for a class of nonlinear systems. *Proceedings of the International Conference on Robotics and Automation (ICRA)*, New Orleans, LA, pp. 2647–2652.

Nakanishi, J., Mistry, M., Peters, J. and Schaal, S. (2007). Experimental evaluation of task space position/orientation control towards compliant control for humanoid robots.

IEEE International Conference on Intelligent Robotics Systems (IROS 2007).

Nguyen-Tuong, D. (2007). Machine learning for robot motor control. *Ph.D. Thesis Proposal*, Max-Planck-Institute for Biological Cybernetics.

Park, J., Chung, W., and Youm, Y. (1999). On dynamical decoupling of kinematically redundant manipulators. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Korea, pp. 1495–1500.

Peters, J., Mistry, M., Udwadia, F., R. Cory, Nakanishi, J. and Schaal, S. (2005). A unifying methodology for the control of robotic systems. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, AB.

Peters, J. and Nguyen-Tuong, D. (2008). Real-time learning of resolved velocity control on a Mitsubishi PA-10. *Proceedings of the International Conference on Robotics and Automation (ICRA)*, submitted.

Peters, J. and Schaal, S. (2007). Reinforcement learning by reward-weighted regression for operational space control. *Proceedings of the International Conference on Machine Learning (ICML).*

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. Cambridge, MA, MIT Press.

Schaal, S., Atkeson, C. G. and Vijayakumar, S. (2002). Scalable techniques from nonparameteric statistics for real-time robot learning. *Applied Intelligence*, **17**(1): 49–60.

Schölkopf, B. and Smola, A. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. Cambridge, MA, MIT Press.

Sciavicco, L. and Siciliano, B. (2007). *Modeling and Control of Robot Manipulators*. Heidelberg, McGraw-Hill.

Sentis, L. and Khatib, O. (2005). Control of free-floating humanoid robots through task prioritization. *Proceedings of the International Conference on Robotics and Automation (ICRA)*. Barcelona, IEEE, pp. 1730–1735.

Snelson, E. and Ghahramani, Z. (2007). Local and global sparse Gaussian process approximations. *Proceedings of Artificial Intelligence and Statistics (AISTATS)*, Vol. 11.

Spall, J. C. (2003). *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Hoboken, NJ, Wiley.

Spong, M. W., Hutchinson, S. and Vidyasagar, M. (2006). *Robot Modeling and Control*. New York, Wiley.

Tevatia, G. and Schaal, S. (2000). Inverse kinematics for humanoid robots. *Proceedings of the International Conference on Robotics and Automation (ICRA)*, San Fransisco, CA.

Ting, J.-A., Mistry, M., Peters, J., Schaal, S. and Nakanishi, J. (2006). A Bayesian approach to nonlinear parameter identification for rigid body dynamics. *Proceedings of the Robotics: Science and Systems Conference*.

Udwadia, F. E. (2005). Discussions on C. F. Gauss, Gauss' principle, and its application to control. Personal communication.

Udwadia, F. E. and Kalaba, R. E. (1996). *Analytical Dynamics: A New Approach*. Cambridge, Cambridge University Press.

Vijayakumar, S., D'Souza, A. and Schaal, S. (2005). Incremental online learning in high dimensions. *Neural Computation*, **17**(12): 2602–2634.

Vijayakumar, S., D'Souza, A., Shibata, T., Conradt, J. and Schaal, S. (2002). Statistical learning for humanoid robots. *Autonomous Robots*, **12**: 59–72.