2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Reinforcement Learning for Operational Space Control

Jan Peters, Stefan Schaal
University of Southern California,
Los Angeles, CA 90089, USA

Abstract— **While operational space control 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 face of modeling errors, which are inevitable in complex robots, e.g., humanoid robots. In such cases, learning control methods can offer an interesting alternative to analytical control algorithms. However, the resulting supervised learning problem is ill-defined as it requires to learn an inverse mapping of a usually redundant system, which is well known to suffer from the property of non-convexity of the solution space, i.e., the learning system could generate motor commands that try to steer the robot into physically impossible configurations. The important insight that many operational space control algorithms can be reformulated as optimal control problems, however, allows addressing this inverse learning problem in the framework of reinforcement learning. However, few of the known optimization or reinforcement learning algorithms can be used in online learning control for robots, as they are either prohibitively slow, do not scale to interesting domains of complex robots, or require trying out policies generated by random search, which are infeasible for a physical system. Using a generalization of the EM-based reinforcement learning framework suggested by Dayan & Hinton, we reduce the problem of learning with immediate rewards to a reward-weighted regression problem with an adaptive, integrated reward transformation for faster convergence. The resulting algorithm is efficient, learns smoothly without dangerous jumps in solution space, and works well in applications of complex high degree-of-freedom robots.**

I. Introduction

In many control problems with complex robotic systems like manipulator robots and anthropomorphic robots, the control task is usually defined in external space, e.g., the Cartesian coordinates of our 3D world. In contrast, motor commands need to be generated at the level of the actuators, i.e., the space internal to the robot, as for instance joint angles. In such scenarios, the goal of control is to find the motor commands that accomplish the task while simultaneously recruiting all degrees-of-freedom of the robot in an optimal way, e.g., by minimizing the amount of movement, the amount of energy consumption, the jerkiness of movement, etc. The most advanced analytical framework to formulate such control problems is *operational space control* [1], [2]. Operational space control has numerous advantageous features [1], [2]; among the most important one is that it allows very compliant control of robots, which is one of the key ingredients that will allow robots to become a non-dangerous part of daily human life. However, the drawback of operational space control, which, so far, has limited its widespread application in robotics, is that the control law is very sensitive to the quality of system identification, i.e., the accurate knowledge of inertia matrix, internal forces and Jacobians. Small errors in these terms can result in very undesirable behavior and even lead to instability of the entire control system [3].

The goal of this paper is to propose a general policy learning[1] solution for operational space control, i.e., a method that does not require prior knowledge of any of the robot's kinematics and dynamics while yielding an accurate control policy. For this purpose, the next section will discuss how operational space control can be seen as an immediate reward reinforcement learning problem. The EM-based framework by Dayan & Hinton [5] provides a suitable starting point for deriving appropriate learning algorithms. Interestingly, reinforcement learning can be reduced to a reward-weighted nonlinear regression problem in this context, which greatly accelerates the speed of learning. This novel algorithm may also be applicable in different robot learning problems, e.g., for motor primitive learning. We evaluate our approach on simulated anthropomorphic robot arms, and compare the results of learning with analytical solutions.

II. Formulating Operational Space Control as a Reinforcement Learning Problem

Let us assume the robot dynamics can be modeled as a rigid body dynamics system

$$\mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} = \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}}, t) + \mathbf{u} \tag{1}$$

where $\mathbf{q}$,$\dot{\mathbf{q}}$,$\ddot{\mathbf{q}}$ are the generalized positions, velocities and accelerations of the robot, $\mathbf{u}$ denotes our motor commands, $\mathbf{M}(\mathbf{q})$ the inertia matrix of the robot, and $\mathbf{F}(\mathbf{q}, \dot{\mathbf{q}}, t)$ all the (potentially time t dependent) forces acting on the system (e.g., Coriolis forces, centripetal force, gravity, friction, etc.). In external space, the task is defined by a desired trajectory $\mathbf{x}_{\mathrm{d}}(t)$, $\dot{\mathbf{x}}_{\mathrm{d}}(t)$, $\ddot{\mathbf{x}}_{\mathrm{d}}(t)$, which is converted into some reference dynamics

$$\ddot{\mathbf{x}}_{\mathrm{ref}} = \ddot{\mathbf{x}}_{\mathrm{d}} + K_D(\dot{\mathbf{x}}_{\mathrm{d}} - \dot{\mathbf{x}}) + K_P(\mathbf{x}_{\mathrm{d}} - \mathbf{x}), \tag{2}$$

where $\mathbf{x} = \mathbf{x}(\mathbf{q})$ and $\dot{\mathbf{x}} = \dot{\mathbf{x}}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}}$ denote the forward kinematics and differential kinematics of the robot end-effector, respectively, and $\mathbf{J}$ the Jacobian of the

[1]Trajectory learning approaches which only work on a specific trajectory while incorporating knowledgee of the system have previously been discussed in the literature, e.g., see [4].

end-effector kinematics. The resulting control law of such a problem are usually formulated in the form of

$$\mathbf{u} = \mu\left(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\text{ref}}, \mathbf{u}_0\right) \tag{3}$$

and, for instance, could look like

$$\mathbf{u} = \mathbf{J}^T(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T)^{-1}(\ddot{\mathbf{x}}_{\text{ref}} - \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}\mathbf{M}^{-1}\mathbf{F}). \tag{4}$$

As stated in Eq.(3), learning operational space control is equivalent to obtaining a mapping $\mathbf{s} = (\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\text{ref}}, \mathbf{u}_0) \rightarrow \mathbf{u} = \mu(\mathbf{s})$ from sampled data using a function approximator. As the dimensionality of the task-space reference behavior $\mathbf{x}_{\text{ref}}$ is lower than that of the motor commands $\mathbf{u}$, infinitely many solutions $\mathbf{u}$ exist for a given $\mathbf{s}$. The solution space of motor commands u achieving the same reference acceleration $\ddot{\mathbf{x}}_{\text{ref}}$ does usually not form a convex set in most robots, a problem first described in the context of learning inverse kinematics [10], [14]. Thus, when learning $\mathbf{s} \rightarrow \mathbf{u}$ as a function approximation or supervised learning problem, the learning algorithm can create physically invalid solutions.

In order to be able to address operational space control as a learning problem, an associated cost function is required that, when optimized, results in an appropriate operational space controller, e.g., Equation (4). A key insight of recent work [6] was that a large class of operational space control laws can be derived as the solution of a constrained immediate reward optimization problem:

$$\begin{aligned} r(\mathbf{u}) &= -(\mathbf{u} - \mathbf{u}_0)^T \mathbf{N}(\mathbf{u} - \mathbf{u}_0) \\ \text{s.t. } \mathbf{J}\ddot{\mathbf{q}} &= \ddot{\mathbf{x}}_{\text{ref}} - \dot{\mathbf{J}}\dot{\mathbf{q}}, \end{aligned} \tag{5}$$

where $\mathbf{u}_0$ is the default or nominal stabilizing behavior, e.g., a force which pulls the robot towards a static rest posture

$$\mathbf{u}_0 = -\mathbf{K}_D\dot{\mathbf{q}} - \mathbf{K}_P(\mathbf{q} - \mathbf{q}_{\text{rest}}), \tag{6}$$

and $\mathbf{u}_1 = \mathbf{u} - \mathbf{u}_0$ corresponds to the control signal that accomplishes the desired task goal, characterized by $\ddot{\mathbf{x}}_{\text{ref}}$. $\mathbf{N}$ is a positive definite metric that decides the relative importance of the motor commands in the optimization. If an accurate model is available, the general analytic solution to this optimization problem is given by

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

where the second summand fulfills the nominal control law $\mathbf{u}_0$ in the null-space of the first term. For example, Equation (4) is derived for $\mathbf{N} = \mathbf{M}^{-1}$ and $\mathbf{u}_0 = \mathbf{0}$. Of course, this solution is only interesting when the dimensionality of $\ddot{\mathbf{x}}_{\text{ref}}$ is smaller than that of $\mathbf{u}$, which we assume in all the following.

The conclusion of this brief summary is that operational space control can be viewed as an immediate reward reinforcement learning problem [7] with high-dimensional, continuous states $\mathbf{s} = [\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\text{ref}}, \mathbf{u}_0] \in \mathbb{R}^n$ and actions $\mathbf{u} \in \mathbb{R}^m$. The goal of learning is to obtain an optimal policy $\mathbf{u} = \mu(\mathbf{s})$ such that the system follows the reference acceleration $\ddot{\mathbf{x}}_{\text{ref}}$ while maximizing the immediate reward $r(\mathbf{u}) = -(\mathbf{u} - \mathbf{u}_0)^T\mathbf{N}(\mathbf{u}-\mathbf{u}_0)$ for any given nominal behavior $\mathbf{u}_0$. In order to incorporate exploration during learning, we need a stochastic control policy $\mathbf{u} = \mu_\theta(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\text{ref}}) + \varepsilon$, modeled as a probability distribution $\pi_\theta(\mathbf{u}|\mathbf{s}) = p(\mathbf{u}|\mathbf{s}, \theta)$ with parameter vector θ. The goal of the learning system is thus to find the policy parameters θ that maximize

$$J_r(\theta) = \int p(\mathbf{s}) \int \pi_\theta(\mathbf{u}|\mathbf{s})\, r(\mathbf{s}, \mathbf{u})\, d\mathbf{u} d\mathbf{s}. \tag{8}$$

$p(\mathbf{s})$ denotes the distribution of states, which is treated as fixed in immediate reward reinforcement learning problems [7].

III. Reinforcement Learning by Reward-Weighted Regression

Previous work in the literature suggested a variety of optimizing methods which can be applied to immediate reward reinforcement learning problems, e.g., 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) [5], [7], [8]. 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, often not suitable 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 & Hinton [5]. As we will show in the following, a generalization of this approach allows us to derive an EM-algorithm which essentially reduces the immediate reward learning problem to a reward-weighted regression problem.

A. Reward Transformation

In order to maximize the expected return (Eq. 8) using samples, we approximate

$$J_r(\theta) \approx \sum\nolimits_{i=1}^{n} \pi_\theta(\mathbf{u}_i|\mathbf{s}_i)\, r_i \tag{9}$$

where $r_i = r(\mathbf{s}_i, \mathbf{u}_i)$. For application of the probabilistic reinforcement learning framework of Dayan & Hinton [5], 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 for bounded rewards, for unbounded rewards as discussed in this paper, a nonlinear 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 $\int_0^\infty U_\tau(r)\, dr = \text{const}$, resulting in the transformed optimization problem

$$J_u(\theta) = \sum\nolimits_{i=1}^{n} \pi_\theta(\mathbf{u}_i|\mathbf{s}_i)\, U_\tau(r_i). \tag{10}$$

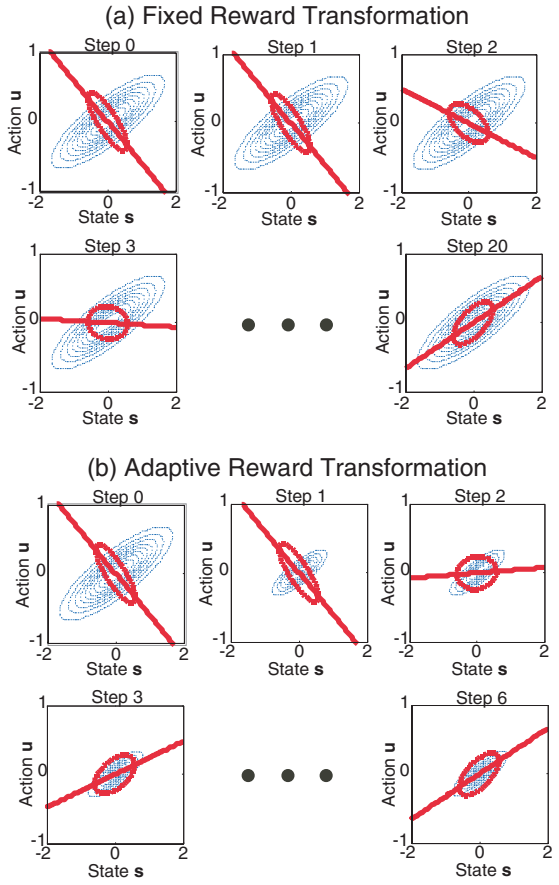

Fig. 1. A comparison of fixed and adaptive reward transformation for learning a linear policy $\pi(a|s) = N(a|\theta_1 s + \theta_2, \sigma^2)$ under the transformed reward $u(r(s,a)) = \exp\left(-\tau\left(q_1 a^2 + q_2 as + s q_3^2\right)\right)$. The transformed reward is indicated by the dotted blue ellipses, the variance of the action distribution is indicated by the red thick ellipse, and the mean of the linear policy is shown by the red 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.

The reward transformation plays a more important role than initially meets the eye: as already pointed out in [5], convergence speed can be greatly affected by this transformation. Making $U_\tau(r)$ an adaptive part of the learning algorithm by means of some internal parameters τ can greatly accelerate the learning speed and help avoid local minima during learning. Figure 1 demonstrates this issue with a 1D continuous state and 1D continuous action example, where the goal is to learn an optimal linear policy. Using the algorithm that we will introduce below, an adaptive reward transformation accelerated the convergence by a factor of 4, and actually significantly helped avoiding local minima during learning.

B. EM-Reinforcement Learning with Adaptive Reward Transformation

To derive our learning algorithm, similar as in [5], we start by establishing the lower bound

$$\log J_u(\theta) \tag{11}$$

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

$$\geq \sum_{i=1}^{n} q(i) \log \frac{\pi_\theta(\mathbf{u}_i|\mathbf{s}_i) U_\tau(r_i)}{q(i)} \tag{12}$$

$$= \sum_{i=1}^{n} q(i) \left[\log \pi_\theta(\mathbf{u}_i|\mathbf{s}_i) + \log U_\tau(r_i) - \log q(i)\right] \tag{13}$$

$$= \mathcal{F}(q, \theta, \tau), \tag{14}$$

due to Jensens inequality. The re-weighting distribution $q(i)$ obeys the constraint

$$\sum_{i=1}^{n} q(i) - 1 = 0. \tag{15}$$

The resulting EM algorithm is given below.

Algorithm 1: An EM algorithm for optimizing both the expected reward as well as the reward-transformation is given by an **E-Step**

$$q_{k+1}(j) = \frac{\pi_{\theta_k}(\mathbf{u}_j|\mathbf{s}_j) U_{\tau_k}(r_j)}{\sum_{i=1}^{n} \pi_\theta(\mathbf{u}_i|\mathbf{s}_i) U_{\tau_k}(r_i)}, \tag{16}$$

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

$$\theta_{k+1} = \arg\max_\theta \sum_{i=1}^{n} q_{k+1}(i) \log \pi_\theta(\mathbf{u}_i|\mathbf{s}_i), \tag{17}$$

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

$$\tau_{k+1} = \arg\max_\tau \sum_{i=1}^{n} q_{k+1}(i) \log U_\tau(r_i). \tag{18}$$

Proof: The E-Step is given by

$$q_{k+1} = \arg\max_q \mathcal{F}(q, \theta, \tau) \tag{19}$$

while fulfilling the constraint

$$0 = \sum_{i=1}^{n} q(i) - 1. \tag{20}$$

Thus, we obtain a constrained optimization problem with Lagrange multiplier λ:

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

Optimizing $L(\lambda, q)$ with respect to q and λ results in Eq.(16). Optimizing $\mathcal{F}(q_{k+1}, \theta, \tau)$ with respect to θ and τ yields Eqs.(17, 18). ■

C. Reinforcement Learning by Reward-Weighted Regression

Let us assume the specific class of normally distributed policies:

$$\pi_\theta(\mathbf{u}|\mathbf{s}) = \mathcal{N}\left(\mathbf{u}|\mu_\theta(\mathbf{s}), \sigma^2\mathbf{I}\right) \tag{21}$$

with a nominal or mean behavior $\mu_\theta(\mathbf{s}) = \phi(\mathbf{s})^T \theta$ where $\phi(\mathbf{s})$ denotes some fixed preprocessing of the state by basis functions and $\sigma^2\mathbf{I}$ determines the exploration[2]. Furthermore, we choose the reward transformation

$$U_\tau(r) = \tau \exp(-\tau r), \tag{22}$$

[2]Note that $\sigma^2\mathbf{I}$ could be replaced by a full variance matrix with little changes in 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 our requirements on a reward transformation (cited from Sec.III-A). Algorithm 1 thus becomes:

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

$$\theta_{k+1} = \left(\mathbf{\Phi}^T\mathbf{W}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{W}\mathbf{Y}, \tag{23}$$

$$\sigma^2_{k+1} = \left\|\mathbf{Y} - \theta^T_{k+1}\mathbf{\Phi}\right\|^2_{\mathbf{W}}, \tag{24}$$

where

$$\mathbf{W} = \left(\sum_{i=1}^{n} U_\tau(r_i)\right)^{-1} \operatorname{diag}\left(U_\tau(r_1), U_\tau(r_2), \ldots, U_\tau(r_n)\right), \tag{25}$$

denotes a diagonal matrix with transformed rewards,

$$\mathbf{\Phi} = \left[\phi(\mathbf{s}_1), \phi(\mathbf{s}_2), \ldots, \phi(\mathbf{s}_n)\right]^T, \tag{26}$$

and

$$\mathbf{Y} = \left[\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\right]^T \tag{27}$$

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

$$\tau_{k+1} = \frac{\sum_{i=1}^{n} U_{\tau_k}(r_i)}{\sum_{i=1}^{n} U_{\tau_k}(r_i)\, r_i}. \tag{28}$$

Proof: When computing $q_{k+1}(j)$ from samples in Eq.(16), we have

$$q_{k+1}(j) = \frac{U_{\tau_k}(r_j)}{\sum_{i=1}^{n} U_{\tau_k}(r_i)} \tag{29}$$

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

$$\pi_\theta(\mathbf{u}|\mathbf{s}) = \left(2\pi\sigma^2\right)^{-\frac{d}{2}}\exp\left(-\frac{(\mathbf{u} - \phi(\mathbf{s})^T\theta)^T(\mathbf{u} - \phi(\mathbf{s})^T\theta)}{2\sigma^2}\right),$$

into Equation (17). By differentiating with respect to θ and equating the result to zero, we obtain

$$\theta = \left(\sum_{i=1}^{n} q_{k+1}(i)\,\phi(\mathbf{s}_i)\,\phi(\mathbf{s}_i)^T\right)^{-1}\left(\sum_{i=1}^{n} q_{k+1}(i)\,\phi(\mathbf{s}_i)\,\mathbf{u}_i\right).$$

In matrix vector form, this corresponds to Eq.(23). Analogously, the reward transformation is obtained from differentiation with respect to τ as

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

which results in Eq.(28). ■

IV. Learning Setup for Operational Space Control

We are now in the position to return to our original objective of learning operational space control. As stated previously, the learning operational space control is equivalent to obtaining a mapping

$$\mathbf{s} = (\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\text{ref}}, \mathbf{u}_0) \rightarrow \mathbf{u} \tag{30}$$

from sampled data using a function approximator. However, due to the difference in dimensionality of task-space and action-space, infinitely many solutions exist but usually do not form a convex set in most robot. Thus, when learning $\mathbf{s} \rightarrow \mathbf{u}$ as a function approximation problem, the learning algorithm can create physically invalid solutions. Nevertheless, the non-convexity issue can be resolved by two insights. The first insight was discussed in Section II where we reformulated operational space control as an immediate reinforcement learning problem.

The second insight is that the non-convexity issue can be addressed locally by employing a spatially localized supervised learning system, an approach that was first introduced in the context of inverse kinematics learning [9], [10]. The feasibility of this idea can be demonstrated by re-writing Eq.(4) in its proper functional form, i.e., not as an inverse function:

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

If we partition the state space of the robot, spanned by $\mathbf{q}, \dot{\mathbf{q}}$, into regions were $\mathbf{q}, \dot{\mathbf{q}}$ is approximately constant, the average over all solutions resulting in a particular $\ddot{\mathbf{x}}_{\text{ref}}$ can be written as:

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

Thus, in the vicinity of same $\mathbf{q}$,$\dot{\mathbf{q}}$ all possible $\mathbf{u}$ that achieve the same $\ddot{\mathbf{x}}_{\text{ref}}$ form a convex solution set, since any average over different solutions $\mathbf{u}_1$, $\mathbf{u}_2$, ..., will be guaranteed to still achieve the given $\ddot{\mathbf{x}}_{\text{ref}}$ [3]. Consequently, our approach to learning operational space control will partition the control law in the form of locally linear controllers

$$\mathbf{u}^i = [\ddot{\mathbf{x}}^T_{\text{ref}}, \dot{\mathbf{q}}^T, 1]\beta^i, \tag{31}$$

which are active only in a region around a particular $\mathbf{q}^i, \dot{\mathbf{q}}^i$ (note that we added constant input in Eq. (31) to account for the intercept of a linear function). From a control engineering point of view, this argument corresponds to the insight that nonlinear control can often be accomplished through local linearizations at the point of interest, and that, 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:

$$\ddot{\mathbf{x}}^i = [\dot{\mathbf{q}}^T, \mathbf{u}^T, 1]\hat{\beta}^i, \tag{32}$$

Learning this forward model is a standard supervised learning problem, as the mapping is guaranteed to be a proper function. A method of learning such a forward model that automatically

[3]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 degrees of freedom i, j; this, however, is not necessarily desirable as it will add new inputs to the local regression problem whose number grows quadratically with the number of degrees of freedom.

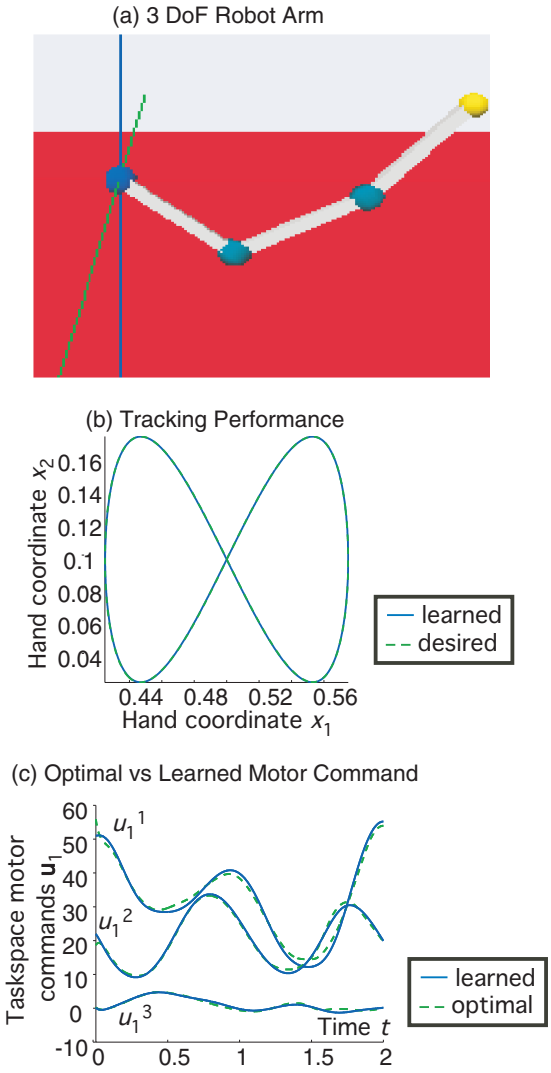

Fig. 2. (a) screen shot of the 3 DOF arm simulator, (b) near ideal tracking performance for a planar figure-8 pattern for the 3 DOF arm, and (c) a comparison between the analytically obtained optimal control commands to the learned ones for one figure-8 cycle of the 3DOF arm exhibits that a near-optimal policy is obtained.

also learns a local linearization is Locally Weighted Projection Regression (LWPR) [11], a fast online learning method which 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. The membership to a local model is determined by a weight generated from a Gaussian kernel:

$$w^i(\mathbf{q}, \dot{\mathbf{q}}) = \exp\left(-\frac{1}{2}\left(\begin{bmatrix}\mathbf{q}\\ \dot{\mathbf{q}}\end{bmatrix} - \mathbf{c}^i\right)^T \mathbf{D}^i \left(\begin{bmatrix}\mathbf{q}\\ \dot{\mathbf{q}}\end{bmatrix} - \mathbf{c}^i\right)\right) \tag{33}$$

centered at a fixed $\mathbf{c}_i$ in $(\mathbf{q}, \dot{\mathbf{q}})$-space, and shaped by a diagonal distance metric $\mathbf{D}_i$. The Gaussian kernel allow the combination of the different local controllers using

$$\mathbf{u} = \frac{\sum_{i=1}^n w^i(\mathbf{q}, \dot{\mathbf{q}})\, [\ddot{\mathbf{x}}_{\text{ref}}^T, \dot{\mathbf{q}}^T, 1]\beta^i}{\sum_{i=1}^n w^i(\mathbf{q}, \dot{\mathbf{q}})}. \tag{34}$$

For a closer description of this statistical learning algorithm see [11].

As learning the forward model provides a suitable partitioning of the robot's state space into regions of local linearity, we re-use this partitioning to learn a local controller in each partition, an approach that resembles several previous forward-inverse model learning approaches [12], [13]. The local controllers are learned using the reward-weighted regression approach described in Section III, with metric $\mathbf{N} = \mathbf{I}$ in Eq.5 – the reinforcement learning ensured that all local controllers learned a globally consistent solution to the operational space control task. Each local controller maintained its own adaptive reward transformation and associated parameter τ_i. The nominal control law $\mathbf{u}_0$ is learned separately; it consists of a gravity compensation component which is obtained directly with LWPR by supervised learning and a joint-space force pushing the robot towards a rest posture as explained in the introduction.

V. Evaluations

We evaluated our approach on two different simulated, physically realistic robots: (i) a three degree-of-freedom (DOF) planar robot arm shown in Figure 2 (a) and (ii) a seven DOF simulated SARCOS master robot arm – an implementation on the real, physical SARCOS master robot arm (Figure 3 (a)) is currently in progress.

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 PD control law. This "motor babbling" exploration was necessary in order bootstrap learning with some initial data, as we would otherwise experience rather slow learning, as typically observed in similar direct-inverse learning approaches [14]. The measured end-effector accelerations served as desired acceleration in Eq.31, 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 2 (b) and 3 (b). For the three DOF arm, we verified the quality of the learned control commands in comparison to the analytical solution, given in Eq.(7): Figure 2 (c) demonstrates that the motor commands of the learned and analytically optimal case are almost identical. Learning results of the simulated seven DOF Sarcos robot achieved almost the same end-effector tracking quality and is shown in Figure 2 (c). It exhibits only slightly increased errors, however, the joint commands were not quite as close to the optimal ones 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

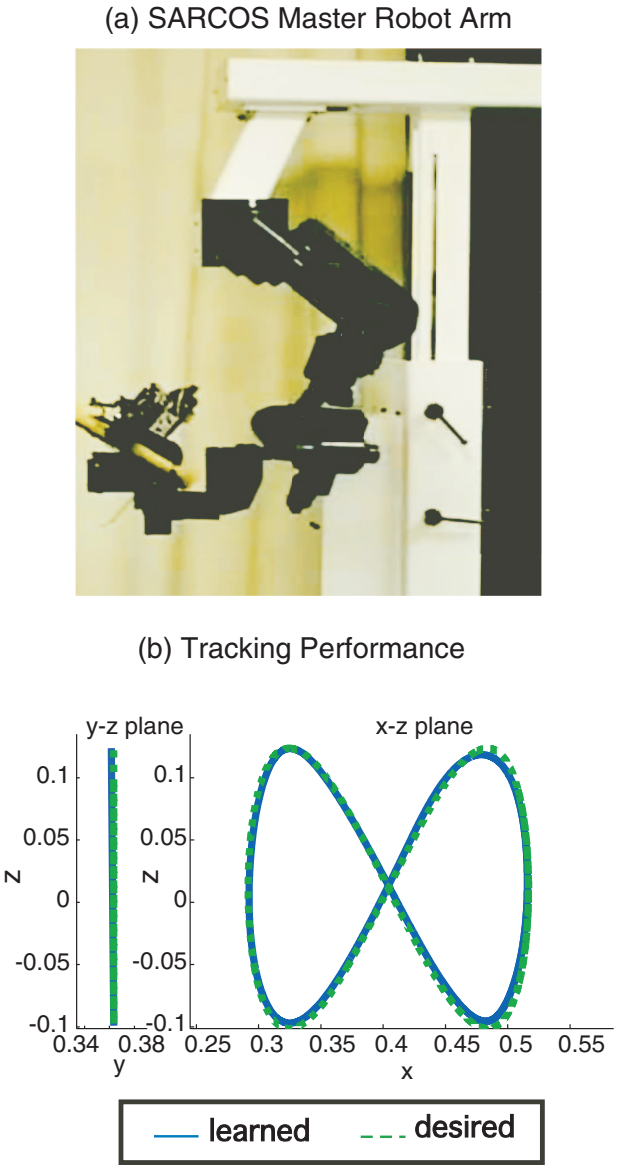

Fig. 3. (a) Anthropomorphic Sarcos Master Arm, used as simulated system and in progress of actual robot evaluations. (b) Tracking performance for a planar figure-8 pattern for the simulated Sarcos Master arm.

the LWPR learning algorithm to achieve local linearizations with very high accuracy and with enough data to find the optimal solution. The 3 DOF required about 2 hours of real-time training, while setup was optimized for the 7 DOF arm where 60 minute run of real-time training was sufficient for achieving the quality exhibited on the test trajectory in Figure 3 (b).

The implementation on the real SARCOS Master Arm is in progress. To date, we have shown that solutions which were pre-trained using data obtained with the real robot allow the offline learning of a controller that works in the SARCOS Master Arm simulator. However, some communication delays between the learning system hosted on an external PowerPC G5 and the real-time system need to be reduced in order to complete the robot experiment which is expected to happen in the next 1-2 months.

VI. Conclusion

This paper contributes in two different ways to advancing the state-of-the-art of learning control. First, we introduced a framework for learning operational space control, a type of controller that has found little practical realizations due to problems with system identification in actual complex robots. Our learning methods avoid system identification entirely. Second, we introduced the idea of reinforcement learning by reward-weighted regression. While we realized this method here for immediate reward problems, i.e., finding optimal solutions in resolving redundancy in operational space control, we believe that there are much broader applications also in the realm of temporally delayed rewards, in particular for learning from trajectories or roll-outs. Reinforcement learning by reward-weighted regression has some of the flavor that was envisioned for modern approaches to reinforcement learning, i.e., the transformation of the reinforcement learning problem into an efficient supervised learning problem. We demonstrated the success of our approach on implementations on complex robot simulations, which will be followed by actual robot implementations in the near future.

References

[1] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal of Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[2] R. Featherstone, *Robot dynamics algorithms*. Kluwer Academic Publishers, 1987.

[3] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Comparative experiments on task space control with redundancy resolution," in *IEEE International Conference on Intelligent Robots and Systems*, 2005.

[4] A. D. Luca and F. Mataloni, "Learning control for redundant manipulators," in *Proceedings of the International Conference in Robotics and Automation (ICRA)*, 1991.

[5] P. Dayan and G. E. Hinton, "Using expectation-maximization for reinforcement learning," *Neural Computation*, vol. 9, no. 2, pp. 271–278, 1997. [Online]. Available: citeseer.ist.psu.edu/dayan97using.html

[6] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *IEEE International Conference on Intelligent Robots and Systems (IROS 2001)*, 2001.

[7] I. M. Jordan and Rumelhart, "Supervised learning with a distal teacher," in *Cognitive Science*, vol. 16, 1992, pp. 307–354.

[8] J. Peters, M. Mistry, F. Udwadia, and S. Schaal, "A unifying methodology for the control of robotic systems," in *IEEE International Conference on Intelligent Robots and Systems*, 2005.

[9] L. P. Kaebling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

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

[11] D. Bullock, S. Grossberg, and F. H. Guenther, "A self-organizing neural model of motor equivalent reaching and tool use by a multijoint arm," *Journal of Cognitive Neuroscience*, vol. 5, no. 4, pp. 408–435, 1993.

[12] S. Schaal, C. G. Atkeson, and S. Vijayakumar, "Scalable techniques from nonparameteric statistics for real-time robot learning," *Applied Intelligence*, vol. 17, no. 1, pp. 49–60, 2002.

[13] M. Haruno, D. M. Wolpert, and M. Kawato, "Multiple paired forward-inverse models for human motor learning and control," in *Advances in Neural Information Processing Systems 11*. Cambridge, MA: MIT Press, 1999.

[14] M. Kawato and D. Wolpert, "Internal models for motor control," *Novartis Found Symp*, vol. 218, pp. 291–304, 1998.