Robotics: Science and Systems 2006
Philadelphia, PA, USA, August 16-19, 2006

Learning Operational Space Control

Jan Peters, Stefan Schaal
Departments of Computer Science & Neuroscience
University of Southern California Los Angeles, CA 90802
Email: {jrpeters, sschaal }@usc.edu

***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 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. A first important insight for this paper is that, nevertheless, a physically correct solution to the inverse problem does exit when learning of the inverse map is performed in a suitable piecewise linear way. The second crucial component for our work is based on a recent insight that many operational space controllers can be understood in terms of a constraint 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 view of machine learning, the learning problem corresponds to a reinforcement learning problem that maximizes an immediate reward and that employs an expectation-maximization policy search algorithm. Evaluations on a three degrees of freedom robot arm illustrate the feasibility of the suggested approach.**

I. Introduction

Operational space control is one of the most elegant approaches to task control due 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 [1], [2] up to balancing and gait execution for humanoid robots [3]. 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 [4]. However, particularly if compliant (i.e., low-gain) control is desired, 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. But learning methods do not easily provide the highly structured knowledge required in traditional operational space control laws, i.e., Jacobians, inertia matrices, and Coriolis/centripetal and gravity forces, as all these terms are not observable and are therefore not suitable for formulating supervised learning as traditionally used in learning control approaches [5].

In this paper, we will 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 will proceed as follows: firstly, we will review operational space control and discuss where learning can be beneficial. Secondly, we will 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 robot arm.

A. Notation and Remarks

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

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

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 $\boldsymbol{\varepsilon}(\mathbf{q},\dot{\mathbf{q}})$ denotes unmodeled nonlinearities.

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 manipulator robot arm where position and orientation of the end-effector are controlled [1], [2]; 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 [3]. 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.,

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

[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 will omit such kind of tasks in this paper.

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

B. Operational Space Control as an Optimal Control Problem

Using the framework of trajectory tracking as an example, the general problem in operational space control[1] can be described as follows: generate a control law $\mathbf{u} = \mathbf{f}_{\text{Control}}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)$ which controls the robot along a joint space trajectory $\mathbf{q}(t), \dot{\mathbf{q}}(t)(t)$ such that the controlled element (e.g., the end-effector) follows a desired trajectory in task space $\mathbf{x}_d(t), \dot{\mathbf{x}}_d(t), \ddot{\mathbf{x}}_d(t)$. This problem has been thoroughly discussed since the late 1980s (e.g., [1], [2]) and, among others, has resulted in a class of well-known control laws [4]. As an important new insight into operational space control it was recently discovered [6], that many of the suggested controllers in the literature can be derived as the solution of a constraint optimization problem given by

$$\min_{\mathbf{u}} C_0(\mathbf{u}) = \mathbf{u}^T \mathbf{N} \mathbf{u} \text{ s.t. } \mathbf{J}\ddot{\mathbf{q}} = \ddot{\mathbf{x}}_{\text{ref}} - \dot{\mathbf{J}}\dot{\mathbf{q}}, \tag{3}$$

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 [6]

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

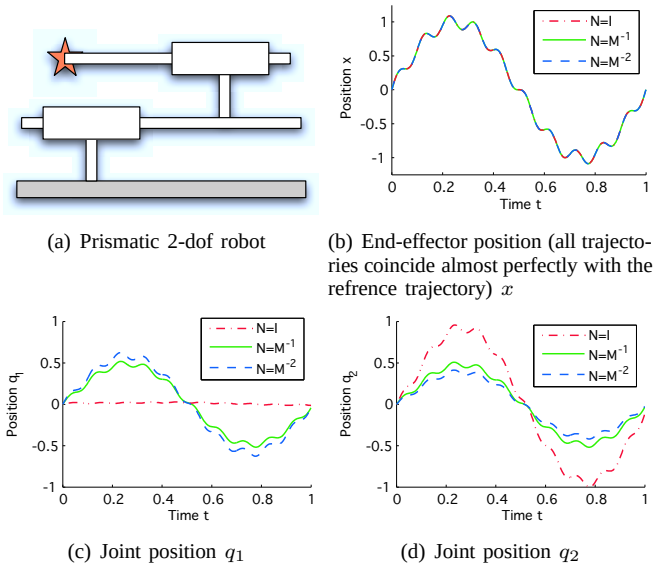
with $\mathbf{F}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \boldsymbol{\varepsilon}(\mathbf{q}, \dot{\mathbf{q}})$, and the notation $\mathbf{D}^{+}$ defining the pseudo inverse of a matrix such that $\mathbf{D}^{+}\mathbf{D} = \mathbf{I}$, $\mathbf{D}\mathbf{D}^{+} = \mathbf{I}$, 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. [2] (without null space optimization) is the result of using the metric $\mathbf{N} = \mathbf{M}^{-2}$, which yields $\mathbf{u} = \mathbf{M}\mathbf{J}^T(\ddot{\mathbf{x}}_{\text{ref}} - \dot{\mathbf{J}}\dot{\mathbf{q}}) + \mathbf{F}$, 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 [1], determined by the metric $\mathbf{N} = \mathbf{M}^{-1}$ and given by

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

Khatib's solution is special as the metric $\mathbf{N} = \mathbf{M}^{-1}$ is the only metric which generated torques that correspond to the ones created by a physical contraint pulling the robot along the trajectory [6], [7], i.e., it is the metric used by nature according to Gauss' principle [7], [8] and it is invariant under change of joint coordinates [9]. Other metrics such as $\mathbf{N} = \text{const}$ can be used to distribute the required forces differently, e.g., such that stronger motors get a higher portion of the generated forces [6].

Even when achieving the task perfectly, the joint-space trajectories can result into unfavorable postures or even joint-space instability (see Example 1 below). For handling such



(a) Prismatic 2-dof robot

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

(c) Joint position q_1

(d) Joint position q_2

Fig. 1. When applied to the prismatic robot from Example 1 shown in (a), the three control laws for the metrics $\mathbf{N} = \mathbf{I}$ (dashed-dot red lines), $\mathbf{N} = \mathbf{M}^{-1}$ (solid green), $\mathbf{N} = \mathbf{M}^{-2}$ (dashed blue) result in (b) the same task-space tracking but (c,d) very different joint-space behavior. See Example 1 for more information.

cases, additional controls which do not affect the tasks 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 which pulls 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 constraint optimization problem

$$\min_{\mathbf{u}} C_1(\mathbf{u}) = (\mathbf{u} - \mathbf{u}_0)^T \mathbf{N}(\mathbf{u} - \mathbf{u}_0) \text{ s.t. } \mathbf{J}\ddot{\mathbf{q}} = \ddot{\mathbf{x}}_{\text{ref}} - \dot{\mathbf{J}}\dot{\mathbf{q}}, \tag{6}$$

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

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

where the second summand fulfill 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 [3], [6].

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 mass matrix will by $\mathbf{M} = \text{diag}(m_1, 0) + m_2\mathbf{1}$ with masses $m_1 = m_2 = 1$ and $\mathbf{1}$ denoting a matrix with all coefficients equal to one. The internal forces are $\mathbf{F} = 0$, the Jacobian is $\mathbf{J} = [1, 1]^T$ and its derivative $\dot{\mathbf{J}} = \mathbf{0}$. If no joint-space control law is selected, i.e., $\mathbf{u}_0 = 0$, the control law in the form of Equation (4) 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 into unstable behavior for most metrics $\mathbf{N}$. 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 with very different properties as can

be observed in Figure 1: (i) metric $\mathbf{N} = \mathbf{I}$ *will result into 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, while (iii) metric* $\mathbf{N} = \mathbf{M}^{-2}$ *simply minimizes the squared acceleration.*

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

C. Why should we learn Operational Space Control?

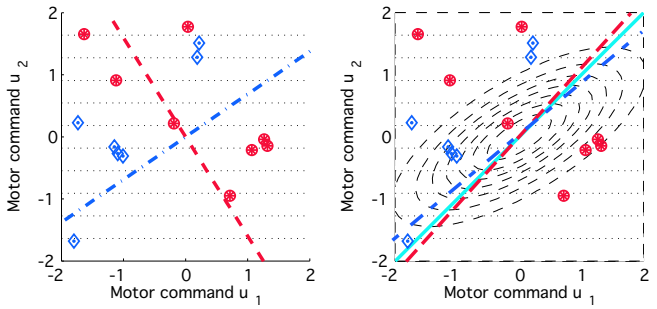
When an accurate analytical model of the robot is available and its parameters can be well-estimated, operational space control laws can be highly successful [1], [3], [4]. However, in many new complex robotic systems, e.g., humanoid robots, space robots, etc., accurate analytical models of the robot dynamics are not available due to significant depatures from idealized theoretical models such as rigid body dynamics. For instance, in our experience with anthropomorphic robots, unmodeled nonlinear effects were caused by complex hydraulic actuator dynamics, hydraulic hoses and cable bundles routed along the light weight structure of the robot as well as complex friction effects. Trying to model such nonlinearities is of little use due 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 nonlinearities can have a drastic effect. If the estimated mass matrix of the robot $\tilde{\mathbf{M}} = \mathrm{diag}\,(m_1, 0) + m_2\mathbf{1}$ *just 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 cables, then the resulting control law will result in unstable and unpredictable null-space behavior despite 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 also task space tracking degrades, and the entire control system goes 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 light-weight robot arms or full-body humanoid robots, such problems become even more frequent and difficult to cope with. Traditionally, this problem is be fixed by the engineer improving the approximation the plant by hand; however, for operational space control of low-gain controlled light-weight robots which are hard to model, learning is a promising novel alternative and will be discussed in Section II.

II. 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 focussed mostly on an inverse kinematics end-effector control [10]–[14], 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. The combination of a learned inverse kinematics and a learned inverse dynamics controller [10], [13], [14] 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.



(a) Unweighted datasets (b) Reward weighted datasets

Fig. 2. This figure illustrates how (a) different data sets result in different solutions if each data point is treated with equal importance (the blue dash-dot line corresponds to the blue diamonds and the red dashed line to the red circles). If these data points are (b) weighted down using the Gaussian cost function (here indicated with the metric $\mathbf{N} = \mathbf{M}^{-1}$ as solid thin black lines) the solutions of different data sets will consistently approximate optimal solutions shown in the solid cyan line. While for the linear prismatic robot one could live with any solution in (a), different local solutions have to create a consistent global solution for nonlinear robots. The horizontal faintly dotted lines in (a) and (b) indicate contour lines of equal task-space acceleration.

A. Can Operational Space Control be learned?

Learning operational space control is equivalent to obtaining a mapping $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\text{ref}}) \rightarrow \mathbf{u}$ 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 the one of motor command $\mathbf{u}$, there are infinitely many solutions for $\mathbf{u}$ for most joint positions $\mathbf{q}$, and joint velocities $\dot{\mathbf{q}}$. For the illustrative linear case in Example 2 without a null-space component, this mapping corresponds to a line in the plane of possible control laws as shown by the two lines in Figure 2(a).

A major problem arises in the case of a robot with rotary joints as the motor commands $\mathbf{u}$ achieving the same reference acceleration $\ddot{\mathbf{x}}_{\text{ref}}$ are no longer form a convex set, a problem first described in the context of learning inverse kinematics [11], [14]. Thus, when learning the inverse mapping $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\text{ref}}) \rightarrow \mathbf{u}$, the learning algorithm will average over unconnected sets of the solutions which can result in invalid solutions to the learning problem. Therefore, the learning problem is ill-conditioned such that directly 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, which, in our case, needs to spatially localized based on both joint space position and velocity – such an approach was first introduced in the context of inverse kinematics learning [12], [14]. The feasibility of this idea can be demonstrated simply by averaging over the combination of Equations (2) and (1)

which yields that by averaging over the exact same spatial position $\mathbf{q}$, and velocity $\dot{\mathbf{q}}$, we have

$$\begin{aligned}\overline{\ddot{\mathbf{x}}} &= \langle \ddot{\mathbf{x}} \rangle = \left\langle \mathbf{J}\mathbf{M}^{-1}\left(\mathbf{u}+\mathbf{F}\right) + \dot{\mathbf{J}}\dot{\mathbf{q}} \right\rangle \\ &= \mathbf{J}\mathbf{M}^{-1}\left\langle \mathbf{u}+\mathbf{F} \right\rangle + \dot{\mathbf{J}}\dot{\mathbf{q}} = \mathbf{J}\mathbf{M}^{-1}\left(\overline{\mathbf{u}}+\mathbf{F}\right) + \dot{\mathbf{J}}\dot{\mathbf{q}},\end{aligned} \tag{8}$$

i.e., in the vicinity of same $\mathbf{q}$,$\dot{\mathbf{q}}$, a particular $\overline{\ddot{\mathbf{x}}}$ will always correspond to exactly one particular $\overline{\mathbf{u}}$ [2]. Therefore, locally linear controllers

$$\mathbf{u}^i = \mathbf{c}^i_{\boldsymbol{\beta}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\text{ref}}) = [\ddot{\mathbf{x}}^T_{\text{ref}}, \dot{\mathbf{q}}^T, 1]\boldsymbol{\beta}^i, \tag{9}$$

can be used if they are only active in a region around $\mathbf{q}, \dot{\mathbf{q}}$ (note that we added constant input in Equation (9) to account for the intercept of a linear function). 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, and, in general, linear system 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

$$\ddot{\mathbf{x}}^i = \mathbf{p}^i_{\hat{\boldsymbol{\beta}}}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}) = [\dot{\mathbf{q}}^T, \mathbf{u}^T, 1]\hat{\boldsymbol{\beta}}^i, \tag{10}$$

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 also learns a local linearization is Locally Weighted Projection Regression (LWPR) [15], 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:

$$w^i(\boldsymbol{q}, \dot{\mathbf{q}}) = \exp\left(\frac{1}{2}\left(\begin{bmatrix}\mathbf{q}\\ \dot{\mathbf{q}}\end{bmatrix} - \mathbf{c}_i\right)^T \mathbf{D}^i \left(\begin{bmatrix}\mathbf{q}\\ \dot{\mathbf{q}}\end{bmatrix} - \mathbf{c}_i\right)\right) \tag{11}$$

centered at $\boldsymbol{c}_i$ in $(\boldsymbol{q}, \dot{\mathbf{q}})$-space, and shaped by a distance metric $\boldsymbol{D}_i$. For a closer description of this statistical learning algorithm see [15].

For each local forward model created by LWPR, we automatically create a local controller. This approach of pairwise combining predictors and controllers is related by the MOSAIC architecture [16] where the quality of predicting a task is used for selecting which local controller should be used for the task.

[2]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 which grows quadratically with the number of degrees of freedom.

B. 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 [15]:

$$\mathbf{u} = \frac{\sum_{i=1}^n w^i\left(\mathbf{q}, \dot{\mathbf{q}}\right)[\ddot{\mathbf{x}}^T_{\text{ref}}, \dot{\mathbf{q}}^T, 1]\boldsymbol{\beta}^i}{\sum_{i=1}^n w^i\left(\mathbf{q}, \dot{\mathbf{q}}\right)}, \tag{12}$$

where each control law $\mathbf{c}^i_{\boldsymbol{\beta}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\text{ref}})$ is just valid in its local region computed by $w^i\left(\mathbf{q}, \dot{\mathbf{q}}\right)$, and $\boldsymbol{\beta}^i$ are the parameters of each local operational space control law.

However, while the mappings $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\text{ref}}) \rightarrow \mathbf{u}$ can properly be learned locally in the neighborhood of some $\mathbf{q}$,$\dot{\mathbf{q}}$, due 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 due to 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 infinity 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 is results in severe problems for any nonlinear robot requiring multiple, consistent linear models. There are two different approaches to tackling such problems: (1) by biasing the system towards using a pre-processed data set such that it can only produce one particular inverse solution [14], and (2) by incorporating a cost/reward function in order to favor a certain kind of solution (an example which will be discussed later and is shown Figure 2 (b)). The first approach lacks generality and can bias the learning system such that the task is not properly accomplished anymore. 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 I-B and previous work [6]. Operational space control can be seen as a constrained optimization problem with a 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 a immediate reward $r\left(\mathbf{u}\right)$ by running it through an exponential function:

$$r\left(\mathbf{u}\right) = \sigma \exp\left(-0.5\sigma^2 C_1\left(\mathbf{u}\right)\right) = \sigma \exp\left(-\sigma^{-2}\mathbf{u}_1^T\mathbf{N}\mathbf{u}_1\right),$$

where σ is a scaling factor and 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 I-B). 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, it can increase the efficiency of the learning algorithm significantly when only sparse data is available for learning (i.e., as for most interesting robots as the high-dimensional action spaces of complex robots will hardly ever

be filled densely with data)[3]. These local rewards allow us the reformulation of our learning problem as an *immediate reward reinforcement learning problem* [18], as will be discussed in Section II-C.

We are now in the position to formulate a supervised learning algorithm for the local operational space controllers. The task constraint in Equation (3) as well as the rigid body dynamics in Equation (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 $\mathbf{c}^i_{\boldsymbol{\beta}}(\mathbf{q}^k, \dot{\mathbf{q}}^k, \ddot{\mathbf{x}}^k_{\text{ref}})$. An intuitive solution is to use reward-weighted regression, i.e., find the solution which minimizes

$$\sum_{k=1}^{N} r\left(\mathbf{u}^k\right) w^i\left(\mathbf{q}^k, \dot{\mathbf{q}}^k\right) \left(\mathbf{u}^k - [\ddot{\mathbf{x}}^{k,T}_{\text{ref}}, \dot{\mathbf{q}}^{k,T}, 1]\boldsymbol{\beta}^i\right)^2 \rightarrow \min, \tag{13}$$

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

$$\boldsymbol{\beta} = \left(\boldsymbol{\Phi}^T \mathbf{W} \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T \mathbf{W} \mathbf{U}, \tag{14}$$

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

C. Rephrased as a Reinforcement Learning Problem

Originally, we derived this algorithm from a weighted regression point of view. However, this point of view is not completely satisfying as 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 reinforcement learning, allows deriving the previous algorithm together with a computation rule for σ^2 by employing an approach similar to the one suggest in Dayan & Hinton [18]. For this purpose, we assume that we have a sampling process or sampling policy $\tilde{\pi}(\mathbf{u})$, e.g., the robot moving along 5-th order polynomial trajectories in joint space towards randomly chosen joint-space targets by employing a simple PD controller in joint space. Additionally, we have local stochastic control policies given by, e.g., $\pi_i(\mathbf{u}) = \mathcal{N}\left(\mathbf{u} \left| [\ddot{\mathbf{x}}^T_{\text{ref}}, \dot{\mathbf{q}}^T, 1]\boldsymbol{\beta}^i, \Sigma^i\right.\right)$, where $\mathcal{N}$ denotes a normal distribution. For simplicity, we assume that the variance Σ of the normal distribution is fixed in this paper, but it could be included in the algorithm below in the exact same way as $\boldsymbol{\beta}^i$. Under this setup, we want to adjust

[3]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 [17].

	Algorithm: Learning for Operational Space Control
1	**for each new data point** $[\ddot{\mathbf{x}}^k_{\text{ref}}, \mathbf{q}, \dot{\mathbf{q}}^k, \mathbf{u}^k]$
2	Add $(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}) \rightarrow \ddot{\mathbf{x}}$ to the forward model regression.
3	Determine the current number of models n and localizations of the forward models $w^i(\mathbf{q}, \dot{\mathbf{q}})$.
4	Compute desired null-space behavior $\mathbf{u}^k_0 = f\left(\mathbf{q}^k, \dot{\mathbf{q}}^k\right)$.
5	Compute costs $C^k_1 = \left(\mathbf{u}^k_1\right)^T \mathbf{N}\left(\mathbf{q}^k\right) \mathbf{u}^k_1$ with $\mathbf{u}^k_1 = \mathbf{u}^k - \mathbf{u}^k_0$.
6	**For each model** $i = 1, 2, \ldots, n$
	Update mean cost:
7	$\sigma^2_i = \sum_{h=1}^k w^k\left(\mathbf{q}^h, \dot{\mathbf{q}}^h\right) C^k_1 \Big/ \sum_{k=1}^N w^k\left(\mathbf{q}^h, \dot{\mathbf{q}}^h\right)$,
	Compute reward:
8	$r(\mathbf{u}) = \sigma_i \exp\left(-0.5\sigma^2_i C^k_1\right)$
	Add data point to weighted regression so that:
9	$\boldsymbol{\Phi}_i = [\mathbf{q}^i, \dot{\mathbf{q}}^i, \ddot{\mathbf{x}}^i_{\text{ref}}]$
10	$\mathbf{U}_i = \mathbf{u}^i$
11	$\mathbf{W} = \text{diag}\left(r\left(\mathbf{u}^1\right) w^1, \ldots, r\left(\mathbf{u}^n\right) w^n\right)$
	Perform policy update by regression
12	$\boldsymbol{\beta}_{k+1} = \left(\boldsymbol{\Phi}^T \mathbf{W} \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T \mathbf{W} \mathbf{U}$,
13	**end**
14	**end**

TABLE I

THIS TABLE SHOWS THE COMPLETE LEARNING ALGORITHM FOR OPERATIONAL SPACE CONTROL. SEE TEXT OF DETAILED EXPLANATIONS.

the parameters of the local controllers such that we minimize the expected return

$$J_i(\theta) = \int \pi_i(\mathbf{u}) r(\mathbf{u}) d\mathbf{u} \approx \sum_{k=1}^{N} \pi_i\left(\mathbf{u}^k\right) r\left(\mathbf{u}^k\right) \tag{15}$$

of the immediate reward (hence the state variable $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_{\text{ref}}$ are dropped from the equation while implicitly present). This maximization is not directly possible, however, we can maximize the lower bound of the $\log J(\theta)$ transformation, i.e.,

$$\log J(\theta) = \log \sum_{k=1}^{N} q\left(\mathbf{u}^k\right) \frac{\pi_i\left(\mathbf{u}^k\right) r\left(\mathbf{u}^k\right)}{q\left(\mathbf{u}^k\right)}, \tag{16}$$

$$\geq \sum_{k=1}^{N} q\left(\mathbf{u}^k\right) \log \frac{\pi_i\left(\mathbf{u}^k\right) r\left(\mathbf{u}^k\right)}{q\left(\mathbf{u}^k\right)}, \tag{17}$$

$$= \underbrace{\sum_{k=1}^{N} q\left(\mathbf{u}^k\right) \left(\log \pi_i\left(\mathbf{u}^k\right) + \log r\left(\mathbf{u}^k\right)\right)}_{Q_i\left(\boldsymbol{\beta}^i, \sigma^2_i\right)} + \varepsilon, \tag{18}$$

where ε denotes the terms which do not depend on the local controllers' parameters $\boldsymbol{\beta}^i$ or the open parameter σ^2_i; we set $q\left(\mathbf{u}^k\right) = \tilde{\pi}(\mathbf{u}) r(\mathbf{u}) w^i(\mathbf{q}, \dot{\mathbf{q}})$ as in [18] as fixed sampling. This yields the two steps of an expectation-maximization algorithm, i.e., computing $Q\left(\boldsymbol{\beta}^i, \sigma^2_i\right)$ and the maximizing the lower bound by maximizing $Q\left(\boldsymbol{\beta}^i, \sigma^2_i\right)$. The maximization step $[\boldsymbol{\beta}_i, \sigma^2_i]^T = \arg\max_{\boldsymbol{\beta}, \hat{\sigma}^2} Q(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2)$ can be obtained by setting $\partial Q(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2)/\partial \hat{\boldsymbol{\beta}} = 0$ and $\partial Q(\hat{\boldsymbol{\beta}}, \hat{\sigma}^2)/\partial\left(\hat{\sigma}^2\right) = 0$, which yields both Equation (14) and a rule for estimating σ^2_i, i.e.,

$$\sigma^2_i = \frac{\sum_{k=1}^N w^i\left(\mathbf{q}^k, \dot{\mathbf{q}}^k\right) \left(\mathbf{u}^k_1\right)^T \mathbf{N}\left(\mathbf{q}^k\right) \mathbf{u}^k_1}{\sum_{k=1}^N w^i\left(\mathbf{q}^k, \dot{\mathbf{q}}^k\right)}. \tag{19}$$

This rule is surprisingly intuitive and has significant importance as it decreases the sensitivity of the learning process

towards regions with too sparse data. Note, that this derivation can also be understood as minimizing the Kullback-Leibler distance $D(r_{\sigma_i^2}(\mathbf{u})\, q(\mathbf{u}), \pi_i(\mathbf{u}))$ with respect to $\boldsymbol{\beta}_i$ and σ_i^2, which is similar to the weighted regression point of view. The complete algorithm is shown in Table I.

D. An Outlook on Future Work: Using Intertia-based Metrics without having the Mass Matrix

In order to learn several important control laws known from analytical robotics, e.g., Khatib-Gauss [1] and Hsu-IDM Control Laws [2], our learning algorithm needs to be modified in order to be able to compute the appropriate rewards. In Section II-C, we have assumed that the reward $r(\mathbf{u}, \mathbf{q}) = \exp\left(-\mathbf{u}^T \mathbf{N}(\mathbf{q})\, \mathbf{u}\right)$ can be computed without difficulty which is the case, e.g., for $\mathbf{N}(\mathbf{q}) = \text{const}$. However, this is not the case for metrics in the form $\mathbf{N}(\mathbf{q}) = \mathbf{M}^{-n}(\mathbf{q})$ as these require the exact determination of the expensive and error-prone inertia tensor. Therefore, when trying to learn an operational space controller with this kind of a metric, we would run into the same kind of difficulties as analytical approaches with modeling errors, or, at least, learn a different control law, which does not fully realize the interesting properties of the desired control law, e.g., the Khatib-Gauss control law. Nevertheless, through a reformulation of the learning problem, we can compute the reward without explicitly using the inertia tensor when employing a forward-inverse modeling approach similar to [16]. For this reformulation, we realize from Equation (1) that

$$\mathbf{M}^{-1}\mathbf{u}_1 = \ddot{\mathbf{q}} - \mathbf{M}^{-1}(\mathbf{F} + \mathbf{u}_0) = \ddot{\mathbf{q}} - \mathbf{g}_{\boldsymbol{\beta}}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}_0) \equiv \boldsymbol{\delta}\ddot{\mathbf{q}}, \quad (20)$$

where $\ddot{\mathbf{q}} = \mathbf{g}_{\boldsymbol{\beta}}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u})$ denotes a learned forward model (or predictor) which predicts acceleration for a given motor command $\mathbf{u}$ at the joint positions $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$. Using this motor command induced acceleration difference $\boldsymbol{\delta}\ddot{\mathbf{q}}$, we can determine the rewards for Khatib-Gauss and Hsu-IDM control laws by

$$r_{\text{K}}(\mathbf{u}) = \exp(-\mathbf{u}_1^T \mathbf{M}^{-1} \mathbf{u}_1) = \exp(-\mathbf{u}_1^T \boldsymbol{\delta}\ddot{\mathbf{q}}), \quad (21)$$

$$r_{\text{H}}(\mathbf{u}) = \exp(-\mathbf{u}_1^T \mathbf{M}^{-2} \mathbf{u}_1) = \exp(-\boldsymbol{\delta}\ddot{\mathbf{q}}^T \boldsymbol{\delta}\ddot{\mathbf{q}}), \quad (22)$$

respectively. This approach has been tested successfully on the prismatic robot.

III. Evaluations

In order to demonstrate the feasibility of our learning approach, we evaluated our learning operational space controller on a three degrees of freedom, planar robot arm similar as in [19]. Evaluations on an anthropomorphic seven degrees of freedom SARCOS master arm robot arm are in progress.

A. Simulated Experiment

We assume the three degrees of freedom planar robot shown in Figure 3. The links have the length $l_1 = l_2 = 35\,\text{cm}$ and $l_3 = 3\,\text{cm}$. The mass of the links is given as $m_1 = m_2 = m_3 = 3\,\text{kg}$. The dynamic equations of the robot used in the simulator have been automatically derived using the Newton-Euler methodology.

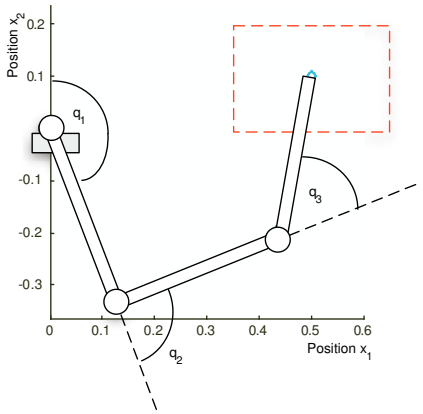


Fig. 3. We consider a three degrees of freedom, rotary robot for tracking control for the experiments. The units are meters and, fully extended, the robot arm extends to 1 m length

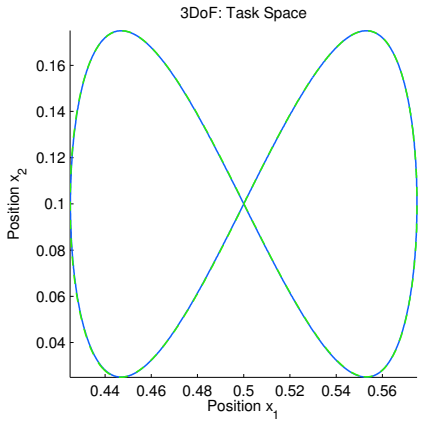


Fig. 4. This figure shows both the analytical controller (blue solid in background) and the learned controller (green dashed). The task space tracking of the learned controller is perfect and can barely be differed from the analytically obtained optimal solution.

The goal of the experiment is to learn how to track operational space trajectories in a limited part of the workspace, i.e., when the end-effector is in the rectangle of horizontal side length of $30\,\text{cm}$, a vertical side length of $20\,\text{cm}$ and a center at $x = 50\,\text{cm}$ and $y = 10\,\text{cm}$. It can be easily verified that in this region the robot dynamics is highly nonlinear, particularly at higher speeds of the robot.

The metric of the cost function is constant and given by

$$\mathbf{N} = \text{diag}(4.44, 1.01, 0.07), \quad (23)$$

and results from the reasoning that in the worst case position $\mathbf{q} = \mathbf{0}$, it will be close to the square of the diagonal of the linearized mass matrix. Furthermore, we assume the presence of a null-space control law $\mathbf{u}_0 = -\mathbf{K}_D^N \dot{\mathbf{q}} - \mathbf{K}_D^N(\mathbf{q} - \mathbf{q}_{\text{rest}})$ pulling the robot towards a rest position

$$\mathbf{q}_{\text{rest}} = [-1.2366, 1.64, 0.95485]^T, \quad (24)$$

which corresponds to a joint-space position which brings the end-effector roughly to the center of the considered workspace.

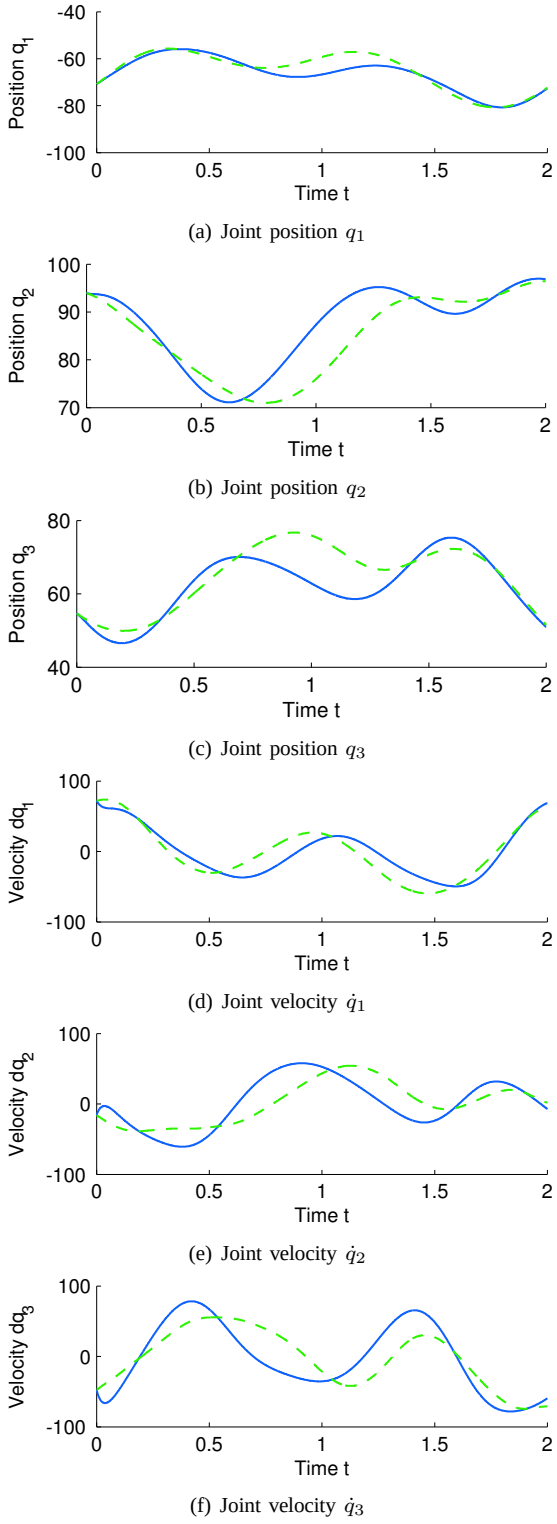
The gains of the null-space component are given by $\mathbf{K}_D^N = \mathrm{diag}(20, 6, 2)$, and $\mathbf{K}_P^N = \mathrm{diag}(0.5, 0.3, 0.1)$. Note, that this null-space term corresponds to a spring in joint-space which is most of the time extended due to the task constraints, hence the lower $\mathbf{K}_P^N$ gains.

B. Results

The experiment consists out of two phases. In the first phase, the control law is trained using data generated by another policy while in the second phase the learned control law is used to generate more data. During the first phase, we generated a sequence of 200 arbitrary joint space positions for which the end-effector was still in the desired workspace rectangle, 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 PD control law, which could not track the trajectories accurately, was used to generate the data. This data was added to the learning system and a first operational space control law was learned. Subsequently, the learned control law was tested on a figure eight reference trajectory of duration 2 s and exhibited relatively good performance in task space. After two to three iterations on the figure eight reference trajectory, the task space tracking performance could not be distinguished from perfect task space tracking as can be observed in Figure 4; differences to the optimal control law computed from the perfect analytical model were negligible, i.e., we obtained nearly perfect task fulfillment. However, as the null-space control law only pulls the robot towards a rest posture but does not prescribe a desired trajectory, small differences in the motor commands of the learned and the analytical operational space control laws will result into different joint-space trajectories as can be observed Figure 5 which shows the all three joint-space position over time in the Figures 5 (a-c) and all three joint-space velocities over time in the Figures 5 (d-f). As a result of differing joint-space trajectories the motor commands do not of both control laws cannot be compared and are different at the same time step, see Figures 6(a-c) which shows all three motor commands over time. In order to verify that we did in fact learn the optimal control law, we compared the outputs of both the learned control law and the analytical optimal control law when using the exact same trajectory as inputs. The outputs of both control laws match nearly perfectly as shown in Figure 6 (d).

IV. Conclusion

In this paper, a general learning framework for operational space for redundant robots has been presented, which is probably the first successful attempt of learning such control laws to date. We overcome the difficulties of having a non-convex data distribution by only learning in the vincinity 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



(a) Joint position q_1

(b) Joint position q_2

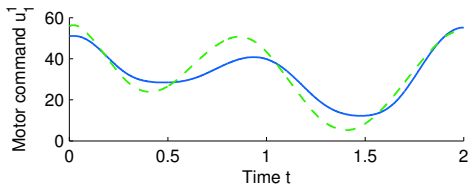
(c) Joint position q_3

(d) Joint velocity $\dot{q}_1$

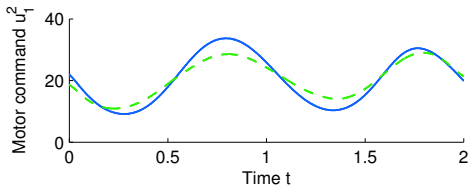
(e) Joint velocity $\dot{q}_2$

(f) Joint velocity $\dot{q}_3$

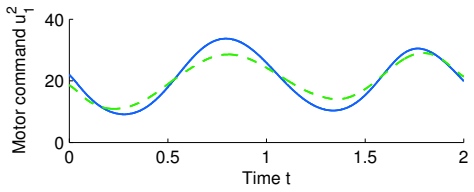
Fig. 5. Figures(a-c) show the joint positions and Figures (d-f) the joint velocities which result from the task space tracking for both the analytical control law (blue solid) and the learned control law (green dashed). The difference between the two controllers in joint-space results from accumulated, very small differences between the analytical and the learned control laws as observable in 6(d).



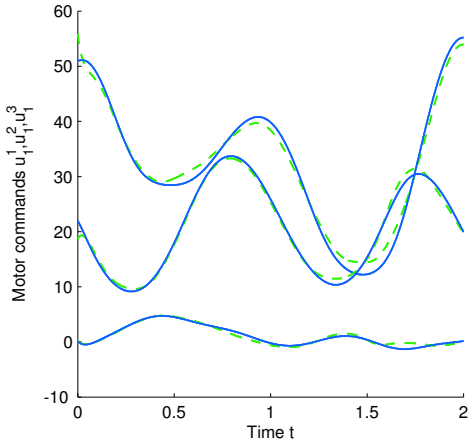
(a) Motor command u_1



(b) Motor command u_2



(c) Motor command u_3



(d) Comparison with ideal output on the same trajectory.

Fig. 6. Figures (a-c) show the motor commands for tracking example for the joint-space trajectories shown in Figure 5. In Figure (d), we compare the outputs of both analytical and learned control laws on the same trajectory and observe only small differences.

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 perspective, 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 two degrees of freedom robot arm as example. As application, we have shown a task-space trajectory following on a three degrees of freedom rotary robot arm, where we could exhibit near-perfect operational space tracking control. As 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, the techniques and theory developed in this paper will be beneficial in developing truly autonomous and self-tuning robotic systems.

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] P. Hsu, J. Hauser, and S. Sastry, "Dynamic control of redundant manipulators," *Journal of Robotic Systems*, vol. 6, no. 2, pp. 133–148, 1989.

[3] L. Sentis and O.Khatib, "Control of free-floating humanoid robots through task prioritization," in *International Conference on Robotics and Automation*, no. 1730-1735. Barcelona, Spain: IEEE, April 2005.

[4] 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.

[5] J. Nakanishi, J. A. Farrell, and S. Schaal, "Learning composite adaptive control for a class of nonlinear systems," in *IEEE International Conference on Robotics and Automation*, 2004, pp. pp.2647–2652.

[6] 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.

[7] F. E. Udwadia and R. E. Kalaba, *Analytical Dynamics: A New Approach*. Cambridge University Press, 1996.

[8] H. Bruyninckx and O. Khatib, "Gauss' principle and the dynamics of redundant and constrained manipulators," *Proceedings of the 2000 IEEE International Conference on Robotics & Automation*, pp. 2563–2569, April 2000.

[9] K. Doty, C. Melchiorri, and C. Bonivento, "A theory of generalized inverses applied to robotics," *International Journal of Robotics Research*, vol. 12, pp. 1–19, 1993.

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

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

[12] 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.

[13] G. Tevatia and S. Schaal, "Inverse kinematics for humanoid robots," in *International Conference on Robotics and Automation (ICRA2000)*, San Fransisco, April 2000, 2000.

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

[15] 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.

[16] 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.

[17] F. E. Udwadia, "Discussions on c.f. gauss, gauss' principle, and its application to control."

[18] 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

[19] J. M. Hollerbach and K. C. Suh, "Redundancy resolution of manipulators through torque optimization," *International Journal of Robotics and Automation*, vol. 3, no. 4, pp. 308–316, 1987.