2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Real-Time Learning of Resolved Velocity Control on a Mitsubishi PA-10

Jan Peters, Duy Nguyen-Tuong
Max Planck Institute for Biological Cybernetics, Spemannstraße 38, 72076 Tübingen
Email: {jrpeters,duy}@tuebingen.mpg.de

Abstract— **Learning inverse kinematics has long been fascinating the robot learning community. While humans acquire this transformation to complicated tool spaces with ease, it is not a straightforward application for supervised learning algorithms due to non-convex learning problem. However, the key insight that the problem can be considered convex in small local regions allows the application of locally linear learning methods. Nevertheless, the local solution of the problem depends on the data distribution which can result into inconsistent global solutions with large model discontinuities. While this problem can be treated in various ways in offline learning, it poses a serious problem for online learning. Previous approaches to the real-time learning of inverse kinematics avoid this problem using smart data generation, such as the learner biasses its own solution. Such biassed solutions can result into premature convergence, and from the resulting solution it is often hard to understand what has been learned in that local region.**

This paper improves and solves this problem by presenting a learning algorithm which can deal with this inconsistency through re-weighting the data online. Furthermore, we show that our algorithms work not only in simulation, but we present real-time learning results on a physical Mitsubishi PA-10 robot arm.

I. Introduction

For most important control tasks, it is easier to specify the task in the Cartesian coordinate systems of the task, e.g., the end-effectors position $\mathbf{p} \in \mathbb{R}^m$, than in the complicated manifold of joint-space configuration $\mathbf{q} \in \mathbb{R}^n$. For this reason, the transformation of task-space trajectories into joint space trajectories, i.e., the inverse kinematics, is an essential step for motor command generation [1], [2].

Learning inverse kinematics has a variety of interesting properties in comparison to analytical approaches as it can deal with camera calibration, sensor offsets and measurements errors [3], [4]. Furthermore, as the kinematics of the robot are implicitly represented by the measured data, singularities no longer pose a difficult problem as the robot cannot steer towards physically impossible configurations for learned models [3], [4].

While various successful approaches exist to the offline learning of inverse kinematics, see [5] for an extensive review, only few methods exist which extend into the realms of real-time learning. A pioneering approach in this direction was presented in [3], where the authors considered fast online-learning approaches using locally linear models to learn differential inverse kinematics for a simulated humanoid robot. However, while local solutions result into locally viable solutions, the data distribution will determine the global solution and consistency of the local models is no longer ensured. In [3], the authors ensure consistency by biasing their data generation such that the learning system would only be presented with a consistency-ensuring set of data. However, this approach is problematic as the learner can get stuck in certain configurations and the bias can largely determine the behavior of the system [6]. Recently, [6], [7], suggested the idea that a cost-related re-weighting can deal with the consistency of models in the related problems such as Operational Space Control. In this paper, we extend the idea of [6], [7] to the problem of learning resolved velocity control [1]–[3]. For doing so, we first reformulate resolved velocity control as an optimization problem following the ideas presented in [8], [9] and subsequently incorporate the resulting cost function into the real-time learning control law. For extracting the regions of localization where the kinematics are dominantly linear, we make use of the multiple paired forward-inverse modeling (MPFIM) approach [10], [11]: as prediction is a well-defined problem for kinematics, we can determine the local region by predicting the end-effector velocity[1].This combined approach results into a feasible learning framework which works on a physical Mitsubishi PA-10 robot shown in Figure 1.

The main contributions of this paper are (i) the transfer of the results in [6], [7] to Resolved Velocity Control, (ii) the correction of misunderstandings in [3], [6] and (iii), to our knowledge, this paper shows the first published results on using the reward-weighting approach suggested in [6], [7] for a real physical robot in real-time kinematics learning.

A. Notation and Review

The forward kinematics given by the transformation

$$\mathbf{p} = \mathbf{f}_{\text{Kinematics}}(\mathbf{q}) \tag{1}$$

is straightforward to compute and can be estimated using stereo vision [1], [2]. However, the inverse of this function $\mathbf{f}^{-1}_{\text{Kinematics}}$ is not unique for redundant robots, i.e., robots with excessive degrees of freedom such that $n > m$. Instead, a multitude of solutions exists and computing inverse kinematics requires that certain solutions are favored depending on their

[1]The idea of using coupled forward and inverse models in order to learn uninvertable functions in [10], [11] can be seen as a local learning form of the distal teacher approach [12]. However, while the distal teacher approach ensures a consistent solution by minimizing global errors which is not the case for local methods.

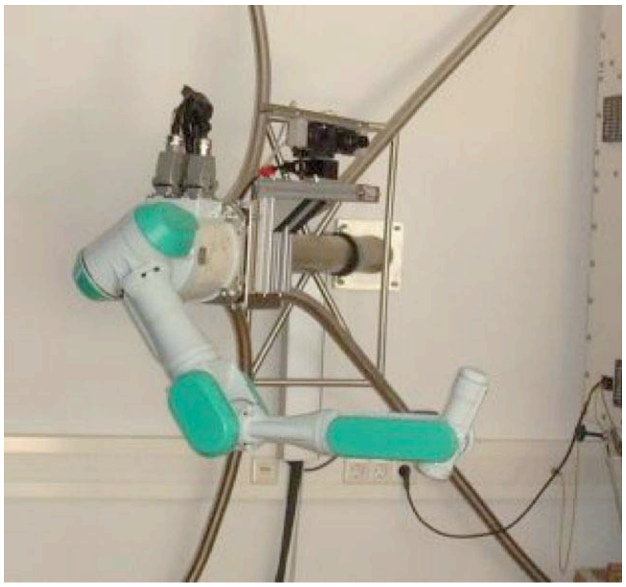


Fig. 1. Mitsubishi PA-10 robot arm with seven degrees of freedom used in the experiments in this paper.

proximity to the current joint configuration. For doing so, the velocity of the end-effector in task space

$$\dot{\mathbf{p}} = \mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}}\,, \tag{2}$$

is considered, where $\mathbf{J}(\mathbf{q}) = d\mathbf{f}_{\text{Kinematics}}(\mathbf{q})/d\mathbf{q}$ denotes the Jacobian. For this problem, we can generate local solutions such as

$$\dot{\mathbf{q}} = \mathbf{J}(\mathbf{q})^{+}\,\dot{\mathbf{p}}\,, \tag{3}$$

in order to determine the joint-space trajectory for a given task-space trajectory with $\mathbf{J}(\mathbf{q})^{+}$ as the pseudo-inverse of $\mathbf{J}(\mathbf{q})$. In Section I-B, we will show how to derive this approach from an optimization point of view.

B. Resolved Velocity Control as Optimization Problem

An important insight into redundant task-space control problems (see [8], [9]) is that resolved motion control can be derived as the solution of a constraint optimization problem given by

$$\begin{aligned} \min_{\dot{\mathbf{q}}} C_0(\dot{\mathbf{q}}) &= \dot{\mathbf{q}}^T \mathbf{N} \dot{\mathbf{q}} \\ \text{s.t. } \mathbf{J}\dot{\mathbf{q}} &= \dot{\mathbf{p}}_{\text{ref}}, \end{aligned} \tag{4}$$

where $\mathbf{N}$ denotes a positive definite metric weighting the joint velocity distributing the movement onto the joints, and $\dot{\mathbf{p}}_{\text{ref}} = \dot{\mathbf{p}}_d(t) + \mathbf{K}_p^R(\mathbf{p}_d(t) - \mathbf{p}(t))$ presents a reference attractor in task space with gain matrix $\mathbf{K}_p$ and desired task-space trajectory $\mathbf{p}_d$, $\dot{\mathbf{p}}_d$. The resulting control laws or solution of this optimization problem (4) obey the general form [8], [9]

$$\dot{\mathbf{q}} = \mathbf{N}^{-1/2}(\mathbf{J}\mathbf{N}^{-1/2})^{+}\dot{\mathbf{p}}_{\text{ref}} \equiv \mathbf{J}_N^{\#}\dot{\mathbf{p}}_{\text{ref}}, \tag{5}$$

where the notation $\mathbf{D}^{+}$ defines the pseudo-inverse and $\mathbf{D}^{1/2}$ the matrix root of a given matrix $\mathbf{D}$. According to Equation (3), $\mathbf{J}_N^{\#} = \mathbf{N}^{-1/2}(\mathbf{J}\mathbf{N}^{-1/2})^{+}$ can be considered as the weighted pseudo-inverse Jacobian.

However, while achieving the task perfectly, the joint-space trajectories can result into unfavorable postures. For handling such cases, additional regularization which do not affect the tasks fulfillment but ensures a favorable joint-space behavior needs to be included. 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

$$\begin{aligned} \min_{\dot{\mathbf{q}}} C_1(\dot{\mathbf{q}}) &= (\dot{\mathbf{q}} - \dot{\mathbf{q}}_0)^T \mathbf{N} (\dot{\mathbf{q}} - \dot{\mathbf{q}}_0)\ , \\ \text{s.t. } \mathbf{J}\dot{\mathbf{q}} &= \dot{\mathbf{p}}_{\text{ref}}, \\ \dot{\mathbf{q}}_0 &= -\mathbf{K}_P(\mathbf{q} - \mathbf{q}_{\text{rest}}). \end{aligned} \tag{6}$$

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 into the general solution given by

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{N}^{-1/2}(\mathbf{J}\mathbf{N}^{-1/2})^{+}\dot{\mathbf{p}}_{\text{ref}} \qquad (7) \\ &\quad + \mathbf{N}^{-1/2}(\mathbf{I} - (\mathbf{N}^{-1/2}\mathbf{J})(\mathbf{J}\mathbf{N}^{-1/2})^{+})\mathbf{N}^{1/2}\dot{\mathbf{q}}_0, \\ &= \mathbf{J}_N^{\#}\dot{\mathbf{p}}_{\text{ref}} + (\mathbf{I} - \mathbf{J}_N^{\#}\mathbf{J})\dot{\mathbf{q}}_0\,, \qquad (8) \end{aligned}$$

where the first term results into task achievement while the second term results into more favorable trajectories. When having more than two tasks, these can be nested in a similar fashion leading to a general framework of hierarchical task control [8], [9].

II. Learning Inverse Kinematics in Real-Time

While the off-line learning of the inverse kinematics has been long studied starting with the early literature in the late 1980s (see [4], [5], [13]–[15] for an overview), the same does not hold true for online real-time learning. For this topic, few approaches exist where [3] appears to be unique in the sense that it can deal also with robots with a large number of degrees of freedom. However, as outlined before, in this paper we attempt to incorporate the strength of [3] in terms of the local learning algorithms while avoiding the shortcomings, e.g., the over-biasing of the data generation and the problematic localization. For doing so, we start by reviewing why the problem is locally convex and show the corrected version of the explanation provided in [3]. Subsequently, we show how cost-related re-weighting can help us to ensure global consistency among the models while avoiding the not real-time capable global regression problem.

A. Local Models of Resolved Velocity Control

Learning resolved velocity control is equivalent to obtaining a mapping $(\mathbf{q}, \dot{\mathbf{p}}_{\text{ref}}) \rightarrow \dot{\mathbf{q}}$ from sampled data using function approximation. However, as the dimensionality of the task-space reference trajectory $\dot{\mathbf{p}}_{\text{ref}}$ is lower than the one of joint-space velocity $\dot{\mathbf{q}}$, there are infinitely many solutions for $\dot{\mathbf{q}}$

for most joint positions $\mathbf{q}$. That means, the joint velocities $\dot{\mathbf{q}}$ no longer forms a convex set and, thus, when learning the inverse mapping $(\mathbf{q}, \dot{\mathbf{p}}_{\text{ref}}) \rightarrow \dot{\mathbf{q}}$, 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 [3], [14].

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 joint space position [3], [15]. The feasibility of this idea can be demonstrated simply by averaging over the combination of Equations (8) which yields that by averaging over the same spatial position $\mathbf{q}$ we have

$$\begin{aligned} \bar{\dot{\mathbf{q}}} &= \langle \dot{\mathbf{q}} \rangle = \left\langle \mathbf{J}_N^{\#} \dot{\mathbf{p}}_{\text{ref}} + (I - \mathbf{J}_N^{\#}\mathbf{J})\dot{\mathbf{q}}_0 \right\rangle \\ &= \mathbf{J}_N^{\#} \langle \dot{\mathbf{p}}_{\text{ref}} \rangle + (I - \mathbf{J}_N^{\#}\mathbf{J})\dot{\mathbf{q}}_0 \,, \end{aligned} \tag{9}$$

i.e., in the vicinity of the same $\mathbf{q}$, a particular $\bar{\dot{\mathbf{p}}} = \langle \dot{\mathbf{p}}_{\text{ref}} \rangle$ will always correspond to exactly one particular $\bar{\dot{\mathbf{q}}}$. The uniqueness only holds if $\dot{\mathbf{q}}_0$ is just a function of $\mathbf{q}$ and $\mathbf{q}_{\text{rest}}$ *but not* $\dot{\mathbf{q}}$. However, while the metric would be set by the engineer in an analytical approach, it will be the a result of the data distribution in learned approaches unless further optimization is employed as in Section II-B.

Therefore, locally linear controllers π_i with parameters θ_i defined by

$$\dot{\mathbf{q}}^i = \pi_i(\mathbf{q}, \dot{\mathbf{p}}_{\text{ref}}) = \theta_i^T [\dot{\mathbf{p}}_{\text{ref}}^T, \mathbf{q}, 1]^T, \tag{10}$$

can be used if they are only active in a region around $\mathbf{q}$ (note that we added constant input in Equation (10) 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 piece-wise linearization for the locally linear controllers. For this purpose, we learn a locally linear forward or predictor model ρ_i with parameters ψ_i defined by

$$\dot{\mathbf{p}}^i = \rho_i(\mathbf{q}, \dot{\mathbf{q}}) = \psi_i^T [\dot{\mathbf{q}}^T, \mathbf{q}^T, 1]^T. \tag{11}$$

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) [16], 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

$$w_i(\mathbf{q}) = e^{-\frac{1}{2}(\mathbf{q}-\mathbf{c}_i)^T \mathbf{D}_i (\mathbf{q}-\mathbf{c}_i)}\,, \tag{12}$$

centered at $\mathbf{c}_i$ in joint-space, and shaped by a distance metric $\mathbf{D}_i$. For a closer description of this statistical learning algorithm see [16].

For each local forward model created by LWPR, we automatically create a local controller. This approach of pair-wise combining predictors and controllers is related by the MPFIM or MOSAIC architecture [11] where the quality of predicting a task is used for selecting which local controller should be used for the task.

B. Globally 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 $\dot{\mathbf{q}}$ which is given by a weighted average [16]:

$$\dot{\mathbf{q}} = \pi(\mathbf{q}, \dot{\mathbf{p}}_{\text{ref}}) = \frac{1}{W_\Sigma} \sum_{i=1}^{n} w_i(\mathbf{q})\, \pi_i(\mathbf{q}, \dot{\mathbf{p}}_{\text{ref}}), \tag{13}$$

with $W_\Sigma = \sum_{i=1}^{n} w_i(\mathbf{q})$ as normalization constant. Each local control law $\pi_i(\mathbf{q}, \dot{\mathbf{p}}_{\text{ref}})$ given in (13) is just valid in its local region computed by $w_i(\mathbf{q})$.

However, while the mappings $(\mathbf{q}, \dot{\mathbf{p}}_{\text{ref}}) \rightarrow \dot{\mathbf{q}}$ can properly be learned locally in the neighborhood of some $\mathbf{q}$, 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. While this problem is not devastating for a prismatic robot, it results in severe problems for any nonlinear robot with rotary joints as these require multiple, consistent linear models.

There are two different approaches 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 [3], and (2) by incorporating a cost/reward function in order to favor a certain kind of solution. 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 find a principled approach to this inconsistency problem is based on the discussion in Section I-B. Resolved velocity control can be seen as a constrained optimization problem with a cost function given in Equation (4). 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 cost-related weight $\varpi(\dot{\mathbf{q}})$ by running the cost $C_1(\dot{\mathbf{q}})$ through an exponential function:

$$\varpi(\dot{\mathbf{q}}) = e^{-\sigma^{-2}(\dot{\mathbf{q}}-\dot{\mathbf{q}}_0)^T \mathbf{N} (\dot{\mathbf{q}}-\dot{\mathbf{q}}_0)},$$

where σ is a scaling factor. The scaling factor σ does not affect the optimality of a solution $\dot{\mathbf{q}}$ as it acts as a monotonic transformation in this cost function. However, it can increase

Algorithm 1 Reward-Weighted Regression for Application in Real-Time Learning of Resolved Velocity Control.

```
• query point $(\mathbf{q}^t, \dot{\mathbf{p}}^t_{\text{ref}})$
• last realized movement $(\mathbf{q}^{t-1}, \dot{\mathbf{p}}^{t-1}, \dot{\mathbf{q}}^{t-1})$

**Online Learning: Incorporate last realized movement**
1. Predict using the current models
   $\dot{\mathbf{p}}^{t-1} = \frac{\sum_{i=1}^n w_i(\mathbf{q}^{t-1})\,\rho_i(\mathbf{q}^{t-1}, \dot{\mathbf{q}}^{t-1})}{\sum_{i=1}^n w_i(\mathbf{q}^{t-1})}.$
2. Update predictors and weighting kernels using LWPR. If necessary, generate new models.
3. Determine the null-space behavior
   $\dot{\mathbf{q}}_0^{t-1} = -\mathbf{K}_P(\mathbf{q}^{t-1} - \mathbf{q}_{\text{rest}}).$
4. Compute the reward-weight
   $\varpi(\dot{\mathbf{q}}) = e^{-\sigma^{-2}(\dot{\mathbf{q}}-\dot{\mathbf{q}}_0)^T\mathbf{N}(\dot{\mathbf{q}}-\dot{\mathbf{q}}_0)}.$
5. Model Update:
**for** each model $i$ **do**
    add data point to reward-weighted regression
        $\mathbf{\Phi}_k = [(\dot{\mathbf{p}}^k_{\text{ref}})^T, (\mathbf{q}^k)^T, 1],$
        $\mathbf{\Lambda}_k^T = \dot{\mathbf{q}}^k,$
        $\mathbf{W}^i = \text{diag}\left(\varpi(\dot{\mathbf{q}}^k)\, w_i(\mathbf{q}^k)\right).$
    perform policy update:
        $\theta_i = \left(\mathbf{\Phi}^T\mathbf{W}^i\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{W}^i\mathbf{\Lambda}.$
**end for**

**Recall: Generate $\dot{\mathbf{q}}$ for current query point**
1. Compute policy output
   $\pi(\mathbf{q}^t, \dot{\mathbf{q}}^t) = \frac{\sum_{i=1}^n w_i(\mathbf{q}^t)\,\pi_i(\mathbf{q}^t, \dot{\mathbf{q}}^t)}{\sum_{i=1}^n w_i(\mathbf{q}^t)}.$
2. Determine exploration $\varepsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$.
3. Returned command $\dot{\mathbf{q}} = \pi(\mathbf{q}^t, \dot{\mathbf{q}}^t) + \varepsilon$.
```

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). This re-weighting factor enforces that we will always use the same metric throughout all models and the same velocity in the null-space which pulls us towards the rest posture.

We are now in the position to formulate a supervised learning algorithm for the local operational space controllers. The task constraint in Equation (4) as well as the physics of the robot 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

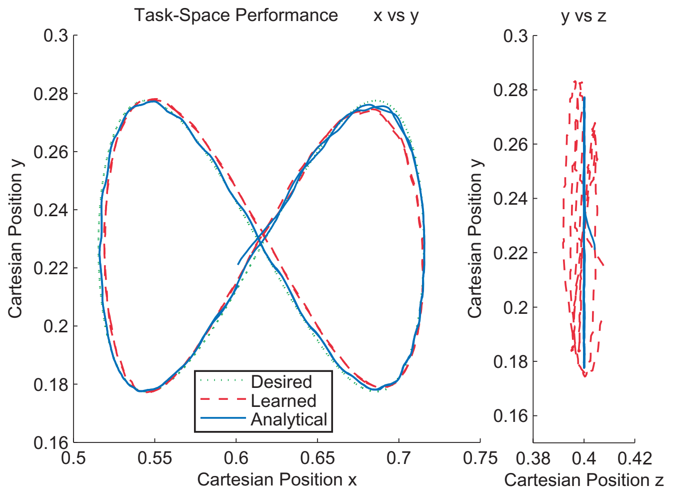


Fig. 2. This figure illustrates the task performance of both the analytical and the learned resolved velocity control laws. Here, the green dotted line shows the desired trajectory which the robot should follow, the red dashed line is the performance of the real-time learning control law while the blue solid line shows the performance of the resolved velocity 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).

attempt to learn primarily from the observed motor commands $\mathbf{u}^k$ which also have a high reward $w(\dot{\mathbf{q}}^k)$ within each active local model $\pi_i(\mathbf{q}^k, \dot{\mathbf{p}}^k_{\text{ref}})$. An intuitive solution is to use reward-weighted regression, i.e., find the solution which minimizes the localized error

$$E_i = \sum_{k=1}^{N} \omega(\dot{\mathbf{q}}^k)\, w_i(\mathbf{q}^k)\left(\dot{\mathbf{q}}^k - \pi_i(\mathbf{q}^k, \dot{\mathbf{p}}^k_{\text{ref}})\right)^2, \tag{14}$$

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

$$\theta_i = \left(\mathbf{\Phi}^T\mathbf{W}^i\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{W}^i\mathbf{\Lambda}, \tag{15}$$

with rows in the matrices $\mathbf{\Phi}$, $\mathbf{W}^i$ and $\mathbf{\Lambda}$ defined by

$$\mathbf{\Phi}_k = [(\dot{\mathbf{p}}^k_{\text{ref}})^T, (\mathbf{q}^k)^T, 1], \tag{16}$$
$$\mathbf{\Lambda}_k^T = \dot{\mathbf{q}}^k, \tag{17}$$
$$\mathbf{W}^i = \text{diag}\left(\varpi(\dot{\mathbf{q}}^k)\, w_i(\mathbf{q}^k)\right). \tag{18}$$

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 Algorithm 1. Note that this step was only possible due to the essential cost function in Equation (6).

III. Experiments & Evaluations

In order to demonstrate the feasibility of our learning approach, we evaluated our learning approach to resolved velocity control on a physical Mitsubishi PA-10 arm shown in Figure 1. We compare it to the analytical solution found in the robotics literature [1], [2].

A. Experimental Setup

The Mitsubishi PA-10 (Figure 1) is general purpose robot arm for low-velocity applications such as surgical tool placement or teleoperated soft-tissue manipulation [17]. This robot has seven degrees of freedom (DoF) and, thus, has four degrees of freedom too much for following a desired trajectory in cartesian space. Note that we do not track orientation as otherwise we could not experimentally show that we can cope with a large number of redundant degrees of freedom. We use the Mitsubishi supplied software to access the robot and command it in an asynchronous mode. This allows us to command joint velocities by giving desired velocities at a 200Hz sampling frequency. These desired increments are then controlled using the internal, joint-level high gain control loop of the Mitsubishi PA-10 control station.

The goal of the experiment is to show that we can learn consistent resolved velocity control laws without observing the task beforehand. For doing so, we choose the standard task of a figure-8 in task space [18], [19] with

$$p_x = p_x^0 + l_x \sin(\omega t), \tag{19}$$

$$p_y = p_y^0 + l_y \sin(\omega t)\cos(\omega t), \tag{20}$$

$$p_z = p_z^0, \tag{21}$$

where $\omega = 6\pi/50 s^{-1}$ and

$$\mathbf{p}_0 = [p_x^0, p_y^0, p_z^0] = \mathbf{f}_{\text{Kinematics}}(\mathbf{q}_{\text{rest}}) \tag{22}$$

is determined by the rest posture $\mathbf{q}_{\text{rest}}$. The rest posture is given by

$$\mathbf{q}_{\text{rest}} = [0.1627, 0.6076, 0.2127, \ldots \\ 1.4407, 0.2626, 1.6965, -0.0138]^T,$$

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 pull towards the rest posture

$$\dot{\mathbf{q}}_0 = -\mathbf{K}_P(\mathbf{q} - \mathbf{q}_{\text{rest}}) \tag{23}$$

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.

B. Results

The experiment consists out 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 velocity control law on the desired trajectory and perturb its out put with a very small amount of exploration $\varepsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ with $\sigma = 0.001$, i.e.,

$$\dot{\mathbf{q}} = \pi(\mathbf{q}, \dot{\mathbf{p}}_{\text{ref}}) + \varepsilon. \tag{24}$$

Such exploration is known under the name motor babbling in the behavioral literature and helps humans learn motor tasks in a similar manner [5]. It is absolutely necessary for fast learning of the resolved velocity 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 2-4, 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 Figures 3,4.

The resulting online-learning is achieved sufficiently fast the robot is capable to learn tracking on the same trajectory which it is executing. Due to the prediction accuracy, the learning system has already determine 7 different local regions and will only learn 5 additional different regions during the execution of the trajectory. Altogether this trajectory requires 12 locally linear regions for accurate tracking. All these models determine the activity of the locally linear control laws which are learned online during the execution of the trajectory. The high gain of the reference attractor compensates for initial model errors and could be reduced once the control law has been achieved a high level of accuracy.

In Figure 2, the resulting task-space performance can be observed. We can see that the resulting task space tracking performance is quite close to the one of the analytical resolved velocity control law. In Figures 3,4, we can see a comparison of the joint-space trajectories of both the analytical resolved velocity control law and the observed one. While both are similar throughout the trajectory, they differ due to the exploration which causes a drift in null-space, the online, real-time learning and model errors.

IV. Conclusion

In this paper, we have shown the first real robot results of the application of the reward-weighted regression framework for online, real-time, learning of resolved velocity control. The robot experiments were conducted on a physical Mitsubishi PA-10 in our lab. For doing so, we had to overcome the difficulties of having a non-convex data distribution by only learning in the vicinity of a local model anchored both in 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 behind resolved velocity control. The combination of these components has resulted into a proper learning setup for resolved velocity control.

The main contributions of this paper are (i) the transfer of the results in [6], [7] to Resolved Velocity Control, (ii) the correction of misunderstandings in [3], [6] and (iii), to our knowledge, this paper shows the first published results on using the reward-weighting approach suggested in [6], [7] for a real physical robot in real-time learning of resolved velocity control.

While we only track the position in this paper in order to show that we can learn with a large number of redundant degrees of freedom, future work will include tracking the orientation as well. However, this work will be executed on systems with a larger number of degrees of freedom in order to remain challenging.

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 developed in this paper will be beneficial in developing truly autonomous and self-tuning robotic systems.

References

[1] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Wiley, 2006.

[2] L. Sciavicco and B. Siciliano, *Modeling and control of robot manipulators*. Heidelberg, Germany: MacGraw-Hill, 2007.

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

[4] D. DeMers and K. Kreutz-Delgado, "Canonical parameterization of excess motor degrees of freedom with self organizing maps," *IEEE Transactions on Neural Networks*, vol. 7, pp. 43–55, 1996.

[5] ——, *Neural Systems for Robotics*. San Diego: Academic Press, 1997, ch. Inverse kinematics of dextrous manipulators, pp. 75–116.

[6] J. Peters and S. Schaal, "Reinforcement learning for operational space," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, Rome, Italy, 2007.

[7] ——, "Reinforcement learning by reward-weighted regression for operational space control," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2007.

[8] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*. Boston, MA: Addison-Wesley, 1991.

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

[10] M. Haruno, D. M. Wolpert, and M. Kawato, "Mosaic model for sensorimotor learning and control," *Neural Comput*, vol. 13, no. 10, pp. 2201–20, 2001.

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

[12] M. I. Jordan, "Computational aspects of motor control and motor learning," in *Handbook of perception and action*, H. Heuer and S. W. Keele, Eds. New York: Academic Press, 1996.

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

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

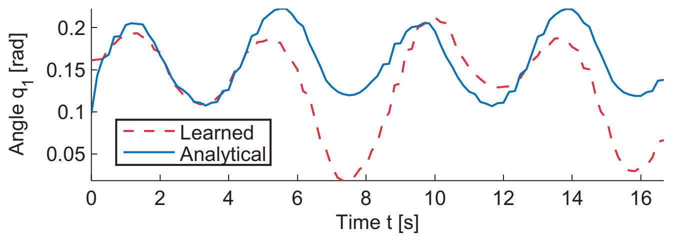
[15] D. Bullock, S. Grossberg, and F. H. Guenther, "A self-organizing neural model of motor equivalent reaching and tool use by a multijoint arm," *Journal of Cognitive Neuroscience*, vol. 5, no. 4, pp. 408–435, 1993.

[16] 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.

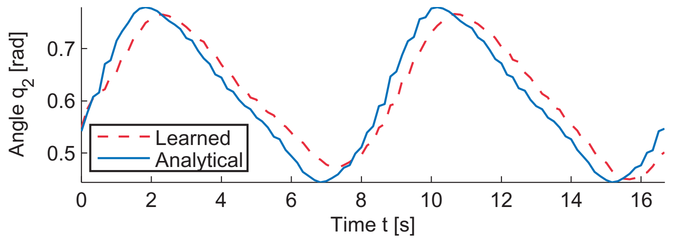
[17] C. W. Kennedy and J. P. Desai, "Modeling and control of the mitsubishi pa-10 robot arm harmonic drive system," *IEEE/ASME Transations on Mechatronics*, vol. 10, no. 3, p. 263, JUNE 2005.

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

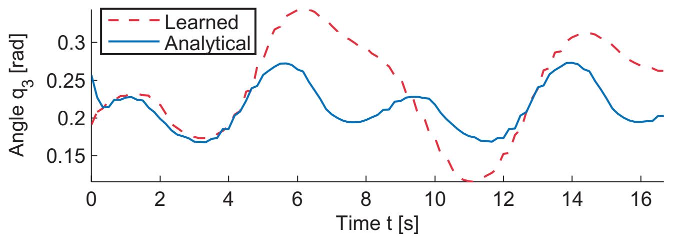
[19] J. Nakanishi, M. Mistry, J. Peters, and S. Schaal, "Experimental evaluation of task space position/orientation control towards compliant control for humanoid robots," in *IEEE International Conference on Intelligent Robotics Systems (IROS 2007)*, 2007.



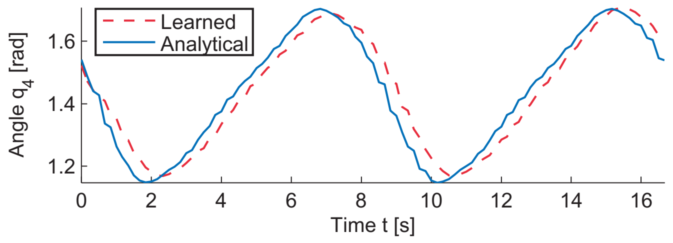
(a) Position of Joint 1



(b) Position of Joint 2



(c) Position of Joint 3



(d) Position of Joint 4

Fig. 3. This figure illustrates the resulting joint-space trajectories for joints 1 to 4 of both the analytical and the learned solution of the resolved velocity control law executed on our physical Mitsubishi PA-10 shown in Figure 1. The remaining joints look analogously. 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 that fast. Additionally, the model error accumulates along the trajectory and the task-space control law needs to compensate for it. Note that the task space trajectory is performance of the learned approach is comparable to the analytical approach as presented in Figure 2. The blue solid line shows the joint-space trajectory of the analytical approach while the red dashed line shows the real-time learning solution.