Using Reward-Weighted Imitation for Robot Reinforcement Learning

Jan Peters and Jens Kober

Abstract— **Reinforcement Learning is an essential ability for robots to learn new motor skills. Nevertheless, few methods scale into the domain of anthropomorphic robotics. In order to improve in terms of efficiency, the problem is reduced onto reward-weighted imitation. By doing so, we are able to generate a framework for policy learning which both unifies previous reinforcement learning approaches and allows the derivation of novel algorithms. We show our two most relevant applications both for *motor primitive learning* (e.g., a complex Ball-in-a-Cup task using a real Barrett WAM™ robot arm) and *learning task-space control.***

I. Introduction

POLICY LEARNING by reward-driven self improvement is an essential ability of future robots. However, despite all the tremendous progress in Reinforcement Learning over the last three decades, most interesting, high-dimensional motor learning problems are often beyond the reach of current methods. Learning by imitation on the other hand has proven to be significantly easier problem than reinforcement learning and efficient methods exist even for humanoid robots. Surprisingly, human reinforcement learning does not appear to differ that strongly from imitation – instead of directly improving a reward, humans will rather attempt to imitate their own successful trials more strongly than the unrewarded ones [10]. This insight originally triggered our research towards reinforcement learning by reward-weighted imitation.

The learning approach basically corresponds to giving the highly rewarded trials proportionally more weight than the unsuccessful ones. While this idea might sound very basic, it can properly be motivated from a statistical learning perspective where the reward simply corresponds to an improper probability distribution [12], [14], [5], [17] and, thus, one can apply most standard methods from machine learning in order to derive novel methods for reinforcement learning. Nevertheless, we can also directly relate existing methods such as policy gradient approaches [6], [7], [9] and the natural actor-critic [9] to this approach.

In this paper, we will give the most extensive overview on these approaches to date. We will start by first deriving the generic framework for reinforcement learning by reward-weighted imitation and show how it relates to previous approaches. Subsequently, we will derive to expectation-maximization algorithms which have been essential in our recent research, i.e., the reward-weighted regression [14] and the Policy Learning by Weighting Exploration with the Returns (PoWER) algorithm [17]. We are especially interested in a particular kind of motor control problems, i.e., (i) learning of task-space control and (ii) learning of dynamic motor primitives [18], [19]. We show that the presented algorithms work well when employed in the context of learning dynamic motor primitives in four different settings, i.e., the two benchmark problems from [9], the Underactuated Swing-Up [16] and the complex task of Ball-in-a-Cup [20], [15]. Both the Underactuated Swing-Up as well as the Ball-in-a-Cup are achieved on a real Barrett WAM™ robot arm as well as in various applications to learning resolved-velocity control.

Jan Peters and Jens Kober are with the Department of Empirical Inference and Machine Learning, Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany (email: {jens.kober,jan.peters}@tuebingen.mpg.de).

II. Policy Learning as Reward-weighted Imitation

In this section, we first discuss the reinforcement learning problem in the general context of motor control and introduce the required notation in Section II-A. Using a generalization of the approach in [12], [14], we show how the general framework is related to policy gradients methods in II-B and derive EM-inspired algorithms such as the Reward-Weighted Regression and the Policy Learning by Weighting Exploration with the Returns (PoWER).

A. Problem Statement & Notation

In this paper, we focus on episodic reinforcement learning [1]. We assume that at time t there is an actor in a state $\mathbf{s}_t$ and chooses an appropriate action $\mathbf{a}_t$ according to a stochastic policy $\pi(\mathbf{a}_t|\mathbf{s}_t, t)$. Such a policy is a probability distribution over actions given the current state. The stochastic formulation allows a natural incorporation of exploration and, in the case of hidden state variables, the optimal time-invariant policy has been shown to be stochastic [7]. Upon the completion of the action, the actor transfers to a state $\mathbf{s}_{t+1}$ and receives a reward r_t.

In motor primitive learning, episodic reinforcement learning tasks of finite length are predominant. Task-space control can be seen as an immediate reward reinforcement learning problem which can be see an episodic task with an episode length of one. Thus, we focus on finite horizons of length T with episodic restarts [1] and learn the optimal parametrized, stochastic policy for such reinforcement learning problems. We assume an explorative (i.e., stochastic) parametrized policy π with parameters $\boldsymbol{\theta} \in \mathbb{R}^n$. The general goal in reinforcement learning is to optimize the *expected return* of the policy π with parameters $\boldsymbol{\theta}$ defined by

$$J(\boldsymbol{\theta}) = \int_{\mathbb{T}} p(\boldsymbol{\tau}) R(\boldsymbol{\tau}) d\boldsymbol{\tau}, \quad (1)$$

978-1-4244-2761-1/09/$25.00 ©2009 IEEE

where $\boldsymbol{\tau} = [\mathbf{s}_{1:T+1}, \mathbf{a}_{1:T}]$ denotes a path of states $\mathbf{s}_{1:T+1} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{T+1}]$ and actions $\mathbf{a}_{1:T} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_T]$. The probability of episode $\boldsymbol{\tau}$ is denoted by $p(\boldsymbol{\tau})$ while $R(\boldsymbol{\tau})$ refers to its return. Using the assumptions of Markovness and additive accumulated rewards, we can write

$$p(\boldsymbol{\tau}) = p(\mathbf{s}_1)\textstyle\prod_{t=1}^{T+1} p(\mathbf{s}_{t+1}|s_t, \mathbf{a}_t)\pi(\mathbf{a}_t|\mathbf{s}_t, t), \quad (2)$$

$$R(\boldsymbol{\tau}) = \textstyle\sum_{t=1}^{T} \gamma^{t-1} r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, t), \quad (3)$$

where $p(\mathbf{s}_1)$ denotes the initial state distribution, $p(\mathbf{s}_{t+1}|s_t, \mathbf{a}_t)$ the next state distribution conditioned on last state and action, $r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, t)$ denotes the immediate reward, and $\gamma \in [0, 1]$ the discount factor.

B. Policy Improvement as Reward-weighted Matching

In this section, we discuss how reward-weighted imitation can be used for policy improvement and derive several approaches resulting from that perspective. For doing so, we first discuss the lower bound on the expected return suggested in [12] for guaranteeing that policy update steps are improvements. While [12], [14] discussed only the immediate reward case, we extend their framework to episodic reinforcement learning and, subsequently, derive a general update rule which yields the policy gradient theorem, a generalization of the reward-weighted regression [14] as well as the PoWER algorithm.

Unlike in reinforcement learning, other machine learning branches have focused on optimizing lower bounds, e.g., resulting in expectation-maximization (EM) algorithms [11]. The reasons for this preference apply in policy learning: if the lower bound also becomes an equality for the sampling policy, we can guarantee that the policy will be improved by optimizing the lower bound. Surprisingly, results from supervised learning can be transferred with ease. For doing so, we follow the scenario suggested in [12], i.e., generate episodes $\boldsymbol{\tau}$ using the current policy with parameters $\boldsymbol{\theta}$ which we weight with the returns $R(\boldsymbol{\tau})$ and subsequently match it with a new policy parametrized by $\boldsymbol{\theta}'$. This matching of the success-weighted path distribution is equivalent to minimizing the Kullback-Leibler divergence $D\left(p_{\boldsymbol{\theta}'}(\boldsymbol{\tau}) \| p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) R(\boldsymbol{\tau})\right)$ between the new path distribution $p_{\boldsymbol{\theta}'}(\boldsymbol{\tau})$ and the reward-weighted previous one $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) r(\boldsymbol{\tau})$. As shown in [12], [14], this results in a lower bound on the expected return using Jensen's inequality and the concavity of the logarithm, i.e.,

$$\log J(\boldsymbol{\theta}') = \log \int_{\mathbb{T}} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})}{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})} p_{\boldsymbol{\theta}'}(\boldsymbol{\tau}) R(\boldsymbol{\tau}) d\boldsymbol{\tau}, \quad (4)$$

$$\geq \int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) R(\boldsymbol{\tau}) \log \frac{p_{\boldsymbol{\theta}'}(\boldsymbol{\tau})}{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})} d\boldsymbol{\tau} + \text{const}, \quad (5)$$

$$\propto -D\left(p_{\boldsymbol{\theta}'}(\boldsymbol{\tau}) \| p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) R(\boldsymbol{\tau})\right) = L_{\boldsymbol{\theta}}(\boldsymbol{\theta}'), \quad (6)$$

where $D\left(p(\boldsymbol{\tau}) \| q(\boldsymbol{\tau})\right) = \int p(\boldsymbol{\tau}) \log(p(\boldsymbol{\tau}) / q(\boldsymbol{\tau})) d\boldsymbol{\tau}$ is the Kullback-Leibler divergence which is considered a natural distance measure between probability distributions, and the constant is needed for tightness of the bound. Note that $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) R(\boldsymbol{\tau})$ is an improper probability distribution as pointed out in [12]. Maximizing the lower bound on the expected return $L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = -D\left(p_{\boldsymbol{\theta}'}(\boldsymbol{\tau}) \| p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) R(\boldsymbol{\tau})\right)$ is the essential step when improving a policy and we will show the relation to different previous policy learning methods in Section II-C.

C. Resulting Policy Updates

In the following part, we will discuss three different policy updates which directly result from Section II-B. First, we show that policy gradients [6], [7], [9] can be derived from the lower bound $L_{\boldsymbol{\theta}}(\boldsymbol{\theta}')$. Subsequently, we show that natural policy gradients can be seen as an additional constraint regularizing the change in the path distribution resulting from a policy update when improving the policy incrementally, and, finally, we will show how an expectation-maximization (EM) algorithm for policy learning can be generated.

1) Policy Gradients.: When differentiating the function $L_{\boldsymbol{\theta}}(\boldsymbol{\theta}')$ which defines the lower bound on the expected return, we directly obtain

$$\partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = \textstyle\int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) R(\boldsymbol{\tau}) \partial_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}'}(\boldsymbol{\tau}) d\boldsymbol{\tau}, \quad (7)$$

where $\mathbb{T}$ is the set of all possible paths and $\partial_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}'}(\boldsymbol{\tau}) = \sum_{t=1}^{T} \partial_{\boldsymbol{\theta}} \log \pi(\mathbf{a}_t|\mathbf{s}_t, t)$ denotes the log-derivative of the path distribution. As this log-derivative only depends on the policy, we can estimate a gradient from roll-outs without having a model by simply replacing the expectation by a sum; when $\boldsymbol{\theta}'$ is close to $\boldsymbol{\theta}$, we have the policy gradient estimator is widely known as Episodic REINFORCE [6], i.e., we have

$$\lim_{\boldsymbol{\theta}' \to \boldsymbol{\theta}} \partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta}). \quad (8)$$

Obviously, a reward which precedes an action in an episode, can neither be caused by the action nor cause an action in the same episode. Thus, when inserting Equations (2) into Equation (7), all cross-products between r_t and $\partial_{\boldsymbol{\theta}} \log \pi(\mathbf{a}_{t+\delta t}|\mathbf{s}_{t+\delta t}, t + \delta t)$ for $\delta t > 0$ become zero in expectation [9]. Therefore, we can omit these terms and rewrite the estimator as

$$\partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = E\left\{\sum_{t=1}^{T} \gamma^{t-1} \partial_{\boldsymbol{\theta}} \log \pi(\mathbf{a}_t|\mathbf{s}_t, t) Q^{\pi}(\mathbf{s}, \mathbf{a}, t)\right\}, \quad (9)$$

where

$$Q^{\pi}(\mathbf{s}, \mathbf{a}, t) = E\left\{\sum_{\tilde{t}=t}^{T} \gamma^{\tilde{t}-t} r(\mathbf{s}_{\tilde{t}}, \mathbf{a}_{\tilde{t}}, \mathbf{s}_{\tilde{t}+1}, \tilde{t}) \,\middle|\, \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a}\right\}$$

is called the state-action value function [1]. Equation (9) is equivalent to the policy gradient theorem [7] for $\boldsymbol{\theta}' \to \boldsymbol{\theta}$ in the infinite horizon case where the dependence on time t can be dropped.

When adding an additional punishment to prevent large steps away from the observed path distribution, this can be achieved by restricting the amount of change in the path distribution and, subsequently, determining the steepest descent for a fixed step away from the observed trajectories. Change in probability distributions is naturally measured using the Kullback-Leibler divergence, thus, after adding the additional constraint of $D\left(p_{\boldsymbol{\theta}'}(\boldsymbol{\tau}) \| p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\right) = \delta$, we can derive the natural policy gradient by simply approximating

Input: initial policy parameters $\boldsymbol{\theta_0}$
repeat
Sample: Perform trial(s) using a stochastic policy of
$\mathbf{a} = \boldsymbol{\theta}^{\mathbf{T}}\boldsymbol{\phi}(\mathbf{s},t) + \boldsymbol{\varepsilon}_t$
for generalized Reward-Weighted Regression (RWR) and
$\mathbf{a} = (\boldsymbol{\theta} + \boldsymbol{\varepsilon}_t)^{\mathbf{T}}\boldsymbol{\phi}(\mathbf{s},t)$
for Policy Learning by Weighting Exploration with Returns (PoWER). Collect all $(t, \mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, \boldsymbol{\varepsilon}_t, r_{t+1})$ for all time-steps $t = 1, 2, \ldots, T+1$.
Estimate: Use unbiased estimate
$\hat{Q}^\pi(\mathbf{s},\mathbf{a},t) = \sum_{\tilde{t}=t}^{T} \gamma^{\tilde{t}-t} r(\mathbf{s}_{\tilde{t}}, \mathbf{a}_{\tilde{t}}, \mathbf{s}_{\tilde{t}+1}, \tilde{t})$.
Reweight: Compute importance weights and reweight trials, discard low-importance trials.
Update policy using the RWR update
$\boldsymbol{\theta}' = \left\langle \sum_{t=1}^{T} \gamma^{t-1} \boldsymbol{\phi}_t^{\mathbf{s}} \boldsymbol{\phi}_t^{\mathbf{s}T} Q_t^{\mathbf{as}} \right\rangle_w^{-1} \left\langle \sum_{t=1}^{T} \gamma^{t-1} \boldsymbol{\phi}_t^{\mathbf{s}} \mathbf{a}^T Q_t^{\mathbf{as}} \right\rangle_w$
or for PoWER use
$\boldsymbol{\theta}' = \boldsymbol{\theta} + \left\langle \sum_{t=1}^{T} \gamma^{t-1} Q_t^{\mathbf{as}} \mathbf{W}_t^{\mathbf{s}} \right\rangle_w^{-1} \left\langle \sum_{t=1}^{T} \gamma^{t-1} Q_t^{\mathbf{as}} \mathbf{W}_t^{\mathbf{s}} \boldsymbol{\varepsilon}_t \right\rangle_w$
with $\mathbf{W}_t^{\mathbf{s}} = \boldsymbol{\phi}(\mathbf{s},t)\boldsymbol{\phi}(\mathbf{s},t)^{\mathrm{T}} / (\boldsymbol{\phi}(\mathbf{s},t)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s},t))$.
until Convergence $\boldsymbol{\theta}_{k+1} \approx \boldsymbol{\theta}_k$

Algorithm 1: Sketch of EM Policy learning with both RWR and PoWER.

$$D\left(p_{\boldsymbol{\theta}'}(\boldsymbol{\tau}) \| p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\right) \approx 0.5(\boldsymbol{\theta}' - \boldsymbol{\theta})^T \mathbf{F}(\boldsymbol{\theta})(\boldsymbol{\theta}' - \boldsymbol{\theta}) \quad (10)$$

with its second-order expansion where $\mathbf{F}(\boldsymbol{\theta})$ denotes the Fisher information matrix [8]. This derivation results in the Natural Actor Critic as discussed in [8], [9].

2) Policy Search via Expectation Maximization.: One major drawback of gradient-based approaches is the learning rate, an open parameter which can be hard to tune in control problems but is essential for good performance. Expectation-Maximization algorithms are well-known to avoid this problem in supervised learning while even yielding second-order convergence [11]. Previously, similar ideas have been explored in immediate reinforcement learning [12], [14]. In general, an EM-algorithm would choose the next policy parameters $\boldsymbol{\theta}_{n+1}$ such that $\boldsymbol{\theta}_{n+1} = \mathrm{argmax}_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}')$. In the case where $\pi(\mathbf{a}_t|\mathbf{s}_t,t)$ belongs to the exponential family, the next policy can be determined analytically by setting Equation (9) to zero, i.e.,

$$E\left\{\sum_{t=1}^{T} \gamma^{t-1} \partial_{\boldsymbol{\theta}'} \log \pi(\mathbf{a}_t|\mathbf{s}_t,t) Q^\pi(\mathbf{s},\mathbf{a},t)\right\} = 0, \quad (11)$$

and solving for $\boldsymbol{\theta}'$. Depending on the choice of a stochastic policy, we will obtain different solutions and different learning algorithms. It allows the extension of the reward-weighted regression to larger horizons as well as the introduction of the PoWER algorithm.

In most learning control problems, we attempt to have a deterministic mean policy $\bar{\mathbf{a}} = \boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{s},\mathbf{t})$ with parameters $\boldsymbol{\theta}$ and basis functions $\boldsymbol{\phi}$, e.g., in linear-quadratic regulation we have gains as $\boldsymbol{\theta}$ and states as $\boldsymbol{\phi}$. In Section III, we will introduce the basis functions of the motor primitives. When learning motor primitives, we turn this deterministic mean policy $\bar{\mathbf{a}} = \boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{s},\mathbf{t})$ into a stochastic policy using additive exploration $\boldsymbol{\varepsilon}(\mathbf{s},t)$ in order to make model-free reinforcement learning possible, i.e., we always intend to have a policy $\pi(\mathbf{a}_t|\mathbf{s}_t,t)$ which can be brought into the form

$$\mathbf{a} = \boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{s},t) + \epsilon(\boldsymbol{\phi}(\mathbf{s},t)). \quad (12)$$

Previous work in this context [6], [4], [9], [14] has focused on state-independent, white Gaussian exploration, i.e., $\epsilon(\boldsymbol{\phi}(\mathbf{s},t)) \sim \mathcal{N}(0,\Sigma)$. It is straightforward obtain the Reward-Weighted Regression for episodic RL by solving Equation (11) for $\boldsymbol{\theta}'$ which naturally yields

$$\boldsymbol{\theta}' = E\left\{\sum_{t=1}^{T} \gamma^{t-1} \boldsymbol{\phi}_t^{\mathbf{s}} \boldsymbol{\phi}_t^{\mathbf{s}T} Q_t^{\mathbf{as}}\right\}^{-1} E\left\{\sum_{t=1}^{T} \gamma^{t-1} \boldsymbol{\phi}_t^{\mathbf{s}} \mathbf{a}^T Q_t^{\mathbf{as}}\right\},$$

i.e., a weighted regression method with the state-action values $Q_t^{\mathbf{as}} = Q^\pi(\mathbf{s},\mathbf{a},t)$ as weights and abbreviating the basis functions by $\boldsymbol{\phi}_t^{\mathbf{s}} = \boldsymbol{\phi}(\mathbf{s},t)$.

However, such unstructured exploration at every step has a multitude of disadvantages: it causes a large variance which grows with the number of time-steps [9], it perturbs actions too frequently 'washing' out their effects and can damage the system executing the trajectory. As a result, all methods relying on this state-independent exploration have proven too fragile for learning the Ball-in-a-Cup task on a real robot system. Rückstieß et al. [13] suggested that as an alternative, one could generate a form of structured, state-dependent exploration $\boldsymbol{\epsilon}(\boldsymbol{\phi}(\mathbf{s},t)) = \boldsymbol{\varepsilon}_t^T \boldsymbol{\phi}(\mathbf{s},t)$ with $[\boldsymbol{\varepsilon}_t]_{ij} \sim \mathcal{N}(0, \sigma_{ij}^2)$, where σ_{ij}^2 are meta-parameters of the exploration that can also be optimized. As suggested in [13], this results into the policy $\mathbf{a} \sim \pi(\mathbf{a}_t|\mathbf{s}_t,t) = \mathcal{N}(\mathbf{a}|\boldsymbol{\theta}\boldsymbol{\phi}(\mathbf{s},\mathbf{t}), \hat{\boldsymbol{\Sigma}}(\mathbf{s},t))$. Inserting Rückstieß's policy into Equation (11), we obtain the optimality condition update in the sense of Equation (11) and can derive the update rule

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + E\left\{\sum_{t=1}^{T} \gamma^{t-1} \mathbf{W}_t^{\mathbf{s}} Q_t^{\mathbf{as}}\right\}^{-1} E\left\{\sum_{t=1}^{T} \gamma^{t-1} Q_t^{\mathbf{as}} \mathbf{W}_t^{\mathbf{s}} \boldsymbol{\varepsilon}_t\right\}$$

with $\mathbf{W}_t^{\mathbf{s}} = \boldsymbol{\phi}(\mathbf{s},t)\boldsymbol{\phi}(\mathbf{s},t)^{\mathrm{T}} / (\boldsymbol{\phi}(\mathbf{s},t)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{s},t))$. Note that for our motor primitives $\mathbf{W}$ reduces to a diagonal, constant matrix and cancels out. Hence the algorithm can be applied in a simplified form in Section III. In order to reduce the number of trials in this on-policy scenario, we reuse the trials through importance sampling as described in the context of reinforcement learning in [1]. To avoid the fragility sometimes resulting from importance sampling in reinforcement learning, samples with very small importance weights are discarded. Replacing the expectations $E\{\cdot\}$ by the importance sampler denoted by $\langle\cdot\rangle_{w(\boldsymbol{\tau})}$. The resulting algorithms is shown in Algorithm 1.

III. Application I: Motor Primitive Learning

In this section, we demonstrate the effectiveness of the presented algorithms in the context of motor primitive learning for robotics. For doing so, we will first give a quick overview how the motor primitives work and how the algorithm can be used to adapt them.

As first evaluation, we will show that the novel presented PoWER algorithm outperforms all previous well-known methods, i.e., 'Vanilla' Policy Gradients, Finite Difference Gradients, the Episodic Natural Actor Critic and the

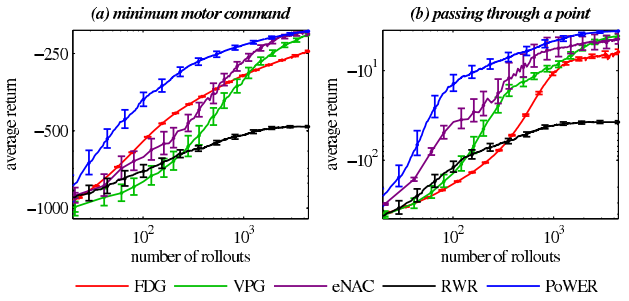


Fig. 1. This figure shows the mean performance of all compared methods in two benchmark tasks averaged over twenty learning runs with the error bars indicating the standard deviation. Policy learning by Weighting Exploration with the Returns (PoWER) clearly outperforms Finite Difference Gradients (FDG), 'Vanilla' Policy Gradients (VPG), the Episodic Natural Actor Critic (eNAC) and the adapted Reward-Weighted Regression (RWR) for both tasks.

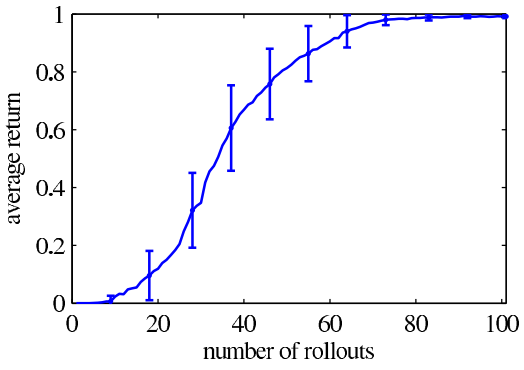


Fig. 2. This figure shows the expected return of the learned policy in the Ball-in-a-Cup evaluation averaged over 20 runs.

generalized Reward-Weighted Regression on the two simulated benchmark problems suggested in [9] and a simulated Underactuated Swing-Up [16].

Real robot applications are done with our best benchmarked method, the PoWER method. Here, we show PoWER can be used to learn learn a high-speed Ball-in-a-Cup [20] movement with motor primitives for all seven degrees of freedom of our Barrett WAM™ robot arm.

A. Using the Motor Primitives in Policy Search

The motor primitive framework [18], [19] can be described as two coupled differential equation, i.e., we have a canonical system $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, \mathbf{z})$ with movement phase $\mathbf{y}$ and possible external coupling to $\mathbf{z}$ as well as a nonlinear system $\ddot{\mathbf{x}} = \mathbf{g}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{y}, \boldsymbol{\theta})$ which yields the current action for the system. Both dynamical systems are chosen to be stable and to have the right properties so that they are useful for the desired class of motor control problems. In this paper, we focus on single stroke movements as they frequently appear in human motor control [10], [19] and, thus, we will always choose the point attractor version of the motor primitives exactly as presented in [19] and not the older one in [18].

The biggest advantage of the motor primitive framework of [18], [19] is that the function $\mathbf{g}$ is linear in the policy parameters $\boldsymbol{\theta}$ and, thus, well-suited for imitation learning as well as for our presented reinforcement learning algorithm. For example, if we would have to learn only a motor primitive for a single degree of freedom q_i, then we could use a motor primitive in the form $\bar{\ddot{q}}_i = g(q_i, \dot{q}_i, y, \boldsymbol{\theta}) = \boldsymbol{\phi}(\mathbf{s})^T \boldsymbol{\theta}$ where $\mathbf{s} = [q_i, \dot{q}_i, y]$ is the state and where time is implicitly embedded in y. We use the output of $\bar{\ddot{q}}_i = \boldsymbol{\phi}(\mathbf{s})^T \boldsymbol{\theta} = \bar{a}$ as a policy mean of Equation (12). The perturbed accelerations $\ddot{q}_i = a = \bar{a} + \varepsilon$ is given to the system. The details of $\boldsymbol{\phi}$ are given in [19].

In Section III-C, we use imitation learning for the initialization. For imitations, we follow [18]: first, extract the duration of the movement from initial & final zero velocity and use it to adjust the time constants. Second, use locally-weighted regression to solve for an imitation from a single example.

B. Benchmark Comparison

As benchmark comparison, we intend to follow a previously studied scenario in order to show which method is most well-suited for our problem class. For doing so, we perform our evaluations on the exact same benchmark problems as [9] (i.e., the discount factor is $\gamma = 1$) and use two tasks commonly studied in motor control literature for which the analytic solutions are known, i.e., a reaching task where a goal has to be reached at a certain time while the used motor commands have to be minimized and a reaching task of the same style with an additional via-point. In this comparison, we mainly want to show the suitability of our algorithm and show that it outperforms previous methods such as Finite Difference Gradient (FDG) methods [9], 'Vanilla' Policy Gradients (VPG) with optimal baselines [6], [7], [9], the Episodic Natural Actor Critic (eNAC) [8], [9], and the episodic version of the Reward-Weighted Regression (RWR) algorithm [14]. For both tasks, we use the same rewards as in [9] but we use the newer form of the motor primitives from [19]. All open parameters were manually optimized for each algorithm in order to maximize the performance while not destabilizing the convergence of the learning process.

When applied in the episodic scenario, Policy learning by Weighting Exploration with the Returns (PoWER) clearly outperformed the Episodic Natural Actor Critic (eNAC), 'Vanilla' Policy Gradient (VPG), Finite Difference Gradient (FDG) and the adapted Reward-Weighted Regression (RWR) for both tasks. The episodic Reward-Weighted Regression (RWR) is outperformed by all other algorithms suggesting that this algorithm does not generalize well from the immediate reward case. While FDG gets stuck on a plateau, both eNAC and VPG converge to the same, good final solution. PoWER finds the same (or even slightly better) solution while achieving it noticeably faster. The results are presented in Figure 1. Note that this plot has logarithmic scales on both axes, thus a unit difference corresponds to an order of magnitude. Please excuse the omission of the first twenty roll-outs to cope with the log-log presentation.

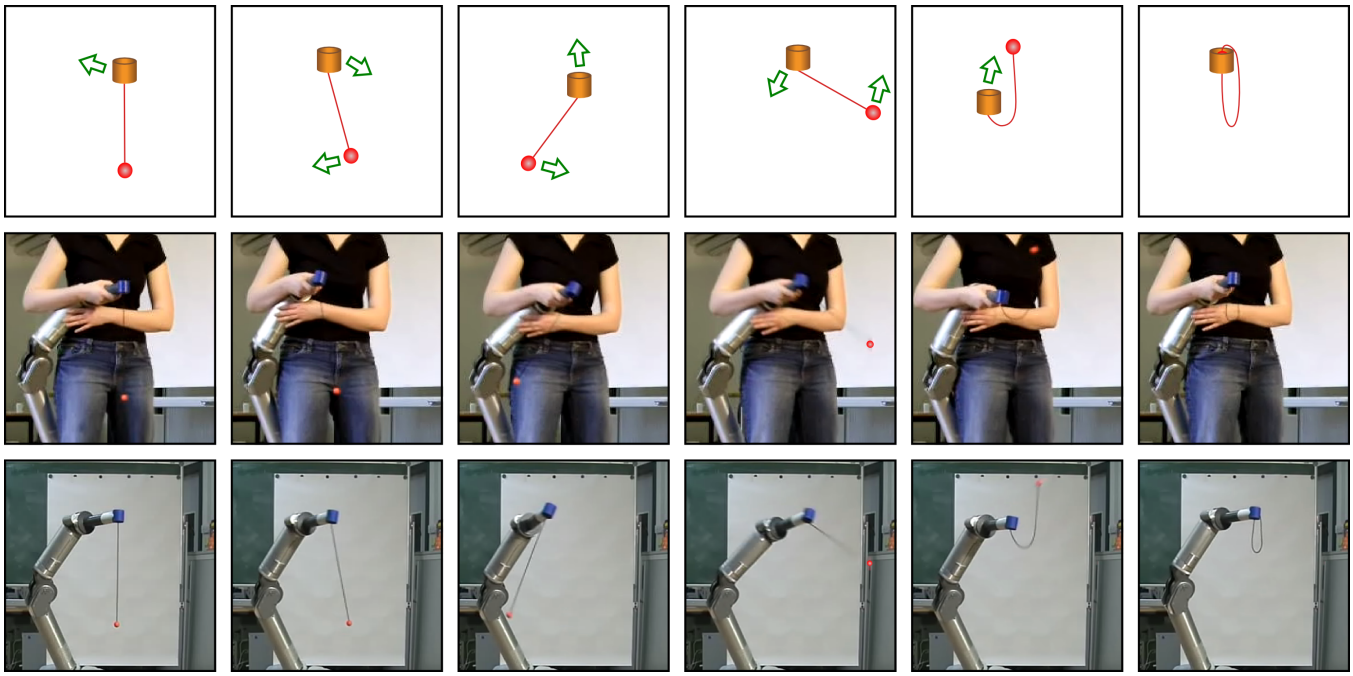


Fig. 3. This figure shows schematic drawings of the Ball-in-a-Cup motion, the final learned robot motion as well as a motion-captured human motion. The green arrows show the directions of the momentary movements. The human cup motion was taught to the robot by imitation learning with 31 parameters per joint for ca. 3 seconds. The inverse dynamic manages to reproduce the simplified motion quite accurately, but the ball misses the cup by several centimeters. After approximately 75 iterations of our Policy learning by Weighting Exploration with the Returns (PoWER) algorithm the robot has improved its motion to a point the ball goes in the cup. Also see Figure 2.

C. Ball-in-a-Cup on a Barrett WAM™

The most challenging application in this paper the children's game Ball-in-a-Cup [20] where a small cup is attached at the robot's end-effector and this cup has a small wooden ball hanging down from the cup on a 40cm string. Initially, the ball is hanging down vertically. The robot needs to move fast in order to induce a motion at the ball through the string, swing it up and catch it with the cup, a possible movement is illustrated in Figure 3 (top row). The state of the system is described in joint angles and velocities of the robot and the Cartesian coordinates of the ball. The actions are the joint space accelerations where each of the seven joints is represented by a motor primitive. All motor primitives are perturbed separately but employ the same joint final reward given by $r(t_c) = \exp(-\alpha(x_c - x_b)^2 - \alpha(y_c - y_b)^2)$ while $r(t) = 0$ for all other $t \neq t_c$ where t_c is the moment where the ball passes the rim of the cup with a downward direction, the cup position denoted by $[x_c, y_c, z_c] \in \mathbb{R}^3$, the ball position $[x_b, y_b, z_b] \in \mathbb{R}^3$ and a scaling parameter $\alpha = 100$. The task is quite complex as the reward is not modified solely by the movements of the cup but foremost by the movements of the ball and the movements of the ball are very sensitive to changes in the movement. A small perturbation of the initial condition or during the trajectory will drastically change the movement of the ball and hence the outcome of the trial.

Due to the complexity of the task, Ball-in-a-Cup is even a hard motor learning task for children who usually only succeed at it by observing another person playing and a lot of improvement by trial-and-error. Mimicking how children learn to play Ball-in-a-Cup, we first initialize the motor primitives by imitation and, subsequently, improve them by reinforcement learning. We recorded the motions of a human player by kinesthetic teach-in in order to obtain an example for imitation as shown in Figure 3 (middle row). From the imitation, it can be determined by cross-validation that 31 parameters per motor primitive are needed. As expected, the robot fails to reproduce the the presented behavior and reinforcement learning is needed for self-improvement.

Figure 2 shows the expected return over the number of episodes where convergence to a maximum is clearly recognizable. The robot regularly succeeds at bringing the ball into the cup after approximately 75 iterations.

IV. Application II: Task-Space Control

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 velocity $\dot{\mathbf{p}} \in \mathbb{R}^m$, than in the complicated manifold of joint-space configuration $\dot{\mathbf{q}} \in \mathbb{R}^n$. For this reason, the transformation of task-space movements into joint space movements, i.e., the differential inverse kinematics, is an essential step for motor command generation [21].

Learning task-space control has a variety of interesting properties in comparison to analytical approaches as it can deal with camera calibration, sensor offsets and measurements errors [22], [23]. 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 [22], [23].

A. Task-Space Control as a Reinforcement Learning Problem

While various successful approaches exist to the offline learning of inverse kinematics, see [23] 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 [22], 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 [22], 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 [14].

The forward kinematics given by the transformation $\mathbf{p} = \mathbf{f}_{\text{Kinematics}}(\mathbf{q})$ is straightforward to compute and can be estimated using stereo vision [21]. 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 proximity to the current joint configuration. For doing so, the velocity of the end-effector in task space $\dot{\mathbf{p}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$ is considered, where $\mathbf{J}(\mathbf{q}) = d\mathbf{f}_{\text{Kinematics}}(\mathbf{q})/d\mathbf{q}$ denotes the Jacobian.

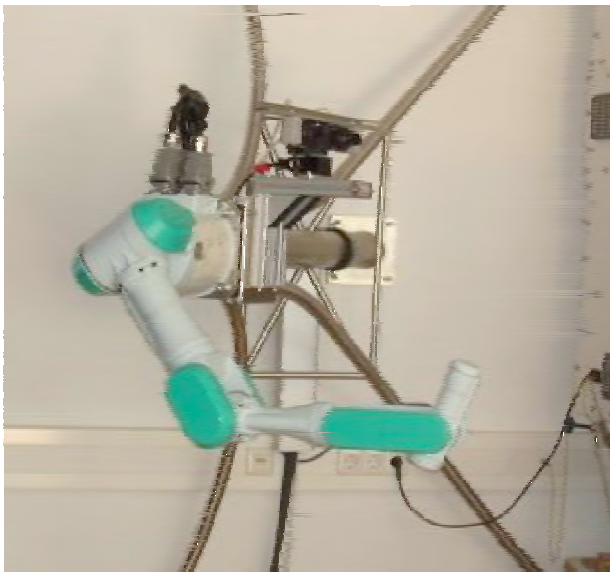
An important insight into redundant task-space control problems (see [25], [24], [14]) is that resolved motion control can be derived as the solution of a constraint optimization problem given by

$$\min_{\dot{\mathbf{q}}} C_0(\dot{\mathbf{q}}) = \dot{\mathbf{q}}^T \mathbf{N} \dot{\mathbf{q}} \text{ s.t. } \mathbf{J}\dot{\mathbf{q}} = \dot{\mathbf{p}}_{\text{ref}}, \tag{13}$$

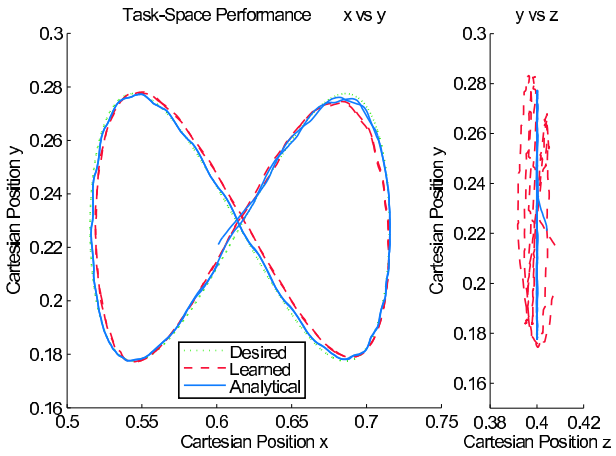
where $\mathbf{N}$ denotes a positive definite metric weighting the joint velocity distributing the movement onto the joints, and $\dot{\mathbf{p}}_{\text{ref}} = \dot{\mathbf{p}}_d(t) + \mathbf{K}_p^R(\mathbf{p}_d(t) - \mathbf{p}(t))$ presents a reference attractor in task space with gain matrix $\mathbf{K}_p$ and desired task-space trajectory $\mathbf{p}_d$, $\dot{\mathbf{p}}_d$. If we treat this problem as a reinforcement learning problem, we have a comparably simple setup (especially in the light of the complex supervised learning solution in [22]). First, all actions which we do will yield us samples which fulfill the constraint. Second, we can turn the cost into a reward by simply using a reward transformation $r(\dot{\mathbf{q}}) = \exp(-h\dot{\mathbf{q}}^T\mathbf{N}\dot{\mathbf{q}})$. As a result, we have an immediate reward reinforcement learning problem for which the reward-weighted regression can be applied perfectly. Future experiments with PoWER are currently under evaluation.

B. Experimental Evaluation

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 4(a). This robot has seven degrees of freedom (DoF) and, thus, has four degrees of freedom too much



(a) Mitsubishi PA 10



(b) Task-Space Tracking

Fig. 4. This figure shows (a) the Mitsubishi PA-10 robot arm with seven degrees of freedom used in the experiments in this paper and (b) illustrates the task performance of both the analytical and the learned resolved motion rate task space 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 motion rate control law. Note, that while the online learning solution is as good as the analytical solution, it still yields comparable performance without any pre-training of the local control laws before the online learning.

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.

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 [24]. 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. In order to turn a cost into

a reward, we again transform it by $r(\dot{\mathbf{q}}) = \exp(-\dot{\mathbf{q}}^T \mathbf{N} \dot{\mathbf{q}})$.

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 $\varepsilon \sim \mathcal{N}\left(0, \sigma^2 \mathbf{I}\right)$ with $\sigma = 0.001$, i.e.,

$$\dot{\mathbf{q}} = \pi(\mathbf{q}, \dot{\mathbf{p}}_{\text{ref}}) + \varepsilon. \tag{14}$$

Such exploration is known under the name motor babbling in the behavioral literature and helps humans learn motor tasks in a similar manner [23]. 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 without a magnifying glass, it nevertheless has an impact as it causes the robot to slightly drift in the null-space of the task during execution.

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 4(b), 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.

V. Conclusion

In this paper, we have presented a new perspective on policy learning methods and an application to a highly complex motor learning task on a real Barrett WAM™ robot arm. We have generalized the previous work in [12], [14]. In the process, we could show that policy gradient methods are a special case of this more general framework. We have derived two reinforcement learning methods from this framework, i.e., the generalized Reward-Weighted Regression and Policy Learning by Weighting Exploration with the Returns. Both have been used successful in the application to robot reinforcement learning. The PoWER algorithm has been used in the context of motor primitive learning of Ball-in-a-Cup on a real Barrett WAM. We have also shown that the Reward-Weighted Regression can be used to for online, real-time reinforcement learning of resolved velocity task-space control on a Mitsubishi PA-10 in our lab.

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 hopefully be beneficial in developing more autonomous and self-tuning robotic systems.

References

[1] R. Sutton and A. Barto. *Reinforcement Learning*. MIT Press, 1998.

[2] J. Bagnell, S. Kadade, A. Ng, and J. Schneider. Policy search by dynamic programming. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[3] A. Ng and M. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *International Conference on Uncertainty in Artificial Intelligence (UAI)*, 2000.

[4] F. Guenter, M. Hersch, S. Calinon, and A. Billard. Reinforcement learning for imitating constrained reaching movements. *RSJ Advanced Robotics*, 21, 1521-1544, 2007.

[5] M. Hoffman, A. Doucet, N. de Freitas, and A. Jasra. Bayesian policy learning with trans-dimensional MCMC. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[6] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[7] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, 2000.

[8] J. Bagnell and J. Schneider. Covariant policy search. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[9] J. Peters and S. Schaal. Policy gradient methods for robotics. In *International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[10] G. Wulf. *Attention and motor skill learning*. Human Kinetics, Champaign, IL, 2007.

[11] G. J. McLachan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley Series in Probability and Statistics. John Wiley & Sons, 1997.

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

[13] T. Rückstieß, M. Felder, and J. Schmidhuber. State-Dependent Exploration for Policy Gradient Methods. In *Proceedings of the European Conference on Machine Learning (ECML), Antwerp, Belgium, 2008.*

[14] J. Peters and S. Schaal. Learning operational space control. In Proceedings of Robotics: Science and Systems (RSS)*, Philadelphia, PA, 2006.*

[15] M. Kawato, F. Gandolfo, H. Gomi, and Y. Wada. Teaching by showing in kendama based on optimization principle. In International Conf. on Artificial Neural Networks*, 1994.*

[16] C. G. Atkeson. Using local trajectory optimizers to speed up global optimization in dynamic programming. In Advances in Neural Information Processing Systems (NIPS)*, 1994.*

[17] J. Kober,J. Peters. Policy Search of Motor Primitives for Robotics. In Advances in Neural Information Processing Systems (NIPS)*, 2008.*

[18] A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In Advances in Neural Information Processing Systems (NIPS)*, 2003.*

[19] S. Schaal, P. Mohajerian, and A. Ijspeert. Dynamics systems vs. optimal control — a unifying view. Progress in Brain Research*, 165(1):425–445, 2007.*

[20] Wikipedia, May 31, 2008. http://en.wikipedia.org/wiki/Ball_in_a_cup

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

[22] 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.*

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

[24] J. Nakanishi, M. Mistry, J. Peters, and S. Schaal. Operational space control: A theoretical and emprical comparison. In International Journal of Robotics Research (IJRR)*, pp.737–757, Vol 27 (6), 2008.*

[25] J. Peters, M. Mistry, F. Udwadia, R.Cory, J. Nakanishi, and S. Schaal. A unifying methodology for robot control with redundant DOFs In Autonomous Robots*, pp.1–12, Vol 24 (1), 2008.*