# Online Incremental Learning of Inverse Dynamics Incorporating Prior Knowledge

Joseph Sun de la Cruz, * Dana Kulić and William Owen†

Department of Electrical and Computer Engineering
†Department of Mechanical and Mechatronics Engineering
University of Waterloo
Waterloo, ON, Canada
{jsundela,dkulic,bowen}@uwaterloo.ca

**Abstract.** Recent approaches to model-based manipulator control involve data-driven learning of the inverse dynamics relationship of a manipulator, eliminating the need for any knowledge of the system model. Ideally, such algorithms should be able to process large amounts of data in an online and incremental manner, thus allowing the system to adapt to changes in its model structure or parameters. Locally Weighted Projection Regression (LWPR) and other non-parametric regression techniques have been applied to learn manipulator inverse dynamics. However, a common issue amongst these learning algorithms is that the system is unable to generalize well outside of regions where it has been trained. Furthermore, learning commences entirely from 'scratch,' making no use of any a-priori knowledge which may be available. In this paper, an online, incremental learning algorithm incorporating prior knowledge is proposed. Prior knowledge is incorporated into the LWPR framework by initializing the local linear models with a first order approximation of the available prior information. It is shown that the proposed approach allows the system to operate well even without any initial training data, and further improves performance with additional online training.

**Keywords:** Robotics, Learning, Control

## 1 Introduction

Control strategies that are based on the knowledge of the dynamic model of the robot manipulator, known as model-based controllers, can present numerous advantages such as increased performance during high-speed movements, reduced energy consumption, improved tracking accuracy and the possibility of compliance [1]. However, this performance is highly dependant upon the accurate representation of the robot's dynamics, which includes precise knowledge of the inertial parameters of link mass, centre of mass and moments of inertia, and friction parameters [2]. In practice, obtaining such a model is a challenging task

* This research is funded in part by the Natural Sciences and Engineering Research Council of Canada

which involves modeling physical processes that are not well understood or difficult to model, such as friction [3] and backlash. Thus, assumptions concerning these effects are often made to simplify the modeling process, leading to inaccuracies in the model. Furthermore, uncertainties in the physical parameters of a system may be introduced from discrepancies between the manufacturer data and the actual system [4]. Changes to operating conditions can also cause the structure of the system model to change, thus resulting in degraded performance.

Traditionally, adaptive control strategies have been used to cope with parameter uncertainty [2], [5]. This allows parameters of the dynamic model such as end effector load, inertia and friction to be estimated online [2]. However, such methods are still reliant upon adequate knowledge of the structure of the dynamic model and are thus susceptible to modeling errors and changes in the model structure. Furthermore, accurate estimation of the model parameters requires sufficiently rich data sets [2],[6], and even when available, may result in physically inconsistent parameters meaning that constraints must be applied to the identified parameters [2],[7].

Newer approaches to manipulator control involve data-driven learning of the inverse dynamics relationship of a manipulator, thus eliminating the need for any a-priori knowledge of the system model. Unlike the adaptive control strategy, these approaches do not assume an underlying structure but rather attempt to infer the optimal structure to describe the observed data. Thus, it is possible to encode nonlinearities whose structure may not be well-known. Solutions to this form of supervised learning approach can be broadly categorized into two types [8]. Global methods such as Gaussian Process Regression (GPR) [9] and Support Vector Regression (SVR) [10], and local methods such as Locally Weighted Projection Regression (LWPR). Recent studies comparing these learning methods [11] show that while SVR and GPR can potentially yield higher accuracy than LWPR, their computational cost is still prohibitive for online incremental learning. Furthermore, a major issue of online learning approaches such as LWPR is the failure to generate appropriate control signals away from the trained region of the workspace [12].

Central to the learning approaches introduced thus far is the need for large amounts of relevant training data in order to yield good prediction performance. These algorithms most often assume that there is no prior knowledge of the system dynamics, and thus learning commences entirely from 'scratch', making no use of the rigid body dynamics (RBD) model from the well-established field of analytical robotics [1], which provide a global characterization of the dynamics. Recent research [13] has been done to incorporate the full RBD model into the GPR algorithm to improve its performance in terms of generalization and prediction accuracy in the context of real-time robot control. However, in [13], the high computational requirements of GPR still prohibit incremental online updates from being made, which is a highly desirable feature of any learning algorithm for robot control [14].

In many cases, partial information about the robot dynamics may be available, such as for example, the link masses, which may be helpful in bootstrapping

a learning model and accelerating the rate of learning. This paper proposes a novel approach for on-line, incremental learning for model-based control, capable of incorporating full or partial a-priori model knowledge. This is done by using first order approximations of the RBD equation to initialize the local models of the LWPR technique. The developed algorithm can be applied both when full knowledge of the RBD model is available, and also when only partial information (e.g. only the gravity loading vector) is known.

The remainder of this paper is organized as follows. In section 2, the dynamic model of a typical rigid body manipulator and model based control strategies are reviewed. Section 3 briefly overviews the previously developed LWPR algorithm for learning control. Section 4 presents the proposed online, incremental algorithm for incorporating full or partial a-priori knowledge into LWPR. Section 5 presents the simulations and results. Finally, in Section 6, the conclusions and next steps are discussed.

## 2 Overview of Model-Based Control

The rigid body dynamic equation (RBD) of a manipulator characterizes the relationship between its motion (position, velocity and acceleration) and the joint torques that cause these motions [1]:

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

where $\mathbf{q}$ is the $n$x1 vector of generalized coordinates consisting of the $n$ joint angles for an $n$-degree of freedom (DOF) manipulator, $\mathbf{M}(\mathbf{q})$ is the $nxn$ inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the $n$x1 Coriolis and centripetal force vector, $\mathbf{G}(\mathbf{q})$ is the $n$x1 gravity loading vector and $\boldsymbol{\tau}$ is the $n$x1 torque vector.

Equation (1) does not include additional torque components caused by friction, backlash, actuator dynamics and contact with the environment. If accounted for, these components are modeled as additional terms in (1).

Model-based controllers are a broad class of controllers which apply the joint space dynamic equation (1) to cancel the nonlinear and coupling effects of the manipulator. A common example of this is the computed torque approach [2],[1] in which the control signal $\mathbf{u}$ is composed of the computed torque signal, $\mathbf{u_{CT}}$, which is set to the torque determined directly from the inverse of the dynamic equations (1). This term globally linearizes and decouples the system, and thus a linear controller can be applied for the feedback term, $\mathbf{u_{FB}}$, which stabilizes the system and provides disturbance rejection. Typically a Proportional-Derivative (PD) feedback scheme is used for this term.

Desirable performance of the computed torque approach is contingent upon the assumption that the values of the parameters in the dynamic model (1) match the actual parameters of the physical system. Otherwise, imperfect cancelation of the nonlinearities and coupling in (1) occurs. Hence, the resulting system is not fully linearized and decoupled and thus higher feedback gains are necessary to achieve good performance.

## 3 Learning Inverse Dynamics

Learning inverse dynamics involves the construction of a model directly from measured data. The problem of learning the inverse dynamics relationship in the joint space can be described as the map from joint positions, velocities and accelerations to torques

$$(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}_\mathbf{d}) \mapsto \boldsymbol{\tau} \tag{2}$$

where $\boldsymbol{\tau}$ is the $n$x1 torque vector, and $\mathbf{q}$ is the $n$x1 vector of generalized coordinates. This means that the mapping has an input dimensionality of $3n$ and an output dimensionality of $n$.

### 3.1 Locally Weighted Projection Regression

Locally Weighted Projection Regression (LWPR) approximates the nonlinear inverse dynamics equation (1) with a set of piecewise local linear models based on the training data that the algorithm receives. Formally stated, this approach assumes a standard regression model of the form

$$\boldsymbol{y} = f(\boldsymbol{X}) + \boldsymbol{\epsilon} \tag{3}$$

where $\mathbf{X}$ is the input vector, $\mathbf{y}$ the output vector, and $\boldsymbol{\epsilon}$ a zero-mean random noise term. For a single output dimension of $\mathbf{y}$, denoted by $y_i$, given a data point $\mathbf{X_c}$, and a subset of data close to $\mathbf{X_c}$, with the appropriately chosen measure of closeness, a linear model can be fit to the subset of data such that

$$y_i = \boldsymbol{\beta}^T \mathbf{X} + \boldsymbol{\epsilon} \tag{4}$$

where $\boldsymbol{\beta}$ is the set of parameters of the hyperplane that describe $y_i$. The region of validity, termed the receptive field (RF) [8] is given by

$$w_{ik} = \exp(-\frac{1}{2}(\mathbf{X} - \mathbf{X_c})^\mathbf{T} \mathbf{D_{ik}} (\mathbf{X} - \mathbf{X_c})) \tag{5}$$

where $w_{ik}$ determines the weight of the $k^{th}$ local linear model of the $i^{th}$ output dimension (i.e. the $ik^{th}$ local linear model), $\mathbf{X_c}$ is the centre of the $k^{th}$ linear model, $\mathbf{D_{ik}}$ corresponds to a positive semidefinite distance parameter which determines the size and shape of the $ik^{th}$ receptive field. Given a query point $\mathbf{X}$, LWPR calculates a predicted output

$$\hat{y}_i(X) = \sum_{k=1}^{K} w_{ik} \hat{y}_{ik} / \sum_{k=1}^{K} w_{ik} \tag{6}$$

where $K$ is the number of linear models, $\hat{y}_{ik}$ is the prediction of the $ik^{th}$ local linear model given by (4) which is weighed by $w_{ik}$ associated with its receptive field. Thus, the prediction $\hat{y}_i(\mathbf{X})$ is the weighted sum of all the predictions of the

local models, where the models having receptive fields centered closest to the query point are most significant to the prediction. This prediction is repeated $i$ times for each dimension of the output vector $\mathbf{y}$.

Determining the set of parameters $\boldsymbol{\beta}$ of the hyperplane is done via regression, but can be a time consuming task in the presence of high-dimensional input data, which is a characteristic of large numbers of DOF in robotic systems. To reduce the computational effort, LWPR assumes that the data can be characterized by local low-dimensional distributions, and attempts to reduce the dimensionality of the input space $\mathbf{X}$ using Partial Least Squares regression (PLS). PLS fits linear models using a set of univariate regressions along selected projections in input space which are chosen according to the correlation between input and output data [15].

The distance parameter, $\mathbf{D}$, (5) can be learned for each local model through stochastic gradient descent given by

$$\mathbf{d}^{n+1} = \mathbf{d}^n - a\frac{\partial J_{cost}}{\partial \mathbf{d}} \tag{7}$$

where $a$ is the learning rate for gradient descent, $\mathbf{D} = \mathbf{d}^\mathbf{T}\mathbf{d}$ and $J_{cost}$ is a penalized leave-one-out cross-validation cost function which addresses the issue of over-fitting of the data [16]. The number of receptive fields is also automatically adapted [8]. Receptive fields are created if for a given training data point, no existing receptive field possesses a weight $w_i$ (5) that is greater than a threshold value of $w_{gen}$, which is a tunable parameter. The closer $w_{gen}$ is set to one, the more overlap there will be between local models. Conversely, if two local models produce a weight greater than a threshold $w_{prune}$, the model whose receptive field is smaller is pruned.

Because training points are not explicitly stored but rather encoded into simple local linear models, the computational complexity of LWPR remains low, and is not affected by the number of training points. Assuming that the number of projection directions found by the PLS algorithm remains small and bounded, the complexity of the basic algorithm varies linearly with input dimension [8].

## 4 Incorporating a-priori knowledge into LWPR

Despite its ability to perform online, incremental updates due to its low computational cost [8],[11], as a result of the local learning approach of LWPR, performance deteriorates quickly as the system moves outside of the region of state space it has been trained in [11],[12]. In order to improve the generalization performance of LWPR, a-priori knowledge from the RBD model (1) is incorporated into the LWPR algorithm as a method of initializing the system. This is done by initializing the receptive fields in the LWPR model with a first order approximation of the available RBD model.

$$\boldsymbol{\beta} \leftarrow \frac{\partial \tau}{\partial x}\Big|_{x=x_*} \tag{8}$$

where $x_*$ is a new query point for which no previous training data has been observed and $\tau(x_*)$ is the known or partially known dynamics. For a query point $x_*$ and known RBD equation $\tau(x_*)$, the algorithm shown in Figure 1 for initializing the LWPR model is added to the standard prediction procedure of LWPR:

**if** there is no existing RF centered near $x_*$ **then**
  Compute $(\beta)$ according to (8)
  Initialize a new RF centered at $x_*$ with hyperparameters $\boldsymbol{\beta}$
**else**
  Compute prediction with standard LWPR procedure (6)
**end if**

**Fig. 1.** LWPR Initialization Pseudocode

The determination of whether an existing RF is centered near $x_*$ is made in the same way as determining whether a new RF should be added, as described in Section 3.1, i.e., if there is no existing RF that produces a weight (5) greater than $w_{gen}$ given $x_*$, the initialization procedure is applied.

By evaluating the partial derivative of the full (1) or partially known RBD equation at the query point, a first-order linear approximation of the system dynamics is obtained, and is used to initialize the model at that point. The size of the receptive field is initially set as a tunable parameter, and is eventually learned optimally through gradient descent [8].

## 5 Simulations

The proposed approach is evaluated in simulation on a 6-DOF Puma 560 robot using the Robotics Toolbox (RTB) [17]. The open-source LWPR [8] was modified to incorporate full or partial initialization with the RBD model. In order to closely simulate the nonlinearities present in a physical robot, the effects of Coloumb and viscous friction were accounted for in simulation by the following model:

$$\tau_f = C_f sign(q) + V_f \dot{q} \tag{9}$$

where $\tau_f$ is the torque due to Coulomb and viscous friction, $C_f$ is the Coulomb friction constant, and $V_f$ is the viscous friction constant. The friction constants were obtained from the defaults for the Puma 560 in the RTB.

Furthermore, to simulate the effects imprecise knowledge of the inertial parameters of the robot, a 10% percent error in the inertial parameters of the a-priori knowledge was introduced.

The LWPR algorithm is applied to learn the joint-space mapping in (2). Full a-priori knowledge from the RBD model in (1), as well as partial knowledge

from the gravity loading vector $\mathbf{G}(\mathbf{q})$, is used to initialize each algorithm. Standard computed torque control is also implemented with the same RBD model used to initialize the learning algorithms. Although LWPR incorporates many algorithms which enable the system to automatically adjust its parameters for optimal performance, initial values of these parameters can significantly impact the convergence rate. The initial value for the distance metric $\mathbf{D}$ (5) dictates how large a receptive field is upon initialization. Too small a value of $\mathbf{D}$ (corresponding to large receptive fields) tends to delay convergence while a larger value of $\mathbf{D}$ results in overfitting of the data [8]. The initial value of the learning rate $\boldsymbol{\alpha}$(7) determines the rate at which prediction error converges. Too high a value leads to instability and too low a value leads to a slow convergence. These parameters were generally tuned through a trial-and-error process which involved monitoring the error of the predicted values during the training phase.

Tracking performance of the controllers is evaluated on a 'star-like' asterisk pattern [7] as seen in Figure 2. This trajectory is obtained by first moving in a straight line outwards from a centre point, then retracing back inwards to the centre, repeating this pattern in eight different directions in a sequential manner. This pattern produces high components of acceleration, and thus requires model-based control for tracking accuracy. Two speeds are tested, slow and fast, with the maximum velocities set to 0.5 and 2.5 m/s respectively, where the fast speed corresponds to roughly 80% of the robot's maximum velocity. At low speeds, it is expected that the gravity term $\mathbf{G}(\mathbf{q})$ (1) will be dominant. The high-speed trajectory is used to ensure that the Coriolis and centripetal components, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, (1) of the dynamics are excited to a sufficient extent so that $\mathbf{G}(\mathbf{q})$ is no longer dominant. A numerical inverse kinematics algorithm [18] is used to convert the workspace trajectories into the joint space.

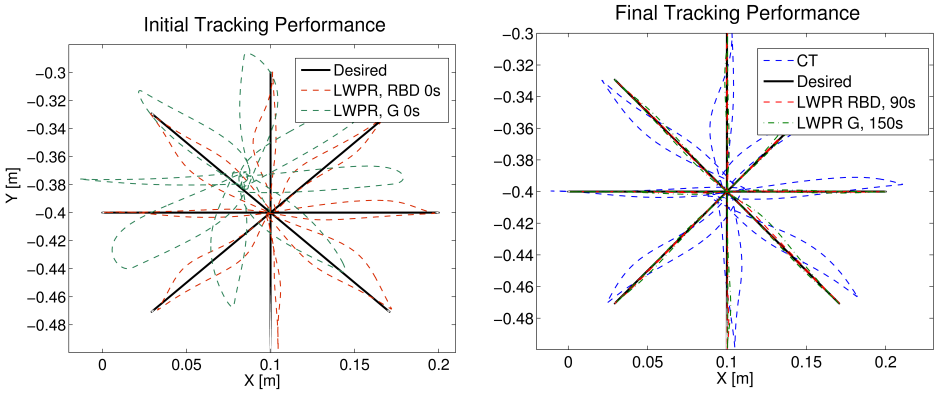


**Fig. 2.** Task Space tracking results for the slow trajectory

Figure 2 depicts the task space tracking performance of the computed torque (CT) controller using a model with 10% error in the inertial and friction param-

eters of the system tracking the slow asterisk trajectory. It is evident that the imprecise knowledge of the RBD model is responsible for poor tracking results. The same RBD model is used to initialize the LWPR model, and the resulting controller is trained online while tracking the slow asterisk trajectory. Table 1 shows the joint space tracking performance after one cycle of the trajectory (approximately 11 seconds). Because the LWPR model is initialized with a set of linear approximations of the RBD equation, the initial performance of LWPR is not as good as the CT method. Table 1 shows the joint space tracking after 90 seconds of training. Due to the high computational efficiency of LWPR, incremental updates are made at a high rate of 400 Hz. As time progresses, LWPR is able to collect more training data and eventually outperforms the CT method by learning the nonlinear behaviour of Coloumb and viscous friction and compensating for the initial inaccurate knowledge of the RBD equation. The model learned on the slow trajectory is then applied to tracking the fast trajectory. The performance suffers only slightly compared to the initial performance on the slow trajectory, as the system was initialized with a full, but slightly inaccurate RBD model. As seen in Table 1, further training for 45 seconds mitigates this issue and results in tracking performance that is nearly as good as compared to that of the slow trajectory. Performance of the CT controller was very similar for both slow and fast trajectories, and is also shown for a comparison.

Table 2 illustrates the joint space tracking performance of the LWPR model when initialized with only the gravity loading vector of the RBD equation while tracking the slow asterisk trajectory. The same table also illustrates the performance after 150 seconds of training. Similarly to the case of full knowledge of the RBD, with sufficient training, LWPR is able to compensate for friction and clearly outperforms the CT controller. However, since the system was initialized with only partial knowledge of the RBD equation, it has taken longer to achieve the same tracking performance in the case of full RBD knowledge. The learned model is once again applied to track the fast trajectory. Compared to the case of full RBD initialization, the performance with only gravity initialization on the faster trajectory is much worse, as seen numerically in Table 2. Performance of the CT controller was very similar for both slow and fast trajectories, and is also shown for a comparison.

**Table 1.** RMS tracking error with full knowledge (deg)

| Joint | 1 | 2 | 3 | 4 | 5 | 6 | Avg |
|---|---|---|---|---|---|---|---|
| Slow Trajectory, Initial | 1.25 | 2.26 | 1.74 | 0.65 | 0.75 | 0.80 | 1.24 |
| Slow Trajectory, 90s | 0.75 | 0.94 | 0.82 | 0.25 | 0.38 | 0.45 | 0.60 |
| Fast Trajectory, Initial | 0.95 | 1.20 | 1.05 | 0.36 | 0.50 | 0.65 | 0.79 |
| Fast Trajectory, 45s | 0.82 | 1.00 | 0.93 | 0.35 | 0.45 | 0.52 | 0.68 |
| CT Controller | 1.10 | 2.05 | 0.95 | 0.55 | 0.65 | 0.62 | 0.98 |

Because the learning of the Coriolis and centripetal term, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, is localized about a specific joint position and velocity, the learned model from the slow trajectory is no longer fully applicable to the fast trajectory. Furthermore, since the system was only initialized with the gravity loading vector, the system performs poorly, and requires significantly more training time to perform well again. This is illustrated in Table 2.

Without the use of a-priori knowledge, learning algorithms were typically initialized with large training data sets obtained through motor babbling [8], [19], [11] in order to achieve decent tracking performance after initialization. By incorporating a-priori knowledge of the RBD equation, whether partial or full, the proposed systems are able to perform reasonably well from the start, even without undergoing such an initialization procedure.

## 6 Conclusions and Future Work

In this paper, an incremental, online learning method is proposed for approximating the inverse dynamics equation while incorporating full or partial knowledge of the RBD equation. The LWPR algorithm was chosen due to its computational efficiency which allows incremental, online learning to occur. In order to improve the generalization performance of the algorithm, prior knowledge was incorporated into the LWPR framework by initializing its local models with a first-order approximation of the RBD equation. In cases where the full RBD is not known due to lack of knowledge of the inertial parameters, partial information (such as the gravity vector) can still be encoded into the model.

**Table 2.** RMS tracking error with partial knowledge (deg)

| | 1 | 2 | 3 | 4 | 5 | 6 | Avg |
|---|---|---|---|---|---|---|---|
| Slow Trajectory, Initial | 2.57 | 4.78 | 3.62 | 0.95 | 0.80 | 0.75 | 2.25 |
| Slow Trajectory, 150s | 0.80 | 0.95 | 0.80 | 0.22 | 0.41 | 0.44 | 0.91 |
| Fast Trajectory, Initial | 2.95 | 5.27 | 4.16 | 1.25 | 1.00 | 0.94 | 2.60 |
| Fast Trajectory, 60s | 0.86 | 1.10 | 0.91 | 0.32 | 0.48 | 0.51 | 0.70 |
| CT Controller | 1.10 | 2.05 | 0.95 | 0.55 | 0.65 | 0.62 | 0.98 |

When this occurs, more training time is required to achieve similar performance as with full RBD initialization. In either case, the generalization performance of LWPR is greatly improved, as the model is able to yield acceptable tracking results without having seen any relevant training data beforehand. It is also shown that this approach is able to compensate for the nonlinear effects of friction, as well as the initial inaccuracy in the inertial parameters. Future work will involve more a more comprehensive evaluation of the proposed methods, including experimental results on physical robots. Secondly, other machine learning algorithms for function approximation will be investigated, such as Gaussian Process Learning [9].

## References


1. L. Sciavicco and B. Scicliano, *Modelling and Control of Robot Manipulators*, 2nd ed. Springer, 2000.
2. J. Craig, P. Hsu, and S. Sastry, "Adaptive control of mechanical manipulators," in *Robotics and Auto. Proc. 1986 IEEE Int Conf on*, vol. 3, Apr. 1986, pp. 190–195.
3. B. Armstrong-Hélouvry, P. Dupont, and C. Canudas de Wit, "A survey of models, analysis tools and compensation methods for the control of machines with friction," *Automatica*, vol. 30, no. 7, pp. 1083–1138, 1994.
4. K. Ayusawa, G. Venture, and Y. Nakamura, "Identification of humanoid robots dynamics using floating-base motion dynamics," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ Int Conf on*, Sept. 2008, pp. 2854–2859.
5. R. Ortega and M. Spong, "Adaptive motion control of rigid robots: a tutorial," in *Decision and Control, 1988., Proc of the 27th IEEE Conf on*, vol. 2, Dec. 1988, pp. 1575–1584.
6. P. Khosla, "Categorization of parameters in the dynamic robot model," *Robotics and Automation, IEEE Transactions on*, vol. 5, no. 3, pp. 261 –268, Jun. 1989.
7. J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational Space Control: A Theoretical and Empirical Comparison," *The Int Journal of Robotics Research*, vol. 27, no. 6, pp. 737–757, 2008.
8. S. Vijayakumar, A. D'souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Comput.*, vol. 17, no. 12, pp. 2602–2634, 2005.
9. C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
10. A. J. Smola and B. Schlkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199–222, 2004.
11. D. Nguyen-Tuong, J. Peters, M. Seeger, B. Schlkopf, and M. Verleysen, "Learning inverse dynamics: A comparison," 2008. [Online]. Available: http://edoc.mpg.de/420029
12. J. Sun de la Cruz, D. Kulić, and W. Owen, "Learning inverse dynamics for redundant manipulator control," in *Autonomous and Intelligent Systems (AIS), 2010 International Conf on*, Jun. 2010, pp. 1–6.
13. D. Nguyen-Tuong and J. Peters, "Using model knowledge for learning inverse dynamics," in *IEEE Int Conf on Robotics and Automation*, 2010, pp. 2677–2682.
14. D. Nguyen-Tuong, B. Scholkopf, and J. Peters, "Sparse online model learning for robot control with support vector regression," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ Int Conf on*, Oct. 2009, pp. 3121–3126.
15. S. Schall, C. Atkeson, and S. Vijayakumar, "Scalable techniques from nonparametric statistics for real time robot learning," *Applied Intelligence*, vol. 16, pp. 49–60, 2002.
16. S. Schaal and C. G. Atkeson, "Constructive incremental learning from only local information," *Neural Computation*, vol. 10, no. 8, pp. 2047–2084, 1998.
17. P. Corke, "A robotics toolbox for MATLAB," *IEEE Robotics and Automation Magazine*, vol. 3, no. 1, pp. 24–32, Mar. 1996.
18. S. Chiaverini, L. Sciavicco, and B. Siciliano, *Control of robotic systems through singularities*, ser. Lecture Notes in Control and Information Sciences. Springer Berlin / Heidelberg, 1991, vol. 162.
19. J. Peters and S. Schaal, "Learning to Control in Operational Space," *The Int Journal of Robotics Research*, vol. 27, no. 2, pp. 197–212, 2008.