Learning Forward Models for the Operational Space Control of Redundant Robots

Camille Salaün, Vincent Padois, Olivier Sigaud

Chapter 1
Learning Forward Models for the Operational Space Control of Redundant Robots

Camille Salaün, Vincent Padois, and Olivier Sigaud

Abstract We present a control approach combining model incremental learning methods with the operational space control approach. We learn the forward kinematics model of a robot and use standard algebraic methods to extract pseudo-inverses and projectors from it. This combination endows the robot with the ability to realize hierarchically organised learned tasks in parallel, using tasks null space projectors built upon the learned models. We illustrate the proposed method on a simulated 3 degrees of freedom planar robot. This system is used as a benchmark to compare our method to an alternative approach based on learning an inverse of the extended Jacobian. We show the better versatility of the retained approach with respect to the latter.

Key words: learning, redundancy, robotics, inverse velocity kinematics

1.1 Introduction

Real-world Robotics applications are evolving from the industrial domain (well-defined tasks in structured environment) to the service domain where it is much harder to model all the aspects of the mission. Service Robotics induces complexity both in terms of the tasks that have to be achieved and in terms of the nature of the environment where robots are supposed to evolve. Part of the answer to the problems raised by this growing complexity lies in the increasing number of sensors with which robots are now equipped as well as in the increasing number of degrees of freedom of the robots themselves

Camille Salaün · Vincent Padois · Olivier Sigaud
Institut des Systèmes Intelligents et de Robotique - CNRS UMR 7222
Université Pierre et Marie Curie, Pyramide Tour 55 - Boîte Courrier 173, 4 Place Jussieu, 75252 Paris CEDEX 5, France. e-mail: firstname.name@upmc.fr

(e.g., Mobile manipulators such as the humanoid robot iCub (Metta et al, 2008) or the wheeled assistant PR2 (Willow Garage, 2009)).

As a matter of fact, the motion controllers developed for such robots have to be either highly robust to uncertainties in the model of the robot and its environment or adaptive, i.e., able to build their own model on-line. The former gets more and more difficult as the complexity of the context grows. When the structure of the model is known, the latter can be acheived with classical parametric identification methods (Ljung, 1986). When this structure is more difficult to obtain, due to different physical phenomena such as friction, internal delays, unmodeled nonlinearities, etc., machine learning methods can be more versatile: they learn a model based only on the inputs and outputs of the real system without making assumptions on their physical relationship.

In this context, learning the model of the robot is achieved using specific representations such as Neural Networks (Van der Smagt, 1994) or Radial Basis Function Networks (Sun and Scassellati, 2004), Gaussian Processes (Shon et al, 2005) or (Nguyen-Tuong et al, 2009a) in this issue, Gaussian Mixture Models (Calinon et al, 2007), Locally Weighted Projection Regression (LWPR) (D'Souza et al, 2001; Natale et al, 2007; Peters and Schaal, 2008), but the control methods used in the corresponding work do not always take advantage of the state-of-the-art techniques developed in recent Robotics research.

Among these techniques, operational space control Khatib (1983) is a model-based approach which provides a mathematical framework giving rise to an easy definition of the tasks and constraints characterising a robotic mission in a hierarchical manner (Liégeois (1977), Nakamura (1991). Readers can refer to Sentis and Khatib (2005) for a more recent work and Nakanishi et al (2008) for a survey. In order to take advantage of this framework, one must develop learning methods and associated representations which fit the needs of the corresponding control techniques.

Actuators of a robot generally act on joints, but the tasks or constraints associated to a mission cannot often be described in the joint space in a natural way. The operational space (also called task space) provides an alternative, more natural space, for such a definition.

The robot being controlled at the level of joints, the operational or task space control approach requires the knowledge of the mapping between the joint space and the task space. More specifically, it is the inverse mapping which is often of interest: given a task, what are the actions required in the joint space to achieve it. Considering minimum representations for the joint and task spaces, it is important to notice that when the dimension of the joint space is larger than the one of the task space, there is an infinite number of inverse mappings and the robot is said redundant with respect to the task. That is the case we are focusing on in this chapter.

More precisely, we examine how one can combine learning techniques and operational space control in such a way that we can hierarchically deal with several tasks and constraints when the robot is redundant with respect to the

task. Our method learns a forward kinematics model using LWPR, a state-of-the-art method already used in the context of learning robot models (Vijayakumar et al, 2005). We show how we can both carefully derive the forward and inverse mappings at the velocity level and the projectors which are necessary to combine several tasks. We compare this approach to an alternative approach presented in the literature where the inversion problem that arises in the redundant case is solved in less generic manner (D'Souza et al, 2001).

The chapter is organised as follows. In section 1.2, we give some background on operational space control and the different levels of forward and inverse mappings which can be used to relate the joint space to the task space and vice versa. We also present different contexts in which several tasks can be combined depending on their compatibility. In section 1.3, we give an overview of the learning methods that have been applied to learn these forward and inverse mappings, before focusing on our approach. In section 1.4, we introduce our experimental apparatus and protocol, as well as the series of simulated experiments that we perform. The corresponding results are presented in section 1.5. Finally, section 1.6 highlights the properties of our approach before concluding on the potential extensions that are unique to the perspectives raised by our work.

1.2 Background in Operational Space Control

In this section, we give some background information (Sciavicco and Siciliano, 2000) on joint to operational space mappings with a focus on the velocity level. We recall the general expression of minimum norm solutions in the redundant case and give an overview of redundancy resolution schemes.

1.2.1 Joint space to operational space mappings

The joint space is the space of the configuration parameters $\mathbf{q}$ of size n, where n is the number of parameters chosen to describe the robot configuration. In the holonomic, fully actuated, minimum representation case, n is also the number of degrees of freedom of the robot as well as the dimension of the actuation torque vector $\mathbf{\Gamma}$.

As stated in the introduction, the tasks or constraints associated to a mission can rarely be described in the joint space in a natural way. The operational space is often associated to the end-effector(s) of the robot but can actually be any point of the robot and more generally any set of parameters of interest which can be described as a function of the robot configuration. This is the case for external collision avoidance where the constraint point can evolve along the robot body. Joint limits avoidance is also a particular

case of constraint where the operational space is a subset of the joint space itself. Independently from their physical meanings, operational spaces can be described by operational space parameters $\boldsymbol{\xi}$ of size m where m is, in the case of a minimum representation, the number of degrees of freedom required to achieve the task.

The joint space to operational space mapping can be described at three different levels. At the geometric level, the forward kinematics model can be described as a non-linear function $\mathbf{f}$ such as:

$$\boldsymbol{\xi} = \mathbf{f}(\mathbf{q}). \tag{1.1}$$

As stated before, if the robot is redundant, there is an infinite number of possible inverses for $\mathbf{f}$. However, there is no simple method to span the set of possible solutions at the geometric level and the mapping is often described at the velocity level by the Jacobian matrix $J(\mathbf{q}) = \partial \mathbf{f}(\mathbf{q}) / \partial \mathbf{q}$ such that:

$$\dot{\boldsymbol{\xi}} = J(\mathbf{q})\,\dot{\mathbf{q}}. \tag{1.2}$$

$J(\mathbf{q})$ is a $m \times n$ matrix and thus can be inverted using linear algebra techniques. Once again, there is an infinity of inverse mappings corresponding to the infinity of possible generalised inverses of $J(\mathbf{q})$ (Ben Israel and Greville, 2003).

The last mapping of interest is the dynamic one. It relates forces applied to the system, among which the control input $\boldsymbol{\Gamma}$, to the resulting acceleration $\ddot{\mathbf{q}}$. It can be written:

$$\boldsymbol{\Gamma} = A(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) + \boldsymbol{\epsilon}(\mathbf{q}, \dot{\mathbf{q}}) - \boldsymbol{\Gamma}_{ext}, \tag{1.3}$$

where $A(\mathbf{q})$, $\mathbf{b}(\mathbf{q}, \dot{\mathbf{q}})$, $\mathbf{g}(\mathbf{q})$, $\boldsymbol{\epsilon}(\mathbf{q}, \dot{\mathbf{q}})$ and $\boldsymbol{\Gamma}_{ext}$ are respectively the $n \times n$ inertia matrix of the system, the vector of Coriolis and centrifugal effects, the vector of gravity effects, the vector of unmodeled effects and the torque resulting from external forces applied to the system.

This equation represents a joint space to joint space mapping at the dynamics level with only one solution. It is of course of interest to learn this mapping since it captures a lot of properties of the system among which effects such as friction which cannot always be easily identified using parametric identification techniques. However, the learning of this mapping is out of the scope of the work presented here (interested readers can refer to Peters and Schaal (2008) and Nguyen-Tuong et al (2008)) and it is supposed to be known in the experiments presented here. We rather focus on the velocity kinematics mapping which is sufficient to capture and characterise the redundancy of the system[1].

[1] A velocity kinematics and dynamic combined mapping known as the Operational Space Formulation is proposed by Khatib (1987).

1.2.2 Model-based control at the velocity level

In the redundant and non singular case, i.e., $\text{rank}\left(J\left(\mathbf{q}\right)\right) = m$ and $m < n$, there is an infinite number of generalised inverses of $J\left(\mathbf{q}\right)$. Among these inverses, weighted pseudoinverses provide minimum norm solutions (Doty et al, 1993) and can be written as

$$J\left(\mathbf{q}\right)^{\sharp} = W_q^{-1} J\left(\mathbf{q}\right)^T \left[J\left(\mathbf{q}\right) W_q^{-1} J\left(\mathbf{q}\right)^T\right]^{-1}, \tag{1.4}$$

where W_q is a symmetric and positive definite matrix of dimension $n \times n$.

Given a desired operational velocity $\dot{\boldsymbol{\xi}}^{\star}$, the inverse mapping of Equation (1.2) which minimises the Euclidean W_q-weighted norm[2] of the solution is given by

$$\dot{\mathbf{q}} = J\left(\mathbf{q}\right)^{\sharp} \dot{\boldsymbol{\xi}}^{\star}. \tag{1.5}$$

The Moore-Penrose inverse or pseudoinverse $J\left(\mathbf{q}\right)^{+}$ of $J\left(\mathbf{q}\right)$ corresponds to the case where $W_q = I_n$.

The system being redundant with respect to the task, Equation (1.5) is not the unique solution to the inverse mapping problem and other solutions of interest are those giving rise to internal motions that do not induce any perturbation on the task. This particular subset of solutions corresponds to the nullspace of $J\left(\mathbf{q}\right)$ and the general form of the minimum norm solutions to Equation (1.2) can be written

$$\dot{\mathbf{q}} = J\left(\mathbf{q}\right)^{\sharp} \dot{\boldsymbol{\xi}}^{\star} + P_J\left(\mathbf{q}\right) \dot{\mathbf{q}}_0, \tag{1.6}$$

where $P_J\left(\mathbf{q}\right)$ is a projector on the nullspace of $J\left(\mathbf{q}\right)$ and $\dot{\mathbf{q}}_0$ is any vector of dimension n. Equation (1.6) is the minimum norm solution that minimises $||\dot{\mathbf{q}} - \dot{\mathbf{q}}_0||_{W_q}$. A commonly used expression for $P_J\left(\mathbf{q}\right)$ is

$$P_J\left(\mathbf{q}\right) = \left(I_n - J\left(\mathbf{q}\right)^{\sharp} J\left(\mathbf{q}\right)\right). \tag{1.7}$$

Efficient computation of $J\left(\mathbf{q}\right)^{\sharp}$ and $P_J\left(\mathbf{q}\right)$ can be done using the SVD (Golub and Van Loan, 1996) of $J\left(\mathbf{q}\right)$. The SVD of $J\left(\mathbf{q}\right)$ is given by $J = UDV^T$ where U and V are orthogonal matrices with dimensions $m \times m$ and $n \times n$ respectively. D is a $m \times n$ diagonal matrix with a diagonal composed of the m singular values of J in decreasing order. Given this decomposition, the pseudoinverse of J can be computed as follows

$$J^{+} = V D^{+} U^T, \tag{1.8}$$

where the computation of D^{+} is straightforward given its diagonal nature. Regarding $P_J\left(\mathbf{q}\right)$, it can be computed using the $m+1$ to n columns of V

[2] $\sqrt{\dot{\mathbf{q}}^T W_q \dot{\mathbf{q}}}$, also noted $||\dot{\mathbf{q}}||_{W_q}$.

which form a basis for the nullspace of $J(\mathbf{q})$

$$P_J(\mathbf{q}) = [\mathbf{v}_{m+1} \dots \mathbf{v}_n] \left[\mathbf{v}_{m+1}^T \dots \mathbf{v}_n^T\right]^T, \tag{1.9}$$

where $\mathbf{v}_i$ is the i^{th} column of V.

A weighted extension of the SVD can be used in the case where $W_q \neq I_n$. Details about this extension can be found in Ben Israel and Greville (2003).

1.2.3 Redundancy resolution schemes

Different possible redundancy resolution schemes can be applied, depending on the compatibility of the tasks or constraints which have to be solved.

Let us consider two tasks of respective dimensions m_1 and m_2 and with associated Jacobian matrices J_1 and J_2 such as $rank(J_1(\mathbf{q})) = m_1$ and $m_1 \leq n$ and $rank(J_2(\mathbf{q})) = m_2$ and $m_2 \leq n$. These two tasks are said to be compatible if $J_{ext} = \left[J_1^T \; J_2^T\right]^T$ is full row rank. This condition is equivalent to saying that the m_{ext} parameters of the augmented operational space are in minimum number and that $rank(J_{ext}) \leq n$.

Given this definition, one has to consider the underconstrained (compatible, infinity of solutions), fully-constrained (compatible, one solution) and over-constrained (incompatible, no exact solution) cases. In these three cases, one can write the solution to the inverse velocity kinematics problem using the solution proposed initially by Maciejewski and Klein (1985)

$$\dot{\mathbf{q}} = J_1^\sharp \dot{\boldsymbol{\xi}}_1^\star + (J_2 P_{J_1})^\sharp \left(\dot{\boldsymbol{\xi}}_2^\star - J_2 J_1^\sharp \dot{\boldsymbol{\xi}}_1^\star\right). \tag{1.10}$$

In the compatible case, tasks 1 and 2 will be achieved perfectly. In the incompatible case task 1 will also be perfectly achieved whereas the error on the achievement of task 2 will be minimised. This solution can present singularities when tasks are highly incompatible, i.e., m_{ext} is much greater than n, but this can be compensated for using a proper damped-least square regularisation (Chiaverini, 1997). This task projection scheme can be extended to several tasks, interested readers can refer to Mansard and Chaumette (2007).

Another method, originally proposed in Baillieul (1985), consists in writing an extended Jacobian J_{ext} in order to reach the fully constrained case ($m_{ext} = n$ and $rank(J_{ext}) = m_{ext}$) and thus to simplify the inversion problem to a square, regular matrix inversion. In the fully constrained case, this is achieved automatically. However, in the under constrained case, this requires to artificially add tasks whereas in the over constrained one, some projections have to be done in order to ensure both a square Jacobian matrix and priorities between tasks.

Similarly to what is shown in the non learning case literature, we will show hereafter that in the case of complex missions where the tasks and

constraints constantly evolve, one cannot ensure compatibility at all time. Thus, the solution provided by Equation (1.10) is more general and should be preferred.

Finally, in the case of constraints such as joints limits, a possibility consists in choosing $\dot{\mathbf{q}}_0$ in Equation (1.6) as the opposite of the gradient of a cost function $Q(\mathbf{q})$. The resulting solution leads to the local maximisation of the cost function as long as this secondary constraint does not induce any perturbation on the first task. The general form of this solution is written

$$\dot{\mathbf{q}} = J(\mathbf{q})^{\sharp} \dot{\boldsymbol{\xi}}^{\star} - \alpha P_J \nabla Q(\mathbf{q}), \tag{1.11}$$

where α is a positive scalar used to tune the steepness of the gradient descent Snyman (2005). This method is often used in the incompatible case, i.e., when it is known in advance that the task will not be perfectly achieved, or when only a global trend has to be followed: minimise the kinetic energy of the system, avoid joint limits or collisions, etc.

1.3 Learning forward and inverse velocity kinematics models

Machine learning researchers have developed several families of methods capable of approximating linear and non-linear functions: perceptrons, multilayer perceptrons, radial basis function networks, gaussian processes, support-vector regression, gaussian process regression, learning classifiers systems and locally weighted regression methods such as LWPR Vijayakumar and Schaal (2000). In this section, we briefly explain basic concepts of some of these methods focusing on LWPR which we use and then mention how they are used to learn forward or inverse kinematics model. Then we expose the advantages of learning forward (instead of inverse) velocity kinematics mappings in redundant cases.

1.3.1 An overview of neural networks function approximation

Neural networks Rojas (1996) are a wide class of general function approximators. They are declined under different forms. The general neural network algorithm may be split into three steps. First, compute an excitation level for each neuron depending on the inputs. Second, apply an activation function on this excitation level to determine if the neuron is active or not. Third, compute an error of the global network output to update all weighted connections. A commonly used feedforward neural network is the Multi-Layer

Perceptron (MLP) where the activation level of each neuron i is calculated as $z_i(\mathbf{x}) = \sum_{i=1}^{N} w_i x_i$ where N is the number of neurons, x_i is an input of the neuron and w_i is the associated weight. Classical activation functions are $y(z_i) = \tanh(z_i)$ or $y(z_i) = (1 + e^{-z_i})^{-1}$. The weights are updated through a backpropagation error method based on the error e_j between the desired value and the real output of each neuron. This method consists in computing an energy function, such as $\mathbf{E}(k) = 1/2 \sum_j e_j^2(k)$, which is minimised to update each weighted connection with a gradient descent: $\delta w_l(k) = -\gamma \partial \mathbf{E}(k)/\partial w_l(k)$. γ is the learning rate which updates the weights $w_{1..l}$ of a layer in the opposite direction with respect to the energy gradient. For a general review on learning non-linear models, readers can refer to Jordan and Rumelhart (1992).

Radial basis function networks are another type of neural networks where weighted sum and activation functions are temporally inverted compared to multi-layer perceptrons. The algorithm consists in calculating the function activation $\phi(r) = \exp(-\beta r^2), \beta > 0$ of the norm $r = \|\mathbf{x} - \mathbf{c}_i\|$ which is then weighted and summed: $y(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi(r)$.

A fundamental problem with those approaches is that neural networks are always considered as black boxes and tuning is usually empirical. Radial basis function networks tend to avoid this limitation and could be treated as grey boxes. They have the advantage of generating differentiable continuous approximations (Sun and Scassellati, 2004).

An extension of radial basis function networks are the locally weighted regression methods (Atkeson, 1991) which combine gaussian and linear models. Some of those methods, such as LWPR, make a projection on the relevant subspace to decrease the dimension of the input space. We focus on this method below since it is the one we use.

1.3.2 Locally Weighted Projection Regression

Locally weighted regressions were first used by Atkeson (1991) to realise supervised learning on robots. As described in Schaal et al (2002), lots of models have followed, such as LWPLS which include partial least square to reduce input dimensionality or RFWR (Schaal and Atkeson, 1997) which transform the algorithm into an incremental regression method, avoiding to store data.

Locally Weighted Projection Regression (LWPR) is an algorithm which perform both incremental regression and inputs projection. It is a function approximator which provides accurate approximation in very large spaces in $O(k)$, where k is the number of data points used to perform this estimation. LWPR uses a combination of linear models that are valid on a zone of the input space. This space, delimited by a gaussian, may change during the training to match the trained data. Each model is called a receptive field. The prediction of an entire LWPR model on an input vector is the weighted sum of the results

of all the active surrounding receptive fields. Receptive fields are created or pruned in order to keep an optimal repartition.

Each receptive field first projects the input vector on the most relevant dimensions to estimate the output vector. This is done by using the covariance matrix of the input/output vectors. At each modification, the projector is updated and the algorithm checks if increasing the complexity, by adding another dimension to the input projection, significantly improves the receptive field results and modifies the projector accordingly. The projected vector is then used in the m dimension linear model (m being the output dimension) to give the output of the receptive field. During prediction, only the significant receptive fields are activated. The algorithm may also computes the gradient of the output with respect to the input.

Different methods listed above have been used to learn various models among the ones listed in Section 1.2. Before presenting our own approach, we provide an overview of these different works.

1.3.3 Learning inverse kinematics

Some authors learn the inverse kinematics with a neural network which realise the mapping between operational and joint velocities, solving an unconstrained optimisation problem formulated as an energy function. This energy function is minimised during the gradient descent and weights are consequently updated. It is thus possible to obtain a desired articular velocity which correspond to a minimum energy function as

$$\dot{q}^* = \arg\min_{\dot{q}} \left(E\left(\dot{q}\right) \right)$$

where E is an energy function which is based on different errors. Pourboghrat (1989) minimises an energy which leads to learn the Moore-Penrose inverse minimising the weighted norm (as seen in Equation 1.5). Barhen et al (1989) resolve redundancy in minimising different Lyapunov function with goal attractors. Guez and Ahmad (1988) and Ahmad and Guez (1990) optimise the manipulability criterion: $H = \sqrt{\|JJ^T\|}$ in each configurations. Lee and Kil (1990) minimise an energy function composed of two types of constraints: one is minimising the angle difference of the first joint and another is locating the joints in the middle of joint-limits. They consider the forward kinematics as known. Brüwer and Cruse (1990) keep the system away from joint limits and compare their results to planar human motion. Finally, DeMers and Kreutz-Delgado (1997) includes the topology to bring flexibilities in his redundant kinematic inverse model.

Based on self-organising maps (Kohonen, 2001), Martinetz et al (1990b,a); Walter and Schulten (1993) learn a mapping between end effector position measured by two cameras and joint positions on a three degrees of freedom

robot. The mapping is made by a three dimensional self-organising map. They automatically resolve redundancy minimising the variation of the joint angles during the learning process, using motor babbling or just learning along target trajectories.

Closer to our approach, D'Souza et al (2001) et al. learn an inverse kinematic model with LWPR. The learned inverse model is an inverse of the extended Jacobian learned on the task with the input $\left(\mathbf{q}, \dot{\boldsymbol{\xi}}\right)$ and the output $(\dot{\mathbf{q}})$

$$\mathcal{M} = LWPR_{learn}\left(\left[\mathbf{q}, \dot{\boldsymbol{\xi}}\right], \dot{\mathbf{q}}\right).$$

Doing so, no inversion is involved and singularity problems are avoided.

1.3.4 Learning forward kinematics

As forward models of serial robots can easily be computed analytically, there are few papers treating this subject.

A few authors have used multi-layer perceptrons networks or self-organising maps to learn the forward kinematics model of various simple systems (Nguyen et al, 1990; Wang and Zilouchian, 1997; Sadjadian and Taghirad, 2004), but in general the evaluation is based on the accuracy of the model itself rather than on its control capabilities. Boudreau et al (1998) and Sang and Han (1999) learn in a more convenient way the forward models of parallel robots which involves highly coupled nonlinear equations and which are difficult to model analytically.

More related to our work, Sun and Scassellati (2004, 2005) learn a forward geometric model with a radial basis function network. They derive each function to obtain a Jacobian, using classical operational space control techniques similarly to what we propose. They specifically use the two cameras of a humanoid robot to obtain the operational position used in their controller. Their approach is very similar to the one we present hereafter, provided that they use radial basis function networks whereas we use LWPR. On a similar line, Butz and Herbort (2008) learn a forward kinematics model and inverse it, using the learning classifier system XCSF as a model learning tool.

In fact, learning forward models for a redundant robot does not raise particular problems. By contrast, as explained in section 1.2, there exists an infinity of possible inverse mappings, thus, unless one always wants to use the same inverse mapping, it does not really make sense to directly learn kinematics or velocity kinematics inverse mappings since this leads to a loss of information regarding the redundant nature of the system. Instead, one can learn the forward mappings and invert them with the methods described in section 1.2.2, keeping the infinity possibilities for the inversion. One may argue that inverting a learned mapping will lead to the amplification of learning errors. This is true. However, the results we present in this chapter demon-

strate that this approach actually provides very good results when combined with a closed-loop controller as well as when keeping the learning active while controlling the robot.

Taking into account these considerations and in order to compare the two approaches, in this paper we propose to learn the forward kinematics model in Equation (1.1) of a 3 degrees of freedom robot, giving as input the joint parameters $\mathbf{q}$ adjusted in $[0, 2\pi[$ and the operational space parameters $\boldsymbol{\xi}$ as output

$$\mathcal{M} = LWPR_{learn}\left(\mathbf{q}, \boldsymbol{\xi}\right).$$

LWPR does not return directly the global model, but only the predicted output for a particular input. However, the Jacobian matrix is the first order derivative of the forward kinematics model relatively to joint space parameters $\mathbf{q}$, thus this matrix is provided "for free" while learning the forward kinematics model. This calculation is made easier by the fact that the learned model is a simple sum of multiple linear functions which are easily differentiated.

1.4 Experimental study

In this section, we present simulation based experiments designed to compare the under, fully and over-constrained cases using both the projection and the extended Jacobian approaches. When using the latter, we do not learn the inverse mapping as in D'Souza et al (2001) but rather the forward mapping which we inverse.

1.4.1 Control architecture

We have chosen to evaluate the compared approaches on a 3 degrees of freedom planar system, shown in Figure 1.1. Sticks lengths are 0.50m, 0.40m and 0.20m. To simulate this system, we use Arboris, a dynamic simulator based on Newton-Euler equations which is implemented in matlab (Barthelemy and Bidaud, 2008). The integration step time of the simulator is chosen to be 10 milliseconds.

Our control scheme uses the resolved motion rate control principle, i.e., the desired task space velocity is computed using the task space parameters error

$$\dot{\boldsymbol{\xi}}^{\star} = K_p\left(\boldsymbol{\xi}^{\star} - \boldsymbol{\xi}\right), \tag{1.12}$$

when $\boldsymbol{\xi}^{\star}$ denotes the desired value of the task space parameters and K_p is a symmetric positive definite matrix. The actual task space parameters are obtained from the simulator model and, in the case of a real robot, they

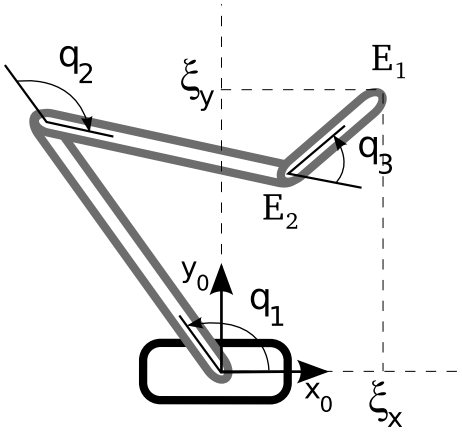

Fig. 1.1 Schematic view of our simulated system

would be measured from exteroceptive sensors. One could think about using LWPR forward kinematics prediction however this would lead to a drift with respect to the real target since no external reference would then be used to close the control loop.

Regarding the projection method, the inverse velocity kinematics is done using solution (1.10) and the estimated Jacobian matrices and projector which can be written as

$$\dot{\mathbf{q}} = \hat{J}_1^+ \dot{\boldsymbol{\xi}}_1^\star + \left(\hat{J}_2 P_{\hat{J}_1}\right)^+ \left(\dot{\boldsymbol{\xi}}_2^\star - \hat{J}_2 \hat{J}_1^+ \dot{\boldsymbol{\xi}}_1^\star\right). \tag{1.13}$$

$\hat{J}_1$ and $\hat{J}_2$ are respectively obtained from LWPR predictions

$$\left[\hat{\boldsymbol{\xi}}_1, \hat{J}_1\right] = LWPR_{predict}\left(\mathbf{q}, \mathcal{M}_1\right)$$

and

$$\left[\hat{\boldsymbol{\xi}}_2, \hat{J}_2\right] = LWPR_{predict}\left(\mathbf{q}, \mathcal{M}_2\right).$$

The extended Jacobian method leads to a solution that can be written:

$$\dot{\mathbf{q}} = \begin{bmatrix} \hat{J}_1 \\ \hat{J}_2 \end{bmatrix}^{-1} \begin{bmatrix} \dot{\boldsymbol{\xi}}_1^\star \\ \dot{\boldsymbol{\xi}}_2^\star \end{bmatrix}. \tag{1.14}$$

$P_{\hat{J}_1}$ and pseudoinverses of $\hat{J}_1$ and $\hat{J}_2 P_{J_1}$ are obtained using their SVD as presented in Section 1.2.2.

$\dot{\mathbf{q}}$ obtained from Equations (1.13) or (1.14) is then differentiated and the resulting joints acceleration vector is used to compute the actuation torque based on the dynamics model in Equation (1.3) which we suppose to know

and is obtained from Arboris (see above). A short version of this control scheme is presented on Figure 1.2.

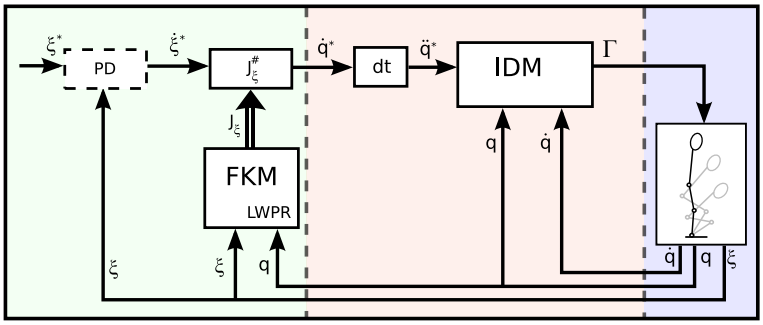

Fig. 1.2 Our control scheme including a Forward Velocity Kinematics Model learned with LWPR and an Inverse Dynamics Model. The dashed line is executed only during the learning process.

1.4.2 Choice of parameters for the LWPR *algorithm*

Before performing our experiments, we start with an initial exploration phase that can be seen as motor babbling, to initialise the model, as suggested in Klanke et al (2008). We generate random configurations taking $q_i \in [0, 2\pi[$. Depending on the corresponding configurations, we measure task parameters and feed the LWPR model with the corresponding $(\mathbf{q}, \boldsymbol{\xi})$ pairs.

Then LWPR comes with some parameters that need to be initialised. We initialise LWPR as proposed by Klanke et al (2008). The $init_D$ coefficient corresponds to the initial size of all receptive fields. This coefficient significantly affects the convergence time of LWPR. $init_D$ is tuned experimentally from comparing the performance of a set of motor babbling phases to find the best value corresponding to the minimal prediction error.

Two important parameters for our simulations are w_{gen} and *penalty*. The first one is a threshold responsible for the creation of a new local model if no model responds high enough. The *penalty* coefficient is critical to the evolution of the size of receptive fields. A small *penalty* term increases precision but decreases the smoothness of the model. We have chosen $w_{gen} = 0.5$ and $penalty = 1e^{-6}$ to have the best precision while avoiding "overlearning". Finally, from our experiments, updating D is not so important once the initialisation is well done but we still keep this option. We set $init_\alpha$ to 10000 and activate meta learning (see Klanke et al (2008)).

1.4.3 Experiments

In this Section, we detail three different constrained cases and associated tasks in order to study the robustness of our control scheme.

1.4.3.1 Under-constrained case

The first studied task is a reaching task. From an initial end-effector position $\boldsymbol{\xi}_1^i = [0.10\ 1.00]^T\ m$, the end-effector ($E_1$) of the robot has to reach a target $\boldsymbol{\xi}_1^\star = [0.20\ 0.50]^T\ m$ with a specified precision of 0.01 meters. Once the task is achieved, the end-effector is sent back to its initial position with the same controller and the same required precision. This point to point movement is repeated until the end of the simulation.

For this simple reaching task, the task space dimension is 2, thus the Jacobian is redundant and there is an infinity of ways to reach the goal. We compare the projection approach presented in Section 1.2.3 without any secondary task to the extended Jacobian approach, where the extension is realised by adding a one dimension constraint on point (E_2):
$\xi_{2x}^\star = 0.40\ m$.

1.4.3.2 Fully constrained case

The second experiment consists in reaching $\boldsymbol{\xi}_1^\star = [0.20\ 0.50]^T\ m$ and keeping the end effector in this position while realising a second task. This second task alternatively requires the parameter ξ_{2x} to reach the values 0.10 m and 0.30 m which are accessible. The first task is a two dimensional task whereas the second one is a one dimensional task. The system is thus fully constrained.

For these two tasks, the same redundancy resolution schemes are tested. In the case of the projection method, the second task is projected in the nullspace of the first one accordingly to Equation (1.13). In the case of the extended Jacobian method, J_{ext} is chosen as in the previous experiment.

1.4.3.3 Over-constrained case

The last experiment is very similar to the previous one. The first task is identical whereas the second one is a two dimensional task for point (E_2) which has to reach $\boldsymbol{\xi}_2^\star = [0.45\ 0.25]^T\ m$. This second task is not compatible with the first one. The system is over constrained.

Regarding this experiment, the projection method is the only one to be tested since the extended Jacobian method would require the same projection in order to obtain a square Jacobian matrix J_{ext}.

1.5 Results

In this section, results from the babbling phase and the experiments described in Section 1.4.3 are presented and analysed. Except for the babbling phase where the presented results are an average over 40 trials, the results presented below correspond to representative trials.

1.5.1 Babbling phase

To evaluate the effectiveness of the forward velocity kinematics model prediction, we use the Normalised Mean Square Error ($NMSE$) computed as:

$$NMSE = \frac{1}{\sigma^2}\frac{1}{N}\sum_{i}^{N}\left(\mathbf{y_i} - \hat{\mathbf{y}}_\mathbf{i}\right)^2$$

where N is the number of points used to compute this error. $\mathbf{y_i}$ is the i^{th} value of the data obtained by the real model of the robot, $\hat{\mathbf{y}}_\mathbf{i}$ is the i^{th} predicted value by the learned model and σ^2 is the sample variance of y: $\sigma^2 = \frac{1}{N}\sum_{k=1}^{N}\left(y_k - \overline{y}_k\right)^2$.

To actually compute this error, we fixed the velocity of each joint to 1.00 $rad.s^{-1}$. As can be seen on Figure 1.3, the $NMSE$ of the predicted velocity decreases during motor babbling. A babbling phase with 5000 samples is, in this case, sufficient for LWPR to cover roughly the joint space, having an output, even bad, in each configurations, and to predict an accurate enough Jacobian matrix.

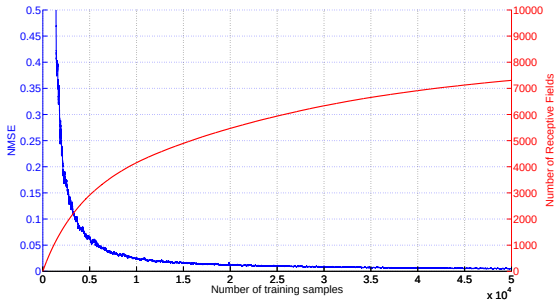

Fig. 1.3 Evolution of the Normalised Mean Square Error of the LWPR operational space velocity prediction (blue, scale left) and of the number of receptive fields for one output (red, scale right) for a 50000 samples babbling phase (average over 40 trials).

Regarding the number of receptive fields, in the end of each experiment, it varies between 2000 and 8000 for each output depending on the length of the babbling phase. For babbling phases with a large number of samples, this number is almost reached at the end of the babbling phase.

The relatively large number of receptive fields required in these experiments is due to two factors. The first one is the precision of the prediction which is asked for and that can be related to our choice of parameters for the LWPR algorithm. The second factor is more specific to the redundancy of the robot which we wish to exploit. This redundancy induces possible internal motions from secondary tasks which cannot be predicted *a priori* and thus require a good coverage of the joint space in complement to specific trajectory learning.

1.5.2 Under-constrained case

In this experiment, in order to highlight the model adaptation during the control phase, we only realise motor babbling using 2000 samples. Figure 1.4(a) represents the two first seconds of simulation. The model of the robot is still quite approximative and the resolved motion rate controller is not sufficiently robust to compensate for inaccuracies in the learned model. Figures 1.4(b) (between 2s and 4s) and 1.4(c) (between 6s and 8s) show the evolution of the trajectories. It can be noticed that the learned model is being adapted during the control phase. Also, the precision requirements (errors smaller than 0.01m) in terms of the point that has to be reached are met. After 20s (see Figure 1.4(d)), the precision is improved and the trajectory of the robot trajectory is almost linear as one would expect when using a resolved motion rate controller.

This is not illustrated here but, as expected, the results obtained using the projection and the extended Jacobian methods are equivalent in the under constrained case.

1.5.3 Fully constrained case

In the fully constrained case, the precision requirements (0.01m) are also met for the two tasks which respectively constrain the position of the end-effector (point (E_1)) and the position along the $\mathbf{x}_0$ axis (see Figure 1.1) of the wrist of the robot (point (E_2)). This is illustrated on Figure 1.5 for the first task. It is shown, that there is no major difference in the precision obtained when controlling redundancy using an extended Jacobian or using the projector approach. From 0 to 1s, the reference operational point is not reached yet, which explains the large error (the initial error, not shown on the figure, is

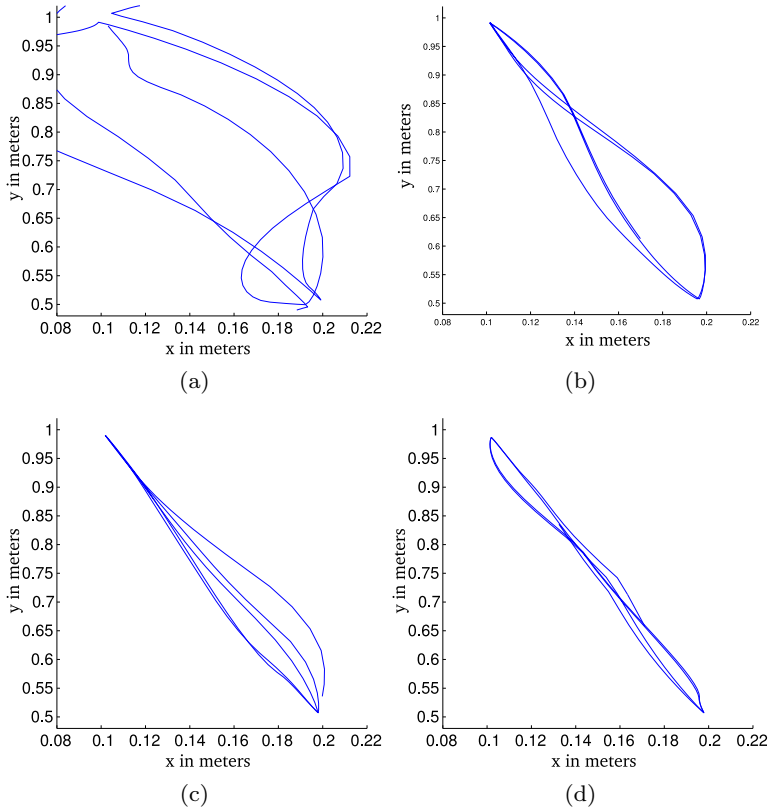

Fig. 1.4 Evolution of operational trajectories while learning. Each graphic represents two seconds of simulation.(a): 0s to 2s.(b): 2s to 4s.(c): 6s to 8s.(d): 20s to 22s.

0.65m). After 1s, the required precision is obtained and errors are due to the cyclic change of reference point for the second task. These errors decrease with time thanks to the on-line improvement of the learned model.

These errors are due to the fact that a learned model is used as well as to the fact that learning errors are propagated when computing the inverse velocity kinematics mapping from the forward one. Using the extended Jacobian approach or ours, if the Jacobian matrices are not accurately predicted, errors disturbs the tasks. Similarly, when specifically using the projection method, an error in the prediction of the first task Jacobian induces an error in the computation of the associated projector leading to disturbances induced by the second task on the first one.

Despite these error propagation effects, the achieved performances are satisfactory.

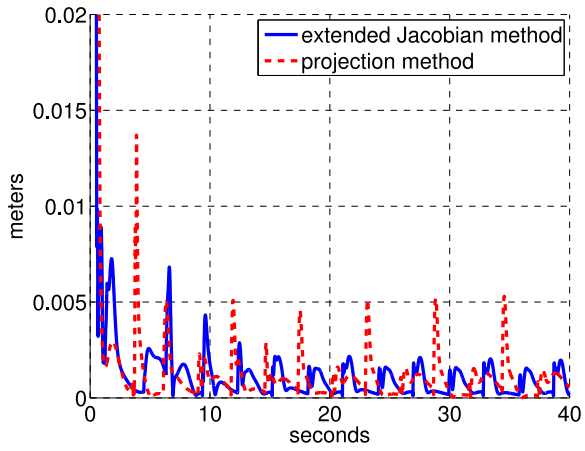

Fig. 1.5 Norm of the end-effector error induced by a second compatible task for two different controllers.

1.5.4 Over-constrained case

The results obtained from the last experiment illustrate the effectiveness of the projection method. The controller maintains the distance between the end effector point (E_1) and the desired reference point (A) under 0.01m. In the same time, the second task is partially achieved as expected from the redundancy resolution scheme which was chosen. It is achieved with the minimum possible error and without inducing any disturbance on the first task: the task hierarchy is respected.

These results are illustrated on Figure 1.6 where the final configuration of the system is shown as well as intermediate configuration, illustrating the convergence of the second task to the best possible result.

Figure 1.7 gives a view of the positioning errors for both tasks. Similarly to the last experiment, error propagation effects are present but once again the end effector error in position is very low.

1.6 Discussion

First, our results demonstrate the necessity of a babbling phase in the redundant case. The number of receptive fields associated to the learned model is quite important but this is explained by the required precision as well as by the necessity to cover the joint space appropriately (see Section 1.5.1).

The results of the experiments for the under constrained case shows that after some initialisation with motor babbling, our method is able to im-

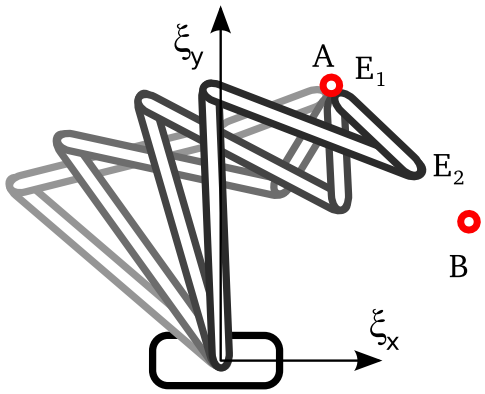

Fig. 1.6 Stroboscopic view of the robot realising two prioritised tasks in the incompatible case. The task hierarchy is respected as the end effector E_1 reach point A while the distance between point E_2 and point B is minimum.

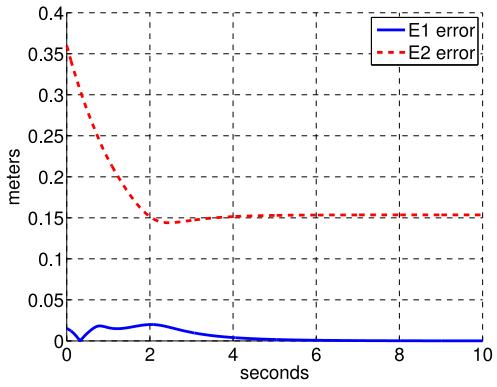

Fig. 1.7 Reaching errors for the first (blue, plain line) and second task (red, dotted line) in the incompatible case.

prove the model of the system while controlling it so that a reaching task is achieved with a prescribed precision. The end-effector trajectory converges to what would be expected in the case of resolved motion rate control. A similar result has already been obtained by D'Souza et al (2001), using the extended Jacobian approach and learning directly the inverse velocity kinematics mapping.

The second series of experiments (fully constrained case) convey more original results. To our knowledge, our approach is original in the sense that the forward velocity kinematics mapping is learned (through the learning of the forward kinematics model) and the obtained Jacobian matrix is used to derive the required inverse and projector allowing to combine several learned

tasks. We show that this method induces error propagations but the performance of the controller remains satisfactory. This is mostly due to the fact that learning is still active while performing the task, inducing on-line model adaptation. Also, closing the control loop at the operational level using exteroceptive sensor information results in the possibility to compensate for model uncertainties.

The last series of experiments (over constrained case) reinforces the results of the fully constrained case by showing that several learned tasks can be combined in a hierarchical manner in the case where those tasks are not compatible. This has been a state-of-the-art result for a while in model-based control in Robotics (Nakamura, 1991). However, to the best of our knowledge, this is the first time that such results are achieved in the case of learned models.

The retained redundancy resolution scheme in that last case is the projection method which leads to the optimal solution for both tasks. In fact, in the over constrained case, the only effective redundancy resolution scheme is the projection method. The extended Jacobian approach can be applied in that case but in a way that requires projections similarly to our approach.

Taking these considerations into account, we draw two conclusions. The first one is that the extended Jacobian approach is not satisfactory in the case where models have to be learned. Combining two tasks in a single one in order to simplify inversion leads to unnecessary constraints on the learning problem to be solved whereas it is simpler to learn elementary tasks separately. Furthermore, tasks combination is easier when the tasks are learned separately. In the extended Jacobian method, the learned inverse velocity kinematics model depends on the supplementary task. The whole model has to be learned again if this task changes whereas learning separately different Jacobian matrices leave them independent and changing one of them does not impact the others.

Our second conclusion is that in the redundant case, learning forward models does not lead to a loss of information about the system. In our method, one can choose to add any secondary task independently from the first one and any weighting matrix W_q can be used[3] when performing the inverse of the Jacobian (see Equation (1.4)). That is not the case when inverse models are learned directly since this corresponds to a specific choice of inverse. Also, learning the nullspace of a given mapping at the velocity level would require a complex learning process and thus it sounds more appropriate to compute them from the learned forward velocity kinematics mapping.

[3] The use of a proper weighting matrix (different from the Identity) can be crucial in the dynamic case (Khatib, 1987).

1.7 Conclusion

In the work presented in this paper, we have used a state-of-the-art function approximation technique, LWPR, to learn the forward kinematics and, by extension, forward velocity kinematics models of a simple robotic system. We have shown that this model learning process could be combined with state-of-the-art operational space control techniques to control a robot. In particular, we demonstrated that we can benefit from the hierarchical combination capabilities of the operational space control framework to achieve several learned tasks in parallel even when those tasks are not fully compatible. This is made possible by learning the unique forward mapping for each task and then inversing it instead of directly learning an inverse mapping. Two methods were tested: the extended Jacobian approach and the projection method. The latter is shown to be more versatile than the former.

There are several possible extensions to this work. The most immediate one consists in dealing with the case of trajectory tracking instead of reaching tasks using resolved motion rate control. This may require faster on-line learning capabilities during the control phase and we will have to demonstrate that benefiting from redundancy and combining tasks is still possible in that more complex case. More generally, we should study the performance of our approach in a wider variety of tasks and combinations (joint limits avoidance, external collision avoidance) as well as in the case of using different types of inversion.

A second extension consists in learning the dynamics of the system and studying the behaviour of our approach under perturbations to validate its on-line adaptation capabilities to external forces, that will make it possible to interact with unknown objects and human users.

Longer term perspectives include an extension of our framework to systems with a larger number of degrees of freedom. Even though our example is complex enough to present our approach to learning for the control of redundant systems, model learning is of interest for complex systems. Increasing the number of dimensions leads to more complex learning problems. In that context, replacing LWPR by the Local Gaussian Process approach described in this volume (Nguyen-Tuong et al, 2009b) seems a natural choice. Finally, the control framework presented in this paper considers the system as deterministic, whereas in a model learning context, regarding it as stochastic seems more adequate. As a consequence, we will examine the option of moving from our deterministic framework to its stochastic equivalent described in (Toussaint and Goerick, 2009) in this volume.

References

Ahmad Z, Guez A (1990) On the solution to the inverse kinematic problem. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), vol 3, pp 1692–1697

Atkeson C (1991) Using locally weighted regression for robot learning. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), vol 2, pp 958–963

Baillieul J (1985) Kinematic programming alternatives for redundant manipulators. In: Proceedings of the International Conference on Robotics and Automation (ICRA), vol 2, pp 722–728

Barhen J, Gulati S, Zak M (1989) Neutral learning of constrained nonlinear transformations. Computer 22(6):67–76

Barthelemy S, Bidaud P (2008) Stability measure of postural dynamic equilibrium based on residual radius. In: Proceedings of the 17th CISM-IFToMM Symposium on Robot Design, Dynamics and Control (RoManSy), Tokyo, Japan

Ben Israel A, Greville T (2003) Generalized Inverses : Theory and Applications, 2nd edn. Springer, ISBN 0-387-00293-6

Boudreau R, Darenfed S, Gosselin C (1998) On the computation of the direct kinematics of parallel manipulators using polynomial networks. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 28(2):213–220

Brüwer M, Cruse H (1990) A network model for the control of the movement of a redundant manipulator. Biological Cybernetics 62(6):549–555

Butz MV, Herbort O (2008) Context-dependent predictions and cognitive arm control with XCSF. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation, ACM New York, NY, USA, pp 1357–1364

Calinon S, Guenter F, Billard A (2007) On Learning, Representing and Generalizing a Task in a Humanoid Robot. IEEE Transactions on Systems, Man and Cybernetics, Part B 37(2):286–298

Chiaverini S (1997) Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. IEEE Transactions on Robotics and Automation 13(3):398–410

DeMers D, Kreutz-Delgado K (1997) Inverse kinematics of dextrous manipulators. Neural Systems for Robotics pp 75–116

Doty K, Melchiorri C, Bonivento C (1993) A theory of generalized inverses applied to Robotics. The International Journal of Robotics Research 12(1):1–19

D'Souza A, Vijayakumar S, Schaal S (2001) Learning inverse kinematics. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), vol 1, pp 298–303, DOI 10.1109/IROS.2001.973374

Golub G, Van Loan C (1996) Matrix computations, 3rd edn. The John Hopkins University Press, ISBN 0-8018-5414-8

Guez A, Ahmad Z (1988) Solution to the inverse kinematics problem in robotics by neural networks. In: Proceedings of the IEEE International Conference on Neural Networks, pp 617–624 vol.2

Jordan MI, Rumelhart DE (1992) Forward models: Supervised learning with a distal teacher. Cognitive science 16(3):307–354

Khatib O (1983) Dynamic control of manipulators in operational space. In: Sixth CISM-IFToMM Congress on Theory of Machines and Mechanisms, pp 1128–1131

Khatib O (1987) A unified approach to motion and force control of robot manipulators: The operational space formulation. IEEE Journal on Robotics and Automation 3(1):43–53

Klanke S, Vijayakumar S, Schaal S (2008) A library for locally weighted projection regression. Journal of Machine Learning Research 9:623–626

Kohonen T (2001) Self-Organizing Maps. Springer, Berlin

Lee S, Kil R (1990) Robot kinematic control based on bidirectional mapping neural network. In: Proceedings of International Joint Conference on Neural Networks (IJCNN), vol 3, pp 327–335

Liégeois A (1977) Automatic supervisory control of the configuration and behavior of multibody mechanisms. Systems, Man and Cybernetics, IEEE Transactions on 7(12):868–871

Ljung L (1986) System identification: theory for the user. Prentice-Hall, Inc. Upper Saddle River, NJ, USA

Maciejewski A, Klein C (1985) Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. The International Journal of Robotics Research 4(3):109–117

Mansard N, Chaumette F (2007) Task sequencing for sensor-based control. IEEE Transactions on Robotics 23(1):60–72

Martinetz T, Bitter H, Schulten K (1990a) Learning of Visuomotor-Coordination of a robot arm with redundant degrees of freedom. In: Proceedings of the Third International Symposium on Robotics and Manufacturing–Research, Education, and Applications, Amer Society of Mechanical, p 521

Martinetz T, Ritter H, Schulten K (1990b) Three-dimensional neural net for learning visuomotor coordination of a robot arm. IEEE Transactions on Neural Networks 1(1):131–136

Metta G, Sandini G, Vernon D, Natale L, Nori F (2008) The iCub humanoid robot: an open platform for research in embodied cognition. In: PerMIS: Performance Metrics for Intelligent Systems Workshop, Washington DC, USA

Nakamura Y (1991) Advanced Robotics: redundancy and optimization. Addison Wesley, ISBN 0-201-15198-7

Nakanishi J, Cory R, Mistry M, Peters J, Schaal S (2008) Operational space control: A theoretical and empirical comparison. The International Journal of Robotics Research 27(6):737–757

Natale L, Nori F, Metta G, Sandini G (2007) Learning precise 3d reaching in a humanoid robot. In: Proceedings of the IEEE International Conference of Development and Learning (ICDL), London, UK, pp 1–6

Nguyen L, Patel R, Khorasani K (1990) Neural network architectures for the forward kinematics problem in robotics. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN), vol 3, pp 393–399

Nguyen-Tuong D, Peters J, Seeger M, Schoelkopf B (2008) Learning inverse dynamics: a comparison. In: Proceedings of the European Symposium on Artificial Neural Networks (ESANN)

Nguyen-Tuong D, Seeger M, Peters J (2009a) Real-time local gp model learning. In: From motor to interaction learning in robots

Nguyen-Tuong D, Seeger M, Peters J (2009b) Real-time local gp model learning. In: From Motor to Interaction Learning in Robots, Springer

Peters J, Schaal S (2008) Learning to control in operational space. The International Journal of Robotics Research 27(2):197–212

Pourboghrat F (1989) Neural networks for learning inverse-kinematics of redundant manipulators. In: Proceedings of the 32nd Midwest Symposium on Circuits and Systems, vol 2, pp 760–762

Rojas R (1996) Neural networks: a systematic introduction. Springer

Sadjadian H, Taghirad HD (2004) Numerical methods for computing the forward kinematics of a redundant parallel manipulator. In: Proceedings of the IEEE Conference on Mechatronics and Robotics, Aachen, Germany

Sang LH, Han MC (1999) The estimation for forward kinematic solution of stewart platform using the neural network. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), vol 1, pp 501–506

Schaal S, Atkeson CG (1997) Receptive field weighted regression. ART Human Inf Process Lab, Kyoto, Japan, Tech Rep TR-H-209

Schaal S, Atkeson CG, Vijayakumar S (2002) Scalable techniques from nonparametric statistics for real time robot learning. Applied Intelligence 17(1):49–60

Sciavicco L, Siciliano B (2000) Modelling and control of robot manipulators, 2nd edn. Springer

Sentis L, Khatib O (2005) Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. The International Journal of Humanoid Robotics 2(4):505–518

Shon A, Grochow K, Rao R (2005) Robotic imitation from human motion capture using gaussian processes. In: Proceedings of the IEEE-RAS/RSJ International Conference on Humanoid Robots (Humanoids)

Van der Smagt PP (1994) Minimisation methods for training feedforward neural networks. Neural Networks 7(1):1–11

Snyman JA (2005) Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms. Springer

Sun G, Scassellati B (2004) Reaching through learned forward model. 4th IEEE/RAS International Conference on Humanoid Robots 1:93–112

Sun G, Scassellati B (2005) A fast and efficient model for learning to reach. International Journal of Humanoid Robotics 2(4):391–414

Toussaint M, Goerick C (2009) A bayesian view on motor control and planning. In: From Motor to Interaction Learning in Robots, Springer

Vijayakumar S, Schaal S (2000) Locally weighted projection regression: An o (n) algorithm for incremental real time learning in high dimensional space. In: Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford, CA

Vijayakumar S, D'Souza A, Schaal S (2005) LWPR: A Scalable Method for Incremental Online Learning in High Dimensions. Tech. rep., Edinburgh University Press

Walter J, Schulten K (1993) Implementation of self-organizing neural networks for visuo-motor control of an industrial robot. IEEE Transactions on Neural Networks 4(1):86–96

Wang D, Zilouchian A (1997) Solutions of kinematics of robot manipulators using a kohonen self-organizing neural network. In: Proceedings of the IEEE International Symposium on Intelligent Control, pp 251–255

Willow Garage (2009) Overview of the PR2 robot. http://www.willowgarage.com/pages/robots/pr2-overview