Control of redundant robots using learned models: an operational space control approach

Camille Salaün, Vincent Padois, Olivier Sigaud

The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Control of redundant robots using learned models: an operational space control approach.

Camille Salaün, Vincent Padois and Olivier Sigaud

Abstract— **We present an adaptive control approach combining forward kinematics model learning methods with the operational space control approach. 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 extended Jacobian. We show the better versatility of the retained approach with respect to the latter.**

I. INTRODUCTION

Real-world Robotics applications are evolving from the industrial domain (simple 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 e.g., Mobile manipulators such as the humanoid robot iCub [1] or the wheeled assistant PR2 [2]).

As a matter of fact, the motion controllers developed for such robots have to be either highly robust to uncertainties on the knowledge of 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 whereas the latter is often achieved using learning techniques. More specifically, a common approach to learning consists in using model-based control techniques in a context where the model of the robot is learned on-line from experience, giving rise to immediate motor adaptation capabilities.

In this context, learning the model of the robot is achieved using specific representations such as Neural Networks or Radial Basis Function Networks [3], Gaussian Processes [4], Gaussian Mixture Models [5], Locally Weighted Projection Regression (LWPR) [6], [7], [8], 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 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 (see [9], [10] and for more recent work [11]). 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 can rarely 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 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 actions have to be taken 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 paper.

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 [12]. 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 a static manner [6].

The paper is organised as follows. In Section II, 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

Camille Salaün (PhD candidate in Robotics), Vincent Padois (Assistant professor in Computer Science and Robotics) and Olivier Sigaud (Professor in Computer Science) are with:

Université Pierre et Marie Curie
Institut des Systèmes Intelligents et de Robotique - CNRS UMR 7222
Pyramide Tour 55 - Boite Courrier 173
4 Place Jussieu, 75252 Paris CEDEX 5, France

Contact: firstname.name@upmc.fr

Section III, we present LWPR and the way we use it to learn the forward kinematics model of our system and derive the inverse mapping and associated projectors at the velocity level. In Section IV, 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 V. Finally, Section VI is dedicated to the discussion, highlighting the properties of our approach before concluding on the potential extensions that are unique to the perspectives raised by our work.

II. Background in Operational Space Control

In this Section, we give some background information on joint to task 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.

A. Joint space to task space mappings

The joint space is the space of the configuration parameters $\boldsymbol{q}$ 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 $\boldsymbol{\Gamma}$.

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 task 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 task space is a subset of the joint space itself. Independently from their physical meanings, task spaces can be described by task 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 task 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 $\boldsymbol{f}$ such as

$$\boldsymbol{\xi} = \boldsymbol{f}(\boldsymbol{q}). \tag{1}$$

As stated before, if the robot is redundant, there is an infinite number of possible inverses for $\boldsymbol{f}$. 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 $J(\boldsymbol{q}) = \frac{\partial}{\partial \boldsymbol{q}} \boldsymbol{f}(\boldsymbol{q})$ such that

$$\dot{\boldsymbol{\xi}} = J(\boldsymbol{q})\,\dot{\boldsymbol{q}}. \tag{2}$$

$J(\boldsymbol{q})$ 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 $J(\boldsymbol{q})$ [13].

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 $\ddot{\boldsymbol{q}}$. It can be written

$$\boldsymbol{\Gamma} = A(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\epsilon}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{\Gamma}_{ext}, \tag{3}$$

where $A(\boldsymbol{q})$, $\boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, $\boldsymbol{g}(\boldsymbol{q})$, $\boldsymbol{\epsilon}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ 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 is a joint space to joint space mapping with only one solution. It is of course of interest to learn this mapping since it captures a lot of properties of the system, but this is not the scope of this work (interested readers can refer to [7] and [14]). Thus the mapping to the dynamic level 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].

B. Model-based control at the velocity level

In the redundant and non singular case, i.e., $\mathrm{rank}(J(\boldsymbol{q})) = m$ and $m < n$, there is an infinite number of generalised inverses of $J(\boldsymbol{q})$. Among these inverses, weighted pseudoinverses provide minimum norm solutions [16] and can be written

$$J(\boldsymbol{q})^{\sharp} = W_q^{-1} J(\boldsymbol{q})^T \left[J(\boldsymbol{q})\, W_q^{-1} J(\boldsymbol{q})^T \right]^{-1}, \tag{4}$$

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

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

$$\dot{\boldsymbol{q}} = J(\boldsymbol{q})^{\sharp}\,\dot{\boldsymbol{\xi}}^{\star}. \tag{5}$$

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

The system being redundant with respect to the task, Equation (5) is not the unique solution to the inverse mapping problem and other solutions of interest are those allowing internal motions that do not induce any perturbation on the task. This particular subset of solutions corresponds to the nullspace of $J(\boldsymbol{q})$ and the general form of the minimum norm solutions to Equation (2) can be written

$$\dot{\boldsymbol{q}} = J(\boldsymbol{q})^{\sharp}\,\dot{\boldsymbol{\xi}}^{\star} + P_J(\boldsymbol{q})\,\dot{\boldsymbol{q}}_0, \tag{6}$$

where $P_J(\boldsymbol{q})$ is a projector on the nullspace of $J(\boldsymbol{q})$ and $\dot{\boldsymbol{q}}_0$ is any vector of dimension n. Equation (6) is the minimum norm solution which minimises $||\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_0||_{W_q}$. A commonly used expression for $P_J(\boldsymbol{q})$ is

$$P_J(\boldsymbol{q}) = \left(I_n - J(\boldsymbol{q})^{\sharp} J(\boldsymbol{q}) \right). \tag{7}$$

[1] A velocity kinematics and dynamic combined mapping known as the Operational Space Formulation is proposed by Khatib in [15]. It is of interest if one wants to learn the dynamic behaviour of a specific task.

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

However, efficient computation of $J(\boldsymbol{q})^\sharp$ and $P_J(\boldsymbol{q})$ can be done using the SVD [17] of $J(\boldsymbol{q})$. The SVD of $J(\boldsymbol{q})$ 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

$$J^+ = VD^+U^T, \tag{8}$$

where the computation of D^+ is straightforward given its diagonal nature. Regarding $P_J(\boldsymbol{q})$, it can be computed using the $m+1$ to n columns of V which form a basis for the nullspace of $J(\boldsymbol{q})$

$$P_J(\boldsymbol{q}) = [\boldsymbol{v}_{m+1} \dots \boldsymbol{v}_n] \left[\boldsymbol{v}_{m+1}^T \dots \boldsymbol{v}_n^T\right]^T, \tag{9}$$

where $\boldsymbol{v}_i$ 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 [13].

C. Redundancy resolution schemes

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

Two tasks with associated Jacobian matrices J_1 and J_2 are 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 task space are in minimum number and $rank(J_{ext}) \leq n$.

Given this definition, one has to consider the under constrained (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 [18]

$$\dot{\boldsymbol{q}} = J_1^\sharp \dot{\boldsymbol{\xi}}_1^\star + (J_2 P_{J_1})^\sharp \left(\dot{\boldsymbol{\xi}}_2^\star - J_2 J_1^\sharp \dot{\boldsymbol{\xi}}_1^\star\right). \tag{10}$$

In the compatible case, tasks 1 and 2 will be achieved perfectly whereas in the incompatible case 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 [19]. This task projection scheme can be extended to several tasks, interested readers can refer to [20].

Another method, originally proposed in [21], 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 automatically achieved. 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, in the remaining of the paper, that in the case of complex missions where the tasks and constraints constantly evolve, one cannot ensure compatibility at each time and. In this regard, the solution provided by Equation (10) should be preferred.

Finally, in the case of constraints such as joints limits, a possibility consists in choosing $\dot{\boldsymbol{q}}_0$ in Equation (6) as the opposite of the gradient of a cost function $Q(\boldsymbol{q})$. 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

$$\dot{\boldsymbol{q}} = J(\boldsymbol{q})^\sharp \dot{\boldsymbol{\xi}}^\star - \alpha P_J \nabla Q(\boldsymbol{q}), \tag{11}$$

where α is a positive scalar used to tune the steepness of the gradient descent. 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.

III. Learning the forward velocity kinematics model

In this section, the LWPR algorithm is briefly introduced and the advantages of learning forward (instead of inverse) velocity kinematics mappings in redundant cases are exposed. We also show how one can easily access the Jacobian matrices representing these mappings from learned forward kinematics models.

A. Locally Weighted Projection Regression

Locally Weighted Projection Regression (LWPR) is a function approximator which provides accurate approximation in very large spaces in $O(k)$, where k is the number of data used to perform this estimation. Here, we use LWPR to learn the forward kinematics model of our robot. LWPR uses a combination of linear models that are valid on a zone of the input spaces. 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 weighed 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 latest version of the algorithm [22] also computes the gradient of the output with respect to the input.

B. Learning the forward kinematics model with LWPR

Learning forward models for a redundant robot does not raise particular problems. By contrast, as explained in Section II, 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 II-B.

Using the extended Jacobian approach, D'Souza *et al.* in [6] propose to directly learn the inverse velocity kinematics model supplying LWPR with the input $\left(\boldsymbol{q}, \dot{\boldsymbol{\xi}}\right)$ and the output $(\dot{\boldsymbol{q}})$

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

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

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) of a 3 degrees of freedom robot, giving as input the joint parameters $\boldsymbol{q}$ adjusted in $[0, 2\pi[$ and the task space parameters $\boldsymbol{\xi}$ as output

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

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 $\boldsymbol{q}$, 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 [23].

IV. Simulations

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 [6] but rather the forward one which we inverse.

A. Protocol

We have chosen to evaluate the compared approaches on a 3 degrees of freedom planar system, shown in Figure 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 [24]. 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

$$\dot{\boldsymbol{\xi}}^{\star} = K_p\left(\boldsymbol{\xi}^{\star} - \boldsymbol{\xi}\right), \tag{12}$$

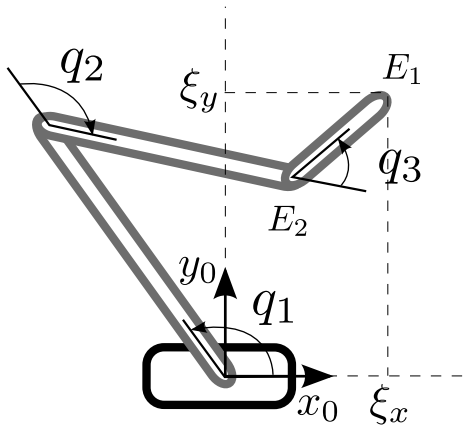

Fig. 1. Schematic view of our simulated system

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 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 of Equation (10) and the estimated Jacobian matrices and projector which can be written

$$\dot{\boldsymbol{q}} = \hat{J}_1^{+}\dot{\boldsymbol{\xi}}_1^{\star} + \left(\hat{J}_2 P_{\hat{J}_1}\right)^{+}\left(\dot{\boldsymbol{\xi}}_2^{\star} - \hat{J}_2\hat{J}_1^{+}\dot{\boldsymbol{\xi}}_1^{\star}\right). \tag{13}$$

For the sake of simplicity in this paper, we keep the weighting matrix W_q equal to identity.

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

$$\left[\hat{\boldsymbol{\xi}}_1, \hat{J}_1\right] = LWPR_{predict}\left(\boldsymbol{q}, model_1\right)$$

and

$$\left[\hat{\boldsymbol{\xi}}_2, \hat{J}_2\right] = LWPR_{predict}\left(\boldsymbol{q}, model_2\right).$$

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

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \hat{J}_1 \\ \hat{J}_2 \end{bmatrix}^{-1} \begin{bmatrix} \dot{\boldsymbol{\xi}}_1^{\star} \\ \dot{\boldsymbol{\xi}}_2^{\star} \end{bmatrix}. \tag{14}$$

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

$\dot{\boldsymbol{q}}$ obtained from Equations (13) or (14) is then differentiated and the resulting joints acceleration vector is used to compute the actuation torque based on the dynamics model described by Equation (3) which we suppose to know and is obtained from Arboris (see above).

B. 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 [25]. 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 $(\boldsymbol{q}, \boldsymbol{\xi})$ pairs.

Then LWPR comes with some parameters that need to be initialised. We initialise LWPR as Klanke *et al.* propose to do in [25]. The $init_D$ coefficient corresponds to the initial size of all receptive field. It is important because it 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 [25]).

C. Experiments

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 II-C without any secondary task to the extended Jacobian approach, where the extension is realised by adding a one dimension constraint on point (E_2): $\xi_{2x}^\star = 0.40\ m$.

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 (13). In the case of the extended Jacobian method, J_{ext} is chosen as in the previous experiment.

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 J_{ext}.

V. Results

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

A. Babbling phase

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

$$NMSE = \frac{1}{\sigma^2}\frac{1}{N}\sum_i^N (\boldsymbol{y_i} - \boldsymbol{\hat{y}_i})^2$$

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

To actually compute this error, we fixed the velocity of each joint to $1.00\ rad.s^{-1}$. As can be seen on Figure 2, 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. The physical time for this babbling phase is about three minutes which is rather short given the fact that babbling is only necessary once.

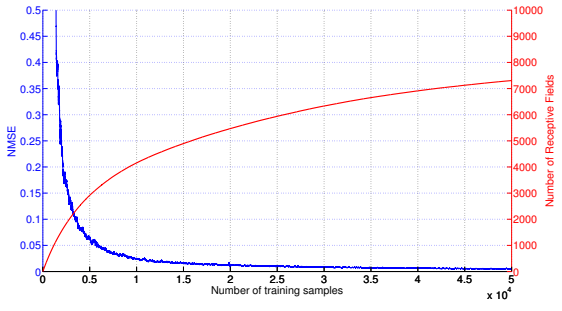

Fig. 2. Evolution of the Normalised Mean Square Error of the LWPR task space velocity prediction (blue, scale left) and of the number of receptive fields for one output (red, scale right) for a 50000 samples babbling phase (average over 40 trials).

Regarding the number of receptive fields, in the end of each experiment, it varies between 2000 et 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.

B. 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 3.(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 3.(b) (between 2s and 4s) and 3.(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 (0.01m) in term of the point that has to be reached are met. After 20s (Figure 3.(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.

C. 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 $\vec{x}_0$ axis (see Figure 1) of the wrist of the robot (point (E_2)). This is illustrated on Figure 4 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 task point is not reached yet, which explains the large error (the initial error, not shown on the figure, is 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 both redundancy resolution schemes, if the Jacobian matrices are not accurately predicted, errors will disturb the tasks. Similarly, when specifically using the projection method, an error in the prediction of the first task Jacobian will induce an error in the computation of the

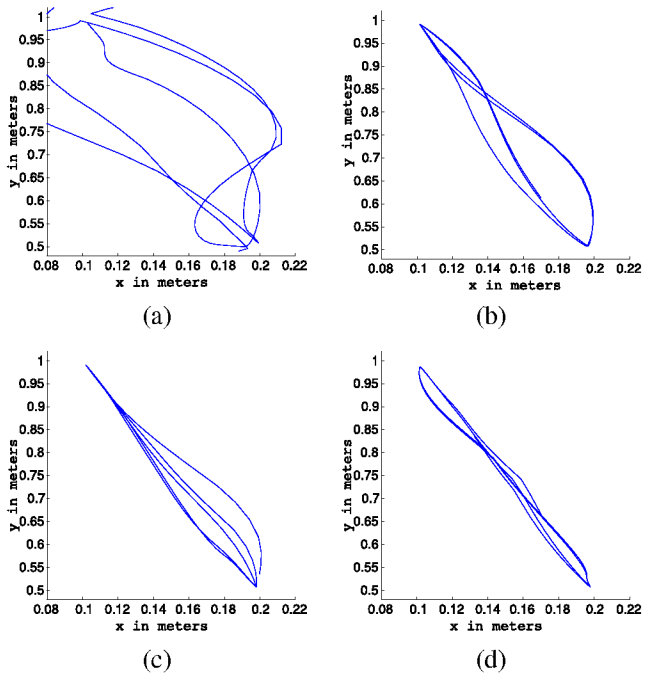

Fig. 3. Evolution of operational trajectories while learning.

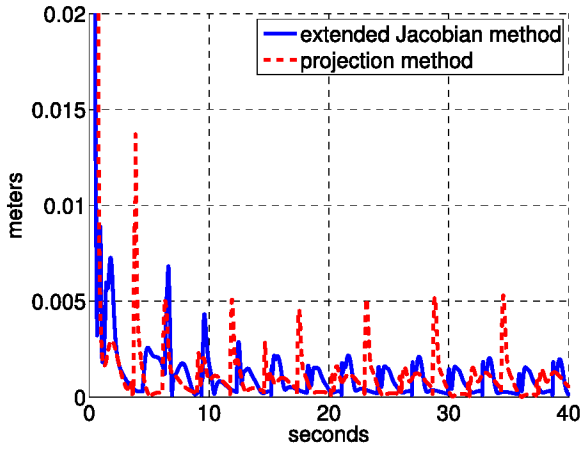

Fig. 4. Norm of the end-effector error induced by a second compatible task for two different controllers.

associated projector leading to disturbances induced by the second task on the first one.

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

D. 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 5 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 6 gives a view of the positioning errors for both tasks. Similarly to the last experiment, error propagation effects are present but once again results are more than satisfactory.

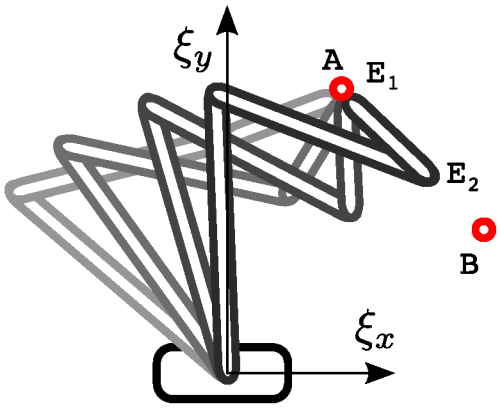

Fig. 5. Robot realising two tasks with priority in the incompatible case.

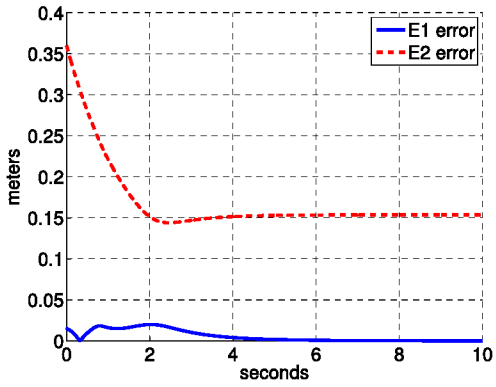

Fig. 6. Reaching errors for the first (blue, plain line) and second task (red, dotted line) in the incompatible case.

VI. 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 V-A).

The results of the experiments for the under constrained case, shows that after some initialisations with motor babbling, our method is able to improve 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 [6], 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 more than 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 task 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. 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. The fact of combining two tasks in a single one in order to simplify the inversion problems leads to an unnecessary growth of 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 is depending 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 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 (4)). That is not the case when inverse models are learned directly since this corresponds to a specific choice of inverse. Also, learning

[3]The use of a proper weighting matrix (different from the Identity) can be crucial in the dynamic case [15].

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.

VII. 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 inverse 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 will be to extend this work to the case of trajectory tracking instead of reaching tasks using resolved motion rate control. This may induce more on-line learning during the control phase and we will try to demonstrate that redundancy usage and tasks combination are possible in that more complex case too.

A second extension will be to study the behaviour of our approach under perturbations to validate its on-line adaptation capabilities. We will also verify that this approach demonstrates good performances in a wider variety of tasks and combination (joint limits avoidance, external collision avoidance) as well as in the case of using different types of inversion.

Long term perspectives include an extension of our framework to systems with a larger number of degrees of freedom as well as a sensivity analysis of the scaling effects on the required length of the babbling phase (see [26] for studies regarding that matter). 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 which can be handled using LWPR.

References

[1] G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori, "The iCub humanoid robot: an open platform for research in embodied cognition," in *PerMIS: Performance Metrics for Intelligent Systems Workshop*, Washington DC, USA, Aug. 2008.

[2] Willow Garage, "Overview of the PR2 robot," http://www.willowgarage.com/pages/robots/pr2-overview.

[3] G. Sun and B. Scassellati, "Reaching through learned forward models," in *Proceedings of the IEEE-RAS/RSJ International Conference on Humanoid Robots (Humanoids)*, Los Angeles, USA, Nov. 2004.

[4] A. Shon, K. Grochow, and R. Rao, "Robotic imitation from human motion capture using gaussian processes," in *Proceedings of the IEEE-RAS/RSJ International Conference on Humanoid Robots (Humanoids)*, 2005.

[5] S. Calinon, F. Guenter, and A. Billard, "On Learning, Representing and Generalizing a Task in a Humanoid Robot," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 37, no. 2, pp. 286–298, 2007.

[6] 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)*, vol. 1, 2001, pp. 298–303.

[7] J. Peters and S. Schaal, "Learning to control in operational space," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 197–212, 2008.

[8] L. Natale, F. Nori, G. Metta, and G. Sandini, "Learning precise 3d reaching in a humanoid robot," in *Proceedings of the IEEE International Conference of Development and Learning (ICML)*, London, UK, July 2007.

[9] A. Liégeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 7, no. 12, pp. 868–871, Dec. 1977.

[10] Y. Nakamura, *Advanced Robotics: redundancy and optimization*. Addison Wesley, 1991, ISBN 0-201-15198-7.

[11] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *The International Journal of Humanoid Robotics*, vol. 2, no. 4, pp. 505–518, Dec. 2005.

[12] S. Vijayakumar, A. DSouza, and S. Schaal, "LWPR: A Scalable Method for Incremental Online Learning in High Dimensions," *Edinburgh University Press*, 2005.

[13] A. Ben Israel and T. Greville, *Generalized Inverses : Theory and Applications*, 2nd ed. Springer, 2003, ISBN 0-387-00293-6.

[14] D. Nguyen-Tuong, J. Peters, M. Seeger, and B. Schoelkopf, "Learning inverse dynamics: a comparison," in *Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*, 2008.

[15] O. Khatib, "A unified approach to motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, Feb. 1987.

[16] K. Doty, C. Melchiorri, and C. Bonivento, "A theory of generalized inverses applied to Robotics," *The International Journal of Robotics Research*, vol. 12, no. 1, pp. 1–19, Feb. 1993.

[17] G. Golub and C. Van Loan, *Matrix computations*, 3rd ed. The John Hopkins University Press, 1996, ISBN 0-8018-5414-8.

[18] A. Maciejewski and C. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *The International Journal of Robotics Research*, vol. 4, no. 3, pp. 109–117, 1985.

[19] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 398–410, June 1997.

[20] N. Mansard and F. Chaumette, "Task sequencing for sensor-based control," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 60–72, Feb. 2007.

[21] J. Baillieul, "Kinematic programming alternatives for redundant manipulators," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, vol. 2, Mar. 1985, pp. 722–728.

[22] Statistical Machine Learning and Motor Control Group, "Locally Weighted Projection Regression," http://www.ipab.informatics.ed.ac.uk/slmc/software/lwpr.

[23] F. Larsson, E. Jonsson, and M. Felsberg, "Visual servoing for floppy robots using LWPR," Workshop on Robotics and Mathematics (RoBoMat), Coimbra, Portugal, Sept. 2007.

[24] S. Barthelemy and P. Bidaud, "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, July 2008.

[25] S. Klanke, S. Vijayakumar, and S. Schaal, "A library for locally weighted projection regression," *Journal of Machine Learning Research*, vol. 9, pp. 623–626, Apr. 2008.

[26] D. Mitrovic, S. Klanke, and S. Vijayakumar, "Learned system dynamics for adaptive optimal feedback control," Neural Information Processing Conference (NIPS) : Workshop on Robotics Challenges for Machine Learning, Dec. 2007.