Statistical Learning for Humanoid Robots

Sethu Vijayakumar (sethu@usc.edu), Aaron D'Souza (adsouza@usc.edu) and Stefan Schaal (sschaal@usc.edu)
Computer Science & Neuroscience and Kawato Dynamic Brain Project
University of Southern California, Los Angeles, CA 90089-2520, USA

Tomohiro Shibata (tom@erato.atr.co.jp)
Kawato Dynamic Brain Project, ERATO
Japan Science & Technology Corp., Kyoto 619-0288, Japan

Jörg Conradt (conradt@ini.phys.ethz.ch)
Univeristy/ETH Zurich, Winterthurerstr. 190, CH-8057 Zurich

Abstract. The unknown complexity of the kinematic and dynamic structure of humanoid robots make conventional analytical approaches to control increasingly unsuitable for such systems. Learning techniques offer a possible way to aid controller design if insufficient analytical knowledge is available, and learning approaches seem mandatory when humanoid systems are supposed to become completely autonomous. While recent research in neural networks and statistical learning has focused mostly on learning from finite data sets without stringent constraints on computational efficiency, learning for humanoid robots requires a different setting, characterized by the need for real-time learning performance from an essentially infinite stream of incrementally arriving data. This paper demonstrates how even high-dimensional learning problems of this kind can successfully be dealt with by techniques from nonparametric regression and locally weighted learning. As an example, we describe the application of one of the most advanced of such algorithms, Locally Weighted Projection Regression (LWPR), to the on-line learning of three problems in humanoid motor control: the learning of inverse dynamics models for model-based control, the learning of inverse kinematics of redundant manipulators, and the learning of oculomotor reflexes. All these examples demonstrate fast, i.e., within seconds or minutes, learning convergence with highly accurate final peformance. We conclude that real-time learning for complex motor system like humanoid robots is possible with appropriately tailored algorithms, such that increasingly autonomous robots with massive learning abilities should be achievable in the near future.

1. Introduction

The necessity for adaptive control is becoming more apparent as the scale of control systems gets increasingly more complex, for instance, as experienced in the fields of advanced robotics, factory automation, and autonomous vehicle control. Humanoid robots, the focus of this paper, are a typical example. Humanoid robots are high dimensional movement systems for which classical system identification and control techniques are often insufficient due to unknown sources of nonlinearities inherent in these systems. Learning techniques are a possible way to overcome such limitations by aiding the design of appropriate control laws (Slotine, 1991), which often involve decisions based on a multitude of sensors and actuators. Learning also seems to be the only viable research approach towards the generation of flexible autonomous robots that can perform multiple tasks (Schaal, 1999), with the hope of creating a completely *autonomous* humanoid robot at some point.

Among the characteristics of the motor learning problems in humanoid robots are high dimensional input spaces with potentially redundant and irrelevant dimensions, nonstationary input and output distributions, essentially infinite training data sets with no representative validation sets, and the need for continual learning. Most learning tasks fall into the domain of regression problems, e.g., as in learning dynamics models, or they at least involve regression problems, e.g., as in learning a policy with reinforcement learning. Interestingly, the class of on-line learning of regression problems with the characteristics above has so far not been conquered by the new developments in statistical learning. Bayesian inference (Bishop, 1995) is usually computationally too expensive for real-time application as it requires representation of the complete joint probablity densities of the data. The framework of structural risk minimization(Vapnik, 1995), the most advanced in form of Support Vector Machines, excels in classification and finite batch learning problems, but has yet to show compelling performance in regression and incremental learning. However techniques from non-parametric regression, in particular the methods of locally weighted learning (Atkeson et al, 1997), have recently advanced to meet all the requirements of real-time learning in high-dimensional spaces (Schaal et al, 2000).

In this paper, we will present one of the most advanced locally weighted learning algorithms, Locally Weighted Projection Regression (LWPR), and its application to three challenging problems of learning in humanoid robotics, i.e., (i) an inverse dynamics model of a 7 DOF anthropomorphic robot, (ii) an inverse kinematics map of a redundant dextrous arm, and (iii) the bio-mimetic gaze stabilization of a humanoid oculomotor system. In the following sections, we will first explain the LWPR algorithm and then introduce the various learning tasks and illustrate learning results from real-time learning on the actual robots for each of the tasks. To the best of our knowledge, this is the first time that real-time learning of such complex models has been accomplished in robot control.

2. Locally Weighted Projection Regression

The core concept of our learning approach is to approximate nonlinear functions by means of piecewise linear models (Atkeson et al, 1997). The learning system automatically determines the appropriate number of local models, the parameters of the hyperplane in each model, and also the region of validity, called receptive field (RF), of each of the model, formalized as a Gaussian kernel:

$$w_k = exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{c}_k)^T \mathbf{D}_k (\boldsymbol{x} - \boldsymbol{c}_k)), \tag{1}$$

Given a query point $\boldsymbol{x}$, each linear model calculates a prediction $y_k = \beta_k \boldsymbol{x}$. The total output of the learning system is the weighted mean of all K linear models:

$$\hat{y} = \sum_{k=1}^{K} w_k y_k / \sum_{k=1}^{K} w_k,$$

also illustrated in Fig. 1. Learning in the system involves determining the linear regression parameter β_k and the distance metric $\mathbf{D}_k$. The center $\boldsymbol{c}_k$ of the RF remains fixed. Local models are created as and when needed as described in Section 2.3.

2.1. Local Dimensionality Reduction

Despite its appealing simplicity, the "piecewise linear modeling" approach becomes numerically brittle and computationally too expensive in high dimensional input spaces. Given the empirical observation that high dimensional data lie often on locally low dimensional distributions, it is possible to develop an efficient approach to exploit this property. Instead of using ordinary linear regression to fit the local hyperplanes, we suggest to employ Partial Least Squares (PLS) (Wold, 1975; Frank & Friedman, 1993). PLS recursively computes orthogonal projections of the input data and performs single variable regressions along these projections on the residuals of the previous iteration step. Table I illustrates PLS in pseudocode for a global linear model where the input data is in the rows of the matrix $\mathbf{X}$, and the corresponding one dimensional output data is in the vector $\boldsymbol{y}$. The key ingredient in PLS is to use the direction of maximal correlation between the residual error and the input data as the projection direction at every regression step. Additionally, PLS regresses the inputs of the previous step against the projected inputs $\boldsymbol{s}_r$ in order to ensure the orthogonality of all the projections $\boldsymbol{u}_r$ (Step 2b). Actually, this additional regression could be avoided by replacing $\boldsymbol{p}_r$ with $\boldsymbol{u}_r$, similar to techniques used in principal

Table I. Pseudocode of PLS projection regression

```
1. Initialize: $\mathbf{X}_{res} = \mathbf{X}$, $\boldsymbol{y}_{res} = \boldsymbol{y}$
2. For $r = 1$ to $R$ (# projections)
    a) $\boldsymbol{u}_r = \mathbf{X}_{res}^T \boldsymbol{y}_{res}$;   $\beta_r = \boldsymbol{s}_r^T \boldsymbol{y}_{res} / (\boldsymbol{s}_r^T \boldsymbol{s}_r)$ where $\boldsymbol{s}_r = \mathbf{X}_{res} \boldsymbol{u}_r$.
    b) $\boldsymbol{y}_{res} = \boldsymbol{y}_{res} - \boldsymbol{s}_r \beta_r$;   $\mathbf{X}_{res} = \mathbf{X}_{res} - \boldsymbol{s}_r \boldsymbol{p}_r{}^T$ where $\boldsymbol{p}_r = \mathbf{X}_{res}^T \boldsymbol{s}_r / (\boldsymbol{s}_r^T \boldsymbol{s}_r)$.
```

component analysis (Sanger, 1989). However, using this regression step leads to better performance of the algorithm. This effect is due to the fact that PLS chooses the most effective projections if the input data has a spherical distribution: with only one projection, PLS will find the direction of the gradient and achieve optimal regression results. The regression step in 2b modifies the input data $\mathbf{X}_{res}$ such that each resulting data vectors have coefficients of minimal magnitude and, hence, push the distribution of $\mathbf{X}_{res}$ to become more spherical.

An incremental locally weighted version of the PLS algorithm is derived in Table II (Vijayakumar & Schaal, 2000). Here, $\lambda \in [0, 1]$ denotes a forgetting factor that determines how quickly older data will be forgotten in the various PLS parameters, similar to the recursive system identification techniques (Ljung & Soderstrom, 1986). The variables SS_r, SR_r and SZ_r are memory terms that enable us to do the univariate regression in step 5 in a recursive least squares fashion, i.e., a fast Newton-like method.

Since PLS selects the univariate projections very efficiently, it is even possible to run locally weighted PLS with only *one* projection direction (denoted as LWPR-1). The optimal projection is in the direction of the local gradient of the function to be approximated. If the locally weighted input data forms a spherical distribution in a local model, the single PLS projection will suffice to find the optimal direction. Otherwise, the distance metric (and hence, weighting of the data) will need to be adjusted to make the local distribution more spherical. The learning rule of the distance metric can accomplish this adjustment, as explained below. It should be noted that Steps 8-10 in Table II become unnecessary for the uni-projection case.

2.2. Learning the Distance Metric

The distance metric $\mathbf{D}_k$ and hence, the locality of the receptive fields, can be learned for each local model individually by stochastic gradient descent in a leave-one-out cross validation cost function. Note that

Table II. Incremental locally weighted PLS for one RF centered at $\boldsymbol{c}$

```
Initialization:
    $\boldsymbol{x}_0^0 = \mathbf{0}$, $\boldsymbol{u}^0 = \mathbf{0}$, $\beta_0^0 = 0$,
    $W^0 = 0$
Given: Training point $(\boldsymbol{x}, y)$
    $w = exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{c})^T\mathbf{D}(\boldsymbol{x}-\boldsymbol{c}))$
Update the means :
    $W^{n+1} = \lambda W^n + w$
    $\boldsymbol{x}_0^{n+1} = \frac{\lambda W^n \boldsymbol{x}_0^n + w\boldsymbol{x}}{W^{n+1}}$
    $\beta_0^{n+1} = \frac{\lambda W^n \beta_0^n + wy}{W^{n+1}}$

Update the local model
Initialize:
    $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{x}_0^{n+1}$, $res_1 = y - \beta_0^{n+1}$
For $r = 1 : R$ (# projections)
    1. $\boldsymbol{u}_r^{n+1} = \lambda \boldsymbol{u}_r^n + w\boldsymbol{z}\ res_r$
    2. $s_r = \boldsymbol{z}^T \boldsymbol{u}_r^{n+1} / (\boldsymbol{u}_r^{n+1\,T} \boldsymbol{u}_r^{n+1})$
    3. $SS_r^{n+1} = \lambda SS_r^n + w\ s_r^2$
    4. $SR_r^{n+1} = \lambda SR_r^n + w\ s_r\ res_r$
    5. $\beta_r^{n+1} = SR_r^{n+1} / SS_r^{n+1}$
    6. $res_{r+1} = res_r - s_r \beta_r^{n+1}$
    7. $MSE_r^{n+1} = \lambda MSE_r^n + w\ res_{r+1}^2$
    8. $SZ_r^{n+1} = \lambda SZ_r^n + w\boldsymbol{z}s_r$
    9. $\boldsymbol{p}_r^{n+1} = SZ_r^{n+1} / SS_r^{n+1}$
    10. $\boldsymbol{z} = \boldsymbol{z} - s_r \boldsymbol{p}_r^{n+1}$

Predicting with novel data $(\boldsymbol{x}_q)$: Initialize: $y = \beta_0, \boldsymbol{z} = \boldsymbol{x}_q - \boldsymbol{x}_0$
    Repeat for r=1:R
        – $y = y + \beta_r s_r$ where $s_r = \boldsymbol{u}_r^T \boldsymbol{z}$
        – $\boldsymbol{z} = \boldsymbol{z} - s_r \boldsymbol{p}_r^n$
```

this update does *not* require competitive learning – only a completely local learning rule is needed, and leave-one-out cross validation can be performed *without* keeping data in memory (Schaal & Atkeson. 1998). The update rule can be written as :

$$\mathbf{M}^{n+1} = \mathbf{M}^n - \alpha \frac{\partial J}{\partial \mathbf{M}} \text{ where } \mathbf{D} = \mathbf{M}^T\mathbf{M} \text{ (for positive definiteness)} \tag{2}$$

and the cost function to be minimized is:

$$J = \frac{1}{\sum_{i=1}^M w_i} \sum_{i=1}^M \sum_{r=1}^R \frac{w_i res_{r+1,i}^2}{(1 - w_i \frac{s_{r,i}^2}{\boldsymbol{s}_r^T \mathbf{W} \boldsymbol{s}_r})^2} + \frac{\gamma}{N} \sum_{i,j=1}^N D_{ij}^2 = \sum_{r=1}^R (\sum_{i=1}^M J_{1,r}) + J_2. \tag{3}$$

where M denotes the number of training data, and N the number of input dimensions. A stochastic version of the gradient $\frac{\partial J}{\partial \mathbf{M}}$ can be derived from the cost function by keeping track of several "memory terms" as shown in Table III.

Table III. Derivatives for distance metric update

$$
\begin{aligned}
\frac{\partial J}{\partial \mathbf{M}} &\approx \sum_{r=1}^{R}\Big(\sum_{i=1}^{M}\frac{\partial J_{1,r}}{\partial w}\Big)\frac{\partial w}{\partial \mathbf{M}} + \frac{w}{W^{n+1}}\frac{\partial J_2}{\partial \mathbf{M}} \ \text{(stochastic update)}\\
\frac{\partial w}{\partial M_{kl}} &= -\frac{1}{2}w(\boldsymbol{x}-\boldsymbol{c})^T\frac{\partial \mathbf{D}}{\partial M_{kl}}(\boldsymbol{x}-\boldsymbol{c});\quad \frac{\partial J_2}{\partial M_{kl}} = 2\frac{\gamma}{N}\sum_{i=1,j=1}^{N} D_{ij}\frac{\partial D_{ij}}{\partial M_{kl}}\\
\frac{\partial D_{ij}}{\partial M_{kl}} &= M_{kj}\delta_{il} + M_{ki}\delta_{jl};\ \text{where } \delta_{ij} = 1 \text{ if } i=j \text{ else } \delta_{ij}=0.
\end{aligned}
$$

Compute the following for each projection direction r:

$$
\begin{aligned}
\sum_{i=1}^{M}\frac{\partial J_{1,r}}{\partial w} &= \frac{e_{cv,r}^2}{W^{n+1}} - 2\frac{(P_r^{n+1}s_re_r)}{W^{n+1}}H_r^n - 2\frac{(P_r^{n+1}s_r)^2}{W^{n+1}}R_r^n - \frac{E_r^{n+1}}{(W^{n+1})^2}\\
&\quad +[\mathbf{T}_r^{n+1} - 2R_r^{n+1}P_r^{n+1}\mathbf{C}_r^{n+1}]\frac{(\mathbf{I}-\boldsymbol{u}_r\boldsymbol{u}_r^T/(\boldsymbol{u}_r^T\boldsymbol{u}_r))\boldsymbol{z}\ res_r}{W^{n+1}\sqrt{\boldsymbol{u}_r^T\boldsymbol{u}_r}}\\
\mathbf{C}_r^{n+1} &= \lambda\mathbf{C}_r^n + ws_r\boldsymbol{z}^T,\quad e_r = res_{r+1},\quad e_{cv,r} = \frac{e_r}{1-wP_r^{n+1}s_r^2}\\
P_r^{n+1} &= \frac{1}{SS_r^{n+1}},\quad H_r^{n+1} = \lambda H_r^n + \frac{w\ e_{cv,r}s_r}{(1-w\ h_r)}\\
R_r^{n+1} &= \lambda R_r^n + \frac{w^2s_r^2e_{cv,r}^2}{(1-w\ h_r)}\ \text{where } h_r = P_r^{n+1}s_r^2\\
E_r^{n+1} &= \lambda E_r^n + we_{cv,r}^2;\quad \mathbf{T}_r^{n+1} = \lambda\mathbf{T}_r^n + \frac{w(2we_{cv,r}^2s_rP_r^{n+1} - e_{cv,r}\beta_r^{n+1})}{(1-w\ h_r)}\boldsymbol{z}^T
\end{aligned}
$$

2.3. The Complete LWPR Algorithm

All the ingredients above can be combined in an incremental learning scheme that automatically allocates new locally linear models as needed. The final learning network is illustrated in Fig. 1 and an outline of the algorithm is shown in Table IV.

In this pseudo-code, w_{gen} is a threshold that determines when to create a new receptive field, and $\mathbf{D}_{def}$ is the initial (usually diagonal) distance metric in Eq.(1). The initial number of projections is set to $r = 2$. The algorithm has a simple mechanism of determining whether r should be increased by recursively keeping track of the mean-squared error (MSE) as a function of the number of projections included in a local model, i.e., step 7 in the incremental PLS pseudocode. If the MSE at the next projection does not decrease more than a certain percentage of the previous MSE, i.e., $\frac{MSE_{i+1}}{MSE_i} > \phi$, where $\phi \in [0,1]$, the algorithm will stop adding new projections locally. For a diagonal distance metric $\mathbf{D}$ and under the assumption that the number of projections R remains small, the computational complexity of the update of all parameters of

Table IV. Psuedocode of the complete LWPR algorithm

```
– Initialize the LWPR with no receptive field (RF);
– For every new training sample ($\boldsymbol{x}$,y):
    • For k=1 to $K$(# of receptive fields):
        * calculate the activation from Eq.(1)
        * update projections & regression (Table II) and Distance Metric (Table III)
        * check if no. of projections needs to be increased (cf. Section 2.3)
    • If no RF was activated by more than $w_{gen}$;
        * create a new RF with $r = 2$, $\boldsymbol{c} = \boldsymbol{x}$, $\mathbf{D} = \mathbf{D}_{def}$
```

LWPR is linear in the number of input dimensions n. For the LWPR-1 variant, this $O(n)$ computational complexity is always guaranteed.

3. Real-Time Learning for Humanoid Robots

One of the main motivations of the development of LWPR was that model-based control of our humanoid robots with analytical models did not result in sufficient accuracy. The following sections describe how LWPR has allowed us to improve model-based control with models that were learned, i.e., they were acquired very rapidly in real-time while the system was trying to accomplish a task. Our results are one of the first in the learning literature that demonstrate the feasibility of real-time statistical learning in high-dimensional systems such as humanoid robots.

3.1. Real-Time Learning of Inverse Dynamics

A common strategy in robotic and biological motor control is to convert kinematic trajectory plans into motor commands by means of an inverse dynamics model. The inverse dynamics takes the desired positions, velocities, and accelerations of all DOFs of the robot and outputs the appropriate motor commands. In the case of learning with our seven DOF anthropomorphic robot arm (see Fig. 2(a)), the inverse dynamics model receives 21 inputs and outputs 7 torque commands. If derived analytically under a rigid body dynamics assumption (An et al, 1988), the most compact recursive formulation of the inverse dynamics of our robot results in about 20 pages of compact C-code, filled with nested sine and cosine terms. For on-line learning, motor commands need to be

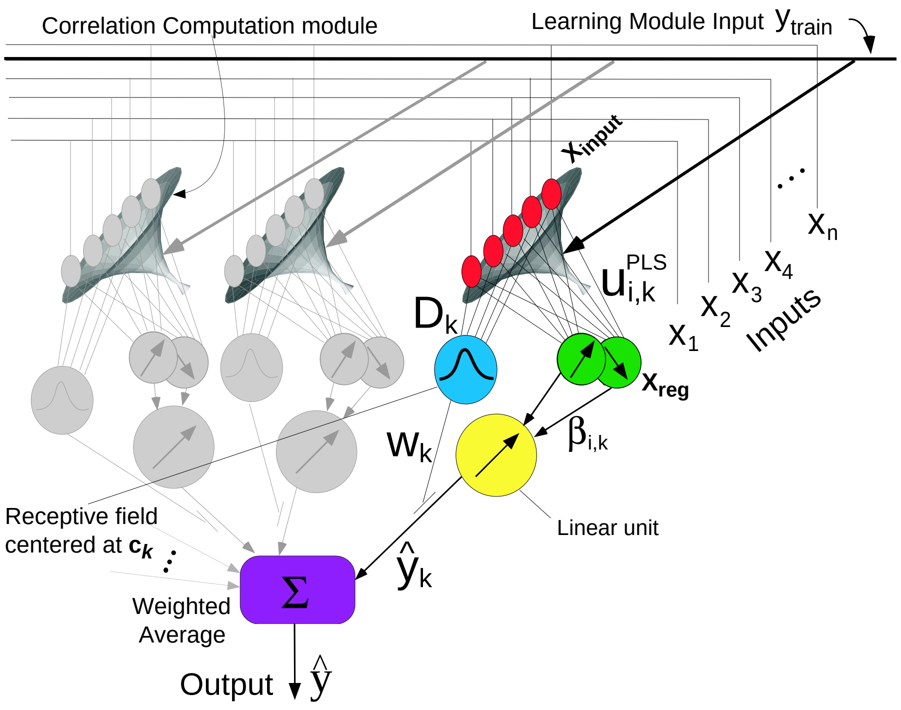

Figure 1. Information processing unit of LWPR

generated from the model at 480Hz in our implementation. Updating the learning system can take place at a lower rate but should remain as high as possible to capture suffcient data in fast movements – we usually achieve about 70Hz updating rate.

Learning regression problems in such high dimensional input space is a daunting problem from the view of the bias-variance trade-off. In learning control, training data is generated by the learning system itself, and it is impossible to assess a priori what structural complexity that data is going to have. Fortunately, actual movement systems do not fill the data space in a completely random way. Instead, when viewed locally, data distributions tend to be low dimensional, e.g., about 4-6 dimensional for the inverse dynamics (Schaal et al, 1998) of our robot instead of the global 21 input dimensions. This property, which is exploited by the LWPR algorithm, is a key element in the excellent real-time performance of our learning scheme.

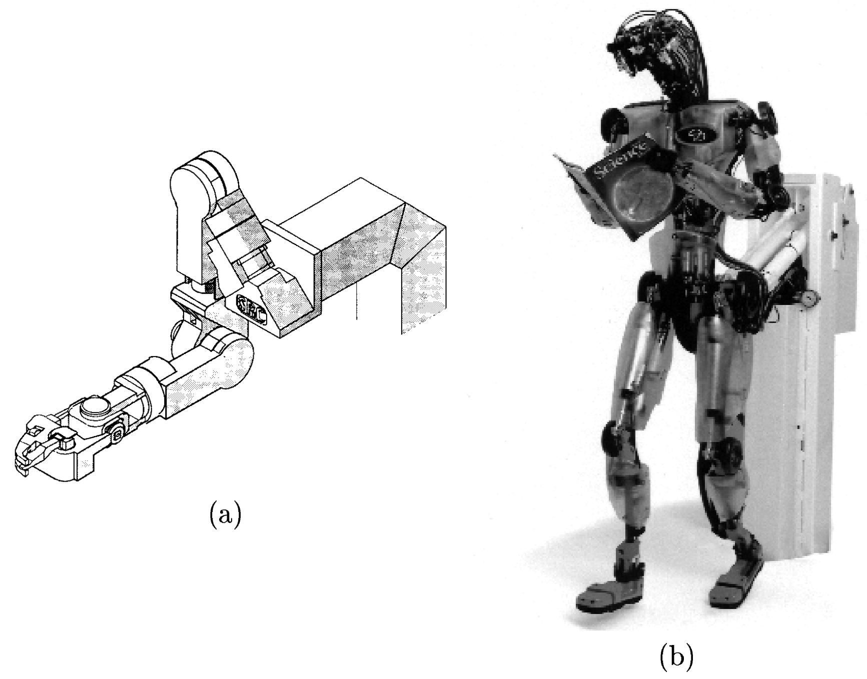

Figure 2. (a) 7-DOF SARCOS dexterous arm (b) 30-DOF Humanoid Robot

3.1.1. *Performance Comparison on a Static Data Set*

Before demonstrating the applicability of LWPR in real-time, a comparison with alternative learning methods will serve to demonstrate the complexity of the learning task. We collected 50,000 data points from various movement patterns from our 7 DOF anthropomorphic robot (Fig. 2a) at 50Hz sampling frequency. 10 percent of this data was excluded as a test set. The training data was approximated by 4 different methods: i) parameter estimation based on an analytical rigid body dynamics model (An et al, 1988), ii) Support Vector Regression (Saunders et al, 1998), iii) LWPR-1, and iv) full LWPR. It should be noted that neither i) nor ii) are incremental learning methods, i.e., they require batch learning. Using a parametric model as suggested in i) and just approximating its open parameters from data results in a global model of the inverse dynamics and is theroretically the most powerful method. However, given that our robot is actuated hydraulically and rather lightweight and compliant, we know that the rigid body dynamics assumption is not fully justified. Method ii), Support Vector Regression, is a relatively new statistical learning approach that was derived from the theory of structural risk minimization. In many re-

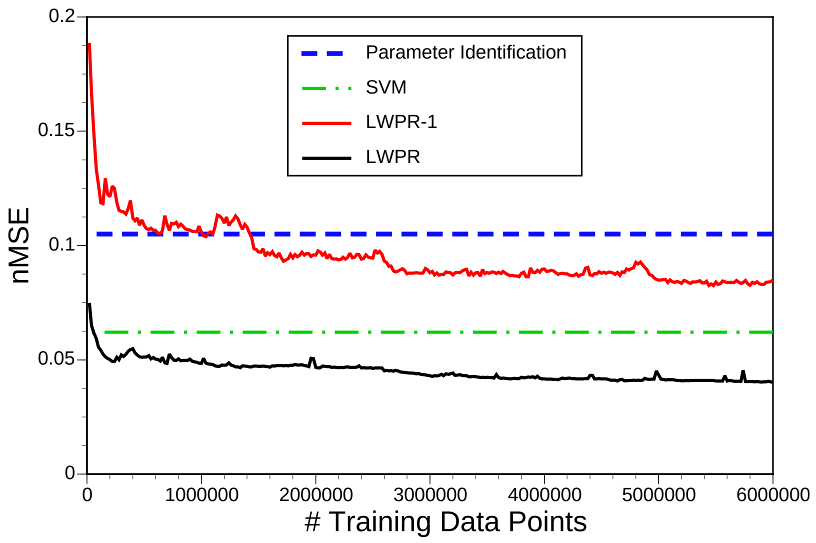

Figure 3. Comparison of generalization error (nMSE) traces for different learning schemes

cent publications, support vector machines have demonstrated superior learning performance over previous algorithms, such that a comparison of this method with LWPR seemed to be an interesting benchmark. LWPR as used in iii) and iv) was exactly the algorithm described in the previous section. Methods ii)-iv) learned a separate model for each output of the inverse dynamics, i.e., all models had a univariate output and 21 inputs. LWPR employed a diagonal distance metric.

Fig. 3 illustrates the function approximation results for the shoulder motor command graphed over the number of training iterations (one iteration corresponds to the update from one data point). Surprisingly, rigid body parameter estimation achieved the worst results. LWPR-1 outperformed parameter estimation, but fell behind SVM regression. Full LWPR performed the best. The results for all other DOFs were analogous. For the final result, LWPR employed 260 local models, using an average of 3.2 local projections. LWPR-1 did not perform better because we used a diagonal distance metric. The abilities of a diagonal distance metric to “carve out” a locally spherical distribution are too limited to accomplish better results – a full distance metric can remedy this problem, but would make the learning updates quadratic in the number of inputs. These results demonstrate that LWPR is a competitive function approximation technique.

3.1.2. *On-line Learning*

We implemented full LWPR on our robotic setup. Out of the four parallel processors of the system, one 366Mhz PowerPC processor was completely devoted to lookup and learning with LWPR. Each DOF had its own LWPR learning system, resulting in 7 parallel learning modules. In order to accelerate lookup and training times, we added a special data structure to LWPR. Each local model maintained a list of all other local models that overlapped sufficiently with it. Sufficient overlap between two local model i and j can be determined from the centers and distance metrics. The point $\boldsymbol{x}$ in input space that is the closest to both centers in the sense of a Mahalanobis distance is $\boldsymbol{x} = (\mathbf{D}_i + \mathbf{D}_j)^{-1}(\mathbf{D}_i\boldsymbol{c}_i + \mathbf{D}_j\boldsymbol{c}_j)$. Inserting this point into Eq.(1) of one of the local models gives the activation w due to this point. Two local models are listed as sufficiently overlapping if $w >= w_{gen}$ (cf. LWPR outline). For diagonal distance metrics, the overlap computation is linear in the number of inputs. Whenever a new data point is added to LWPR, one neighborhood relation is checked for the maximally activated RF. An appropriate counter for each local model ensures that overlap with all other local models is checked exhaustively. Given this "nearest neighbor" data structure and the fact that a movement system generates temporally highly correlated data, lookup and learning can be confined to only few RFs. For every lookup (update), the identification number of the maximally activated RF is returned. The next lookup (update) will only consider the neighbors of this RF. It can be proved that this method performs as good as an exhaustive lookup (update) strategy that excludes RFs that are activated below a certain threshold w_{cutoff}.

The LWPR models were trained on-line while the robot performed a pseudo randomly drifting figure-8 pattern in front of its body. Lookup proceeded at 480Hz, while updating the learning model was achieved at about 70Hz. At certain intervals, learning was stopped and the robot attempted to draw a planar figure-8 at 2Hz frequency for the entire pattern. The quality of these drawing patterns is illustrated in Fig. 4a,b. In Fig. 4a, X_{des} denotes the desired figure-8 pattern, X_{sim} illustrates the figure-8 performed by our robot simulator that uses a perfect inverse dynamics model (but not necessarily a perfect tracking and numerical integration algorithm), X_{param} is the performance of the estimated rigid body dynamics model, and X_{lwpr} shows the results of LWPR. While the rigid body model has the worst performance, LWPR obtained the results comparable to the simulator.

Fig. 4b illustrates the speed of LWPR learning. The X_{nouff} trace demonstrates the figure-8 patterns performed without any inverse dynamics model, just using a low gain PD controller. The other traces

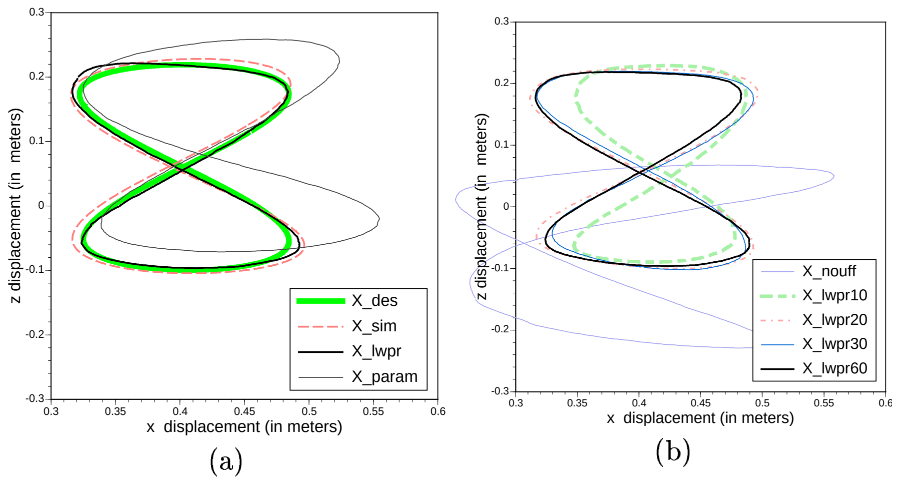

Figure 4. (a) Robot end effector motion traces under different control schemes (b) Progress of online learning with LWPR control

show how rapidly LWPR learned the figure-8 pattern during training: they denote performance after 10, 20, 30, and 60 *seconds* of training. After 60 seconds, the figure-8 is hardly distinguishable from the desired trace.

3.2. Inverse kinematics learning

Since most movement tasks are defined in coordinate systems that are different from the actuator space of the robot, coordinate transformation from task to actuator space must be performed before motor commands can be computed. On a system with redundant degrees-of-freedom (DOFs), this inverse kinematics transformation from external plans to internal coordinates is often ill-posed as it is under-constrained. If we define the intrinsic coordinates of a manipulator as the n-dimensional vector of joint angles $\boldsymbol{\theta} \in \mathcal{R}^n$, and the position and orientation of the manipulator's end effector as the m-dimensional vector $\boldsymbol{x} \in \mathcal{R}^m$, the forward kinematic function can generally be written as:

$$\boldsymbol{x} = f(\boldsymbol{\theta}) \tag{4}$$

while what we need is the inverse relationship:

$$\boldsymbol{\theta} = f^{-1}(\boldsymbol{x}) \tag{5}$$

For redundant systems, like our Sarcos robots (see Fig. 2), solutions to the above equation are non-unique. Traditional inverse kinematics algorithms address how to determine a particular solution in face of multiple solutions by optimizing an additional cost criterion $g = g(\boldsymbol{\theta})$. Most approaches favor local optimizations that compute an optimal change in $\boldsymbol{\theta}$, $\Delta\boldsymbol{\theta}$, for a small change in $\boldsymbol{x}$, $\Delta\boldsymbol{x}$ and then integrate $\Delta\boldsymbol{\theta}$ to generate the entire joint space path. Resolved Motion Rate Control (RMRC) is one such local method which uses the Jacobian $\mathbf{J}$ of the forward kinematics to describe a change of the endeffector's position as:

$$\dot{\boldsymbol{x}} = \mathbf{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{6}$$

This equation can be solved for $\dot{\boldsymbol{\theta}}$ by taking the inverse of $\mathbf{J}$ if it is square i.e. $m = n$, and non-singular, or by using pseudo-inverse computations that minimize g in the null space of $\mathbf{J}$ (Liegeois, 1977):

$$\boldsymbol{\theta} = \mathbf{J}^{\#}\dot{\boldsymbol{x}} - \alpha(\mathbf{I} - \mathbf{J}^{\#}\mathbf{J})\frac{\partial g}{\partial \boldsymbol{\theta}} \tag{7}$$

3.2.1. *Motivation for Learning*

Learning of inverse kinematics is useful when the kinematic model of the robot is not known accurately, when Cartesian information is provided in uncalibrated camera coordinates, or when the computational complexity of analytical solutions becomes too high. For instance, in our humanoid robot we observed that offsets in sensor readings and inaccurate knowledge of the exact kinematics of the robot can lead to significant error accumulations for analytical inverse kinematics computations, and that it is hard to maintain an accurate calibration of the active vision system of the robot. Instead of re-calibrating the entire robot frequently, we would rather employ a self-calibrating, i.e., learning approach. An additional appealing feature of learning inverse kinematics is that it avoids problems due to kinematic singularities– learning works out of experienced data, and such data is always physically correct and does not demand impossible postures as can result from an ill-conditioned matrix inversion.

A major obstacle in learning inverse kinematics, is that the inverse kinematics of a redundant kinematic chain has infinitely many solutions. In the context of eq. (6), this means that multiple $\dot{\boldsymbol{\theta}}_i$, are mapped to the same $\dot{\boldsymbol{x}}$. Algorithms that learn the mapping $\dot{\boldsymbol{\theta}} \leftarrow f^{-1}(\boldsymbol{x})$ average over all the solutions $\dot{\boldsymbol{\theta}}_i$, assuming that different $\dot{\boldsymbol{\theta}}_i$ for the same $\dot{\boldsymbol{x}}$ are due to noise. This may result in an invalid solution if the multiple $\dot{\boldsymbol{\theta}}_i$ lie in a non-convex set, as is frequently the case in robot kinematics (Jordan & Rumelhart, 1992).

This problem can be avoided by a specific input representation to the learning network (Bullock et al, 1993) which allows local averaging over $\dot{\boldsymbol{\theta}}_i$. This can be shown by averaging eq. (6) over multiple $\dot{\boldsymbol{\theta}}_i$ that map to the same $\dot{\boldsymbol{x}}$, for a fixed $\boldsymbol{\theta}$.

$$\langle \dot{\boldsymbol{x}} \rangle = \left\langle \mathbf{J}(\theta)\,\dot{\boldsymbol{\theta}}_i \right\rangle_i \Rightarrow \dot{\boldsymbol{x}} = \mathbf{J}(\boldsymbol{\theta}) \left\langle \dot{\boldsymbol{\theta}}_i \right\rangle = \mathbf{J}(\boldsymbol{\theta})\,\bar{\dot{\boldsymbol{\theta}}}_i \tag{8}$$

Since the Jacobian relates the $\dot{\boldsymbol{x}}$ and $\dot{\boldsymbol{\theta}}_i$ in linear form, even for redundant systems the average of the solutions will result in the desired $\dot{\boldsymbol{x}}$ as long as the averaging is carried out in the vicinity of a particular $\boldsymbol{\theta}$.

Thus we propose to learn the inverse mapping function with our spatially localized LWPR learning system based on the input/output representation $(\dot{\boldsymbol{x}}, \boldsymbol{\theta}) \rightarrow (\dot{\boldsymbol{\theta}})$. This approach will automatically resolve the redundancy problem without resorting to any other optimization approach: the local average solution picked is simply the local average over the solutions that were experienced. The algorithm will also perform well near singular posture since, as mentioned before, it cannot generate joint movements that it has never experienced.

3.2.2. *Applying LWPR to inverse kinematics learning*

In order to apply LWPR to inverse kinematics learning for our humanoid robot, we learn a separate model to generate each of the joint angles such that each of the models performs a 29 (26 degrees of freedom neglecting the 4 degrees of freedom for the eyes, plus 3 Cartesian inputs) to 1 mapping $(\dot{\boldsymbol{x}}, \boldsymbol{\theta}) \rightarrow (\dot{\theta}_l)$, and we have 26 such models $(l = 1, \ldots, 26)$.

The resolution of redundancy requires creating an optimization criterion that allows the system to choose a *particular* solution to the inverse kinematics problem. Given that our robot is a humanoid robot, we would like the system to assume a posture that is as "natural" as possible. Our definition of "natural" corresponds to the posture being as close as possible to some default posture $\boldsymbol{\theta}_{opt}$, as advocated by behavioral studies (Cruse & Brüer, 1987). Hence the total cost function for training LWPR can be written as follows:

$$Q = \frac{1}{2}\left(\dot{\boldsymbol{\theta}} - \hat{\dot{\boldsymbol{\theta}}}\right)^T \left(\dot{\boldsymbol{\theta}} - \hat{\dot{\boldsymbol{\theta}}}\right) + \frac{1}{2}\alpha \left(\hat{\dot{\boldsymbol{\theta}}} - \frac{\Delta\boldsymbol{\theta}}{\Delta t}\right)^T \mathbf{W} \left(\hat{\dot{\boldsymbol{\theta}}} - \frac{\Delta\boldsymbol{\theta}}{\Delta t}\right) \tag{9}$$

where $\Delta\boldsymbol{\theta} = \boldsymbol{\theta}_{opt} - \boldsymbol{\theta}$ represents the distance of the current posture from the optimal posture $\boldsymbol{\theta}_{opt}$, $\mathbf{W}$ is a diagonal weight matrix, and $\hat{\dot{\boldsymbol{\theta}}}$ is the current prediction of LWPR for $\mathbf{z} = (\dot{\mathbf{x}}, \boldsymbol{\theta})$. Minimizing Q can be achieved by presenting LWPR with the target values:

$$\dot{\boldsymbol{\theta}}_{target} = \dot{\boldsymbol{\theta}} - \alpha\mathbf{W}\left(\hat{\dot{\boldsymbol{\theta}}} - \Delta\boldsymbol{\theta}\right) \tag{10}$$

These targets are commposed of the self-supervised target $\dot{\boldsymbol{\theta}}$, slightly modified by a component to enforce the optimization of the cost function within the null space of the Jacobian (c.f. eq. (7)).

As an exploration strategy, we initially bias the output of LWPR with a term that creates a motion towards $\boldsymbol{\theta}_{opt}$:

$$\tilde{\dot{\boldsymbol{\theta}}} = \hat{\dot{\boldsymbol{\theta}}} + \frac{1}{n_r}\Delta\boldsymbol{\theta} \tag{11}$$

The strength of the bias decays with the number of data points n_r seen by the largest contributing local model of LWPR. This additional term allows creating meaningful (and importantly, data-generating) motion even in regions of the joint space that have not yet been explored. This enables us to learn inverse kinematics "on the fly", i.e., while attempting to perform the required task itself.

An important aspect of our formulation of the inverse kinematics problem is that although the inputs to the learning problem comprise $\dot{\mathbf{x}}$ and $\boldsymbol{\theta}$, the locality of the local model is a function of only $\boldsymbol{\theta}$, while the linear projection directions (given this locality in $\boldsymbol{\theta}$) are solely dependent on $\dot{\mathbf{x}}$ (cf. Eq.(8)). We encode this prior knowledge into LWPR's learning process by setting the initial values of the diagonal terms of the distance metric $\mathbf{D}$ in Eq. (1) that correspond to the $\dot{\mathbf{x}}$ variables to zero. This bias ensures that the locality of the receptive fields in the model is solely based on $\boldsymbol{\theta}$.

LWPR has the ability to determine and ignore inputs that are locally irrelevant to the regression, but we also provide this information by normalizing the input dimensions such that the variance in the relevant dimensions is large. This scaling results in larger correlations of the relevant inputs with the output variables and hence biases the projection directions towards the relevant subspace. We use this feature to scale the dimensions corresponding to the $\dot{\mathbf{x}}$ variables so that the regression within a local model is based primarily on this subspace.

3.2.3. *Experimental Evaluations*

The goal task in each of the experiments was to track a figure-eight trajectory in Cartesian space created by simulated visual input to the robot. In each of the figures in this section, the performance of the system is plotted along with that of an analytical pseudo-inverse solution (c.f. Eq. (7) that was available for our robot from previous work (Tevatia & Schaal, 2000).

The system was first trained on data generated from small sinusoidal motions of each degree of freedom about a randomly chosen mean in $\boldsymbol{\theta}$ space. Every few seconds this mean is repositioned. The performance

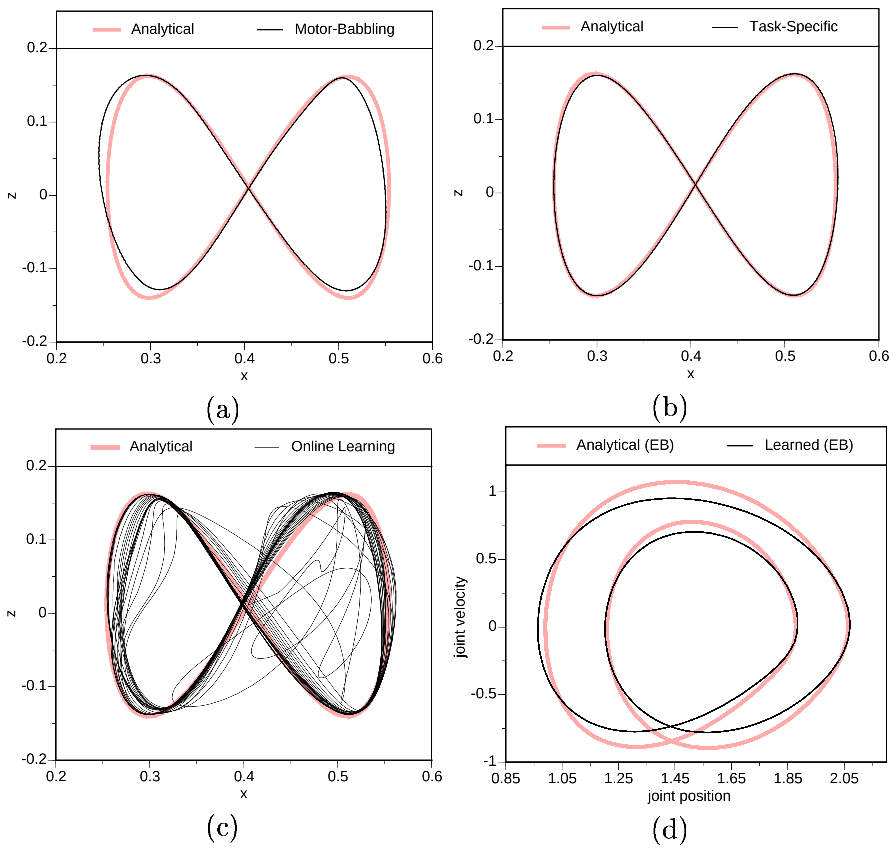

Figure 5. Tracking a figure eight with learned inverse kinematics. (a) Performance after training with motor babbling (b) Results after improving performance using the data seen on the task (c) Performance during the first 3 minutes of learning from scratch on the task (d) Phase plot of joint position and joint velocity

of the system after training the system on this “motor babbling” for 10 minutes is shown in Fig. 5(a).

In the second experiment, the robot executed the figure-eight again, using the trained LWPR from the first experiment. In this case however, the system was allowed to improve itself with the data collected while performing the task. As shown in Fig. 5(b), after merely 1 minute of additional learning, the system performs as well as the analytical pseudo-inverse solution.

The final experiment started with an untrained system, and endeavored to learn the inverse kinematics from scratch, while performing the figure-eight task itself. Fig. 5(c) shows the progression of the system’s performance from the beginning of the task to about 3 minutes into the learning. The system initially starts out making slow inaccurate movements. As it collects data however, it rapidly converges towards the desired trajectory. Within a few more minutes of training, the performance approached that seen in Fig. 5(b).

It is important to note that for redundant manipulators, following a periodic trajectory in operational space does not imply consistency in joint space, i.e., the trajectory followed in joint space may not be cyclic since there could be aperiodic null space motion that does not affect tracking accuracy. Fig. 5(d) shows a phase plot of one of the joints (elbow flexion and extension), over about 30 cycles of the figure-eight trajectory after learning had converged. The presence of a single loop over all cycles shows that the inverse kinematics solution found by our algorithm is indeed consistent.

3.3. Learning for Biomimetic Gaze Stabilization

Oculomotor control in a humanoid robot faces similar problems as biological oculomotor systems, i.e., the stabilization of gaze in face of unknown perturbations of the body, selective attention, stereo vision, and dealing with large information processing delays. Given the nonlinearities of the geometry of binocular vision as well as the possible nonlinearities of the oculomotor plant, it is desirable to accomplish accurate control of these behaviors through learning approaches. Here, we describe the application of LWPR to a learning control system for the phylogenetically oldest behaviors of oculomotor control, the stabilization reflexes of gaze.

In our recent work (Shibata & Schaal, 2001), we described how control theoretically reasonable choices of control components result in an oculomotor control system that resembles the known functional anatomy of the primate oculomotor system. The resulting control circuitry for such a system is shown in Fig. 6. The core of the learning system is derived from the biologically inspired principle of feedback-error learning combined with the LWPR algorithm. There are essentially three blocks in the system (c.f. Fig. 6): (1) the middle block which is the vestibular (head velocity) input based linear feedforward controller with conservatively low gains (2) the top block that makes up the non-linear feedforward controller (continuously adapted using LWPR) with vestibular inputs and (3) a lower block which is the retinal slip based negative feedback controller that generates a delayed error signal to both the linear (fixed) feedforward control and the non-linear (continuously learned) feedforward circuit.

Feedback Error Learning (FEL) is a principle of learning motor control. It employs an approximate way of mapping sensory errors into

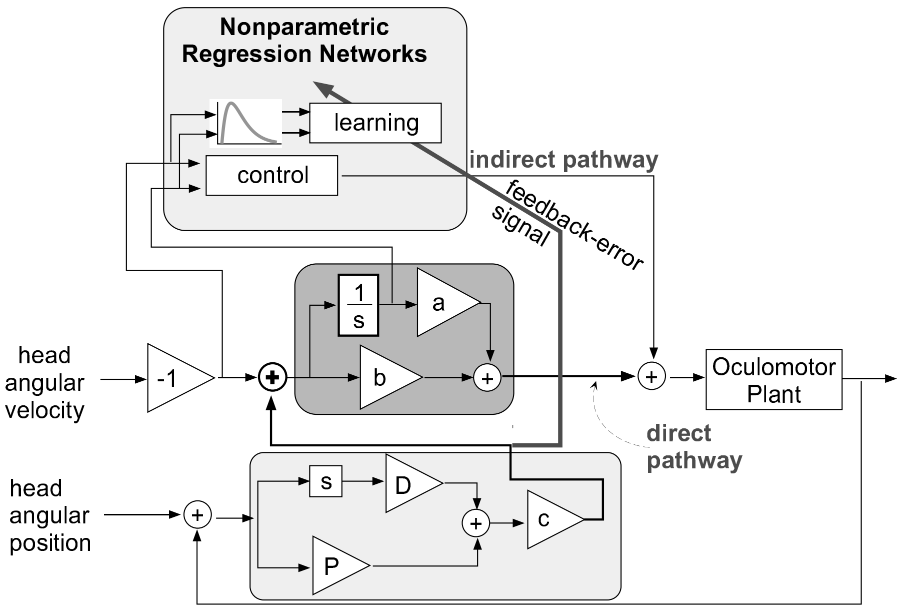

Figure 6. A control diagram of the VOR-OKR learning system. The lowest box corresponds to the OKR-like negative feedback circuit, the middle box corresponds to the linear feedforward model and the top box corresponds to the continuously learned non-linear feedforward circuitry

motor errors that, subsequently, can be used to train a neural network by supervised learning. From the viewpoint of adaptive control, FEL is a model-reference adaptive controller. The controller is assumed to be equipped a priori with a stabilizing linear feedback controller whose performance, however, is not satisfactory due to nonlinearities in the plant and delays in the feedback signals. Therefore, the feedback motor command of this controller is employed as an error signal to train a neural network controller. Given that the neural network receives the correct inputs, i.e., usually current and desired state of the plant, it can acquire a nonlinear control policy that includes both an inverse dynamics model of the plant and a nonlinear feedback controller. (Kawato, 1990) proved the convergence of this adaptive control scheme and advocated its architecture as an abstract model of learning in the cerebellum.

In order to employ LWPR for learning under the FEL scheme, we require the presence of a target output y (See Table II). In motor learning, target values for motor commands rarely exist since errors are usually generated in sensory space, not in motor command space. The FEL strategy can be interpreted as generating a pseudo target for

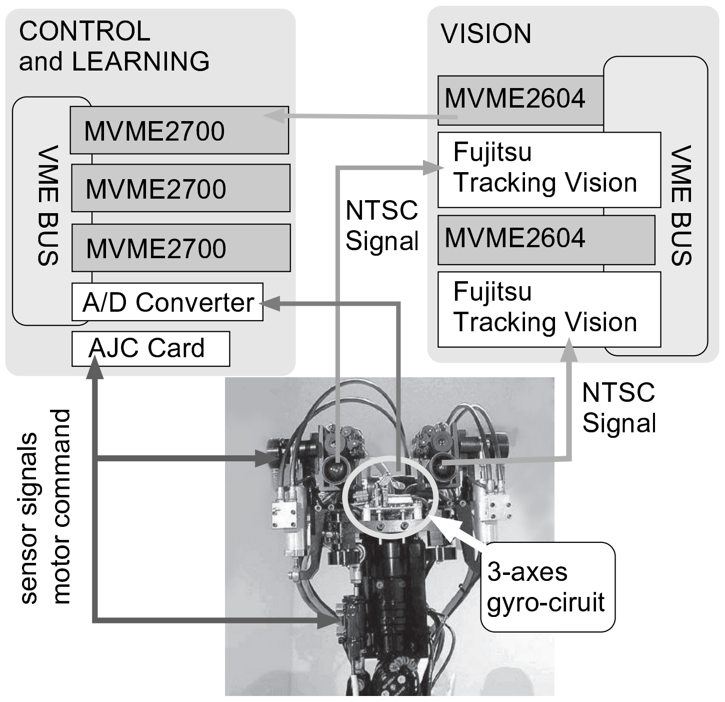

Figure 7. The vision head subsystem of our humanoid experimental setup

the motor command $y(t-1) = \hat{y}(t-1) + \tau_{fb}(t)$, where τ_{fb} denotes the feedback error signal and $\hat{y}$ is the predicted output. Using these principles and by employing the LWPR algorithm for on-line learning, we demonstrate that our humanoid robot is able to acquire high performance visual stabilization reflexes after about 40 seconds of learning despite significant nonlinearities and processing delays in the system.

3.3.1. *Experimental setup*

Fig. 7 depicts our experimental system. Each DOF of the robot is actuated hydraulically out of a torque control loop. Each eye of the robot's oculomotor system consists of two cameras, a wide angle (100 degrees view-angle horizontally) color camera for peripheral vision, and second camera for foveal vision, providing a narrow-view (24 degrees view-angle horizontally) color image. This setup mimics the foveated retinal structure of primates, and it is also essential for an artificial vision system in order to obtain high resolution vision of objects of interest while still being able to perceive events in the peripheral environment. Each eye has two independent degrees of freedom, a pan and a tilt motion.

The controllers are implemented in two subsystems, a learning control subsystem and a vision subsystem, each operated out of a VME rack using the real-time operating system VxWorks. Three CPU boards (Motorola MVME2700) are used for the learning control subsystem, and two CPU boards (Motorola MVME2604) are provided for the vision subsystem. In the learning control subsystem, CPU boards are used, respectively, for: i) low level motor control of the eyes and other joints of our robot (compute torque mode), ii) visuomotor learning, and iii) receiving data from the vision subsystem. All communication between the CPU boards is carried out through the VME shared memory communication which, since it is implemented in hardware, is very fast. In the vision subsystem, each CPU board controls one Fujitsu tracking vision board in order to calculate retinal slip and retinal slip velocity information of each eye. NTSC video signals from the binocular cameras are synchronized to ensure simultenous processing of both eyes' vision data. Vision data are sent via a serial port (115200 bps) to the learning control subsystem. For the experimental demonstrations of this paper, only one peripheral camera is used for VOR-OKR in its horizontal (pan) degree-of-freedom. Multiple degrees of freedom per camera, and multiple eyes just require a duplication of our control/learning circuits. If the image on a peripheral camera is stabilized, the image on the mechanically-coupled foveal vision is also stabilized. In order to mimic the semicircular canal of biological systems, we attached a three-axis gyro-sensor circuit to the head. From the sensors of this circuit, the head angular velocity signal is acquired through a 12 bit A/D board. The oculomotor and head control loop runs at 480 Hz, while the vision control loop runs at 30 Hz.

We use both visual-tracking and optical flow calculation in order to acquire the retinal slip and the retinal slip velocity, respectively. Spatial averaging of multiple optical flow detectors were used to reduce the noise. To maintain a 30 Hz vision processing loop rate, pixels were sampled only every three dots. Due to this sampling, the effective angular resolution around the center of the image was about 0.03 rad.

3.3.2. *Experimental results of online gaze stabilzation*

There are three sources of nonlinearities both in biological and artificial oculomotor systems: i) muscle nonlinearities or nonlinearities added by the actuators and the usually heavy cable attached to the cameras, ii) perceptual distortion due to foveal vision, and iii) off-axis effects. Off-axis effects result from the non-coinciding axes of rotation of eye-balls and the head and require a nonlinear adjustment of the feedforward controller as a function of focal length, eye, and head position. Note that

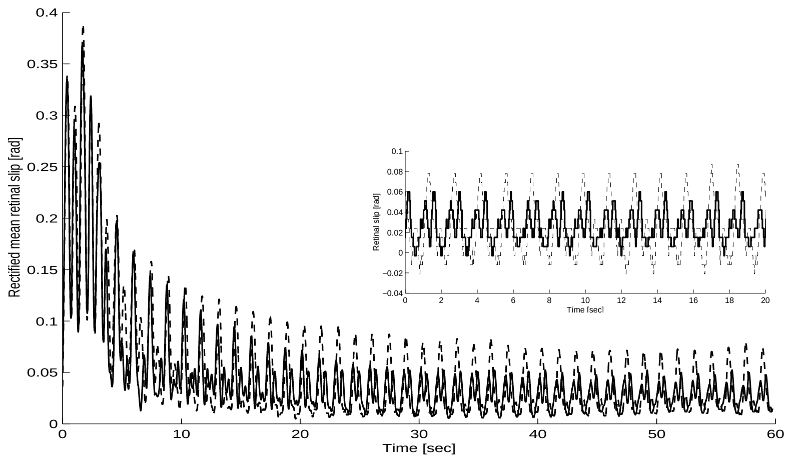

Figure 8. Time course of the mean retinal slip: the dashed line corresponds to linear learning result and the solid line corresponds to non-linear learning with LWPR; (inset) retinal slip during the last part of learning

this off-axis effect is the most significant nonlinearity in our oculomotor system.

In the learning experiment, we will compare the learning performance of our LWPR non-linear online learning algorithm against Recursive Least Squares (RLS) regression, a linear learning system (Ljung & Soderstrom, 1986). For this purpose, a large board with texture appropriate for vision processing was placed in front of the robot. The distance between a camera and the board was around 50 cm, i.e., a distance that emphasized the off-axis nonlinearities. In this experiment, the head was moved horizontally according to a sinusoidal signal with frequency 0.8 Hz and amplitude 0.25 rad.

Fig. 8 shows the time course of the rectified retinal slip, smoothed with a moving average over a one second time window. The dashed line corresponds to RLS learning, while the solid line presents the learning performance of LWPR. The benefits for a nonlinear learning system are clearly demonstrated in this plot: both learning curves show rapid improvement over time, but the final retinal slip out of LWPR is almost half of the remaining slip from linear learning. Fig. 8(inset) shows the time course of the raw retinal slip signals at the end of learning. Since, as mentioned in Section 3.3.1, the effective angular resolution around the center of the image was 0.03 rad, the learning results shown in Fig. 8 are satisfactory as their amplitude is also about 0.03 rad, i.e., the best result achievable with this visual sensing resolution.

The nonlinear component generated by the off-axis effect is around 0.05 rad when the head is rotated 0.25 rad and the visual stimulus is at

0.5 m distance (based on analytical computations from the geometry of off-axis vision head system). This difference is consistent with the average difference between the results obtained by RLS and LWPR, suggesting that LWPR was able to learn the nonlinear component generated by the off-axis effect.

4. Conclusions

This paper introduced locally weighted projection regression (LWPR), a statistical learning algorithm, for applications of real-time learning in highly complex humanoid robots. The *O(n)* update complexity of LWPR in the number of inputs n, together with its statistically sound dimensionality reduction and learning rules allowed a reliable and successful real-time implementation of various learning problems in humanoid robotics, including inverse dynamics learning, inverse kinematics learning, and oculomotor learning. These results demark one of the first times that complex internal models for model-based control could be learned autonomously in real-time on sophisticated robotic devices. We hope that algorithms like LWPR will allows us in the near future to equip robots with massive on-line learning abilities such that we come one step closer to realizing the dream of completely autonomous humanoid robots.

References

An,C.H., Atkeson,C. & Hollerbach, J. (1988) *Model Based Control of a Robot Manipulator* MIT Press.

Atkeson, C., Moore, A. & Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review*, *11*, 76–113.

Bishop, C. (1995) *Neural Networks for Pattern Recognition.* Oxford Press.

Bullock, D., Grossberg, S., and Guenther F. H (1993). A self-organizing neural model of motor equivalent reaching and tool use by a multijoint arm. *Journal of Cognitive Neuroscience*, 5(4):408–435.

Cruse, H. and Brüwer, M. The human arm as a redundant manipulator: The control of path and joint angles. *Biological Cybernetics*, 57:137–144, 1987.

Frank, I. E. & Friedman, J. H. (1993). A statistical view of some chemometric regression tools. *Technometrics*, *35*, 109–135.

Jordan, I.M. & Ruemelhart (1992) Supervised learning with a distal teacher, *Cognitive Science*, pp. 307-354.

Kawato, M. (1990). Feedback-error-learning neural network for supervised motor learning. In R. Eckmiller, editor, *Advanced Neural Computers*, pages 365–372. North-Holland: Elsevier.

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

Ljung, L. & Soderstrom, T. (1986) *Theory and practice of recursive identification.* Cambridge MIT Press.

Sanger, T. D. (1989). Optimal unsupervised learning in a single layer liner feedforward neural network. *Neural Networks*, *2*, 459–473.

Saunders,C., Stitson, M.O., Weston,J., Bottou,L., Schoelkopf,B., Smola,A. (1998) Support Vector Machine - Reference Manual. *TR CSD-TR-98-03*, Dept. of Computer Science, Royal Holloway, Univ. of London.

Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, *3*, 233–242.

Schaal, S. & Atkeson, C. G. (1998). Constructive incremental learning from only local information. *Neural Comp*, *10*, 2047–2084.

Schaal, S., Atkeson, C. G. & Vijayakumar, S. (2000). Real-Time Robot Learning with Locally Weighted Statistical Learning. *Proc. International Conference on Robotics and Automation ICRA2000*,288–293.

Schaal, S., Vijayakumar, S. & Atkeson, C. G. (1998). Local dimensionality reduction. *Proc. Neural Information Processing Systems 10*,633–639.

Shibata, T. & Schaal, S. (2001). Biomimetic gaze stabilization based on feedback-error-learning with nonparametric regression networks. *Neural Networks.* Vol. 14. No. 2., pp. 201–216.

Slotine, J. E. & Li, W. (1991). *Applied Nonlinear Control.* Prentice Hall.

Tevatia, G. and Schaal, S. Inverse kinematics for humanoid robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA2000)*, San Francisco, CA, Apr. 2000.

Vapnik, V. (1995) *The Nature of Statistical Learning Theory.* Springer, New York.

Vijayakumar, S. & Schaal,S. (2000). Locally Weighted Projection Regression : An *O(n)* algorithm for incremental real time learning in high dimensional space. Proc. International Conference on Machine Learning ICML2000, pp.1079-1086.

Wold, H. (1975). Soft modeling by latent variables: the nonlinear iterative partial least squares approach. *Perspectives in Probability and Statistics.*