D'Souza, A., Vijayakumar, S., Schaal, S., (submitted). Learning inverse kinematics, International Conference on Intelligence in Robotics and Autonomous Systems (IROS 2001). Maui, Hawaii, Oct 2001.

Learning Inverse Kinematics

Aaron D'Souza
adsouza@usc.edu
http://www-slab.usc.edu/adsouza

Sethu Vijayakumar
sethu@usc.edu
http://www-slab.usc.edu/sethu

Stefan Schaal
sschaal@usc.edu
http://www-slab.usc.edu/sschaal

Abstract

Real-time control of the endeffector of a humanoid robot in external coordinates requires computationally efficient solutions of the inverse kinematics problem. In this context, this paper investigates learning of inverse kinematics for resolved motion rate control (RMRC) employing an optimization criterion to resolve kinematic redundancies. Our learning approach is based on the key observations that learning an inverse of a non uniquely invertible function can be accomplished by augmenting the input representation to the inverse model and by using a spatially localized learning approach. We apply this strategy to inverse kinematics learning and demonstrate how a recently developed statistical learning algorithm, Locally Weighted Projection Regression, allows efficient learning of inverse kinematic mappings in an incremental fashion even when input spaces become rather high dimensional. The resulting performance of the inverse kinematics is comparable to Liegeois ([9]) analytical pseudo-inverse with optimization. Our results are illustrated with a 30 degree-of-freedom humanoid robot.

1 Introduction

Most movement tasks are defined in coordinate systems that are different from the actuator space in which motor commands must be issued. Hence, movement planning and learning in task space ([1], [2], [11]) require appropriate coordinate transformations from task to actuator space before motor commands can be computed. We will focus on the case where movement plans are given as external kinematic trajectories — as opposed to complete task-level control laws — on systems with many redundant degrees-of-freedom (DOFs), as typical in humanoid robotics (Figure 1). The transformation from kinematic plans in external coordinates to internal coordinates is the classic inverse kinematics problem, a problem that arises from the fact that inverse transformations are often ill-posed. If we define the intrinsic coordinates of a manipulator as the n-dimensional vector of joint angles $\boldsymbol{\theta} \in \Re^n$, and the position and orientation of the manipulator's end effector as the m-dimensional vector $\mathbf{x} \in \Re^m$, the forward kinematic function can generally be written as:

Figure 1: Humanoid robot in our laboratory.

$$\mathbf{x} = f(\boldsymbol{\theta}) \tag{1}$$

while what we need is the inverse relationship:

$$\boldsymbol{\theta} = f^{-1}(\mathbf{x}) \tag{2}$$

For redundant manipulators, i.e., $n > m$, solutions to Equation (2) are usually non-unique (excluding the degenerate case where no solutions exist at all), and even for $n = m$ multiple solutions can exist (e.g., [6]). Therefore, inverse kinematics algorithms need to address how to determine a particular solution to (2) in face of multiple solutions. Heuristic methods have been suggested, such as freezing DOFs to eliminate redundancy. However, redundant DOFs are not necessarily disadvantageous as they can be used to optimize additional constraints, e.g., manipulability, force constraints, etc. Thus it is useful to solve the inverse problem (2) by imposing an optimization criterion:

$$g = g(\boldsymbol{\theta}) \tag{3}$$

where g is usually a convex function that has a unique global optimum.

There are two generic approaches to solving inverse kinematics problems with optimization criteria ([4]). Global methods find an optimal path of $\boldsymbol{\theta}$ with respect to the entire trajectory, usually in computationally expensive off-line calculations. In contrast, local methods,

Computer Science and Neuroscience, HNB-103, Univ. of Southern California, Los Angeles, CA 90089-2520
Kawato Dynamic Brain Project (ERATO/JST), 2-2 Hikaridai, Seika-cho, Soraku-gun, 619-02 Kyoto, Japan

which are feasible in real time, only compute an optimal change in $\boldsymbol{\theta}$, $\Delta\boldsymbol{\theta}$, for a small change in $\mathbf{x}$, $\Delta\mathbf{x}$ and then integrate $\Delta\boldsymbol{\theta}$ to generate the entire joint space path. Resolved Motion Rate Control (RMRC) ([15]) is one such local method. It uses the Jacobian $\mathbf{J}$ of the forward kinematics to describe a change of the endeffector's position as

$$\dot{\mathbf{x}} = \mathbf{J}\left(\boldsymbol{\theta}\right)\dot{\boldsymbol{\theta}} \tag{4}$$

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. For a redundant manipulator n is greater than m, e.g., $n = 26$ and $m = 3$ for our humanoid reaching for an object (neglecting the 4 degrees-of-freedom for the eyes), which necessitates the use of additional constraints, e.g., the optimization criterion g (3), to obtain a unique inverse. For instance, Liegeois ([9]) suggested a pseudo-inverse solution by minimizing g in the null space of $\mathbf{J}$:

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^{\#}\dot{\mathbf{x}} - \alpha\left(\mathbf{I} - \mathbf{J}^{\#}\mathbf{J}\right)\frac{\partial g}{\partial \boldsymbol{\theta}} \tag{5}$$

which, for certain cost functions g, is a special case of Baillieul's extended Jacobian method ([3], [13]) which has the general form

$$\dot{\boldsymbol{\theta}} = \mathbf{J}_E^{-1}\dot{\mathbf{x}}_{aug} \tag{6}$$

where $\dot{\mathbf{x}}_{aug}$ is an augmented input vector ([3]).

The goal of our research is to accomplish solutions to RMRC inverse kinematics with statistical learning approaches that approximate (5) and (6). In the next sections, we will first discuss the problems of inverse kinematics learning and how they can be overcome. Afterwards, we briefly describe a learning algorithm that we developed that is ideally suited for the inverse kinematics learning. In the last section will provide evaluations of inverse kinematics learning algorithm with a humanoid robot.

2 Learning Inverse Kinematics

Learning of inverse kinematics is useful when the kinematic model of a robot is not accurately available, 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. Instead of re-calibrating the entire system 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 will not demand impossible postures as can result from an ill-conditioned matrix inversion.

The major obstacle in learning inverse kinematics lies in the problem that the inverse kinematics of a redundant kinematic chain has infinitely many solutions. Thus, the learning algorithm has to acquire a particular inverse, and moreover, has to make sure that the inverse is actually a valid solution. This latter issue was characterized in Jordan and Rumelhart ([8]) as the problem of non-convex mappings. In the context of equation (4), the forward kinematics of a redundant system maps multiple $\dot{\boldsymbol{\theta}}_i$ to the same $\dot{\mathbf{x}}$. When learning an inverse mapping $\dot{\mathbf{x}} \rightarrow \dot{\boldsymbol{\theta}}$, learning algorithms average over all the solutions $\dot{\boldsymbol{\theta}}_i$, assuming that different $\dot{\boldsymbol{\theta}}_i$ for the same $\dot{\mathbf{x}}$ are due to noise. Thus, for $\dot{\mathbf{x}} \rightarrow \dot{\boldsymbol{\theta}}$ to be a valid inverse, it is required that all $\dot{\boldsymbol{\theta}}_i$ encountered during training form a convex set-otherwise the average $\bar{\dot{\boldsymbol{\theta}}}_i$ could become an invalid solution to the inverse problem. Unfortunately, as shown in Jordan and Rumelhart ([8]), inverse kinematics has the non-convexity property and therefore, does not permit direct learning of the inverse mapping.

As noted by Bullock et al. ([5]), it is possible to transform the non-convex problem of inverse kinematics learning into a convex problem by spatially localizing the learning task: within the vicinity of a particular $\boldsymbol{\theta}$, inverse kinematics is actually convex. This can be proven easily by averaging equation (4) for multiple $\dot{\boldsymbol{\theta}}_i$ that map to the same $\dot{\mathbf{x}}$:

$$\langle\dot{\mathbf{x}}\rangle = \left\langle \mathbf{J}\left(\boldsymbol{\theta}\right)\dot{\boldsymbol{\theta}}_i \right\rangle_i \Rightarrow \dot{\mathbf{x}} = \mathbf{J}\left(\boldsymbol{\theta}\right)\left\langle\dot{\boldsymbol{\theta}}_i\right\rangle = \mathbf{J}\left(\boldsymbol{\theta}\right)\bar{\dot{\boldsymbol{\theta}}}_i \tag{7}$$

Equation (7) simply demonstrates that a local average over $\dot{\boldsymbol{\theta}}_i$ will still result in a valid solution to the inverse kinematics problem, i.e., $\bar{\dot{\boldsymbol{\theta}}}_i$ is guaranteed to map to the correct $\dot{\mathbf{x}}$. Importantly, local averaging has to be done within the vicinity of $\boldsymbol{\theta}$. Thus, inverse kinematics learning for a redundant system can theoretically be accomplished properly by learning a mapping $(\dot{\mathbf{x}}, \boldsymbol{\theta}) \rightarrow \dot{\boldsymbol{\theta}}$ if a spatially localized learning algorithm is employed.

3 Locally Weighted Projection Regression

Locally weighted projection regression (LWPR) ([14]) is a supervised learning algorithm that is well suited for learning the inverses kinematics mapping $(\dot{\mathbf{x}}, \boldsymbol{\theta}) \rightarrow \dot{\boldsymbol{\theta}}$. The key concept of LWPR is to approximate nonlinear functions by means of piecewise linear models. The region of validity, called a *receptive field*, of each linear model is computed from a Gaussian function:

$$w_k = \exp\left(-\frac{1}{2}\left(\mathbf{x}-\mathbf{c}_k\right)^T \mathbf{D}_k \left(\mathbf{x}-\mathbf{c}_k\right)\right) \tag{8}$$

where $\mathbf{c}_k$ is the center of the k^{th} linear model, and $\mathbf{D}_k$ corresponds to a distance metric that determines the size and shape of region of validity of the linear model. Given an input vector $\mathbf{x}$, each linear model calculates a prediction y_k. The total output of the network is the weighted mean of all linear models:

$$\hat{y} = \frac{\sum_{k=1}^{K} w_k y_k}{\sum_{k=1}^{K} w_k} \tag{9}$$

In order to avoid numerical problems due to matrix inversions and to minimize the computational complexity, the linear models in each receptive field are not computed by linear regression but rather by applying a sequence of one-dimensional regressions along selected projections $\mathbf{u}_r$ in input space (note that we will drop the index k from now on unless it is necessary to distinguish explicitly between different linear models):

$$\begin{aligned} &\text{Initialize: } y = \beta_0, \mathbf{z} = \mathbf{x} - \mathbf{x}_0 \\ &\quad \text{For } i = 1:r \\ &\qquad s = \mathbf{u}_i^T \mathbf{z} \\ &\qquad y = y + \beta_i s \\ &\qquad \mathbf{z} \leftarrow \mathbf{z} - \mathbf{p}_i s \end{aligned} \tag{10}$$

The projections $\mathbf{u}_i$, the univariate regression parameters β_i, the mean $\mathbf{x}_0$, and the number of projections r are determined by the learning algorithm. Additionally, the learning algorithm also finds a projection vector $\mathbf{p}_i$that reduces the input space for the next univariate regression. As will be explained below, this step allows finding more efficient projections $\mathbf{u}_i$ at subsequent univariate regression steps.

In order to determine the open parameters in Equation (10), the technique of partial least squares (PLS) regression can be adapted from the statistics literature ([16]). The important ingredient of PLS is to choose projections according to the correlation of the input data with the output data. The following algorithm, Locally Weighted Projection Regression (LWPR), uses an incremental locally weighted version of PLS to determine the linear model parameters:

Given: A training point $(\mathbf{x}, y)$

Update means of inputs and outputs:

$$\mathbf{x}_0^{n+1} = \frac{\lambda W^n \mathbf{x}_0^n + w\mathbf{x}}{W^{n+1}}$$

$$\beta_0^{n+1} = \frac{\lambda W^n \beta_0^{n+1} + wy}{W^{n+1}}$$

where $W^{n+1} = \lambda W^n + w$

Update the local model:

Initialize: $\mathbf{z} = \mathbf{x}, res = y - \beta_0^{n+1}$

For $i = 1:r$,

$$\begin{aligned} &\text{a)} \quad \mathbf{u}_i^{n+1} = \lambda \mathbf{u}_i^n + w\mathbf{z} \cdot res \\ &\text{b)} \quad s = \mathbf{z}^T \mathbf{u}_i^{n+1} \\ &\text{c)} \quad SS_i^{n+1} = \lambda SS_i^n + ws^2 \\ &\text{d)} \quad SR_i^{n+1} = \lambda SR_i^n + ws \cdot res \\ &\text{e)} \quad SZ_i^{n+1} = \lambda SZ_i^n + w\mathbf{z}s \\ &\text{f)} \quad \beta_i^{n+1} = SR_i^{n+1}/SS_i^{n+1} \\ &\text{g)} \quad \mathbf{p}_i^{n+1} = SZ_i^{n+1}/SS_i^{n+1} \\ &\text{h)} \quad \mathbf{z} \leftarrow \mathbf{z} - s\mathbf{p}_i^{n+1} \\ &\text{i)} \quad res \leftarrow res - s\beta_i^{n+1} \\ &\text{j)} \quad MSE_i^{n+1} = \lambda MSE_i^n + w \cdot res^2 \end{aligned} \tag{11}$$

In the above equations, $\lambda \in [0, 1]$ is a forgetting factor that determines how much older data in the regression parameters will be forgotten, similar as in recursive system identification techniques ([10]). The variables SS, SR, and SZ are memory terms that enable us to do the univariate regression in step f) in a recursive least squares fashion, i.e., a fast Newton-like method. Step g) regresses the projection $\mathbf{p}_i$ from the current projected data s and the current input data $\mathbf{z}$. This step guarantees that the next projection of the input data for the next univariate regression will result in a $\mathbf{u}_{i+1}$ that is orthogonal to $\mathbf{u}_i$. Thus, for $r = n$, the entire input space would be spanned by the projections u_i and the regression results would be identical to that of a traditional linear regression. Step j) will be discussed below.

There are several important properties in PLS. First, if all the input variables are statistically independent, PLS will find the optimal projection direction $\mathbf{u}_i$ in a *single* iteration — the optimal projection direction corresponds to the gradient of the assumed locally linear function to be approximated. Second, choosing the projection direction from correlating the input and the out-

put data in Step a) automatically excludes irrelevant input dimensions, i.e., inputs that do not contribute to the output. And third, there is no danger of numerical problems in PLS due to redundant input dimensions as the univariate regressions will never be singular.

The above update rule can be embedded in an incremental learning system that automatically allocates new locally linear models as needed ([12]):

Initialize the LWPR with no receptive field (RF);
For every new training sample $(\mathbf{x}, y)$:
  For k=1 to #RF:
    calculate the activation from (8)
    update according to
  end;
  If no linear model is activated more than w_{gen};
    create a new RF with $r = 2$, $\mathbf{c} = \mathbf{x}$, $\mathbf{D} = \mathbf{D}_{def}$
  end;
end;

In this pseudo-code algorithm, 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 (8). 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 j) in (11). If the MSE at the next projection does not decrease more than a certain percentage of the previous MSE, i.e.,

$$\frac{MSE_{i+1}}{MSE_i} > \phi, \text{ where } \phi \in [0, 1] \tag{12}$$

the algorithm will stop adding new projections to the local model.

It is even possible to learn the correct parameters for the distance metric $\mathbf{D}$ in each local model. The algorithm for this update was derived in ([12]) for normal locally linear regression based on an incremental cross validation technique. This algorithm is directly applicable to LWPR, and is strongly simplified, as it only needs to be done in the context of univariate regressions. Due to space limitations, we will not provide the update rules in this paper as they can be derived from ([12]).

3.1 Applying LWPR to Inverse Kinematics Learning

By using spatially localized receptive fields, LWPR has all the prerequisites to learn inverse kinematics. The inputs to the learning system are $\mathbf{z} = (\dot{\mathbf{x}}, \boldsymbol{\theta})$, and the outputs are $\mathbf{y} = \dot{\boldsymbol{\theta}}$. $\dot{\mathbf{x}}$ can be in Cartesian coordinates if a calibrated 3D tracking system for the endeffector exists, but it could also be in uncalibrated image coordinates of two or more cameras — since LWPR can handle redundant inputs there is not restriction on the dimensionality of $\dot{\mathbf{x}}$. For our humanoid robot, the dimensionality of the input $\mathbf{z}$ is 29 (26 degrees-of-freedom neglecting the 4 degrees-of-freedom for the eyes, plus 3 Cartesian inputs), while the dimensionality of the output $\mathbf{y}$ is 26. By moving the robot while reading values for $\mathbf{z}$ and $\mathbf{y}$ from the sensors, training data is generated that can be added incrementally to the learning system — this processed often termed self-supervised learning.

3.1.1 Creating a cost function

In the introduction we alluded to the notion that 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 ([7]). Additionally, each degree of freedom is given a weight, which determines the extent of its contribution to the cost function. Hence the total cost function for training LWPR can be written as follows:

$$Q = \frac{1}{2}\left(\dot{\boldsymbol{\theta}} - \hat{\dot{\boldsymbol{\theta}}}\right)^T \left(\dot{\boldsymbol{\theta}} - \hat{\dot{\boldsymbol{\theta}}}\right) + \frac{1}{2}\alpha \left(\hat{\dot{\boldsymbol{\theta}}} - \frac{\Delta\boldsymbol{\theta}}{\Delta t}\right)^T \mathbf{W} \left(\hat{\dot{\boldsymbol{\theta}}} - \frac{\Delta\boldsymbol{\theta}}{\Delta t}\right) \tag{13}$$

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:

$$\dot{\boldsymbol{\theta}}_{target} = \dot{\boldsymbol{\theta}} - \alpha\mathbf{W}\left(\hat{\dot{\boldsymbol{\theta}}} - \Delta\boldsymbol{\theta}\right) \tag{14}$$

These targets are commposed of the self-supervised target $\dot{\boldsymbol{\theta}}$, slightly modified by a component to enforce the null space optimization criterion. Note that the null space optimization will sacrifice some performance in

tracking accuracy to accomplish the desired null space motion towards the optimal posture $\boldsymbol{\theta}_{opt}$.

3.1.2 Learning on the task

Our emphasis in this paper is towards learning the inverse kinematics "on the fly", i.e., while attempting to perform the task itself. The problem however, is that initially, LWPR has very little (or no) data upon which to base its regression. We still however, require a command to be sent to create an output motion of the robot. As an exploration strategy, we initially bias the output of LWPR with a term that creates a motion towards $\boldsymbol{\theta}_{opt}$:

$$\tilde{\dot{\boldsymbol{\theta}}} = \hat{\dot{\boldsymbol{\theta}}} + \frac{1}{n_r}\Delta\boldsymbol{\theta} \tag{15}$$

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.

LWPR learns extremely quickly from even very sparse data. This can result in jerky and inaccurate movement during the initial stages of exploration and learning. In order to ensure smoother trajectories during the learning process, we initialize the SS variable of each local model in Equation (11)c with a value of 10^{10}. By inspecting (11)c, it can be seen that this bias causes the regression coefficients to have very small values initially, which results in very slow movement of the robot. While making these slow movements however, data is continuously added to the LWPR algorithm, and eventually the initial bias is overcome due to the forgetting factor λ, which effectively bases the statistics computed in Equation (11) on the last $(1-\lambda)^{-1}$ data points. As the system acquires more data, it gradually increases its "trust" in its own approximation to the inverse kinematics, eventually allowing the full strength of the regression to command the output.

3.1.3 Localization space vs. regression space

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 $\dot{\mathbf{x}}$. 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 Equation (8) 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.

4 Experimental Evaluations

In the following experiments, we use a simple Cartesian controller to generate the desired accelerations in Cartesian space for tracking a target $\mathbf{x}_t$. Given the position, velocity and acceleration information of the target, the control law is:

$$\tilde{\ddot{\mathbf{x}}} = \ddot{\mathbf{x}}_t + k_v\left(\dot{\mathbf{x}}_t - \dot{\mathbf{x}}\right) + k_p\left(\mathbf{x}_t - \mathbf{x}\right) \tag{16}$$

where $k_p = 1250$, and $k_v = 70$. This desired acceleration is numerically integrated to obtain a desired Cartesian velocity:

$$\tilde{\dot{\mathbf{x}}}^{n+1} = \tilde{\dot{\mathbf{x}}}^{n} + \tilde{\ddot{\mathbf{x}}}\Delta t \tag{17}$$

where $\Delta t = 1/420$ in our experiments. It is this value of $\tilde{\dot{\mathbf{x}}}$ that we use as the Cartesian space input to our algorithm to generate $\dot{\boldsymbol{\theta}} \leftarrow f^{-1}(\tilde{\dot{\mathbf{x}}}, \boldsymbol{\theta})$, which is then integrated and differentiated to obtain $\ddot{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$ respectively, as inputs to the inverse dynamics controller. Training data, on the other hand, is created from Equation (14).

4.1 Experiments

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 (c.f. Equation (5)) that was available for our robot from previous work ([13]).

4.1.1 Learning from "motor babbling"

We first trained the system on data collected from "motor babbling". We created small sinuslidal motions of each degree of freedom about a randomly chosen mean in $\boldsymbol{\theta}$ space. Every few seconds, this mean is repositioned within the workspace. After training the system in this manner for approximately 10 minutes, we tested its performance on the figure-eight task. The trajectory followed by the system is shown in fig. 2 along with the solution obtained using an analytical pseudo-inverse solution. The tracking inaccuracies seen in the figure are not surprising, since given the high dimensionality of the joint space, the data obtained from motor babbling is sparse in the region required by the figure-eight task. Thus, LWPR's predictions are based on too few points to achieve high accuracy.

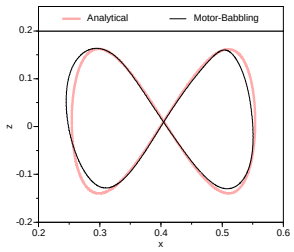


Figure 2: System performance after being trained on data collected from motor babbling.

4.1.2 Improving performance on the task

In order to demonstrate that more task-specific training data leads to better inverse kinematics learning, our second experiment trained the inverse kinematics on the task. 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. 3, after merely 1 minute of additional learning, the system performs as accurately as the analytical solution.

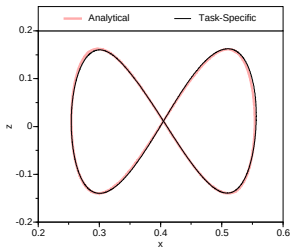


Figure 3: System performance after 1 minute of executing the figure-eight task, with learning on the task enabled.

4.1.3 Learning from scratch on the task

The final experiment endeavored to start with an untrained system, and learn the inverse kinematics from scratch, while performing the figure-eight task itself. Figure 4 shows the progression of the system's performance from the beginning of the task to about 3 minutes into the learning. One can see that 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 on the task, the performance approached that seen in Figure 3.

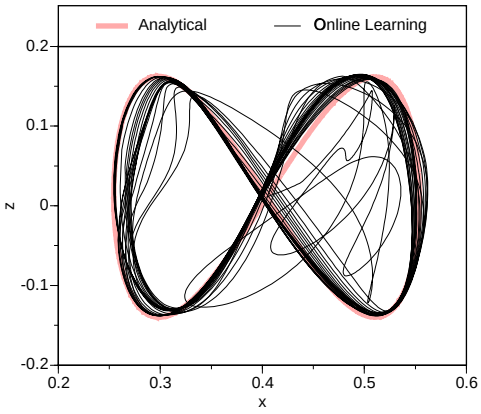
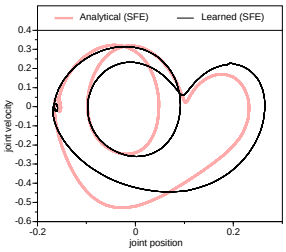


Figure 4: Trajectory followed in the first 3 minutes when learning the inverse kinematics from scratch while attempting to perform the figure-eight task.

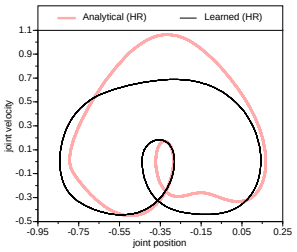
4.2 Consistency of the learned inverse kinematics

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 the accuracy of the tracked trajectory in operational space. Figures 5(a), 5(b), and 5(c) show phase plots of three degrees of freedom — shoulder flexion and extension (SFE), humeral rotation (HR), and elbow (EB) flexion and extension respectively — plotted over about 30 cycles of the figure-eight trajectory after learning had converged. The presence of a single loop for the phase plot over all cycles in each case shows that the inverse kinematics solution found by our algorithm is indeed consistent.

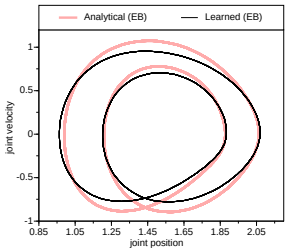
Comparing the analytical solution with LWPR's solution in the above figures, it is clear that we learn an inverse kinematics solution that is qualitatively similar to that obtained by an analytical pseudo-inverse. The quantitative discrepancies in the two wolutions are due to an imperfect approximation of the null space, which is a result of enforcing the null space optimization only



(a) Shoulder flexion and extension (SFE).



(b) Humeral rotation (HR).



(c) Elbow (EB) flexion and extension.

Figure 5: Phase plots

implicitly in the cost function (Equation (13)).

5 Discussion

This paper presented how inverse kinematics for redundant manipulators can be learned with modern statistical learning algorithms. The key element of our approach was to augment the input space to the learning system such that averaging over redundant solutions of the inverse mapping could be done safely without creating physically impossible results. Using a specific optimization criterion for training the learning system, performance comparable to Liegeois' analytical pseudo-inverse could be accomplished ([9]). We demonstrated the functionality of our learning methods on a full body humanoid robot learning to trace a figure-eight in Cartesian space after only a few minutes of training.

Despite these encouraging results, we need to address a variety of issues in future work. Most importantly, our suggested algorithm only finds an approximate solution to optimizing the null space motion of the robot due to a cost function that causes a slight amount of interference between task goals and null space optimization. We noted that under unfortunate training data distributions, this interference can cause a slight amount of unwanted movement in unconstrained endeffectors of our humanoid, e.g., the left hand moved while only the right hand was supposed to track a target. Additionally, it is not favorable to "hard code" the optimization criterion for the null space motion in the learning system, as is currently the case in our approach. Different tasks may favor different optimization criteria for for the resolution of redundancy, and the learning system should be flexible enough to accommodate this requirement. Our future work will address learning algorithms that learn the null space and range space of the local inverse kinematics mapping explicitly in order to allow for such flexibility.

6 Acknowledgements

This work was made possible by Award #9710312 of the National Science Foundation, the ERATO Kawato Dynamic Brain Project funded by the Japanese Science and Technology Cooperation, and the ATR Human Information Processing Research Laboratories.

References

[1] E. W. Aboaf, C. G. Atkeson, and D. J. Reinkensmeyer. Task-level robot learning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Philadelphia, PA, 1988.

[2] E. W. Aboaf, S. M. Drucker, and C. G. Atkeson. Task-level robot learning: Juggling a tennis ball more accurately. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Scottsdale, AZ, 1989.

[3] J. Baillieul. Kinematic programming alternatives for redundant manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 722–728, 1985.

[4] J. Baillieul and D. P. Martin. Resolution of kinematic redundancy. In *Proceedings of Symposia in Applied Mathematics*, volume 41, pages 49–89. American Mathematical Society, 1990.

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

[6] J. J. Craig. *Introduction to Robotics*. Addison-Wesley, Reading, MA, 1986.

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

[8] M. I. Jordan and Rumelhart. Supervised learning with a distal teacher. *Cognitive Science*, 16:307–354, 1992.

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

[10] L. Ljung and T. Söderström. *Theory and practice of recursive identification*. Cambridge MIT Press, 1986.

[11] E. Saltzman and S. J. A. Kelso. Skilled actions: A task-dynamic approach. *Psychological Review*, 94(1):84–106, 1987.

[12] S. Schaal and C. G. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10(8):2047–2084, 1998.

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

[14] S. Vijayakumar and S. Schaal. Locally weighted projection regression: An $O(n)$ algorithm for incremental real time learning in high dimensional spaces. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, Stanford, CA, 2000.

[15] D. E. Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems*, 10(2):47–53, 1969.

[16] H. Wold. Soft modeling by latent variables: The nonlinear iterative partial least squares approach. In J. Gani, editor, *Perspectives in Probability and Statistics, Papers in Honour of M. S. Bartlett*, pages 520–540. Academic Press, London, 1975.