A FAST AND EFFICIENT MODEL FOR LEARNING TO REACH

GANGHUA SUN & BRIAN SCASSELLATI

Computer Science Department, Yale University
New Haven, Connecticut 06511, United States
{*ganghua.sun, brian.scassellati*}*@yale.edu*

Abstract.
This paper proposes a self-supervised model which enables a humanoid robot to learn to reach to visual targets. Only 400 training samples are used to learn a forward kinematic model of the 6 degree-of-freedom (DOF) arm. The forward model is represented compactly with just 150 hidden neurons and enables high accuracy reaching in real-time. We provide an optimization process for the learning parameters and a careful analysis of reaching errors. An extension of the model is presented to address additional DOFs in the neck. The consistency of the model with physiological and psychological observations is elaborated.

Keywords: Reaching, Kinematics, Humanoid Robots.

1. Introduction

In robotics, early work on reaching focused primarily on inverse kinematics. Many of these solutions are based on the Resolved Motion Rate Control (RMRC) algorithm which requires the forward kinematics of the arm to be known for the computation of Jacobian matrices.[1] In contrast to high-precision robotic manipulators, the arm kinematics of a human changes over his/her lifespan and is difficult to describe as an analytical function. This need for humans to acquire their own kinematic models was discussed by Piaget in 1936 and has been attracting the attention of numerous physiologists and psychologists since then.[2]

What must be learned in order to achieve accurate reaching movements if the values of the arm parameters are not readily available to the learning system? One seemingly straightforward approach is to learn the inverse kinematic mapping from x_{target} to θ directly, where x_{target} denotes the task vector describing the target and θ denotes the joint vector describing the corresponding arm posture. This approach is problematic in the case of human arm movements and many humanoid robot arm movements because the dimension of the joint vector is larger than that of the task vector such that there can exist multiple values of θ which correspond to the same x_{target}. Choosing an appropriate θ from all possible values of θ can be difficult.[3]

Bullock et al. suggested that learning a directional mapping from $(\theta, \dot{x})$ to $\dot{\theta}$ can circumvent this decision problem.[4] After training, the directional mapping can be used to incrementally generate a reaching trajectory similar to RMRC. In addition to the directional mapping, a model of the forward kinematics of the arm is learned which makes reaching possible without the aid of visual feedback during the course of the movement. There is evidence that this "forward" model exists in humans.[5,6] The advantages of using both the forward model and the directional mapping for reaching have been demonstrated.[4] Arguments supporting the learning of both a forward model and a direction mapping have also been put forth by Shadmehr and Wise.[7,8]

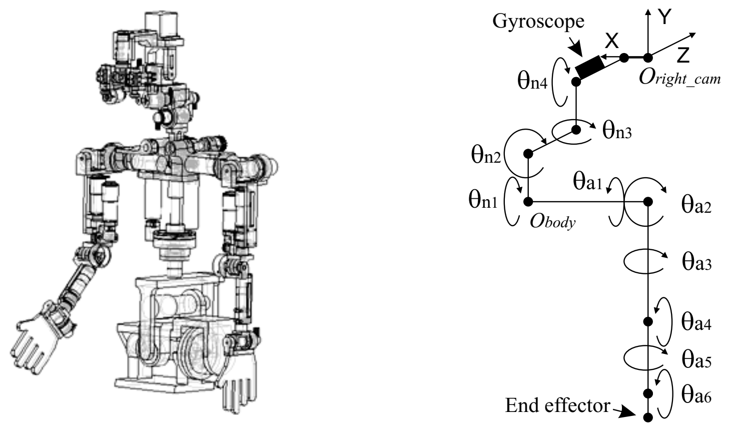
A variety of learning algorithms have been suggested for acquiring these functions, including Self Organizing Maps (SOM), Multiple Layer Perceptrons (MLP), Reinforcement Learning (RL) and more recently Locally Weighted Projection Regression (LWPR).[9,5,10,11] The choice of learning algorithm determines directly the compactness of the representations of the forward model and the directional map, as well as the number of training samples required to ensure good convergence. The larger the dimension of the joint vector θ, the greater the need for compact representations and reduced training set sizes. Bernstein identified this as the degree of freedom (DOF) problem[12] and it is known in statistical learning theory as the curse of dimensionality. While many have argued about which mappings should be learned and which learning algorithms were most appropriate for this task, the dimensionality problem remains. As an example of scale, Bullock et al. employed 15625 neurons to learn a forward model of a 3-DOF arm. The required number of neurons in this work increases exponentially with each additional DOF in the arm.[4]

In this paper, we present a model for learning arm movements on a humanoid robot that requires far fewer training examples and has a more compact representation than Bullock et al. A Radial Basis Function Network (RBFN) is used to learn a forward model of the arm. The directional mapping needed for incremental trajectory generation is extracted automatically from the learned forward model and does not need to be learned separately. This paper has two goals: (1) to introduce a practical model that can be used without major changes for learning to reach across different robotic platforms, and (2) to show that it allows for a biologically plausible implementation and is consistent with physiological and psychological observations.

This paper is organized as follows. Section 2 gives an introduction to the humanoid robot we use as our experimental platform. Section 3 describes in detail the model used for learning to reach with emphasis placed on the forward model learning and the trajectory generation. The performance of the model is demonstrated in Section 4. Section 5 extends the model to incorporate the additional DOFs in the robot's neck. Section 6 relates our model to several important observations of human reaching behavior. Section 7 summarizes the contributions of this paper.

2. Hardware and Software Platform

Nico is an upper-torso humanoid robot developed to match the body dimension of an average one-year-old infant. It has four DOFs in the neck and six DOFs in each arm. A gyroscope is mounted in the head on top of all neck joints. Fig. 1(A) shows a dimetric view of Nico. Fig. 1(B) shows the kinematic structure of the neck joints and the joints in the right arm viewed from behind. The vision system of Nico consists of four miniature CCD video-cameras divided into two sets, one for each eye. In each set, there is one long focal length camera for fovial vision and one short focal length camera for peripheral vision. Each joint of Nico is driven independently by a DC motor with an integrated high-resolution optical encoder. All motors and sensors on Nico are connected to a 16-node computation cluster running the QNX real-time operating system. Nodes are connected through a 100Mbit backbone switch and a number of direct point-to-point network links.



(A) (B)

Fig. 1. (A) A dimetric view of Nico, an upper-torso humanoid designed to match the size of an average one-year-old infant. (B) The kinematic structure of the neck and the right arm viewed from behind. The origins of the eye and the body-centered coordinate system are indicated with O_{right_cam} and O_{body} respectively.

A set of modular software components have been implemented on the robot, ranging from low-level device drivers to selected higher-level cognitive functions. During run-time, selected modules are allocated to processing nodes based on their computation requirements. Active modules can selectively communicate with one another through a common communication interface. Whether a data exchange takes place on the same node or across the network is transparent to an individual module.

3. A Model for Learning to Reach

Fig. 2 provides an overview of the model used in this work. The basis of the model is a forward model of the arm that is learned autonomously through motor babbling. It is important that the parameters for learning the forward model are set appropriately in order to keep the required number of training samples and the size of the representation space as low as possible. After the forward model is learned, it is used during a reaching movement to predict the position of the end effector from the current arm posture. The pseudo-inverse of the Jacobian corresponding to the current arm posture is computed directly from the forward model. Δx is a direction vector of small magnitude in the task space pointing from the predicted end effector position to the target position. It is transformed into a joint vector increment $\Delta\theta$ that is relayed to the arm motors. If the end effector can be perceived by the stereo vision system during the reaching movement, x_{pred} is replaced by x_{perc} and Δx becomes a direction vector pointing from the perceived end effector position to the target position. A description of the stereo vision component is given in Section 3.1. The training of the forward model and the algorithm for incremental trajectory generation are described in detail in Section 3.2 and 3.3 respectively.

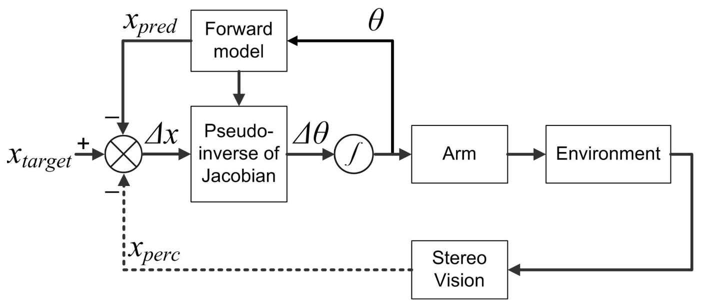


Fig. 2. Overview of a model for learning to reach. It is based on a learned forward model of the arm kinematics. During a reaching movement, it is used both for the prediction of the end effector position and the computation of the pseudo-inverse of the current Jacobian. The dashed line representing the data flow from the stereo vision component means visual feedback is optional during reaching.

3.1. *Stereo vision*

The stereo vision component shown in Fig. 2 retrieves video data from the two short focal length cameras as input. The two long focal length cameras prove to be impractical for the stereo vision needed for the reaching behavior because their common vision field has only a small overlap with the reachable space of the robot arm. The radial distortion coefficients K_1 and K_2 and other parameters are measured for each camera through the Camera Calibration Toolbox for Matlab developed by J.-Y. Bouguet.[13] The pixel value of position (x, y) in the corrected frame is filled with the pixel value of position (x', y') in the original frame through the following equation[14] —

$$\begin{cases} x' = x(1 + K_1 r^2 + K_2 r^4) \\ y' = y(1 + K_1 r^2 + K_2 r^4) \end{cases} \tag{1}$$

If values of x' and y' are not integers, they are substituted by their respective integer parts. During the startup of the vision subsystem, a lookup table is built for each camera consisting of the mappings from (x, y) to (x', y') for all possible (x, y). The pre-built lookup tables enable an efficient distortion correction in real time.

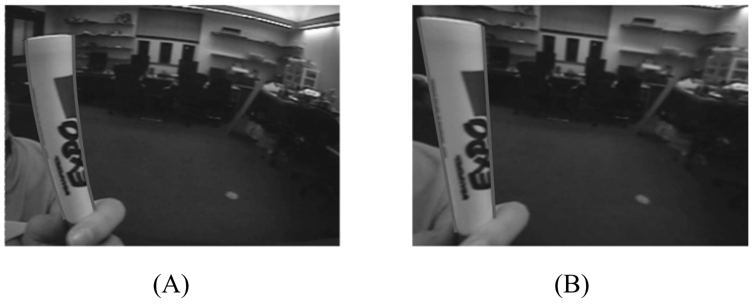


Fig. 3. Original image (A) and image (B) corrected for radial distortion through Eq. (1)

Currently, a ϕ19.05mm wooden ball is attached to the distal end of the robot arm as the end effector and a wooden ball of the same diameter is used as the reaching target. The projection of either the end effector or the target on each camera image plane is replaced by its centroid. For the experiments described in this paper, the two eye cameras are positioned parallel to each other. These simplifications make it straightforward to determine the position of both the end effector and the target in the eye-centered coordinate system.[15]

3.2. *Learning a forward model*

The forward kinematic function of the arm is defined as a mapping $f_{arm} : \theta \rightarrow x$. Each of Nico's arms has 6 degrees of freedom, so we have $\theta \in R^6$. At the current stage, we require our robot to touch a presented target without putting any restriction on the orientation of its end effector. This makes $x \in R^3$. A forward model of the arm is learned through motor babbling, during which the arm is repeatedly moved into random postures. If the end effector can be perceived by the stereo vision system at the end of an arm movement, the joint vector θ corresponding to the current arm posture and the perceived 3D position x of the end effector is recorded as a training sample. The arm stops moving only when a pre-specified number of training samples are gathered.

Learning a forward model of the arm is essentially approximating the function f_{arm} through training samples of the form $(\theta_i, x_i)_{i=1,2,\ldots,n}$, where n is the size of the training set. Each x_i in (θ_i, x_i) contains noise introduced by the stereo vision system. Neural networks such as MLP and RBFN are commonly used function approximation techniques.[16] The most important reason for our adoption of RBFN for learning the forward model is that the only weights to be learned are those connecting the hidden layer and the output layer. They can be determined directly by the linear least square method, which avoids the problem of local minima that MLPs often encounter. Arguments favoring RBFN from the perspective of biological plausibility will be given in Section 3.2.2.

3.2.1. *Optimization of learning parameters*

The Gaussian function is used as the basis function in the hidden layer of our RBFN. It can be expressed as $g(x) = exp(\|x - c\| \cdot 0.8326/spread)$, where x is the input vector and c is the center of the Gaussian. The parameter *spread* controls the extent of g's influence in its neighborhood. Since it has been shown that a RBFN with a set of basis functions that have a common form but different centers can approximate any continuous input-output mapping,[17] the same value is assigned to the *spread* of each Gaussian. We use the Orthogonal Least Squares (OLS) algorithm to determine the number and the centers of the Gaussian functions automatically from the training data.[18] The training stops when the root mean square error (*rmse*) of the network falls below a certain *margin*. The optimal values for both *spread* and *margin* must be determined before training. In addition, we are also interested in keeping the size of the training set as small as possible to save the time spent on gathering training samples. Computer simulations are used to optimize the three learning parameters (*spread*, *margin* and training set size) because it is very difficult to measure accurately the quality of the learned forward model through physical experiments.

Since the optimal *spread* of the Gaussian functions in the hidden layer depends primarily on the actual function to be approximated, the effect of noise is excluded in the simulations conducted to optimize *spread*, which means the value of x_i in the training sample (θ_i, x_i) is for the time being the true end-effector position corresponding to the joint vector θ_i. Fig. 4 shows four *rmse*-*spread* curves for four different training set sizes. The line style of each curve indicates which size it corresponds to. Each data point is generated by averaging the *rmse* values measured on 40 random training sets of the same size. It can be observed that for all of the four curves displayed, the *rmse* falls sharply at the beginning. As *spread* grows larger, the *rmse* appears to stay constant although it actually rebounds very gently after reaching a minimum point. The average of the optimal *spread* values of the four *rmse*-*spread* curves is about 130.

A large part of the noise in the perceived 3D position of the end effector is the stereo perception error, which is dependent on the resolution of the camera images. To find the optimal values of *margin* for different camera resolutions, simulations were conducted that use training samples whose task vector components contain stereo perception errors. *spread* was set to a constant value of 130. The results of the simulations to optimize *margin* are shown in Fig. 5, where the curves are plotted in different styles according to the same convention used in Fig. 4. As expected, the optimal *margin* for a 160x120 camera resolution is more than 2mm higher

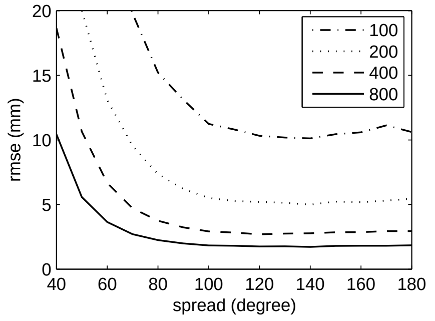


Fig. 4. *rmse-spread* curves for four different training set sizes. The averaged optimal *spread* is 130.

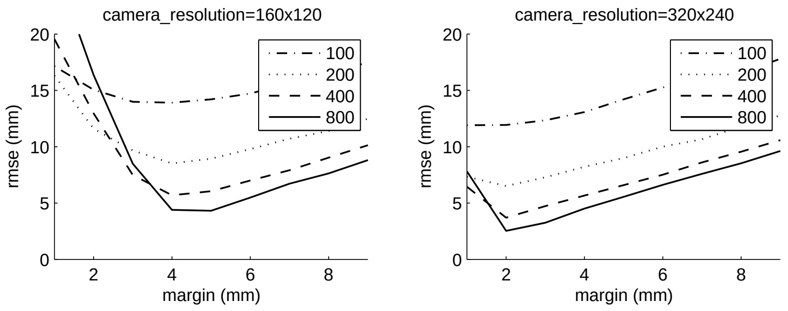


Fig. 5. *rmse-margin* curves for four different training set sizes and two camera resolutions. For each camera resolution, the optimal *margin*s for different training set sizes lie quite close to each other.

than that for a 320x240 camera resolution. Lower-resolution images result in higher stereo perception error which in turn requires a larger value for *margin* to prevent overfitting. Fig. 5 also shows that the more training samples we use, the higher the quality of the learned forward model. However, a 400-sample training set already leads to a small *rmse* very close to that achieved by a 800-sample training set. 400 samples require a very moderate amount of time to gather on a robotic platform, less than 30 minutes in our case.

In the real world, the stereo perception error only partially contributes to the noise in the perceived position of the end effector. The end effector of a robotic arm is a 3D structure. Its projection on the camera image plane is not a point, but a blob. Using the centroid of the blob as we do for the calculation of the end effector position introduces additional error since the two centroids on the right and left image planes do not represent the same point on the end effector. Putting a special

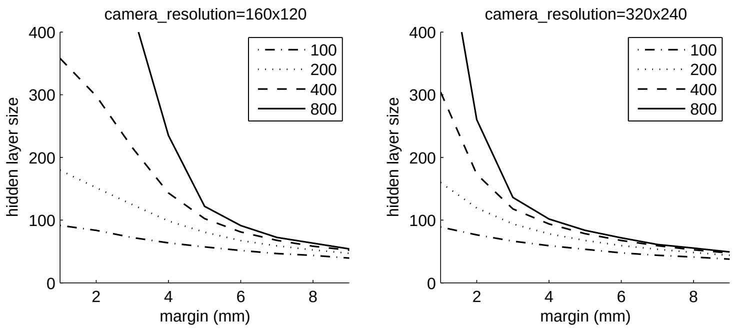


Fig. 6. Hidden layer size versus *margin* curves for four different training set sizes and two camera resolutions. Unlike the *rmse-margin* curves that are difficult to generate on a physical robotic platform, these curves can be easily produced.

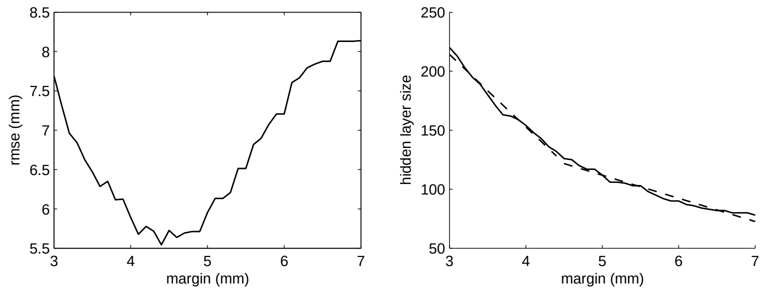


Fig. 7. The left plot shows the *rmse-margin* curve of a RBFN trained on a 400-sample set. The right plot shows that the hidden layer size versus *margin* curve of the same network can be fitted very well with two straight lines (dashed). The intersection of these two lines corresponds roughly to the optimal *margin*. A 160x120 camera resolution is used during motor babbling.

marking on the end effector does not solve the problem effectively because it is hard to guarantee that this marking is visible for all arm postures. From Fig. 5, we can see that *rmse* rises significantly as *margin* moves away from its optimal value. For a RBFN learned through motor babbling on a physical robot, it is difficult, if not impossible, to measure the true error vector $f_{arm}(\theta_i) - \tilde{f}_{arm}(\theta_i)$ accurately, where $\tilde{f}_{arm}$ is the learned forward model represented by a RBFN. Thus, finding out the optimal *margin* on a physical robot through gathering data to plot the *rmse-margin* curve is impractical. Fortunately, the representation of the RBFN provides a measurement that is easy to gather and useful for determining the optimal *margin*. Fig. 6 shows the curves of the hidden layer size of the RBFN versus *margin*. Many of those curves appear to be consisted of two straight segments whose intersection corresponds approximately to the optimal *margin*. Some curves exhibit a third segment in the middle, but this phenomenon is caused by the coarse increment for *margin* used for generating these curves. This observation leads to the following

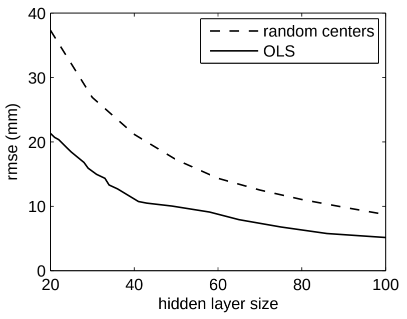


Fig. 8. The solid curve is generated by training a RBFN repeatedly on a 400-sample set with OLS, each time with slightly larger *margin*. Each data point on the dashed curve is the averaged *rmse* of a RBFN trained on 40 different 400-sample sets. Each time, the number of the basis functions in the RBFN is kept the same, but their centers are randomly set to the joint vectors of randomly selected samples from the training set. A 320x240 camera resolution is used during motor babbling.

practical strategy: First generate a curve of hidden layer size versus *margin* by training a new RBFN each time with a slightly larger *margin* on the same training set. Then find the splitting point e for which the sum of the errors for fitting the left and the right part of the curve with two different straight lines is minimal. e can be found with a simple exhaustive search. Fig. 7 shows an example.

3.2.2. *A biologically plausible implementation*

Radial basis function networks have a solid theoretic foundation and close ties to regularization theory and Support Vector Machine (SVM).[19,20,21,22] Poggio has suggested that RBFN is one of the learning mechanisms in the brain.[23] Pouget et al. saw RBFNs as a form of population coding that could play an important role in sensorimotor transformation.[24,25]

Our implementation of RBFN uses the OLS algorithm to determine the number and the centers of the basis functions in the hidden layer. OLS is a sophisticated algorithm that tries to approximate an unknown function with a minimum number of hidden neurons; it is highly unlikely to be used by a biological learning system. However, OLS is not essential for RBFN training; other algorithms or even heuristics can be used as substitutes. To consider the worst-case performance, we compared the performance of two RBFNs, one of which is trained with OLS while for the other, the centers of the hidden layers are set to the joint vectors of randomly selected samples from the training set. The results of the simulation are shown in Fig. 8. It can be observed that the *rmse* of the second network is within a factor of 2 of that of the first network if both networks use the same number of hidden neurons. Compared with the overall range of motion of the arm, this error is still quite small.

Once the number and the centers of the basis functions in the hidden layer have been determined, the weights from the hidden layer to the output layer can be calculated directly. The output of the hidden layer for the task vector θ_i of the training sample (θ_i, x_i) can be described with vector h_i. Using $H = [h_1\ h_2\ ...]$ and $X = [x_1\ x_2\ ...]$ respectively, we would like to find a weight matrix W such that

$$WH = X. \tag{2}$$

Usually no exact solution can be found. A minimum norm solution can be computed through the linear least square algorithm as

$$W = XH^T(HH^T)^{-1}. \tag{3}$$

From another perspective, W in Eq. (2) can be seen as an associative memory. It can be formed through an incremental learning rule such as the biologically plausible Hebbian learning, which is the only learning mechanism needed to learn a forward model of the arm with a RBFN since the centers of the basis functions in the RBFN can be randomly selected as discussed in the previous paragraph.[9,16]

3.3. *Incremental trajectory generation*

3.3.1. *Algorithm description*

The forward kinematics equation $x = f_{arm}(\theta)$ can be transformed into

$$\dot{x} = J(\theta)\dot{\theta} \tag{4}$$

by taking derivative on both sides. RMRC solves Eq. (4) with

$$\dot{\theta} = J^{\#}\dot{x}, \tag{5}$$

where $J^{\#}$ is the pseudo-inverse of the Jacobian matrix J.[1] $\dot{\theta}$ in Eq. (5) is the minimum norm solution to $\dot{x}$ satisfying Eq. (4). Liegois proposed an extension

$$\dot{\theta} = J^{\#}\dot{x} + \alpha(J^{\#}J - I_n)\nabla H \tag{6}$$

that exploits the null space of J to incorporate an additional optimization criterion H into Eq. (5).[26] Typical applications of Eq. (6) include singularity avoidance and obstacle avoidance.[27,28] We use the original form of RMRC as the basis for our incremental trajectory generation algorithm (ITGA). However, it should be noted that our ITGA can be easily adapted to any extension of RMRC based on Eq. (6).

Eq. (5) can be approximated by

$$\Delta\theta = J^{\#}\Delta x. \tag{7}$$

$J^{\#}$ in Eq. (7) can be understood as a directional mapping which transforms the direction vector Δx in the task space into direction vector $\Delta\theta$ in the joint space. It has been suggested that an approximation $\tilde{J}^{\#}$ of $J^{\#}$ can be learned independent of the forward model $\tilde{f}_{arm}$. However, the most direct way to obtain $\tilde{J}^{\#}$ is to extract $\tilde{J}$ from $\tilde{f}_{arm}$ and transform it into $\tilde{J}^{\#}$. (Recall that $\tilde{f}_{arm}$ can be easily learned as was shown in Section 3.2.) One way to extract $\tilde{J}$ is to replace the basis functions in the hidden layer of the RBFN representing $\tilde{f}_{arm}$ with their appropriate partial derivatives.[29] For instance, in order to get an approximation of $[J_{11}, J_{21}, J_{31}]^T$, we simply replace the basis functions by their partial derivatives with respect to θ_1 and use the output of the network as the approximation for $[J_{11}, J_{21}, J_{31}]^T$. In this way, a complete Jacobian approximation $\tilde{J}$ can be constructed. Another way to derive $\tilde{J}$ is simply to use numerical differentiation. Our ITGA that relies on both $\tilde{f}_{arm}$ and $\tilde{J}$ is shown below. This version assumes that visual feedback of the end effector position is not available throughout the reaching movement.

```
1   **initialize** $\theta_{start}, x_{target}, step\_size$
2   $\tilde{\theta}(0) \leftarrow \theta_{start}$
3   $\tilde{x}(0) \leftarrow \tilde{f}_{arm}(\tilde{\theta}(0))$
4   $i \leftarrow 0$
5   **loop**
6       $i \leftarrow i + 1$
7       **if** $\|x_{target} - \tilde{x}(i-1)\|_2 > step\_size$
8           $\alpha(i) \leftarrow step\_size / \|x_{target} - \tilde{x}(i-1)\|_2$
9       **else**
10          $\alpha(i) \leftarrow 1$
11      **end**
12      Calculate $\tilde{J}$ and $\tilde{J}^{\#}$ for $\tilde{\theta}(i-1)$
13      $\Delta x \leftarrow \alpha(i)(x_{target} - \tilde{x}(i-1))$
14      $\tilde{\theta}(i) \leftarrow \tilde{\theta}(i-1) + \tilde{J}^{\#}\Delta x$
15      Output $\tilde{\theta}(i)$ to the motor controller
16      $\tilde{x}(i) \leftarrow \tilde{f}_{arm}(\tilde{\theta}(i))$
17      **if** $\alpha(i)$ **equal** 1
18          **break**
19      **end**
20  **end**
```

Based on the predicted current position of the end effector $\tilde{x}(i-1)$ and the target x_{target}, the joint vector $\tilde{\theta}(i)$ at step i is calculated using

$$\tilde{\theta}(i) = \tilde{\theta}(i-1) + \tilde{J}^{\#}\alpha(i)(x_{target} - \tilde{x}(i-1)), \tag{8}$$

where $x_{target} - \tilde{x}(i-1)$ represents the direction vector in the task space toward the reaching target. $\alpha(i)$ ensures that the magnitude of $\alpha(i)(x_{target} - \tilde{x}(i-1))$ is equal to *step_size* before the final reaching step. Eq. (8) can be seen as a further approximation of Eq. (7).

3.3.2. *Analysis*

To make it clear that both the learned forward model $\tilde{f}_{arm}$ and the extracted Jacobian $\tilde{J}$ are involved in Eq. (8), we can rewrite Eq. (8) as

$$\tilde{\theta}(i) = \tilde{\theta}(i-1) + \tilde{J}^{\#}\alpha(i)(x_{target} - \tilde{f}_{arm}(\tilde{\theta}(i-1))). \tag{9}$$

According to Eq. (9), the errors in both $\tilde{J}$ and $\tilde{f}_{arm}$ contribute to the final position error. In order to isolate the effects of these two kinds of errors, we consider three variants of Eq. (9):

$$\tilde{\theta}(i) = \tilde{\theta}(i-1) + J^{\#}\alpha(i)(x_{target} - \tilde{f}_{arm}(\tilde{\theta}(i-1))). \tag{10}$$

$$\tilde{\theta}(i) = \tilde{\theta}(i-1) + \tilde{J}^{\#}\alpha(i)(x_{target} - f_{arm}(\tilde{\theta}(i-1))) \tag{11}$$

and

$$\tilde{\theta}(i) = \tilde{\theta}(i-1) + J^{\#}\alpha(i)(x_{target} - f_{arm}(\tilde{\theta}(i-1))) \tag{12}$$

In contrast to Eq. (9), Eq. (10) only uses $\tilde{f}_{arm}$ to generate a reaching trajectory while Eq. (11) only uses $\tilde{J}$. So the final position error resulting from an ITGA based on Eq. (10) or (11) is caused solely by the error in $\tilde{f}_{arm}$ or $\tilde{J}$ respectively. Eq. (12) can be seen as an exact reformulation of Eq. (7). Because Eq. (12) relies on the exact forward kinematics and Jacobian, among the four equations listed above, it should achieve the best reaching performance and can serve as a benchmark. For the sake of simplicity, the four different versions of ITGA based on Eq. (9), (10), (11) and (12) are referred to below as ITGA($\tilde{f}_{arm}, \tilde{J}$), ITGA($\tilde{f}_{arm}, J$), ITGA($f_{arm}, \tilde{J}$) and ITGA($f_{arm}, J$) respectively.

Error histograms of the four ITGAs defined above are shown in Fig. 9. They are generated by simulations based on the same forward model. A 320x240 camera resolution is used during motor babbling. It is assumed that x_{target} contains no perceptual noise. 10mm is assigned to the variable *step_size*. The starting posture is selected such that all joints assume values in the middle of their motion ranges. Since ITGA($\tilde{f}_{arm}, \tilde{J}$) uses no visual feedback at all, it is very satisfying to see that almost all of its errors are below 5mm. Despite the good reaching performance achieved, there are two very interesting questions to be answered: What are the most important factors that determine the reaching accuracy? Does our model scale to tasks requiring very high reaching accuracy, e.g. threading a needle?

Fig. 10 shows a second set of error histograms for the four ITGAs with reduced camera resolution of 80x60. With all other simulation parameters held constant, the error histograms of ITGA($\tilde{f}_{arm}, \tilde{J}$) and ITGA($\tilde{f}_{arm}, J$) look much worse than their counterparts in Fig. 9. However, the error histogram of ITGA($f_{arm}, \tilde{J}$) does

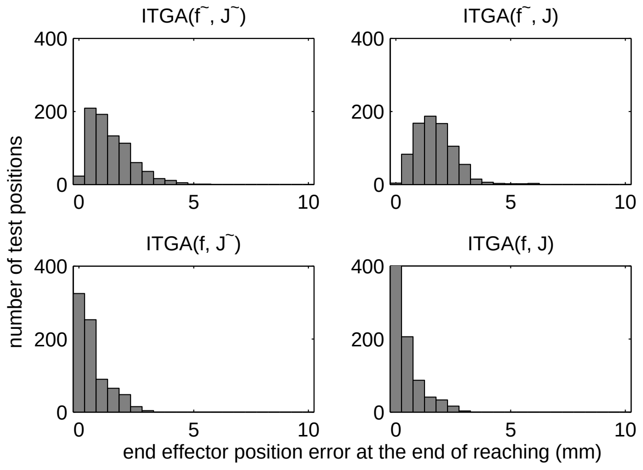


Fig. 9. Error histograms of the four different versions of ITGA. A 320x240 camera resolution is used during motor babbling. 10mm is assigned to *step_size*.

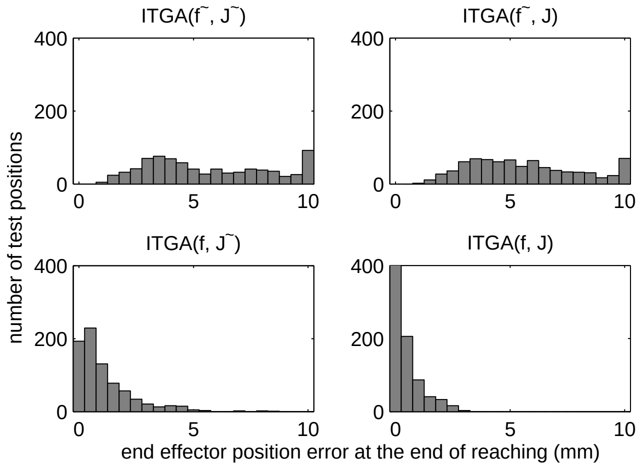


Fig. 10. Error histograms of the four different versions of ITGA. A 80x60 camera resolution is used during motor babbling. 10mm is assigned to *step_size*.

not deteriorate significantly. More interestingly, it can observed in both Fig. 9 and Fig. 10 that the reaching accuracy of ITGA($f_{arm}, \tilde{J}$) is much higher than that ITGA($\tilde{f}_{arm}, J$), which seems to indicate that the quality of $\tilde{J}$ is much better than $\tilde{f}$. Through a careful examination of the original ITGA algorithm listed in Section 3.3.1 and Eq. (10) and (11), it can be discovered that while the reaching accuracy ITGA($\tilde{f}_{arm}, J$) is almost single-handedly determined by the errors in $\tilde{f}_{arm}$, the

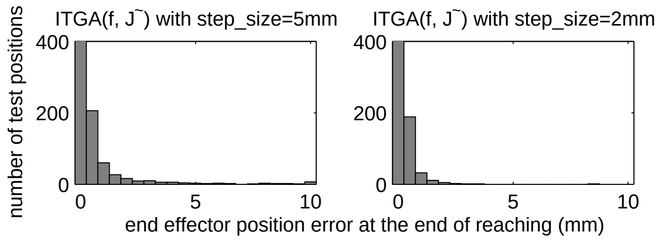


Fig. 11. Error histograms of ITGA$(f, \tilde{J})$ based on reduced *step_size*s. A 80x60 camera resolution is used during motor babbling.

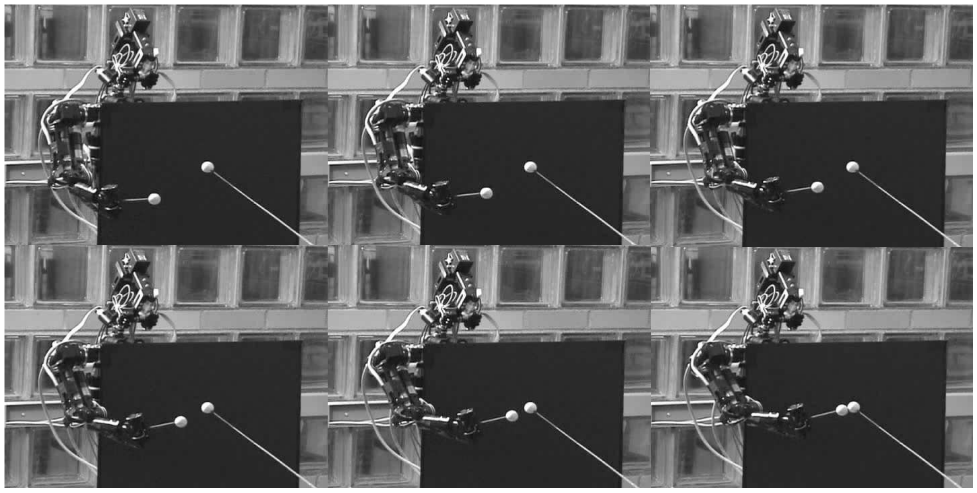
reaching accuracy of ITGA$(f_{arm}, \tilde{J})$ is determined by both the errors in $\tilde{J}$ and the variable *step_size*. Furthermore, the reaching accuracy can be improved by reducing *step_size* while keeping $\tilde{J}$ fixed. Fig. 11 shows two error histograms produced by ITGA$(f_{arm}, \tilde{J})$ with reduced *step_size*s while inheriting the rest of the parameters from Figure 10(c). The mean reaching error achieved by ITGA$(f_{arm}, \tilde{J})$ with *step_size* = 2 is only 0.26mm, enough for threading a needle considering the average size of a needle eyelet. This result is surprising because the forward model used for Fig. 11 is learned with a very crude camera resolution. For physical experiments on a robotic platform, we have no other choice than using ITGA$(\tilde{f}_{arm}, \tilde{J})$ for trajectory generation. But if we use the feedback from the stereo vision to track the position of the end effector during reaching movements, we no longer need $\tilde{f}_{arm}$ for prediction, as long as the end effector is visible to the stereo vision system. In fact, if x_{pred} is substituted by x_{perc} in Fig. 2, the actual reaching movements are controlled by a new ITGA. Its core equation can be described as

$$\tilde{\theta}(i) = \tilde{\theta}(i-1) + \tilde{J}^{\#}\alpha(i)(x_{target} - st(f_{arm}(\tilde{\theta}(i-1)))), \tag{13}$$

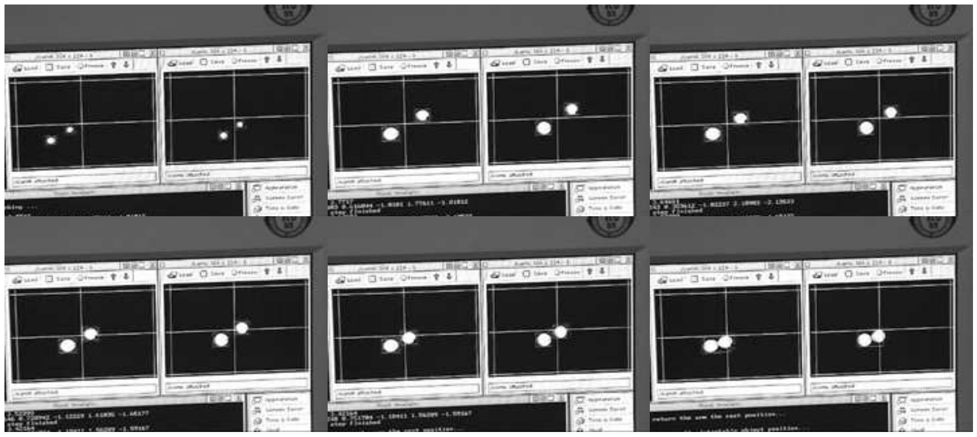
where the function $st()$ is the stereo perception function. The higher the resolution of visual feedback, the closer $st(x)$ is to x. With a very high camera resolution for visual feedback, the term $st(f_{arm}(\tilde{\theta}(i-1)))$ in Eq. (13) is virtually indistinguishable from $f_{arm}(\tilde{\theta}(i-1))$ such that the new ITGA in effect becomes ITGA$(f_{arm}, \tilde{J})$. With the discussion above and the results shown in Fig. 11, it can be concluded that a very high reaching accuracy can be achieved with a small *step_size* and the aid of visual feedback of a high resolution. This combination can compensate for the relatively large errors in a forward model trained on samples that are gathered under a crude camera resolution.

4. Physical Experiments on Nico

Experiments have been carried out on Nico to test the performance of our model. A 320x240 camera resolution is used during motor babbling. 400 samples are collected to train a forward model of the arm kinematics. It takes less than 30 minutes to



(A)



(B)

Fig. 12. Please refer to Section 4 for detailed description.

collect these samples. *spread* is set to 130. *margin* is determined by the heuristics described in Section 3.2.1. The reaching target is attached to the tip of a modified retractable TV antenna to facilitate its positioning. A fixed starting posture is used to generate all reaches. At the beginning of a reaching movement, *step_size* is set to 10mm and a 320x240 camera resolution is used to locate the target. After the difference between the target position and the end effector position estimated by the forward model becomes smaller than 50mm, *step_size* is reduced to 5mm and the camera resolution is switched to 640x480. Visual feedback of the end effector is exploited whenever available after the resolution switch. Fig. 12 shows two groups of pictures. Each group consists of six pictures organized from left to right with the pictures in the second row following the pictures in the first row. Group (A) shows a successful reach by Nico that moves its end-effector toward the target with the guidance of visual feedback. The pictures in Group (B) shows a reaching movement from the perspective of Nico's stereo vision system. Segmentation results presented in these pictures are computed using prior knowledge about color. The camera images in the first picture are captured before the resolution switch. The images in the rest of the pictures are captured with a 640x480 resolution. It can be seen from the last picture that at the end of the movement, the end effector already touches the target from the perspective of Nico's stereo vision system.

5. Extending the Model to More Degrees of Freedom

To extend our model to more realistic situations, we allow the four neck joints in Nico are allowed to move freely during motor babbling. With the neck joints activated, the forward kinematics becomes $f_{neck\_arm} : (\theta_{arm}, \theta_{neck}) \rightarrow x$. Our previous work demonstrated that the number of training samples needed to learn a forward model of a 4-DOF arm using the method described in Section 3.2 is around 120,[30] while in this paper, about 400 samples are used for a 6-DOF arm. Using a larger training set to learn the expanded forward kinematics does not constitute a good strategy because the number of training samples needed rises almost exponentially with each additional joint. However, sensory information other than proprioceptive feedback can be exploited to alleviate the degree of freedom problem.

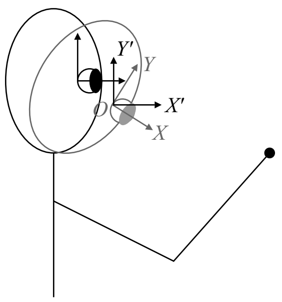


Fig. 13. Illustration showing the effect a shifted head posture on the eye-centered coordinate system. The original and the shifted head posture is painted in black and grey respectively. The sensory feedback from the vestibular system (the gyroscope in the case of a robot) allows for the correction of the eye-centered coordination system OXY into $OX'Y'$.

Fig. 13 illustrates a simplified situation where a shift of the head posture leads to changes in both the position and the orientation of the head. Although the eye-centered coordinate system's positional change can not be perceived directly, its change in orientation can be sensed by the vestibular system (a gyroscope in case of a robot). The information delivered by the vestibular system/gyroscope can then be used to correct this orientation shift caused by the postural change of the head, i.e. the coordinate system OXY can be corrected into $OX'Y'$ as shown in Fig. 13.

The original input and output of the kinematics function f_{neck_arm} is $\theta = (\theta_{arm}, \theta_{neck})$ and $x = T_{body}^{eye}(\theta_{neck}) \cdot T_{ee}^{body}(\theta_{arm}) \cdot [0,0,0,1]^t$ respectively, where T_{body}^{eye} and T_{ee}^{body} are homogenous transformation matrices. ee stands for end-effector. T_{body}^{eye} can be expressed as

$$T_{body}^{eye} = \begin{bmatrix} R_{3\times 3} & T_{3\times 1} \\ 0_{1\times 3} & 1_{1\times 1} \end{bmatrix}, \tag{14}$$

where R is the rotation matrix and T is the translation vector. R can be reconstructed from the readings from the vestibular system/gyroscope. We denote T'^{eye}_{body} as

$$T'^{eye}_{body} = \begin{bmatrix} R_{3\times 3} & 0_{3\times 1} \\ 0_{1\times 3} & 1_{1\times 1} \end{bmatrix}. \tag{15}$$

Using T'^{eye}_{body}, the perceived position x of the end-effector can be transformed into $x' = (T'^{eye}_{body})^{-1} \cdot x$. This has the same effect as the correction of OXY into $OX'Y'$ shown in Fig. 13. If we assume that the overall effect of the neck joints is purely rotational so that it is fully eliminated in x', (θ_{arm}, x') is just a lossless transformation of $((\theta_{arm}, \theta_{neck}), x)$. In reality, the assumption just mentioned does not hold true, which can be easily seen in Fig. 13. Learning a forward model with the transformed training samples is equivalent to regarding the translational effect of the neck joints as noise. Simulations based on the kinematics of Nico have determined that the standard deviation of possible end effector positions is more than three times the standard deviation of possible eye camera positions. Fig. 14 shows a scatter plot for 1000 random positions of the end-effector and a scatter plot for 1000 random positions of the right eye camera. The large difference in the ranges of motion between the end-effector and the eye camera means x' is determined to a much greater extent by the translational effect of the arm joints than that of the neck joints. Therefore, learning a forward model on samples (θ_{arm}, x') will not degrade the quality of the forward model significantly.

This learning strategy does not increase the number of dimensions of the kinematic models. It has the additional advantage that the diagram shown in Fig. 2 does not need to be changed for generating reaching trajectories. Simulations show that the average position error of blind reaching based on a forward model learned with the new strategy is about 20mm. Although this is significantly larger than the average position error ($< 5mm$) achieved by a system that assumes a fixed head posture, it can be compensated by exploiting visual feedback of the end effector during the actual reaching movements.

The essential sensory information delivered by the vestibular system/gyroscope can be summarized into a three dimensional vector θ_{euler}. It is worth noting that on our robotic platform, if a forward model is learned on samples in form of $((\theta_{euler}, \theta_{arm}), x)$, the dimension of the input is nine, only one less than that of

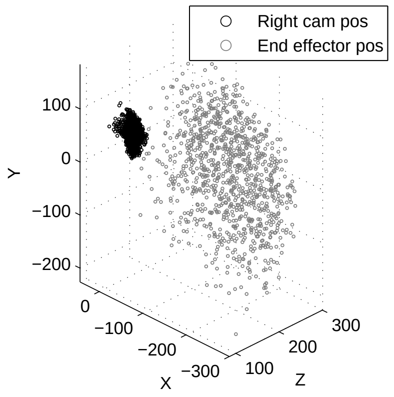


Fig. 14. 1000 random positions for both the right eye camera (in black) and the end effector (in gray) are plotted in the body-centered coordinate system. It can be easily seen that the range of motion of the end effector is much larger than that of the eye camera.

the input of the original f_{arm_neck}. While this seems to be only a small dimensionality reduction on our robotic platform, the advantage will scale with additional joints in the neck. Because the human neck has more than four DOFs at even the joint level, the dimensionality reduction can be substantially more. On the level of muscle activation, using θ_{euler} to represent the principle effect of the head posture can lead to an enormous saving of both the time spent on learning and the size for model representation.

6. Discussion

It has been suggested by some psychologists that the representation of a reaching target undergoes a series of transformation stages from the eye-centered coordinate system via the head-centered and body-centered coordinate systems to the hand-centered coordinate system before an arm movement is initiated.[31,32] However, studies of hemineglect patients and recording experiments in the brain areas associated with reaching have not found convincing evidence for the existence of neural representations unique to the intervening stages.[33,34] So far, only substantial neural correlation of the difference vector between the target and the hand position in the eye-centered coordinate system has been reported.[35] Our model can use this difference vector directly for generating reaching movements without any additional transformational stages. A forward model can be learned on training samples in form of either (θ_{arm}, x) or $((\theta_{euler}, \theta_{arm}), x)$ depending on whether the neck is activated during motor babbling. Figure 6 demonstrates that a forward model with only about 100 hidden neurons can already represent the forward kinematics of the arm reasonably well (examine the value of the curve corresponding to 400 training samples in the right plot at *margin*=4.5mm). Even if the neck is activated during motor babbling, a population with at most several thousands hidden neurons will be sufficient. This number is far smaller than previously reported results on a 3-DOF arm.[4,9]

One interesting paper that studies motor learning on a robotic platform tries to solve the degree of freedom problem with a sophisticated statistical learning algorithm called Locally Weighted Projection Regression (LWPR).[11] LWPR is used to learn the direction mapping from Δx to $\Delta\theta$ with effort put into locally reducing the dimensionality. However it does not make use of any additional sensory information other than proprioceptive feedback. Our experiments suggest that the dimensionality of motor learning can be reduced globally by completely substituting the proprioceptive feedbacks from all neck joints with the feedback from the vestibular system that can be found in all human beings.

In Section 3.3.2 we came to the conclusion that a very high reaching accuracy can be achieved with high resolution visual feedback and small *step_size* even if the forward model is learned with a crude visual resolution. *step_size* can be seen as the counterpart of speed in the discrete domain. This conclusion is supported by observations of human behavior. It has long been known that for humans, the accuracy of a reaching movement is inversely related to the speed of the movement.[36] However, if high resolution vision can be used for feedback during reaching movements, why not use it for forward learning as well? Why bother to consider using a crude visual resolution at all? As mentioned in Section 3.2.1, because we replace the projections of the end effector on the two camera image planes with their centroids for the calculation of its 3D position, the effective resolution can be crude even if a high camera resolution is used. When an infant starts to make spontaneous reaches, its vision is quite poor compared with that of an adult. The conclusion drawn in Section 3.3.2 ensures that a forward model learned with crude vision can be used later on for generating accurate reaches when high resolution vision becomes available.

Reaching is one of the most thoroughly investigated sensorimotor tasks. Because the human arm contains more joints than the dimensionality of an ordinary task space, there are in principle infinite postures that reach the same target in space. The experiments carried out by Morasso demonstrated the remarkable result that human reaching movements invariably possess two characteristics: the trajectory is gently curved and the velocity profile is bell-shaped.[37] Many theories have been proposed to explain the observations made by Morasso with the assumption that motor learning is governed by some optimization measure. The optimization measures suggested include minimum jerk, maximum smoothness, minimum torque and, recently, minimum variance of the final hand position.[38,39,40,41] For a comprehensive review, please refer to a recent review by Todorov.[42]

Interestingly, the strategy of substituting sensory feedback from the vestibular system for the neck kinematics also causes curved reaching trajectories. This substitution leads to errors in the forward model and the extracted Jacobian. These errors, while still relatively small, invariably result in a trajectory that does not follow the optimal shortest path through Cartesian space. Fig. 15 shows a sample trajectory for both blind reaching and reaching aided by visual feedback. Both trajectories are visibly curved. This observation serves as a support for dimensionality reduction as a complementary explanation to the observed curvature in human reaching trajectories. We have recently begun to explore this effect both in simulation and with the robot Nico.[43]

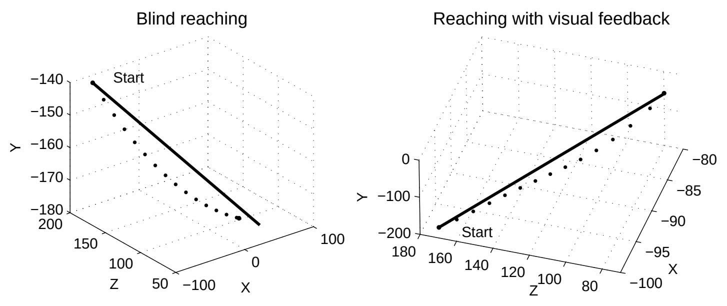


Fig. 15. Sample reaching trajectories for blind reaching and reaching with visual feedback. The straight lines in both plots connect the starting position of the effector effector and the target position. The dotted lines are the actual reaching trajectories. For blind reaching, the position of the end effector at the end of the movement is a small distance away from the target because of the error in the learned forward model. These two trajectories are based on a forward model trained on samples of the form (θ_{arm}, x').

7. Conclusion

In this paper, we have presented a model for learning to reach. A forward model of the arm kinematics is learned on autonomously gathered training samples. We have described in detail an optimization process for the learning parameters because the quality of the learned forward model heavily influences the position error of a blind reach. With the learned forward model, reaching movements towards targets can be generated based on the derived directional mapping. We have shown through careful analysis that if high resolution visual feedback is available during reaching, tasks requiring high accuracy can be achieved despite the residual errors in the forward model. For a 6-DOF arm, our model requires only about 400 training samples, which takes a very modest amount time to collect on a modern robotic platform.

In addition, the consistency of our model with physiological and psychological observations of human arm movements has been discussed. We have demonstrated that through the substitution of a portion of the proprioceptive feedback by other sensory feedback, motor learning can take place in a space with a dimension that is much lower than the number of actual degrees of freedom involved. Even when all the degrees of freedom in the neck and the arm are taken into account, a maximum of only a few thousand neurons are needed to represent a forward model for generating accurate reaching trajectories. This number is far fewer than previous conjectures.

8. Acknowledgement

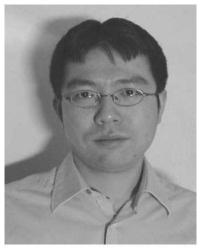
Support for this work was provided by a National Science Foundation CAREER award (#0238334). Some parts of the architecture used in this work was constructed under NSF grants #0205542 (ITR: A Framework for Rapid Development of Reliable Robotics Software) and #0209122 (ITR: Dance, a Programming Language for the Control of Humanoid Robots) and from the DARPA CALO/SRI project.

References

1. D. E. Whitney, Resolved motion rate control of manipulators and human prostheses, *IEEE Transactions on Man-Machine Systems* **10**(2), 47-53 (1969).
2. J. Piaget, *La naissance de l'intelligence chez l'enfant*, (Delachaux et Niestlé, Geneva, 1936).
3. D. DeMers, K. Kreutz-Delgado, Inverse kinematics of dextrous manipulators, in *Neural Systems for Robotics*, ed. O. Omidvar and P. van der Smagt (Academic Press, New York, 1997), pp. 75-116.
4. 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).
5. M. I. Jordan and D. Rumelhart, Supervised learning with a distal teacher, *Cognitive Science* **16**, 307-354 (1992).
6. R. C. Miall and D. M. Wolpert, Forward models for physiological motor control, *Neural Networks* **9**, 1265-1279 (1996).
7. R. Shadmehr and S. P. Wise, Motor learning and memory for reaching and pointing, in *The Cognitive Neurosciences III*, ed. M. S. Gazzaniga (MIT Press, Cambridge, Massachusetts, 2004), pp. 511-524.
8. R. Shadmehr and S. P. Wise, *The Computational Neurobiology of Reaching and Pointing*, (MIT Press, Cambridge, Massachusetts, 2005).
9. H. Ritter, T. Martinetz and K. Schulten, *Neural Computation and Self-Organizing Maps: An Introduction*, (Addison-Wesley, New York, 1992).
10. N. E. Berthier, M. T. Rosenstein and A. G. Barto, Approximate optimal control as a model for motor learning, *Psychological Review* **112**, 329-346 (2005).
11. A. D'Souza, S. Vijayakumar and S. Schaal, Learning inverse kinematics, in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (IEEE Press, Piscataway, New Jersey, 2001), pp. 298-303.
12. N. A. Bernstein, *The Co-ordination and Regulation of Movements*, (Pergamon, New York, 1967).
13. J.-Y. Bouguet, Camera calibration toolbox for Matlab, *http://www.vision.caltech.edu /bouguetj/calib_doc/*.
14. J. Heikkilä and O. Silvén, Calibration procedure for short focal length off-the-shelf CCD cameras, in *Int. Conf. on Pattern Recognition* (IEEE Computer Society, Washington, DC, 1996), pp. 166-170.
15. L. G. Shapiro and G. C. Stockman, *Computer Vision*, (Prentice-Hall, Upper Saddle River, New Jersey, 2001), pp. 397-398.
16. S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd edn. (Prentice Hall, Upper Saddle River, New Jersey, 1999).
17. J. Park and I. W. Sandberg, Universal approximation using radial-basis-function networks, *Neural Computation* **3**(2), 246-257 (1991).
18. S. Chen, C. F. N. Cowan and P. M. Grant, Orthogonal least squares learning algorithm

for radial basis function networks, *IEEE Transactions on Neural Networks* **2**(2), 302-309 (1991).
19. M. J. D. Powell, The theory of radial basis function approximation in 1990, in *Advances in Numerical Analysis Vol. II: Wavelets, Subdivision Algorithms, and Radial Basis Functions*, ed. W. Light (Oxford Science Publications, Oxford, 1992), pp. 105-210.
20. T. Poggio and F. Girosi, Networks for approximation and learning, *Proceedings of the IEEE* **78**, 1481-1497 (1990).
21. F. Girosi, M. Jones and T. Poggio, Regularization theory and neural networks architectures, *Neural Computation* **7**, 219-269 (1995).
22. V. N. Vapnik, *The Nature of Statistical Learning Theory*, 2nd edn. (Springer Verlag, New York, 1999).
23. T. A. Poggio, A theory of how the brain might work, *Cold Spring Harbor Symp. Quant. Biol.* **55**, 899-910 (1990).
24. A. Pouget and L. H. Snyder, Computational approaches to sensorimotor transformations, *Nature Neuroscience* **3**, 1192-1198 (2000).
25. S. Deneve, P. E. Latham and A. Pouget, Efficient computation and cue integration with noisy population codes, *Nature Neuroscience* **4**(8), 826-831 (2001).
26. A. Liegeois, Automatic supervisory control of the configuration and behavior of multibody mechanisms, *IEEE Transactions on Systems, Man and Cybernetics* **7**(12), 868-871 (1977).
27. Y. Nakamura and H. Hanafusa, Inverse kinematic solutions with singularity robustness for robot manipulator control, *ASME Journal of Dynamic Systems, Measurement and Control* **108**, 163-171 (1986).
28. A. A. Maciejewski and C. A. Klein, Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments, *The International Journal of Robotics Research* **4**(3), 109-117 (1985).
29. C. M. Bishop, *Neural Networks for Pattern Recognition*, (Oxford University Press, Oxford, 1995), pp. 148-150.
30. G. Sun and B. Scassellati, Reaching through learned forward model, in *IEEE-RAS/RSJ Int. Conf. on Humanoid Robots (Humanoids)*, (Los Angeles, California, 2004).
31. M. Flanders, S. I. Helms-Tillery and J. F. Soechting, Early stages in a sensorimotor transformation, *Behavioral and Brain Sciences* **15**, 309-362 (1992).
32. J. McIntyre and F. Stratta, Short-term memory for reaching to visual targets: psychophysical evidence for body-centered reference frames, *Journal of Neuroscience* **18**, 8423-8435 (1998).
33. A. Pouget, S. Deneve and T. Sejnowski, Frames of reference in hemineglect: a computational approach, *Progress in Brain Research* **121**, 81-97 (1999).
34. J. R. Duhamel, F. Bremmer, S. BenHamed and W. Graf, Spatial invariance of visual receptive fields in parietal cortex neurons, *Nature* **389**, 845-848 (1997).
35. C. A. Buneo, M. R. Jarvis, A. P. Batista and R. A. Andersen, Direct visuomotor transformations for reaching, *Nature* **416**, 632-636 (2002).
36. P. M. Fitts, The information capacity of the human motor system in controlling the amplitude of movements, *Journal of Experimental Psychology* **47**, 381-391 (1954).
37. P. Morasso, Spatial control of arm movements, *Experimental Brain Research* **42**(2), 223-227 (1981).
38. N. Hogan, An organizing principle for a class of voluntary movements, *Journal of Neuroscience* **4**(11), 2745-2754 (1984).
39. T. Flash and N. Hogan, The coordination of arm movements: an experimentally confirmed mathematical model, *Journal of Neuroscience* **5**(7), 1688-1703 (1985).

40. Y. Uno, M. Kawato and R. Suzuki, Formation and control of optimal trajectory in human multijoint arm movement: minimum torque-change model, *Biological Cybernetics* **61**(2), 89-101 (1989).
41. C. M. Harris and D. M. Wolpert, Signal-dependent noise determines motor planning, *Nature* **394**(6695), 780-784 (1998).
42. E. Todorov, Optimality principles in sensorimotor control, *Nature Neuroscience* **7**(9), 907-915 (2004).
43. G. Sun and B. Scassellati, Exploiting vestibular output during learning results in naturally curved reaching trajectories, in *Proceedings of the Fifth International Workshop on Epigenetic Robotics (EpiRob)*, (Nara, Japan, 2005).



Ganghua Sun received his Diploma degree in Computer Science (Diplom.-Inform) from University of Bonn in 2001. He is currently a Ph.D. candidate in the Computer Science Department of Yale University. His research interests include developmental learning of human cognitive functions, statistical learning theory and computer vision. He is the chief designer of the humanoid robot Nico.

Brian Scassellati received both his M.Eng. degree and a Ph.D. in Computer Science and Electrical Engineering from the Massachusetts Institute of Technology in 1995 and 2001, respectively. Since 2001, he has been an Assistant Professor of Computer Science at Yale University.