1993-10

A Network for Learning Kinematics with Application to Human Reaching Models

Fiala, John C.

Boston University Center for Adaptive Systems and Department of Cognitive and Neural Systems

http://hdl.handle.net/2144/2032

Boston University
A NETWORK FOR LEARNING KINEMATICS WITH APPLICATION TO HUMAN REACHING MODELS

John C. Fiala

October 1993


Permission to copy without fee all or part of this material is granted provided that: 1. the copies are not made or distributed for direct commercial advantage, 2. the report title, author, document number, and release date appear, and notice is given that copying is by permission of the BOSTON UNIVERSITY CENTER FOR ADAPTIVE SYSTEMS AND DEPARTMENT OF COGNITIVE AND NEURAL SYSTEMS. To copy otherwise, or to republish, requires a fee and/or special permission.

Copyright © 1993

Boston University Center for Adaptive Systems and
Department of Cognitive and Neural Systems
111 Cummington Street
Boston, MA 02215
A Network for Learning Kinematics with Application to Human Reaching Models

John C. Fiala
Department of Cognitive and Neural Systems, Boston University
111 Cummingston St., Rm 244, Boston, MA 02215-2411

Abstract
A model for self-organization of the coordinate transformations required for spatial reaching is presented. During a motor babbling phase, a mapping from spatial coordinate directions to joint motion directions is learned. After learning, the model is able to produce straight-line spatial trajectories with characteristic bell-shaped spatial velocity profiles, as observed in human reaches. Simulation results are presented for transverse plane reaching using a two degree-of-freedom arm.

1. Introduction
A basic problem for any controller of a complex kinematic mechanism, whether it be a multijoint robotic manipulator or a human arm, is how to generate reaches to targets perceived in visual or tactile sensory spaces. This problem is difficult because the relationship between sensory coordinates and motor coordinates, i.e., the kinematics, is highly nonlinear. This paper presents a general method for learning kinematic relationships that can be used for any sensory-motor system. A related algorithm for short-term adaptations was described previously (Fiala & Lumia, 1991).

Studies of human reaching in a transverse plane, (a horizontal plane located directly in front of the subject,) have found characteristic unimodal velocity profiles (Abend et al. 1982; Ghez et al. 1990). These studies have also shown the motions to be approximately straight lines. Models have been developed to explain these characteristics of reaching to spatial targets. The Vector Integration To Endpoint (VITE) model (Bullock & Grossberg 1988) produces joint trajectories with bell-shaped velocity profiles like those observed in reaching studies. In the Vector Associative Map (VAM) model (Gaudiano & Grossberg 1991), the transformation between a spatial coordinate representation of hand position and the corresponding joint angle position is learned. A recent model, DIRECT (Bullock et al. 1993), self-organizes the transformation from spatial directions to joint rotation directions. The DIRECT model produces trajectories which are straight lines in spatial coordinates, but does not produce bell-shaped velocity profiles as in the VITE model. In the following, a DIRECT model is proposed which produces straight-line motions in spatial coordinates with characteristic bell-shaped velocity profiles as observed in experimental studies. Section 2 describes the basic components of a new network for the transformation of spatial directions to joint rotations. The equations for the network are described in Section 3. Demonstration of network learning performance is given in Section 4. Section 5 shows how the network can be incorporated into a model for human reaching.

2. Basic Network Components
Coordinate transformations of directions (or velocities) form a linear field over arm configuration. That is, at a given set of joint angles the differential relationship between hand motions in spatial coordinates and joint rotations is a linear mapping. The mapping varies with the positions of the joints. Using $0$ to represent the vector of joint angles, and $s$ to represent joint and spatial directions, respectively, the transformations can be represented as

$$ r = J(\theta)^{-1} s $$

$$ s = J(\theta) r $$

At a particular configuration, the basic form of (EQ 1) can be rewritten

$$ r_i = \sum_{j=1}^{m} z_{ji} s_j $$

Each $z_{ji}$ represents the element of the inverse mapping which multiplies the $i^{th}$ spatial component to contribute to the $j^{th}$ joint component. Here, the spatial representation is taken to contain $m$ components and there are $n$ joint angles to
be considered. The mapping, then, has the form of a linear summation appropriate for implementation by a neuron-like component as depicted in Figure 1.

In the direction mapping network, there are \( m \) number of \( S \) cells which form the spatial direction vector input to the network and \( n \) number of \( R \) cells which form the joint rotation direction output of the network. The activities of \( S \) cells encode the vector of spatial directions. The set of \( R \) cells encode the vector of joint rotation directions. A set of \( V \) cells receive input from the \( S \) cells and send output to the \( R \) cells. Each \( V \) cell receives the complete set of spatial inputs \( S_i \), \( i=1,..,m \), but connects to only one \( R \) cell. The mechanism used for learning to ensure weights converge to the correct linear mapping is similar to the VAM learning construction (Gaudiano & Grossberg 1991). The \( V \) cells compute a difference of activity between the two vector representations via feedback from the \( R \) cells, as shown in Figure 2. During learning, this difference drives the adjustment of the weights. During performance, the difference drives the \( R \) cell activity to the value encoded in the learned mapping.

Since the direction mapping is actually a field over joint configuration, some means is needed to account for the joint configuration context in which the mapping is computed. To do this, a construction called a context field is used. A context field is a set of tonically active inhibitory cells which receive broad-based inputs that determine the context of a motor action. A context field cell pauses when it recognizes a particular motor state on its inputs, and thereby disinhibits its target cells. This is depicted in Figure 3. Since a context field cell is tonically active, it can be thought of as responding to all contexts except for one particular one to which it is tuned. Thus, for most states, the target cells of a context cell are inhibited. It is assumed that target cells are completely shut-off when their context cells are active. In the direction mapping problem, each cell in the context field pauses for a particular joint angle configuration. The cells of the context field span the joint configuration space at some resolution. A cell is "off" for a compact region of the joint space. It is assumed for simplicity that only one context field cell turns "off" at a time. Figure 3 depicts a two degree-of-freedom joint configuration space. The center context field cell is "off" (open circle) when the joint angles are in the center region of the joint space, in this example.

3. Direction Mapping Construction

The overall construction for learning a spatial-to-joint direction mapping combines a context field with the \( V, S \) and \( R \)
cells. The V cells are the target cells of a context field. The context field is a set of tonically active cells which encode the joint configuration. Each context field cell projects to a set of V cells, one for each joint vector component. Each joint vector component has a set of V cells associated with it, one for each context. The context field cells select which V cells will be active at any time. There are nXo V cells in the network, one for each joint rotation in each context situation. The V_k, i=1..n, are inhibited by the k^th context field cell. Each V_k cell receives additive inhibitory feedback from the R_i cell to which it projects.

The spatial direction cells S_j are driven by the visual input of spatial direction s_j,

\[ \dot{S}_j = \delta (s_j - S_j) \]  

The V cells are driven to the difference of the weighted spatial cell input and the joint-related R cell activity,

\[ \dot{V}_{ik} = \alpha (-V_{ik} + c_k \sum_j z_{ijk}S_j - R_i) \]  

The c_k represents inhibition from the context field. The k^th context field cell contacts the set \{V_k, i=1..n\} of V cells. When the context cell is active, the entire input current to the soma is shunted away such that there remains only activity in the axon hillock, which decays to zero. This is modeled as c_k=0. When the k^th context cell shuts off, c_k=1, the V_k receive normal input.

The joint direction cells R_i are driven by the V_k during performance and by joint rotation velocities r_i during learning.

\[ \dot{R}_i = \delta \left[ (e-1) \sum_k V_{ik} - R_i + e (r_i - R_i) \right] \]  

An endogenous generator of babbling movements gates the input to the R_i (Gaudiano & Grossberg 1991). When e=1, motor babbling is active and the R_i are driven to sensed joint velocities r_i. When motor babbling is inactive, e=0, input is the sum of the V_k, only one of which will be actively processing input.

Learning is obtained by decreasing weights in proportion to the product of the presynaptic and postsynaptic activities.

\[ z_{ijk} = -\gamma c_k V_{ik} S_j \]  

It is probably not necessary to include the e and c_k gates in (6) but this is used in the simulations which follow. Note also that the implementation allows positive and negative values for the sj and ri, as well as for the z_{ijk} weights. The network can be redesigned to have only positive activations and weights by using the appropriate push-pull mechanisms as in (Gaudiano & Grossberg 1991).

During learning in a particular context k, with e=1,

\[ S_j \rightarrow s_j \]
\[ R_i \rightarrow r_i \]
\[ V_{ik} \rightarrow \sum_j z_{ijk}S_j - R_i \]

Therefore, the learning rule could be restated in discrete form as

\[ z_{ijk} (t+1) = z_{ijk} (t) \eta (r_i - \sum_j z_{ijk} (t) S_j) S_j \]  

where \( \eta \) is a constant gain. This is a gradient descent algorithm (Hertz et al. 1991) for the solution of

\[ r_i = \sum_j z_{ijk} S_j \]  

Thus, the learning law will result in the desired weights as is demonstrated by simulation in the next section.
4. Learning Algorithm Performance

To test the convergence of the direction mapping learning algorithm, simulations were performed using a two-link planar arm model. A two-link planar arm allows explicit computation of the inverse of the Jacobian for comparison to the learned mapping represented by the $z_{ijk}$ weights. The two-link planar mechanism is defined as shown in Figure 4. Here, $\theta_1$ and $\theta_2$ are the joint angles, and $l_1$ and $l_2$ are the link lengths. The base of the mechanism is located a distance $a$ from the origin of the spatial coordinate frame. For the simulations discussed below, $l_1=0.25\text{m}$, $l_2=0.31\text{m}$, and $a=0.2\text{m}$.

In Cartesian coordinates, the spatial position of the end effector is given by

$$
\begin{align*}
    x_1 &= l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) + a \\
    x_2 &= l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2)
\end{align*}
$$

(9)

The ideal value for the learned spatial-to-joint direction mapping is the inverse of the Jacobian of this coordinate transformation.

Training is done by generating random movements, and using the resulting joint velocities and observed spatial velocities of the hand as training vectors to the direction mapping network. A joint configuration target $\theta_4$ is randomly selected and a joint-space movement is generated to this goal. As the arm slows to some minimum velocity in the region of the goal position, a new target is randomly generated and the arm driven to that goal. The motor babbling gate is set, $\omega=1$, throughout these motions. The spatial velocities are obtained in the simulation by multiplying $r_i$ by the actual Jacobian computed from (9). This simulates spatial velocities obtained, for example, from vision.

The magnitude of the difference between the numerically computed inverse of the Jacobian and the $z_{ijk}$ at a particular joint configuration is the error in the direction mapping. In Figure 5 the error is plotted for every hundredth trial, where one trial is the trajectory to a randomly generated target configuration. The error is shown over a 20,000 trial epoch. Initially, the weights are all zero. The parameters used for this simulation were $\alpha=16$, $\gamma=8$, $\delta=32$. Integration used Euler's method with a time step of 0.025s. The joint angles were constrained to $\theta_1 \in (0.8, 1.6)$ and $\theta_2 \in (1.6, 2.4)$ in radians. The context field cells had fixed rectangular receptive fields. The joint space was divided into 9 bins on the $\theta_1$ axis and 9 bins on the $\theta_2$ axis. Figure 5 shows that the network exponentially learns the inverse of the Jacobian for a particular joint position. Note that the discretization of the joint space is into bins about 0.09 radians wide. Within a bin of this size the value of the inverse Jacobian can vary about the amount shown as the steady-state error in the figure. Thus, the inverse of the Jacobian is learned to the accuracy determined by the resolution of the context field.

5. Performance of Reaching Trajectories

Since the spatial-to-joint direction mapping will accurately relate the spatial difference magnitude to a corresponding joint difference magnitude, the model can successfully generate characteristic unimodal trajectory profiles. All that is required is the addition of the GO signal as shown in Figure 6(A).

As indicated in the figure, the spatial direction vector $DV_s$ is computed by taking the difference between the spatial target position and the spatial hand position, which can be obtained from visual input. This difference is continually computed during the motion. The $DV_s$ is transformed to the $DV_m$, the joint-space direction vector, using the direction

![Figure 4. Planar Two-Link Arm.](image)

![Figure 5. Error in Learned Representation.](image)
A model of spatial reaching based on the DIRECT model (Bullock et al. 1993) has been developed. This model uses a new construction for learning the direction mapping between spatial and joint coordinates. The performance of this construction has been successfully demonstrated in simulations with a two degree-of-freedom planar manipulator. The method can be applied to non-planar and kinematically redundant arms. In the case of a redundant arm, the direction mapping learns something analogous to the pseudoinverse of the Jacobian. The direction mapping is also well-
Figure 7. Three degree-of-freedom reaching in a sagittal plane. Reaches to Cartesian targets result in somewhat curved spatial trajectories (A) with characteristic unimodal velocity profiles (B).

behaved near kinematic singularities. Figure 7 shows the model applied to sagittal plane reaches with a three degree-of-freedom arm. A Cartesian coordinate system is used to represent spatial targets. As in experiments (Atkeson & Hollerbach 1985), trajectories have bell-shaped velocity profiles but are not always straight lines. To more closely reproduce human reaches the model would need to include the arm mass dynamics as well as a method for internal feedback of hand movements when visual feedback is unavailable, such as implemented in the DIRECT model (Bullock et al. 1993).

The context field used in the network separates the determination of context from the computation of the transformations which depend on it. The context field could incorporate a wide range of inputs. For example, an additional context input specifying the coordinate system of the command could be used to select between direction mapping learned for different coordinate systems. The context field can also drive a number of independent functions which depend on context. Each function does not require its own separate context field. For example, the joint space context used for selecting the direction mapping could simultaneously select the appropriate inertia compensation mapping as well. Such a learned compensation method would complement short-term adaptation for inertia changes as described by Fiala and Lumia (1991).

Acknowledgments
The author thanks Dan Bullock and Frank Guenther for helpful discussions on this subject and on the manuscript. This work was funded in part by the Office of Naval Research (ONR N00014-92-J-1309).

References