N90-29030

"Computational" Neural Learning Formalisms for Manipulator Inverse Kinematics

Sandeep Gulati Jacob Barhen S. Sitharama Iyengar[§]

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

ABSTRACT

An efficient, adaptive neural learning paradigm for addressing the inverse kinematics of redundant manipulators is presented. The proposed methodology exploits the infinite local stability of terminal attractors - a new class of mathematical constructs which provide unique information processing capabilities to artificial neural systems. For robotic applications, synaptic elements of such networks can rapidly acquire the kinematic invariances embedded within the presented samples. Subsequently, joint-space configurations, required to follow arbitrary end-effector trajectories, can readily be computed. In a significant departure from prior neuromorphic learning algorithms, this methodology provides mechanisms for incorporating an in-training "skew" to handle kinematics and environmental constraints.

1. INTRODUCTION

Space telerobots envisioned for exacting scientific and military applications in unstructured and hazardous space environments, e.g., satellite servicing, space system construction and maintenance, planetary missions etc., must be able to dexterously and adaptively manipulate objects in a nonstationary task workspace. Redundancy in the design of robot manipulators has been suggested as one means to enhance their dexterity and adaptability. In contradistinction to other engineering contexts where redundancy implies fault-tolerance or superfluity, redundancy in robotics is determined relative to the task [4]. It refers to a manipulator with more than the minimum number of degrees of freedom necessary to accomplish general tasks. The major objective motivating introduction of redundancy in robot design and control is to use the additional degrees of freedom to improve performance in complex and unstructured environments. It helps overcome kinematic, mechanical and other design limitations of non-redundant manipulators. Also, the extra degrees of freedom can be used during real-time manipulator operation to simultaneously achieve end-effector trajectory control while satisfying additional constraints.

There are two primary goals in developing control strategies which take advantage of redundancy. First, given the initial and final end-effector task coordinates, simultaneously generate, in real time, a Cartesian-space trajectory that enables the robot to achieve a goal (*the path planning problem*) and a set of joint space configurations, which cause the end-effector to follow the desired trajectory *(inverse kinematics problem)* while satisfying additional constraints, such as obstacle avoidance, servo-motor torque minimization and joint limit avoidance. Developing algorithms to use the additional degrees of freedom to satisfy constraints is known as the ***redundancy resolution problem*** [1,4,7,16]. Secondly, provide adaptive mechanisms for responding to any unforeseen changes in the workspace or the manipulator geometry. Despite a tremendous growth in research activity on "model-based" adaptive control algorithms, the above problems entail a level of computational and paradigmatic complexity far exceeding that which can be provided by the existing strategies.

Artificial neural networks on the other hand, defined as massively parallel, adaptive dynamical systems modeled on the general features of biological networks, interact with the objects of the real world and its

[§] Robotics Research Lab., Dept. of Computer Science, Louisiana State University, Baton Rouge, LA 70803.

statistical characteristics in the same way as living beings do. The potential advantages of neuronal processing arise as a result of their ability to perform parallel, asynchronous and distributed information processing in a dynamic self-organizing manner typical of living systems. Neurons with simple properties and interacting according to relatively simple rules can accomplish collectively complex functions such as generalization, error correction, pattern classification, learning etc. However, their paradigmatic strength for potential applications, which require solving intractable computational problems or adaptive modeling, arises from a spontaneous emergent ability to achieve ***functional synthesis*** and thereby learn topological mappings [8] i.e., establish relationships between multiple continuous-valued inputs and outputs, based on a presentation of a large number of examples. Once the underlying invariances have been learned and encoded in the strengths of the synaptic interconnections, the neural network can generalize to solve arbitrary problem instances. In addition, the operational versions of these trained networks can be dynamically "regularized" at run-time to satisfy one or more task-specific constraints, without any explicit retraining or reprogramming. Since the inverse mappings are acquired from real-world examples, network functionality is not limited by assumptions regarding parametric or environmental uncertainty [3]. Thus, neural networks provide an attractive alternate algorithmic basis towards designing real-time manipulator control architectures for automating "man-out-of-the-telerobot-loop" tasks beyond the existing technology. In this paper we introduce a powerful neural learning methodology for addressing the inverse kinematics problem commonly encountered during the design of real-time, adaptive systems operating in redundant environments.

2. MANIPULATOR INVERSE KINEMATICS

A forward kinematic function, Φ, is defined as a nonlinear differentiable function which uniquely relates a set of N_Q joint variables, $\bar{q}$, to a set of N_X task-space coordinates, $\bar{x}$: $\bar{x} = \Phi(\bar{q})$. For serial chain robot manipulators the forward kinematic function is easily derived [11]. The more difficult problem, which is of primary practical interest in robot manipulation, is the inverse transformation: $\bar{q} \quad = \quad \Phi^{-1}(\bar{x})$. In other words, determine one or more sets of joint configurations which take the end-effector into a desired task position and orientation in the operational workspace. While the inverse kinematic function is highly nonlinear, closed-form analytical solutions can be found for a number of non-redundant manipulators with special architecture. Complete positioning capability in Cartesian space can be nominally achieved by using only six degrees of freedom. However, most manipulators have degenerate configurations, or kinematic singularities, near which small displacements of the end-effector require physically unrealizable joint speeds. These singularities effectively lead to a loss of usable workspace and capability, and there is a strong incentive to design robots with additional degrees of freedom to overcome this and other problems. However, incorporation of redundancy injects additional complexity into the inverse kinematic problem. For redundant manipulators, the inverse kinematics problem has an infinity of solutions, which can be mapped into a finite set of manifolds [4].

Because of this infinity of solutions, many redundant manipulator investigators have chosen to focus on the instantaneous or differential kinematics [15], in which the instantaneous end-effector velocity is related to the instantaneous joint velocities by the manipulator Jacobian matrix. For redundant robots the manipulator Jacobian is not uniquely invertible, and pseudo-inverse techniques can be used to select a solution from the infinity of possible solutions. But pseudo-inverse resolution techniques are generally not-cyclic, i.e., do not generate closed joint-space trajectories corresponding to closed end-effector trajectories, thereby posing a serious limitation for practical implementations. So, in the absence of satisfactory closed-form solutions, offline iterative approximation techniques based on "local-methods" have been used, e.g., "augmented task method" proposed by Goldenberg et al. [5]. The latter however suffers from algorithmic singularities and is computationally prohibitive for manipulators with large degrees of freedom. In addition, the existing algebraic and geometric strategies provide limited mechanisms for resolution of kinematic redundancy with respect to multiple criteria [3] and have little susceptibility to unforeseen changes in the workspace or the manipulator geometry, etc. Since no mechanisms are provided for resolving redundancy over more than one internal self-motion manifold, each different application requirement may entail reprogramming the control algorithm, thereby severely limiting manipulator functionality.

In contrast, neuromorphic approaches to the inverse kinematics problem entail systems composed of many simple processors ("neurons"), fully or sparsely interconnected, whose functions are determined by the topology and strength of the interconnections. The synaptic elements of such neural systems must capture the transcendental kinematic transformations by using *a priori* generated examples enabling subsequent generalization to other points in the workspace. Thus, the inverse transformation equations do not need to be explicitly programmed or derived. Once they have been learned, the network's inherent self-organizing abilities enable it to adapt to changes in the environment, e.g. planning joint trajectories in the presence of obstacles or to any unforeseen changes in the mechanical structure of the manipulator, with little effort [8]. Within a neuromorphic framework, a solution of the inverse kinematic problem involves two phases: a training phase and a recall phase. The training phase involves encoding the inverse mapping in the network's synaptic weight space, through repeated presentations of a finite set of *a priori* generated examples, linking Cartesian space end-effector coordinates to the corresponding joint angles. Once the network has acquired the nonlinear mapping imbedded within the training set, it can be used to rapidly recall, or generalize the joint configuration corresponding to any arbitrary Cartesian-space orientation within its workspace of training, thereby eliminating the intensive computational overheads that plague the existing iterative techniques. Also, once the training cycle is completed, the time required to obtain a solution practically depends in a weak fashion on the number of degrees of freedom. In the past, Josin [8], Guez et al. [6] and Tawel et al. [14] have applied this generic neuromorphic paradigm to the inverse kinematics problem for a 3-DOF redundant manipulator. In particular, they train a heteroassociative, multi-layered feed-forward neural network by using the backpropagation algorithm (for details refer to [13]).

Despite its conceptual simplicity, there are a number of non-trivial issues, both from the kinematics perspective and from the computational cost perspective that have hitherto limited the efficacy of such neuromorphic solutions to the inverse kinematics problem for redundant motion control. The major limitations, as discerned from the existing implementations, include an unacceptably large number of training iterations ($O(10^6)$ even for generalizing over small manifolds, see Tawel et al. [14]). Also the interpolated angular coordinates have relatively poor precision as compared to their algebraic or iterative counterparts. Besides, the backpropagation algorithm fails to efficiently scale-up to configurations with large number of degrees of freedom. For example, manipulators with seven or more degrees of freedom could not be satisfactorily trained by use of the standard back-propagation algorithm even after several million iterations. Furthermore, the back-propagation algorithm *per se* does not provide any intrinsic mechanism to simultaneously exploit redundancy to increase the task workspace (design constraints) and satisfy additional requirements inherent to operations in an unstructured environment such as obstacle avoidance in real-time. Since the latter flexibility is essential to the purpose of redundant manipulators [16], there is a strong incentive to develop an alternative neural network methodology that alleviates the above limitations to provide an efficient and accurate solution to the inverse kinematics problem.

3. NEURODYNAMICAL FORMULATION

3.1. Training Network Specification

Consider a fully connected neural network with N graded-response neurons, implementing a nonlinear functional mapping from the N_X-dimensional input space to the N_Q-dimensional output space. The network is topographically partitioned into three mutually exclusive regions comprising a set of input neurons, S_X, that receives the input coordinates, an output set, S_Q, which provides the output coordinates required to achieve the desired output, and a set of "hidden" neurons, S_H, that partially encode the input / output mapping. The network is presented with K randomly sampled training pairs of input-output, $\{ \bar{x}^k, \bar{q}^k \mid k = 1, \ldots, K\}$ obtained by solving the well-posed forward kinematics formulation (see Paul [11]).

The neuromorphic reformulation of the inverse kinematics problem requires determining synaptic interconnection strengths that can accurately capture the transcendental transformations imbedded within the training samples. Our approach is based upon the minimization of a constrained Hamiltonian ("neuromorphic energy"), given by the following expression:

$$E \;=\; \frac{1}{2K}\sum_{k=1}^{K}\left(\frac{1}{N_X}\sum_{l\in S_X}\,[\,u_l^k \;-\; x_l^k\,]^2 \;+\; \frac{1}{N_Q}\sum_{l\in S_Q}\,[\,u_l^k \;-\; q_l^k\,]^2\right) \;+\; \sum_r \lambda_r\, g_r(\cdot) \qquad (3.1.1)$$

where u_l^k denotes the l−th neuron's activity when processing the k−th training sample, $g_r(\cdot)$ reflects network constraints and the design considerations related to specific applications, e.g., singularity avoidance [4], obstacle avoidance [10], joint availability etc., and λ_r denotes the Lagrangian multiplier corresponding to the r−th application of design requirement. The proposed objective function therefore includes contributions from two sources. Firstly, it enforces convergence of every neuron in S_X and S_Q to attractors corresponding to the presented input-output coordinates and joint coordinates respectively for every sample pair in the training set, thereby enforcing the network to learn the underlying kinematic invariances. Secondly, it enforces the synaptic elements to satisfy network constraints of the type

$$g_r(\cdot) \;=\; \frac{1}{2}\,(i \;-\; j)^2\, T_{ij}^2$$

which minimize the interconnection strengths in line with the Gauss's least-constraint principle. For a more detailed treatment of redundancy resolution refer to [2,3]. We now proceed with the derivation of the learning equations (time evolution of the synaptic weights) by minimizing the energy function given in eqn. (3.1.1).

Lyapunov's stability criteria require an energy function to be monotonically decreasing in time. Since in our model the internal dynamical parameters of interest are the synaptic interconnection strengths, T_{nm}, and the Lagrange multipliers λ_r, this implies that

$$\dot{E} \;=\; \sum_n\sum_m \frac{\partial E}{\partial T_{nm}}\,\dot{T}_{nm} \;+\; \sum_r \frac{\partial E}{\partial \lambda_r}\,\dot{\lambda}_r \;<\; 0 \qquad (3.1.2)$$

One can choose

$$\tau_T\,\dot{T}_{nm} \;=\; -\frac{\partial E}{\partial T_{nm}} \qquad (3.1.3)$$

where τ_T is an arbitrary but positive time-scale parameter. Then substituting in Eqs. (3.1.2) we have

$$\sum_r \frac{\partial E}{\partial \lambda_r}\,\dot{\lambda}_r \;<\; \tau_T\,\dot{T} \oplus \dot{T}. \qquad (3.1.4)$$

In the above expression $\oplus$ denotes tensor contraction, i.e., $\dot{T} \oplus \dot{T} \;\equiv\; \sum_i \sum_j \dot{T}_{ij}\,\dot{T}_{ij}$. This will be true *a fortiori* if for some $\theta \;>\; 0$,

$$\sum_r \dot{\lambda}_r\,\frac{\partial E}{\partial \lambda_r} \;+\; \theta \;<\; \tau_T\dot{T} \oplus \dot{T} \qquad (3.1.5)$$

The equations of motion for the Lagrange multipliers λ_r must now be constructed in such a way that Eq. (3.1.4) is strictly satisfied. Noting that the analytic expression for the energy function results in $\frac{\partial E}{\partial \lambda_r} \;=\; g_r(\cdot)$, we adopt the following model:

$$\dot{\lambda}_r \;=\; \tau_T\,\frac{\dot{T}\oplus\dot{T} \;-\; \theta}{\bar{g}\oplus\bar{g} \;+\; \theta}\,g_r \qquad (3.1.6)$$

where $\bar{g} \oplus \bar{g} \;\equiv\; \sum_r g_r(\cdot)\, g_r(\cdot)$, and θ is an arbitrary positive constant. It is easy to see that $\dot{E} \;<\; 0$ is strictly satisfied. Also on differentiating Eqs. (3.1.1) with respect to T_{nm} we get

$$\frac{\partial E}{\partial T_{nm}} \;=\; \frac{1}{K}\sum_k\left\{\frac{1}{N_X}\sum_{l\in S_X}\,[\,u_l^k \;-\; x_l^k]\,\frac{\partial u_l^k}{\partial T_{nm}}\right.$$
$$\left. +\; \frac{1}{N_Q}\sum_{l\in S_Q}\,[\,u_l^k \;-\; q_l^k \;+\; N_Q\sum_r \frac{\partial g_r}{\partial u_l^k}\,]\,\frac{\partial u_l^k}{\partial T_{nm}}\right\} \qquad (3.1.7)$$

If we define

$$\hat{I}_l^k \;=\; \begin{cases} \frac{1}{KN_Q}\left[\, u_l^k \;-\; q_l^k \;+\; N_Q \sum_r \frac{\partial q_r}{\partial u_l^k} \right] & \text{if } l \in S_Q \\ 0 & \text{if } l \in S_H \\ \frac{1}{KN_X}\left[\, u_l^k \;-\; x_l^k \,\right] & \text{if } l \in S_X \end{cases} \tag{3.1.8}$$

then we can rewrite Eqn. (3.1.7) as

$$\frac{\partial E}{\partial T_{nm}} \;=\; \sum_l \sum_k \hat{I}_l^k \, \frac{\partial u_l^k}{\partial T_{nm}} \tag{3.1.9}$$

where the index l is defined over the entire set of neurons. Eqs. [3.1.3, 3.1.8 and 3.1.9] constitute a dissipative nonlinear dynamical system, the flow of which generally converges to a manifold of lower dimensionality in the phase space. In this paper we focus on network convergence to point attractors, i.e., state-space vector locations corresponding to the presented, joint- and Cartesian-space coordinates. Of crucial importance is to know how stable these attractors are, and, starting from arbitrary network configurations how fast they can be reached. In this vein, we first briefly review a novel mathematical concept in dynamical systems theory, the ***terminal attractor***, and its properties that subsequently will enable us to formalize efficient algorithms for learning the inverse kinematics mapping.

Artificial neural networks store memory states or patterns in terms of the fixed points of the network dynamics, such that initial configurations of neurons in some neighborhood, or ***basin of attraction***, of that memory state will be attracted to it [9]. But the static attractors considered so far in nonlinear dynamical system formulations in general, and in neural networks in particular, have represented regular solutions to the differential equations of motion. Such solutions can never intersect the transients. The theoretical relaxation time of the system to these "regular attractors" can theoretically be infinite, and they suffer from convergence to spurious states and local minima. The concept of ***terminal attractors*** in dynamical systems, was initially introduced by Zak [17,18] to obviate some of the above limitations, thereby significantly improving the performance characteristics of associative memory neural network models.

The existence of terminal attractors was established by Zak [17,18] using the following argument: At equilibrium, the fixed points, $\bar{p}$, of an N-dimensional, dissipative dynamical system

$$\dot{u}_i \;-\; f_i(u_1,\; u_2,\; ,\cdots,\; u_N\,) \;=\; 0 \qquad i \;=\; 1,\; 2,\; \cdots,\; N \tag{3.1.10}$$

are defined as its constant solutions $\bar{u}^{\infty}(\bar{p})$. If the real parts of the eigenvalues, η_μ, of the matrix $M_{ij} \;=\; \left[\frac{\partial f_i}{\partial x_j}(\bar{p})\right]$ are all negative, i.e., Re $\{\eta_\mu\} \;<\; 0$, then these points are globally asymptotically stable. Such points are called static attractors since each motion along the phase curve that gets close enough to $\bar{p}$, i.e., enters a so-called basin of attraction, approaches the corresponding constant value as a limit as $t \;\rightarrow\; \infty$. An equilibrium point represents a repeller if at least one of the eigenvalues of the matrix $\mathbf{M}$ has a positive real part. Usually, nonlinear neural networks deal only with systems which satisfy the Lipschitz condition, i.e., $\left|\, \frac{\partial f_i}{\partial u_j} \,\right| \;<\; \infty$. This condition guarantees the existence of a unique solution for each of the initial phase space configurations. That is why a transient solution cannot intersect the corresponding constant solution to which it tends, and therefore the theoretical time of approaching the attractors is always infinite.

In contrast, Zak's notion of terminal attractors is based upon the violation of the Lipschitz condition. As a result of this violation the fixed point becomes a singular solution which envelops the family of regular solutions, while each regular solution approaches the terminal attractor in finite time. To formally exhibit a terminal attractor which is approached by transients in finite time, consider the following one-dimensional example:

$$\dot{u} \;=\; -u^{1/3} \tag{3.1.11}$$

This equation has an equilibrium point at $u \;=\; 0$ at which the Lipschitz uniqueness condition is violated, since

$$\frac{d\dot{u}}{du} \;=\; -\frac{1}{3}u^{-2/3} \;\longrightarrow\; -\infty \;\; at \;\; u \;\longrightarrow\; 0 \tag{3.1.12}$$

Since here $\mathrm{Re}\{\eta\} \rightarrow -\infty < 0$, this point is an attractor with "infinite" local stability. As a consequence, the dynamical system is bestowed with "infinite attraction power", enabling rapid clamping of neuronal potentials to the fixed points; in this case the above phenomena imply immediate relaxation to the desired attractor coordinates, x_l and q_l. Also the relaxation time for the solution corresponding to initial conditions $u = u_0$ of this attractor is finite. It is given by

$$t_0 = - \int_{u_0}^{u \to 0} \frac{du}{u^{1/3}} = \frac{3}{2} u_0^{2/3} < \infty \tag{3.1.13}$$

i.e., this attractor becomes terminal. It represents a singular solution which is intersected by all the attracted transients. In particular static terminal attractors occur for $k = (2n+1)^{-1}$ and $n \geq 1$, while for $k = 2n+1$ all attractors are regular. It has been shown that incorporation of terminal attractor dynamics leads to the elimination of all spurious states. This property is critical to providing an accurate generalization ability during the operational phase. It ensures that interpolations / extrapolations of joint configurations are not based on false attractors, i.e., attractor coordinates not obtainable by the forward kinematics mapping. In our proposed neuromorphic framework, terminal attractor dynamics then provides a mechanism that can implicitly exploit the time-bounded terminality of phase trajectories and the locally infinite stability, thereby enabling an efficient and accurate solution to the manipulator inverse kinematics.

3.2. Virtual Attractor Computation

The Hamiltonian defined in Eqs. (3.1.1) specified the functionality of our fully connected neural network, i.e., learn the inverse kinematics mapping. We now need to select the network dynamics for evolving the synaptic elements, such that the latter's convergence to steady state leads towards the above function. So to capture the kinematic invariances consider the following neurodynamics.

$$\tau_u \dot{u}_l^k + u_l^k = \varphi_\gamma \, [\sum_{l'} T_{ll'} u_{l'}^k \,] - I_l^k \tag{3.2.1}$$

Here u_l represents the mean soma potential of the l-th neuron (u_l^k is the neuron's activity when processing the k-th training sample), $T_{ll'}$ denotes the synaptic coupling from the l'-th to the l-th neuron, and I_l^k captures the varying input/output contribution in a terminal attractor formalism. Though I_l^k influences the degree of stability of the system and the convergence to fixed points in finite time, it does not further affect the location of existing static attractors. And, $\varphi_\gamma(\cdot)$ denotes the sigmoidal neural response function with gain γ; typically, $\varphi_\gamma(z) = \tanh(\gamma \cdot z)$. In topographic maps N_T neurons are generally used to compute a single value of interest in terms of spatially-coded response strengths. Here we use the simplest possible model (where $N_T = 1$), but encode the information through terminal attractors. Thus, the topographic map is given by

$$I_l^k = \begin{cases} (\ u_l^k - x_l^k\ )^{1/3} & \text{if } l \in S_X \\ 0 & \text{if } l \in S_H \\ (\ u_l^k - q_l^k\ )^{1/3} & \text{if } l \in S_Q \end{cases} \tag{3.2.2}$$

where x_l^k and q_l^k are the attractor coordinates provided by the training sample, to be denoted for brevity as a_l^k. Our basic operating assumption for the dynamical system defined by (3.2.1) is that at equilibrium

$$\dot{u}_n \longrightarrow 0 \quad \text{and} \quad u_n \longrightarrow a_n$$

This yields the fixed point equations :

$$a_n = \varphi_\gamma \left[\sum_m T_{nm} \, a_m \right] \tag{3.2.3}$$

In associative memory applications, these equations can in principle be used to determine the synaptic coupling matrix T, resulting in each memory pattern being stored as a fixed point. The key issue is that some of these fixed points may actually be repellers. The terminal attractors are thus used to guarantee that each fixed point becomes an attractor, i.e., spurious states are suppressed. In this case however, we are in the process of learning a continuous mapping between two spaces and attractor coordinates have been defined for only two of the three topographic regions of the network, i.e., the input set S_X, and the output set S_Q. Consequently, the fixed point equation $\bar{a} = \varphi(T\bar{a})$ may not necessarily be defined, since for $| S_H | > 0$, $\{ a_n \mid n \in S_H \}$ are not defined, and cannot be used for directly computing T.

This necessitates a strategy whereby ***virtual attractor*** coordinates are first determined for the hidden units. These coordinates are virtual since they correspond to a current estimate $\hat{T}$ of the synaptic connectivity matrix. This is achieved by considering the fixed point equations as ***adaptive conservation equations*** which use the extra degrees of freedom made available by the hidden neurons in S_H. Let $\{ \hat{u}_j = a_j \mid j \in S_H \}$ denote the virtual attractors to which the unknowns, $\{ u_j \mid j \in S_H \}$, are expected to converge. Then at equilibrium Eqn. (3.2.3) yields

$$\begin{aligned}
\varphi^{-1}(x_i) &= \sum_{i' \in S_X} \hat{T}_{ii'} x_{i'} + \sum_{j' \in S_H} \hat{T}_{ij'} \hat{u}_{j'} + \sum_{l' \in S_Q} \hat{T}_{il'} q_{l'} \qquad \forall i \in S_X \\
\varphi^{-1}(\hat{u}_j) &= \sum_{i' \in S_X} \hat{T}_{ji'} x_{i'} + \sum_{j' \in S_H} \hat{T}_{jj'} \hat{u}_{j'} + \sum_{l' \in S_Q} \hat{T}_{jl'} q_{l'} \qquad \forall j \in S_H \\
\varphi^{-1}(q_l) &= \sum_{i' \in S_X} \hat{T}_{li'} x_{i'} + \sum_{j' \in S_H} \hat{T}_{lj'} \hat{u}_{j'} + \sum_{l' \in S_Q} \hat{T}_{ll'} q_{l'} \qquad \forall l \in S_Q
\end{aligned} \tag{3.2.4}$$

where $\hat{T}_{jl}$ denotes the current estimate of synaptic coupling from l-th neuron to the j-th neuron, and $\hat{u}_j$ represents a virtual attractor whose value is isomorphic to the current level of knowledge in the network. Now define

$$\begin{aligned}
\psi_i &= \varphi^{-1}(x_i) - \sum_{i'} \hat{T}_{ii'} x_{i'} - \sum_{l'} \hat{T}_{il'} q_{l'} \qquad \forall\, i \in S_X \\
\psi_j &= \sum_{i'} \hat{T}_{ji'} x_{i'} + \sum_{l'} \hat{T}_{jl'} q_{l'} \qquad \forall\, j \in S_H \\
\psi_l &= \varphi^{-1}(x_l) - \sum_{i'} \hat{T}_{li'} x_{i'} - \sum_{l'} \hat{T}_{ll'} q_{l'} \qquad \forall\, l \in S_Q .
\end{aligned} \tag{3.2.5}$$

Then consistency with the terminal attractor dynamics assumptions requires that $\{ \hat{u}_j \mid j \in S_H \}$ be simultaneous solutions to the following "conservation" equations

$$\begin{aligned}
\sum_{j' \in S_H} \hat{T}_{ij'} \hat{u}_{j'} &= \psi_i \qquad \forall\, i \in S_X \\
\varphi^{-1}(\hat{u}_j) - \sum_{j' \in S_H} \hat{T}_{jj'} \hat{u}_{j'} &= \psi_j \qquad \forall\, j \in S_H \\
\sum_{j' \in S_H} \hat{T}_{lj'} \hat{u}_{j'} &= \psi_l \qquad \forall\, l \in S_Q
\end{aligned} \tag{3.2.6}$$

The above system of equations for $\hat{u}$ is generally overdetermined. A number of standard algorithms exist to obtain a good approximate solution to such a system. In our implementation we use an iterative approach (e.g. conjugate gradient descent) to minimize the function

$$\begin{aligned}
\hat{E} = {} & \frac{1}{2N_X} \sum_i \Big(\psi_i - \sum_{j'} \hat{T}_{ij'} \hat{u}_{j'} \Big)^2 + \frac{1}{2N_H} \sum_j \Big(\hat{u}_j - \varphi[\sum_{j'} \hat{T}_{jj'} \hat{u}_{j'} + \psi_j] \Big)^2 \\
& + \frac{1}{2N_Q} \sum_l \Big(\psi_l - \sum_{j'} \hat{T}_{lj'} \hat{u}_{j'} \Big)^2
\end{aligned} \tag{3.2.7}$$

We can now return to the computation of $\partial u_l^k \,/\, \partial T_{nm}$ in Eq. (3.1.9). Let us define $z_l^k \;=\; \sum_{l'} T_{ll'}\, u_{l'}$ and denote $\varphi'_{lk} \;=\; \frac{\partial \varphi(\cdot)}{\partial z_l^k}$. Then at equilibrium, as $\dot{u}_l^k \longrightarrow 0$ and $I_l^k \longrightarrow 0$, we have

$$\frac{\partial u_l^k}{\partial T_{nm}} \;=\; \varphi'_{lk} \left[\sum_{l'} \frac{\partial T_{ll'}}{\partial T_{nm}}\, u_{l'}^k \;+\; \sum_{l'} T_{ll'}\, \frac{\partial u_{l'}^k}{\partial T_{nm}} \right] \tag{3.2.8}$$

which can be rewritten as

$$\sum_{l'} [\, \delta_{ll'} \;-\; \varphi'_{lk}\; T_{ll'}\,]\, \frac{\partial u_{l'}^k}{\partial T_{nm}} \;=\; \varphi'_{lk} \delta_{ln} u_m^k \tag{3.2.9}$$

In the above expression δ_{ij} denotes the Kronecker symbol. We now define, following [12], a weighted coupling matrix $A_{ll'}^k \;=\; \delta_{ll'} \;-\; \varphi'_{lk}\; T_{ll'}$. Then substituting $A_{ll'}^k$ in (3.2.9), and premultiplying both sides with $[A^{-1}]_{ln}^k$ and summing over l yields

$$\sum_l [A^{-1}]_{pl}^k\, A_{ll'}^k\, \frac{\partial u_{l'}^k}{\partial T_{nm}} \;=\; \sum_l [A^{-1}]_{pl}^k\, \varphi'_{lk}\; \delta_{ln}\; u_m^k. \tag{3.2.10}$$

Carrying out the algebra and relabeling the dummy indices results in

$$\frac{\partial u_l^k}{\partial T_{nm}} \;=\; [A^{-1}]_{ln}^k\; \varphi'_{nk}\; u_m^k. \tag{3.2.11}$$

The above expression can now be substituted in Eq. (3.2.10); the learning equation thus takes the form

$$\tau_T \dot{T}_{nm} \;=\; -\sum_l \sum_k \hat{I}_l^k\; [\; A^{-1}\;]_{ln}^k\; \varphi'_{nk}\; u_m^k \tag{3.2.12}$$

where the indices l and k run over the complete sets of neurons and training samples.

3.3. Adjoint Network Dynamics

A computation of the synaptic interconnection matrix as suggested by Eq. (3.2.12) would involve a matrix inversion. Since direct matrix inversion is typically nonlocal, we adopt the relaxation procedure suggested by Pineda [12] to compute the synaptic updates defined by (3.2.12). Consider the following change of variable

$$v_n^k \;=\; \sum_l [\; A^{-1}\;]_{ln}^k\; \hat{I}_l^k\; \varphi'_{nk} \tag{3.3.1}$$

Then substituting (3.3.1) in (3.2.12) we have

$$\sum_n A_{np}^k\, \frac{v_n^k}{\varphi'_{nk}} \;=\; \hat{I}_p^k \tag{3.3.2}$$

One can also use the explicit form of A_{np}^k and by substitution in (3.2.12), we obtain

$$\sum_n A_{np}^k\, \frac{v_n^k}{\varphi'_{nk}} \;=\; \frac{v_p^k}{\varphi'_{pk}} \;-\; \sum_n T_{np}\, v_n^k \tag{3.3.3}$$

Regrouping the previous equations and (3.3.2), and relabeling the dummy indices yields

$$v_n^k \;=\; \varphi'_{nk} \cdot [\, \sum_p T_{pn} v_p^k \;+\; \hat{I}_n^k\,]. \tag{3.3.4}$$

where τ_v denotes the relaxation constant. We see that v_n^k represents a fixed point solution of an "adjoint" neural network having the following coupled dynamics:

$$\tau_v \dot{v}_n^k + v_n^k = \varphi'_{nk} \cdot [\sum_p T_{pn} v_p^k + \hat{I}_n^k] \tag{3.3.5}$$

By comparing Eqs. (3.2.12, 3.3.1 and 3.3.5) we see that the resulting neural learning equations couple the terminal attractor dynamics for u_m^k with the adjoint dynamics for v_n^k, i.e.,

$$\tau_T \dot{T}_{nm} = - \sum_k v_n^k u_m^k \tag{3.3.6}$$

During run-time, i.e., after the kinematic invariances have been learned, the above neurodynamics can be used to generate joint-configurations corresponding to arbitrary task end-effector positions, with two primary modifications. Firstly, in the operational phase, terminal attractor coordinates are specified only on the input neurons. Secondly, adaptive virtual attractor computation is no longer required. The pseudo-code for the complete neural learning algorithm, criteria for selecting different time-scales and the results of our simulation on 3-dof and 7-dof redundant manipulators are reported in [3,19].

4. CONCLUSIONS

In this paper we have attempted to address a complex problem in robotics research, which enables the enhancement of manipulative capability and reliability. Our novel learning paradigm for neural network models, based on the terminal attractor concept, is shown [19] to be computationally competitive with iterative methods currently used in robotics to solve the inverse kinematics of redundant manipulators. In addition, this strategy does not appear to suffer from non-cyclicity of motion, as encountered in the pseudo-inverse resolution techniques, or the algorithmic singularities common to augmented task approaches. Furthermore, unlike the feed forward, backpropagation neural learning approaches, the adaptive dynamical system formulation presented here provides the flexibility for incorporating arbitrary combinations of kinematic optimization criteria, without imposing high computational overheads. Two options are available for including the redundancy resolution criteria in the algorithm to resolve the nonuniqueness of joint configurations that may satisfy a given end-effector configuration. The constraints may either be included *a priori* , i.e., while generating the training samples themselves, thereby forcing the network to learn only limited aspects of inverse kinematics mapping with a bias towards a particular criterion [2,3]; or they could be selectively applied in real-time to an operational version of the network (trained to encode the emergent invariants of the inverse kinematic mapping), to regularize the solutions (i.e. provide unique best answers) [3].

Despite the emphasis on real-time performance, the dexterous nature of applications envisaged for the next-generation robots imposes uncompromising demands on the resultant end-effector trajectory. Consequently, this entails the generation of intermediate joint angles with a high degree of precision, currently achievable only through off-line programming techniques (e.g., acceptable error tolerances are below 0.05%). In this context, our future directions for research include development of true neural topographic map techniques, enabling the much higher resolution needed to achieve the desired precision in interpolated joint angles.

ACKNOWLEDGEMENTS

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. The authors wish to acknowledge fruitful discussions with Michail Zak, Ken Kreutz and Joel Burdick.

REFERENCES

[1] J. Baillieul, "Kinematic Programming Alternatives for Redundant Manipulators ", in *Proc. IEEE Int'l Conf. on Robotics and Automation* , San Francisco, April 1986, pp. 1698-1704.

[2] J. Barhen, S. Gulati and M. Zak, "Neural Learning Algorithm for Inverse Kinematics of Redundant Manipulators in Unstructured Environments", *IEEE Computer* , Vol. 22, June 1989 (in press).

[3] J. Barhen and S. Gulati, "Neuromorphic Formalisms for Resolution of Kinematic Redundancy" submitted for publication.

[4] J.W. Burdick IV, " Kinematic Analysis and Design of Redundant Robot Manipulators," *Ph. D. Thesis,* Dept. of Mech. Eng., Stanford Univ., 1988.

[5] A.A. Goldenberg, B. Benhabib and R.G. Fenton, "A Complete Generalized Solution to the Inverse Kinematics of Robots," *IEEE Journal of Robotics and Automation* , Vol. RA-1, No. 1, March 1985, pp. 14-20.

[6] A. Guez, "Solution to the Inverse Kinematics Problem in Robotics by Neural Networks," in *Proc. of 2nd Int'l Conf. on Neural Networks* , San Diego, 1988, Vol. 2, pp. 617-624.

[7] J.M. Hollerbach and K.C. Suh, " Redundancy Resolution of Manipulators Through Torque Optimization," in *Proc. of IEEE Int'l Conf. on Robotics and Automation* , St. Louis, 1985, pp. 1016-1021.

[8] G. Josin, " Integrating Neural Networks with Robots ", *AI Expert* , Vol. 3, 8, 1988, pp. 50-60.

[9] J.D. Keeler, " Basins of Attraction of Neural Network Models ", *Proc. of AIP Conference* , Vol. 151, Neural Networks for Computing, Snowbird, UT, 1986, pp. 259-265.

[10] A.A. Maciejewski and C.A. Klein, "Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments", *Int'l Journal on Robotics Research* , Vol. 4, No. 3 , 1985, pp. 109-117.

[11] R.P. Paul, *Robot Manipulators : Mathematics, Programming and Control* , Cambridge, Mass., MIT Press, 1981.

[12] F.J. Pineda, "Generalization of Back-Propagation to Recurrent Neural Networks," *Physical Review Letters* , Vol. 59, No. 19, 1987, pp. 2229-2232.

[13] D.E. Rumelhart, J.L. McClelland and the PDP Research Group, *Parallel Distributed Processing* , Vol. I, Cambridge, Mass.: MIT/Bradford Book, 1986.

[14] R. Tawel, S. Eberhardt and A.P. Thakoor, "Neural Networks For Robotic Control," in *Proc. Conf. on Neural Networks for Computing* , Snowbird, Utah, 1988.

[15] D.E. Whitney, " Resolved Motion Rate Control of Manipulators and Human Prosthesis," *IEEE Trans. on Man-Machine Systems* , Vol. MMS-10, 1969, pp. 47-53.

[16] T. Yoshikawa, " Analysis and Control of Robot Manipulators with Redundancy ", *Robotics Research: First Int'l Symp.*, ed. M. Brady and R. Paul, Cambridge, Massachusetts: MIT Press, 1984, pp. 735-748.

[17] M. Zak, " Terminal Attractors for Addressable Memory in Neural Networks," *Physics Letters A*, Vol. 133, No. 1,2, 1988.

[18] M. Zak, " Terminal Attractors in Neural Networks", *Int'l Journal on Neural Networks*, Vol. 2, 1989 (in press).

[19] J. Barhen and S. Gulati, "Self-Organizing Neuromorphic Architecture for Manipulator Inverse Kinematics", *NATO-ASI Series*, Vol. F44, 1989 (in press).