N93-25964

Parametric Motion Control of Robotic Arms: A Biologically Based Approach Using Neural Networks

O. Bock
Institute for Space and Terrestrial Science, and York University

G.M.T. D'Eleuterio
Institute for Aerospace Studies, University of Toronto

J. Lipitkas
Institute for Space and Terrestrial Science
and, Institute for Aerospace Studies, University of Toronto

J.J. Grodski
Defence and Civil Institute of Environmental Medicine

January 19, 1993

1 Abstract

A neural network based system is presented which is able to generate point-to-point movements of robotic manipulators. The foundation of this approach is the use of prototypical control torque signals which are defined by a set of parameters. The parameter set is used for scaling and shaping of these prototypical torque signals to effect a desired outcome of the system. This approach is based on neurophysiological findings that the central nervous system stores generalized cognitive representations of movements called *synergies, schemas*, or *motor programs*. It has been proposed that these *motor programs* may be stored as torque-time functions in central pattern generators which can be scaled with appropriate time and magnitude parameters. The central pattern generators use these parameters to generate stereotypical torque-time profiles, which are then sent to the joint actuators. Hence, only a small number of parameters need to be determined for each point-to-point movement instead of the entire torque-time trajectory. This same principle is implemented for controlling the joint torques of robotic manipulators where a neural network is used to identify the relationship between the task requirements and the torque parameters. Movements are specified by the initial robot position in joint co-

PRECEDING PAGE BLANK NOT FILMED

ordinates and the desired final end-effector position in Cartesian coordinates. This information is provided to the neural network which calculates six torque parameters for a two-link system. The prototypical torque profiles (one per joint) are then scaled by those parameters. After appropriate training of the network, our parametric control design allowed the reproduction of a trained set of movements with relatively high accuracy, and the production of previously untrained movements with comparable accuracy. We conclude that our approach was successful in discriminating between trained movements and in generalizing to untrained movements.

2 Introduction

An important problem in space robotics is point-to-point control of the robotic arm end-effector in an unstructured environment. Many attempts have been made to solve this problem: the usual methods are tedious and computationally intensive to solve in real-time, even with the most advanced computational methods ([4], [11], [13]). This paper introduces a different strategy based on motor control principles used by humans.

In many studies on human movements, consistent and stereotypical hand and joint trajectories have been observed across movement speeds, extents, directions, and external loads. Such findings support the notion that movements are controlled by prototypical *motor programs* which are stored in the central nervous system and scaled to fit the requirements of each particular movement task before playback [1], [2], [5], [7], [12], [15], [16]. In particular, it has been proposed that these motor programs may be stored as muscle force-time functions and that different movements along the same path, but with varying speed or paylod, can be executed by playing back those functions with appropriate time and magnitude scaling. Therefore, the human motor system replaces the explicit calculation of the entire muscle-force profile by the calculation of just a few scaling parameters which are used to control central pattern generators (CPG) where the motor programs are stored.

A problem emerging from the motor program concept is that, since an infinite number of possible movements exist, the nervous system must have some way to calculate or to look up an infinite number of possible scaling parameters. Recently, engineering solutions for similar problems have been introduced in the form of artificial neural networks (ANN's [14]). Essentially, an ANN consists of processing elements, interconnection topologies, and a learning algorithm governing the modification of connection strengths depending on mapping performance. Generally, an ANN allows the mapping of input values into output values based on previously established mapping rules. These rules are determined via a repetitive trial-and-error *learning* procedure rather than by explicit calculations. An important characteristic of ANN's is that once a correct mapping has been learned for a number of input values, the network can *generalize* and provide correct output values even for untrained input values. Thus the above problem of representing an infinite number of parameters can be overcome by using neural networks to find suitable solutions.

To summarize, control by motor programs appears to be potentially useful for manipulator control because the controller would only have to calculate a limited number of scaling parameters before movement onset rather than calculating the entire joint torque-time profiles in real-time. This results in a robotic manipulator control system that can be referred to as a *Parametric Control System*, and is presented here as a means of controlling the joint torques of a two degree-of-freedom planar robotic manipulator. Furthermore, this approach is used in conjunction with a neural network which identifies the relationship between the task requirements and the torque parameters. Therefore, the approach presented here combines the motor program concept with neural networks to determine the joint torque-time functions necessary to drive a robotic manipulator end-effector from an initial to a desired final configuration.

3 Control Strategy

The control problem is to move a two-link planar robotic arm, as shown in Figure 1, from an initial position to a desired final position within the workspace shown. The robot manipulator control system, which was used, is designed to utilize the benefits of the motor program concept, and is illustrated in Figure 2. The adaptive controller is an ANN, trained to map inputs x_d, θ_a into outputs p. The parameters p are applied to a function generator which generates a prototypical time-function. This time-function is scaled by p to yield the force-time functions $T_d(t)$, one per joint, to be applied by the plant. In the work reported here, the plant is the *Radius* robotic manipulator at the University of Toronto Institute for Aerospace Studies [3]. This manipulator is a two degree-of-freedom planar manipulator with rigid links, where the links are supported by airjets in the horizontal plane. The airjets allow the *Radius* robotic manipulator to move in the horizontal plane without friction. The two joint actuators are harmonic-drive servomotors with the joint position θ_a being measured by precision potentiometers.

The ANN was implemented using the structure shown in Figure 3. Each neuron is a logistic unit having a working range of - 1.0 to + 1.0 with all of the neurons being fully forward-connected. Inputs to the ANN structure are x_d, the two Cartesian coordinates of the desired final gripper position, and θ_a, the actual initial angles of both joints, with θ_a and x_d being sampled once before a movement.

The input signals x_d and θ_a pass through a layer of 127 coarse-coding neurons [6] (each neuron being tuned to a range of input values with overlapping ranges for neighboring neurons), then through two hidden layers of 20 units per layer (the first layer containing 20 neurons and the second layer containing 20 neurons) and finally through a layer of six output neurons. The output signals provided by the last layer represent the values of the six parameters p which were previously described. Three of these parameters are used for each joint, p_1 to p_3 for the *shoulder* joint and p_4 to p_6 for the *elbow* joint.

The parameters p serve as inputs to the Function Generator (Figure 2), which in turn provides two output signals $T_d(t)$, one for each joint, which are applied to the plant. Both output signals are triggered synchronously when p changes after a new x_d has been entered, and each output signal consists of two successive sinusoidal half-waves having an overall duration of 4 sec. Figure 4 illustrates that p_1 and p_4 represent the percentage of movement time taken by the first lobe of the two torque profiles, one for each joint, and p_2, p_3, p_5, and p_6 represent the maximum torque amplitudes for each lobe of the two torque profiles.

The function of the ANN is, essentially, to map four discrete input signals x_d, θ_a for the two joints into six discrete output signals p_1, p_2, and p_3 (the torque parameters for the *shoulder* joint) and p_4, p_5, and p_6 (the torque parameters for the *elbow* joint). Only a single mapping action per movement is needed. The modifiable ANN weights are adjusted in order to achieve satisfactory mapping by a modified version of Direct Inverse Modeling [8], a known training procedure.

In this modified Direct Inverse Modeling training procedure, the initial *Radius* joint positions θ_a are first noted and an operator then moves the gripper into a selected final position x_s along an approximately straight path with an approximately bell-shaped, single-peaked velocity profile of 4 second duration. The joint trajectories $\theta(t)$ during that movement are recorded on a disk and subsequently transformed into predicted joint torque profiles $T_p(t)$ using *Radius's* Inverse Dynamics equations. Next, predicted joint torque profiles $T_p(t)$ are approximated by two sinusoidal half-waves of variable relative duration and amplitude and the corresponding parameters p are noted. Then, *Radius* having been reset to θ_a, p is provided as inputs to the function generator which supplies outputs $T_d(t)$ to the actuators in order to drive *Radius* to a final position noted as x_d. Since the parameterization is only an approximation, $T_d(t)$ and $T_p(t)$ will be somewhat different and x_d will be somewhat different from x_s.

The noted values of x_d, θ_a and p characterize one movement of a training set. The above steps are repeated for 225 different movements of various amplitudes and directions within the workspace shown in Figure 1 to yield a set of 225 training movements characterized by their respective values of x_d, θ_a and p.

Training of the ANN commences by initializing the modifiable weights with random values. Then x_d and θ_a of the training set are used as the ANN inputs and the corresponding outputs p are recorded. The difference between p as calculated by the ANN and the corresponding p as noted for the training set is the ANN performance error and is used for incremental weight changes according to the backpropagation rule. ANN performance is considered satisfactory when the output values p_1 to p_6, which are applied to the function generator, result in a gripper movement to the desired final position x_d such that $x_a \approx x_d$.

4 Results

An illustrative representation of network performance is given by Figure 5, where the final position error of the end-effector is plotted. The errors are coded as lines from the actual final position to the desired final position. Performance before training is shown in Figure 5A, and after training (10,000 iterations) in Figure 5B. As can be seen, the errors between the desired and actual final end effector positions are greatly reduced. In fact, the average error drops from 0.75 m before training, to 0.03 m after training: in comparison, the robotic arm is 2.12 m long. Therefore, the error after learning was almost an order of magnitude smaller than the inter-target distances which ranged from 0.1 m to 0.85 m. Thus, the system was able to discriminate between targets. Figure 5C shows the final position errors of the trained neural-network controller using a set of movements that were not previously trained. As can be seen, the average final position error of 0.07

m was slightly higher than the trained data set, but was again less than the shortest intertarget distance. Therefore, generalization within the workspace was successful.

5 Conclusions

We have described a solution to the control of point-to-point movements of a two joint planar robotic arm. This parametric control concept is qualitatively different from traditional approaches described earlier. Instead of explicitly calculating the torques for the entire trajectory, the new concept specifies only a limited number of characteristic parameters. In addition, the control system presented in this paper provides the following advantages over most other known types of systems:

1. No explicit knowledge of the manipulator's dynamics is required.

2. The nonlinear (ANN) stage is not in a control loop which will avoid any problems due to computational delays of the type generally caused by nonlinear stages.

3. The ANN can be easily retrained for different robotic manipulators and/or changing robot dynamics.

4. The design of a controller for a multi-link robotic manipulator with $n > 2$ is not qualitatively different than that described in this paper since the nonlinear stage is designed by trial-and-error rather that by an analytical solution.

In addition, our control concept discriminates between targets, and generalizes to unlearned target positions. This parametric control should be particularly useful for real-time robot control in unstructured environments since only a limited number of variables need to be updated, therefore placing less of a computational burden on the controller. Moreover, our control concept may be improved to achieve a more accurate terminal approach to the targets by the addition of sensory feedback, as found in humans. Also, this concept could be easily expanded to allow for velocity control by direct scaling of the torque profiles, and better control of movement paths could be achieved by adding more parameters (p_i).

6 Acknowledgments

We gratefully acknowledge the financial support provided by the Defence and Civil Institute of Environmental Medicine under contract W7711-0-7118. Further support was provided by the Institute for Space and Terrestrial Science, and the University of Toronto Institute for Aerospace Studies. In addition, we would like to thank the many employees and students in the Space Robotics Laboratory at the University of Toronto Institute for Aerospace Studies for their help.

References

[1] Atkeson C.G., Hollerbach J.M. (1985). *Kinematic features of unrestrained vertical arm movements.* J. Neurosci 5, 2318-2330.

[2] Bock O. (1990). *Load Compensation in human goal-directed arm movements.* Behav Brain Res 41, 167-177.

[3] Carusone J., D'Eleuterio G.M.T. (1991). *Experiments in the control of structurally flexible manipulators with the Radius facility.* Proc. of the Second Joint Japan/U.S. Conference on Adaptive Structures, Nagoya, Japan, 589-605.

[4] Freund E. (1982). *Fast Nonlinear Control with Arbitrary Pole-Placement for Industrial Robots and Manipulators.* Int J Robotics Res 1,65-78.

[5] Gordon J., Ghez C. (1987). *Trajectory control in targeted force impulse. II. Pulse height control.* Exp Brain Res 67, 241-252.

[6] Hinton G.E., McClelland J.L., Rumelhart D.E. (1986). Distributed Representations. In Rumelhart D.E., McClelland J.L. (Eds). *Parallel Distributed Processing: Vol 1* (pp 77-109). Cambridge:MIT Press.

[7] Hollerbach J.M., Flash T. (1982). *Dynamic interactions between limb segments during planar arm movements.* Biol Cybernetics 44, 67-77.

[8] Jordan M.I., Rumelhart D.E. (1991). *Forward models: Supervised learning with a distal teacher.* Submitted to Cognitive Science.

[9] Lacquaniti F., Soechting J.F., Terzuolo C.A. (1982). *Some factors pertinent to the organization and control of arm movements.* Brain Res 252, 394-397.

[10] Lipitkas J. (1992). *Organizational Principles Underlying Movements of the Upper-Limb.* unpublished Ph. D. thesis, Univ. of Toronto.

[11] Luh J.Y.S., Walker M.W., Paul R.P.C. (1980). *Resolved-Acceleration Control of Mechanical Manipulators.* IEEE Trans on Automatic Control 25, No. 3, 468-474.

[12] Meyer D.E., Smith J.E.K., Wright C.E. (1982). *Models for the speed and accuracy of aimed movements.* Pyschol Rev 89, 449-482.

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

[14] Rumelhart D.E., McClelland J.L. (1986). *Parallel Distributed Processing.* Cambridge:MIT Press.

[15] Schmidt R.A., Zelaznik H.N., Hawkins B., Frank J.S., Quinn Jr. J.T., (1979). *Motor output variability: a theory for the accuracy of rapid motor acts.* Psychol Rev 86, 415-451.

[16] Sherwood D.E., Schmidt R.A., Walter C.B. (1988). *Rapid movements with reversals in direction II: Control of movement amplitude and inertial load.* Exp Brain Res 69, 355-367.

[17] Stein R.B., Cody F.W.J., Capaday C. (1988). *The trajectory of human wrist movements.* J Neurophysiol 59, 1814-1830.

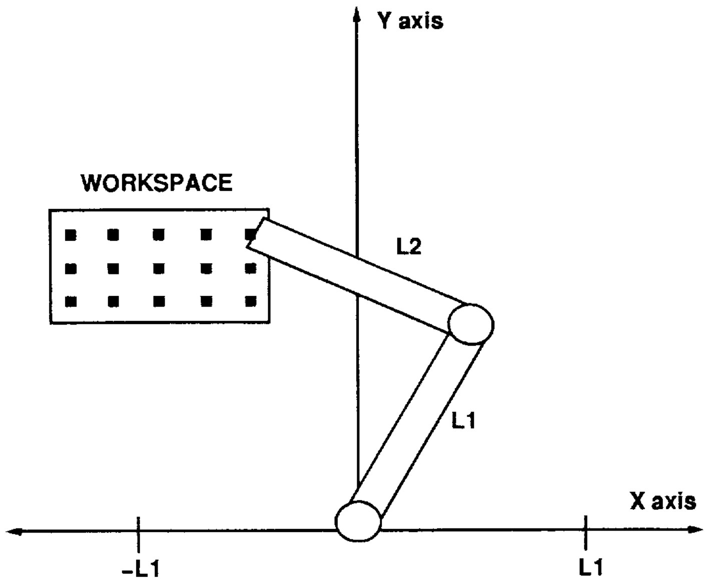

Figure 1: Two-link planar manipulator and workspace (L1 and L2 are the link lengths of the first and second links, where L1 = L2 = 1.06 m).

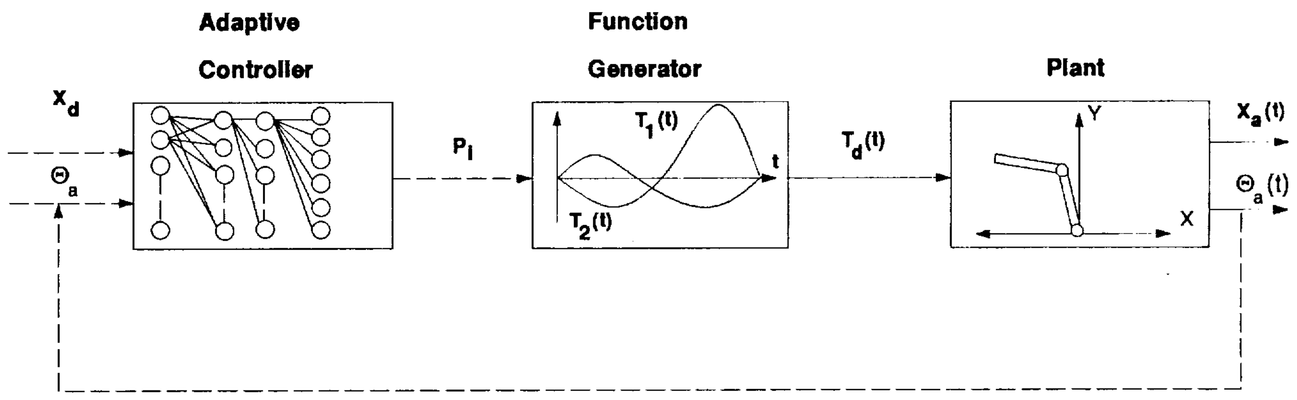

Figure 2: Block diagram of the control system utilized, where solid lines represent time varying actions and hatched lines represent a single mapping actions per movement (X_d and X_a are the desired and actual end-effector positions, θ_a the initial robot configuration in joint coordinates, $P_i's$ are the torque scaling parameters, $T_1(t)$ and $T_2(t)$ are the joint torque-time profiles for the shoulder and elbow joints, and $T_d(t)$ represents the input torques to the plant).

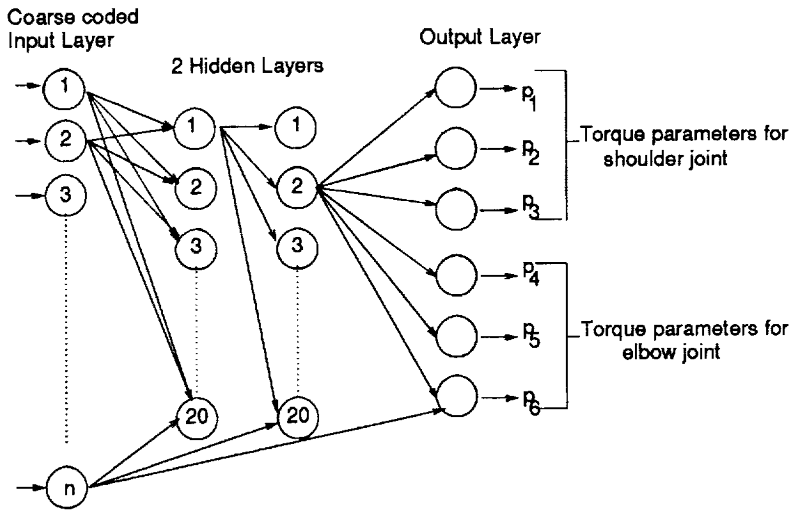

Figure 3: Artificial neural network architecture used in the simulations reported here (n = 127). All neurons are fully forward- connected to the neurons in the layers in front.

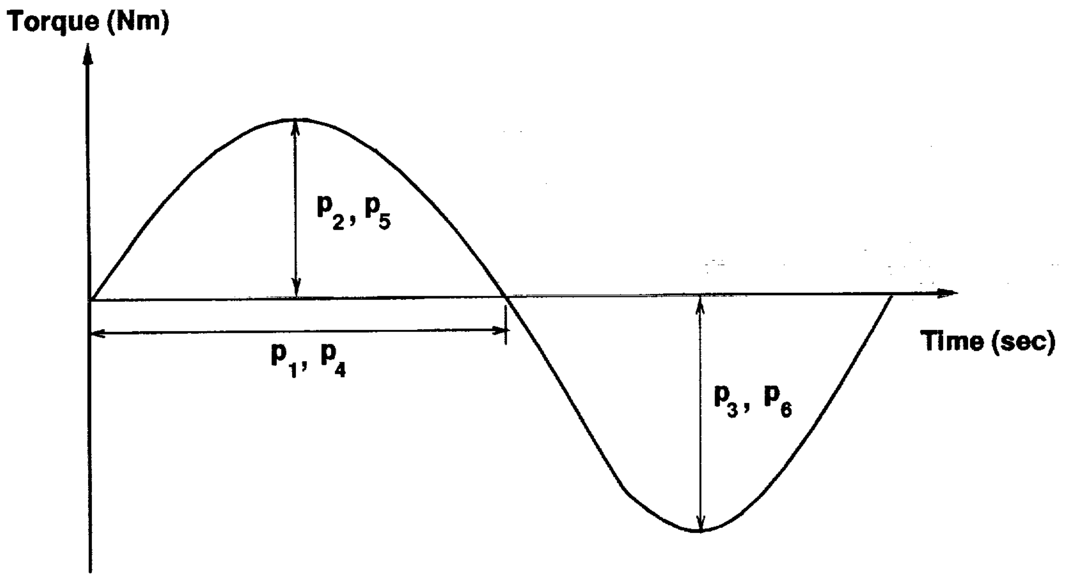

Figure 4: Torque parameterization scheme employed. Where p_1, and p_4 are the time of switching from the first lobe to the second lobe for torque profiles T_1 and T_2 respectively, p_2 and p_5 are the amplitudes of the first lobe, and p_3 and p_6 are the amplitudes of the second lobe for torque profiles T_1 and T_2.

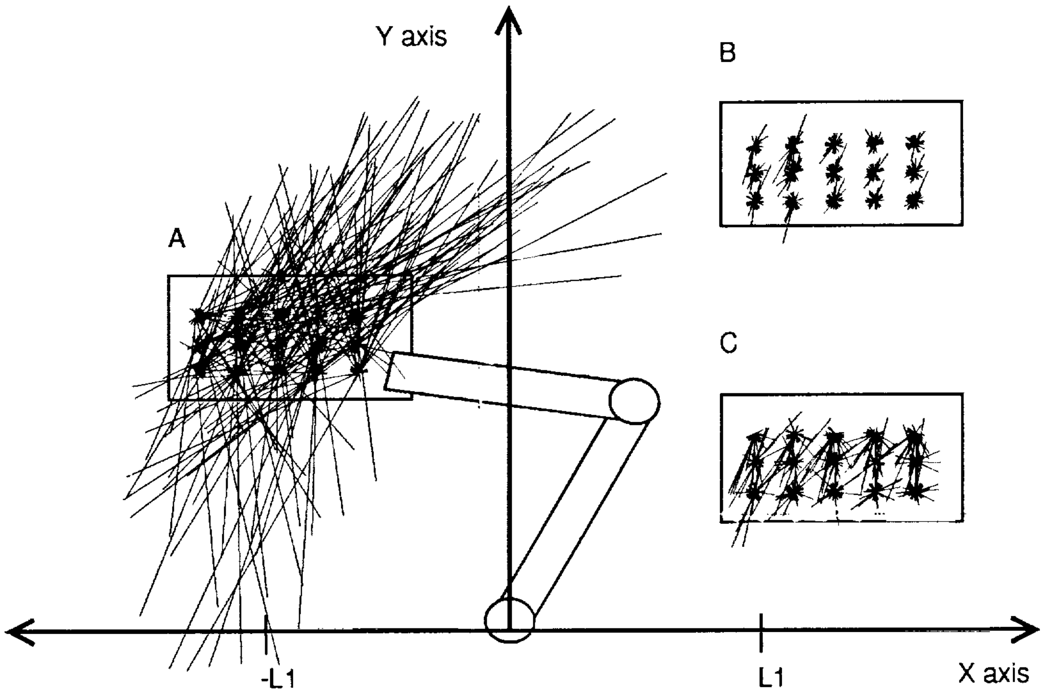

Figure 5: Graphical illustration of the final position errors from the actual to the desired final end-effector positions (A - final position errors before training, B - final position errors after training, for same workspace as A, C - final position errors for an untrained data set, for same workspace as A also).