01 Jan 1991

Hierarchical Neurocontroller Architecture for Intelligent Robotic Manipulation

Xavier J. R. Avula
Missouri University of Science and Technology, avula@mst.edu

Luis C. Rabelo

Proceedings of the 1991 IEEE
International Conference on Robotics and Automation
Sacramento, California - April 1991

HIERARCHICAL NEUROCONTROLLER ARCHITECTURE FOR INTELLIGENT ROBOTIC MANIPULATION

Luis C. Rabelo+ and Xavier J. R. Avula*

+Center for Technology Transfer
***Dept. of Mechanical and Aerospace Eng. and Engineering Mechanics**
University of Missouri-Rolla
Rolla, Missouri 65401

Abstract

In this paper, a hierarchical neurocontroller architecture consisting of two artificial neural networks systems for the manipulation of a robotic arm is presented. The higher level neural system participates in the delineation of the robot arm workspace, coordinates transformation and the motion decision making process. The lower one provides the correct sequence of control actions. The capabilities including speed, adaptability, and computational efficiency of the developed architecture is illustrated by an example..

1. Introduction

In recent years, there have been several attempts to apply artificial neural networks (ANN's) to the engineering problem of robotic control. Guez and Ahmad [6] presented a hybrid approach using a multi-layered feedforward network to the iterative solution of robotic manipulators which resulted in accelerated convergence in the inverse kinematics. Guo and Cherkassky [7] presented a solution algorithm, using an analog neural computational scheme to implement the Jacobian control technique in real time which is desirable in practical control problems. Sobajic et al. [16] investigated the control of a constrained robot manipulator using backpropagation, and showed that the manipulator could be moved toward a target in the presence of different disturbances. Bassi and Bekey [4] provided a simulated control strategy which indicated that it is practical to control a manipulator to an arbitrary degree of precision by using a neural network whose transformation has a relatively low precision. Liu et al. [11] employed an adaptive neural network in building a generic architecture for robot hand control. This architecture allows device-independent control and separates the low level control problems from high level functionality. In our study we have developed an architecture which can yield a strategy for dynamic decision-making that allows the robot end-effector to reach it's goal using apriori and on-line contextual information.

The artificial neural systems which are proposed in this wokr cooperate with the entire architecture to achieve the necessary flexibility to adapt to unforeseen changes in the robot workspace, environmental changes (e.g., temperature), coordinates transformation relationships, contextual rules (e.g., payload, direction of travel, stiffness), and interactions with other intelligent systems. In these artificial neural systems, the knowledge necessary to adapt to a new environment is learned from experience. This allows the implementation of self-organization strategies and, therefore, of the evolving systems.

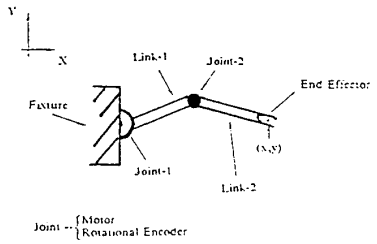
2. The Physical System

The computer simulated robotic manipulator model consists of a 2-D arm with two jointed links of equal length. Figures 1-a and 1-b illustrate the simulated robot arm and its abstract model. The workspace is constrained by the combined lengths of the two members (200 cm) in the simulation and by a maximum rotational displacement limit of 170 degrees at the elbow joint. The robotic manipulator's dimensions can also change due to the effects of temperature induced link expansion or contraction. The temperature is assumed to vary randomly in the range 25 C to 125 C. A coefficient of thermal expansion similar to that of aluminum is used.

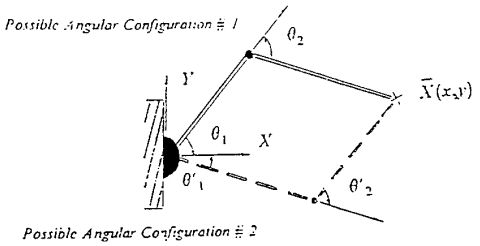
Due to the simplicity of the simulated manipulator, algorithmic relationships yield most of the feedback needed for the supervision of the learning of the different networks.

3. Artificial Neural Systems

In this paper, two different artificial neural systems associated with the prototype of a scheme which uses the integration of artificial neural networks and knowledge-based systems for robotics motion control is presented. These artificial neural systems participate in the tasks of



(a) Robot Arm Structure



(b) Possible Angular Positions

FIGURE 1. Modeled Robot Arm

the motion analysis process at the higher hierarchical level (see Fig. 2) and the process of control-emulation at the lower level. (see Figure 3)

The contributions of the artificial neural system at the higher hierarchical level to the motion analysis process is divided into three distinct tasks, each with its own associated neural network arrangement. The first part, consisting of the preliminary motion feasibility analysis, uses a Restricted Coulomb Energy (RCE) network [2,14] to delineate the robot-arm workspace and evaluate whether the end-effector could reach the proposed location. If the goal is feasible, a second system of artificial neural networks is employed to map the angular coordinates of the two possible robot configurations that are consistent with the end-effector. Standard Backpropagation (BP) is utilized for the coordinates transformation scheme. Special emphasis is placed on finding the best approach to obtain an efficient network architecture. Finally, the third network arrangement using BP analyzes both alternatives and select the most adequate one in accordance with the initial position. This third network makes an initial proposition to the motion decision making mechanism which uses the cooperation

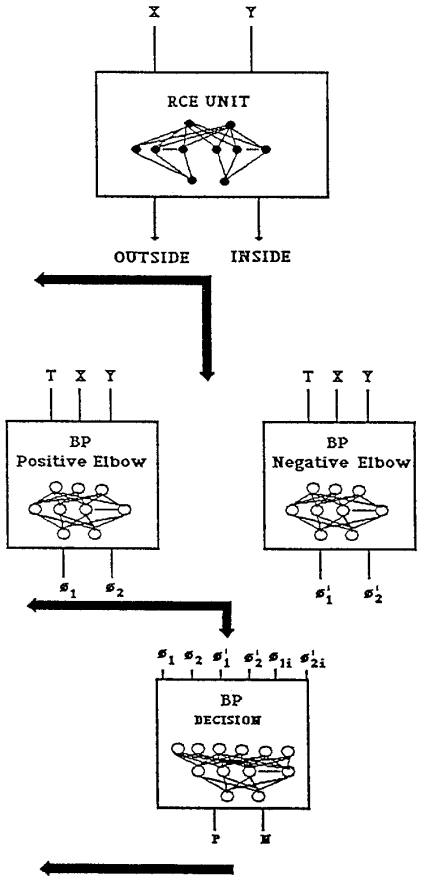


FIGURE 2. Artificial Neural System at the Higher Hierarchical Level

of other knowledge sources (e.g. knowledge bases, algorithms, procedures) with more contextual information leading to the final decision. This final decision will create a plan for the control actions which will be implemented by the lower elements of the hierarchy.

The process of control-emulation is implemented at the lower part of the hierarchy. This part is based on previous developments by Werbos [18], Barto et al. [3], Nguyen and Widrow [13], Jordan [9], and Kong and Kosko [10]. The problem to be solved is to provide the correct sequence of control actions to incite the robot arm to go from an initial position to a target position. This "correct sequence" of control actions is decided by a plan generated by the higher order element which includes the kind of desired trajectory to follow. In this study we have utilized forward modelling because the training data can

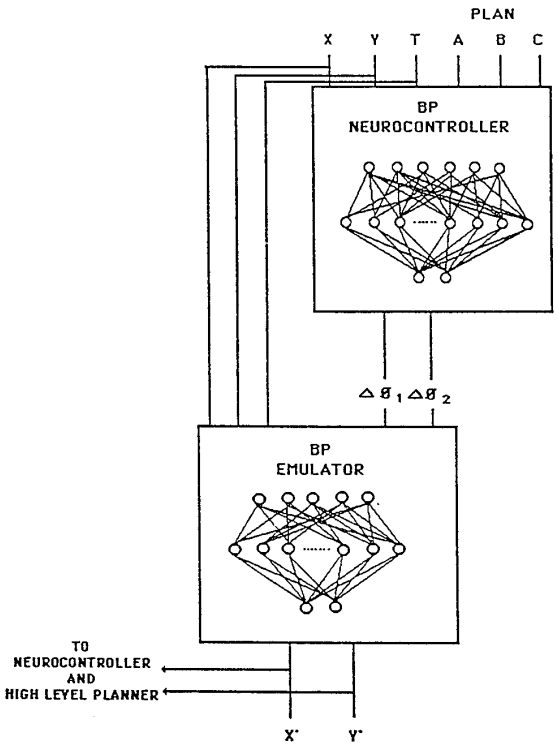


FIGURE 3. Artificial Neural System at the Lower Hierarchical Level

be easily generated. The forward model is taught by the high level planner using simulated sensor feedback and implemented in a BP network. Then, the emulator network of the robot arm dynamics is developed and the controller implementation is started. The controller is implemented using a BP network driven by the high level planner which takes the decision on the kind of trajectory to be followed (i.e., linear, circular, linear/circular). The frequency of this trajectory changes is totally handled by the high level planner. In addition, the controller receives input from the emulator at a higher frequency. The methodology which has been successfully used to train BP networks in the motion analysis networks is utilized to train the emulator-controller artificial neural system. This methodology is proven to be very efficient in the development of the required networks which especially involve large training data sets.

3.1 RCE Mapping of 2-D Robotic Arm Workspace

The NDS 500 network package from Nestor Inc. was used to obtain the workspace mapping. The training data is clustered according to the class boundaries to help the learning process. In addition, to reduce the level of overlapping of the cells in the internal layer, very small minimum influence fields were selected.

The RCE network used in this study was of the most liberal type since it was desired to prompt it to decide the categorization of the X-Y pair even at relatively high uncertainty levels due to the fact that a response was needed. The nearest neighbor approximation was used whenever an input vector fell outside the influence region of any hidden layer cell. A training data file consisting of 4500 points and a testing file of 499 points were utilized. The final network had 426 units. Of these, 115 had overlapping influence fields, while the rest possessed influence fields that were exclusive. The learning process showed a healthy input subspace growth. This growth, represented by the number of internal layer cells versus the number of training samples reached an asymptotic value as training was completed and the network converged. This is a good indication of the quality of the categorization (99.9%).

3.2 Robot Arm Coordinates Transformation

The coordinates transformation for the robot used in this research is not a one-to-one mapping. Two configurations, with a positive and a negative elbow rotation, are valid if no arbitrary constraints are placed on the solution space. Consequently, it is important to obtain both positions in order to develop a methodology for choosing one which satisfies the motion and contextual constraints.

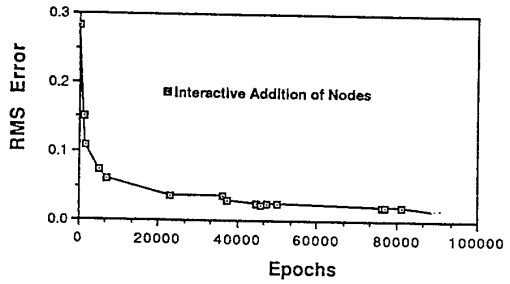
The system used to produce this transformation is based on two artificial neural networks, each responsible for one of the possible configurations. The mapping assigns a 3-D input vector consisting on the X and Y location of the end-effector as well as the temperature, T, (X, Y, T) to a 2-D vector containing the required shoulder and elbow angular position (ø1, ø2).

3.3 Mapping with Standard Backpropagation

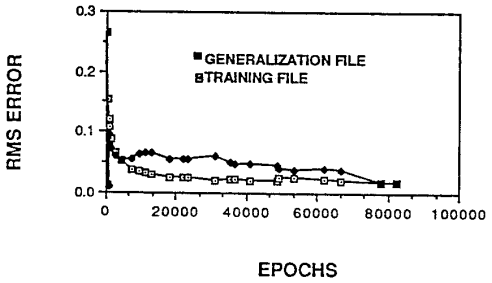
The artificial neural networks for coordinate transformation, configuration selection, arm emulator, and neurocontroller were developed using the standard propagation paradigm. [15]. The training process using BP is a difficult problem [1,5,8,12,15,17]. It is needed to find an appropriate architecture (e.g. number of hidden units, number of hidden layers, etc.), adequate size and quality of training data, satisfactory initialization learning parameter values (e.g., initial weights, learning rates), and avoid over-training effects (performance degradation due to prolonged training).

In this research the following the following approaches were utilized:

1. To help to find an appropriate architecture (i.e., number of hidden units) will be the utilization of Dynamic Node Creation (DNC). DNC is a methodology developed at the University of California-San Diego by Ash [1] which add nodes to the hidden layer(s) of the network during training (see Figure 4-a).



(a) Interactive Addition of Nodes During Training



(b) Utilization of a Validation ("Generalization") File

FIGURE 4. Learning Curves

2. To speed up the training time, the utilization of momentum is recommended [15]. Momentum is defined by

$$\mathbf{Wijm(t) = Wijm(t\text{-}1) + \Delta Wijm(t) + \alpha\, Wijm(t\text{-}1)}$$

Wijm = weight value located between nodes i and j at layer m
t = present iteration
Δ**W** = weight update
α = momentum value

The momentum is changed dynamically, because each problem has a range of optimal momentum values to avoid oscillations. In addition, dynamical adjustments of the learning rate are desired to speed up training [8].

3. To reinforce learning, Combined Subset Training (CST) was utilized [17]. CST combines old and new training sets. First, a random subset to train the network is selected. When the network has learned it fairly well, a new subset (of the same size as the previous one) is added to the first training set, and the network is trained with the combined set. If this can be learned successfully, the training set is doubled in the same fashion. This training method is for large and "uniform" training sets.

4. To avoid over-training effects, a validation file (a file with unseen data to make on-line tests about network performance was utilized [12] (see Fig. 4-b).

3.4 Configuration Selection

In this stage, an ANN based on BP is used. It has six inputs representing both possible configurations and the initial position of the arm, two outputs specifying the rank given to each configuration (The higher value identifies the winning combination of shoulder and elbow angles), and six hidden units (see Fig. 2). This network was developed using the training techniques described above.

3.5 Arm Emulator

The arm emulator is needed in order to identify the arm dynamic behavior. As expressed by Nguyen and Widrow [13], "This process is roughly analogous to the steps that would be taken by a human designer to identify the plant"and this "identification is done automatically by a neural network". The knowledge, in order to develop the emulator, should be encoded in the higher order element. The higher order element could examine the arm responses in the cartesian plane with different motor actions. This process was implemented repetitively until an efficient emulator was developed. Artificial neural networks has several advantages for this "automatic identification process" due to their non-linear modelling and learning capabilities.

An ANN is developed successfully using the techniques previously mentioned. This ANN has five inputs identifying the current X, Y cartesian positions, the elbow and shoulder angle increments (which could define the width of the motors drive pulses in our discrete-time model), and the temperature (see Fig. 3). The ANN has two outputs which correspond to the X, Y position (i.e., next state) after the movement has been performed. The ANN developed has 42 hidden units.

3.6 The Neurocontroller

The neurocontroller has the functions to provide a sequence of orders in order to drive the arm from an initial position to a target position. The emulator is used to teach the controller by the higher order element which provides the plans. The plans in this applications define the type of trajectory desired. The trajectory could be defined as linear or circular. The higher element decides according to sensory feedback or emulator responses to modify the plans given to the neurocontroller. To define the type of trajectory the coefficients of a line (B * x + C) or circle (A * x + B * x + C) are provided to the neurocontroller. The neurocontroller taking into consideration the present state vector (X, Y, T) and the plan (A, B, C) generates the

discrete increments for the elbow and shoulder angles (see Fig. 3). Then the "new" present state vector is used repeatedly till the targeted position is achieved. The necessary comparison is done by the higher element which will decide to send the STOP order A=B=C=0).

An ANN is developed using BP to implement the neurocontroller. This ANN has six inputs corresponding to X, Y, T, A, B, and C. The outputs correspond to the increments of the shoulder and elbow angles.

4. An Example

Here we consider an example to illustrate the integration of the different artificial neural networks toward the development and accomplishment of a motion task. The goal of this task is to drive the manipulator from the initial position X = 155.6 cm, Y = 109.0 cm, ø1 = 16.8, ø2=36.4, and a temperature T = 30 C to the final position X = 58.1 cm, Y = 159.7 cm at temperature T = 40 C using a linear trajectory.

After the RCE Unit declares that the targeted position is indeed inside of the workspace, the two ANNs developed for coordinate transformation mappings are executed.

A selection from the two possible configurations is made by using a BP network which takes into consideration the initial position. The high level planner develops the plan for the neurocontroller which in this example are the coefficients of the trajectory between the initial and the goal position (A = 0, B = -0.5, and C = 190). The artificial neural system at the lower hierarchical level will be monitored by the high level planner in order to decide when it is going to be reasonable to stop. Finally, the Neurocontroller drives the manipulator arm from the initial position to the final position using a quasi-linear trajectory. In spite of the temperature changes, the neurocontroller is able to adapt and the manipulator reaches a point near the final position. See Fig. 5 for details of the eight movements required by the systems in order to reach the targeted position.

5. Conclusions

Artificial neural networks (ANN's) have capabilities to learn, perform massively parallel processing, and to adapt to complex environmental changes. These capabilities are especially significant in robotics for providing enhanced systems with abilities of learning and self-organization, and efficient real-time operation. In this paper, we have demonstrated the utilization of several ANN's to support the robot motion decision analysis and drive a manipulator arm. The supervised learning models which are introduced here have the following capabilities:

* Perform automated modeling of workspaces in spite of their arbitrary shape,
* Support the decision-making using contextual information in order to provide an initial solution,
* Provide systems which are adaptable to environmental changes, and even constructional imperfections and other faults,
* Develop controllers which can create their own knowledge bases about the system dynamics.

The results indicate an advancement in the applicability of ANN's to robotics in particular, and to nonlinear control in general. The hierarchical structure introduced here significantly enhances the system capabilities because it takes into consideration not only the knowledge about the manipulator but also about the controller. Using arms with more than 2-D and 3D work envelopes are not limiting factors to this architecture due to its concurrent coordination capabilities. Further research will concentrate in the coordination of several robotic end-effectors and optimal strategies as functions of the planning horizon.

References

[1] T. Ash, "Dynamic Node Creation," Technical Report, ICS, University of California-San Diego, 1989.

[2] C. Bachman, L. Cooper, A. Dembo, O. Zeitouni, "A Relaxation Model for Memory with High Storage Density," Proc. of the National Academy of Sciences USA, Vol. 21, 1984, pp 2088-3092.

[3] A. Barto, R. Sutton, C. Anderson, "Neuronlike Adaptive Elements that Can Solve Difficult Learning Control Problems," IEEE Transactions on Systems, Man, and Cybernetics, SMC-13, pp 834-846.

[4] D. Bassi, G. Bekey, "High Precision Position Control by Cartesian Trajectory Feedback and Connectionist Inverse Dynamics Feedforward," Proceedings, International Joint Conference on Neural Networks (IJCNN), Washington, DC, 1989, pp. II-325-332.

[5] Y. Chauvin, "Dynamic Behavior of Constrained Back-Propagation Networks," Advances in Neural Information Processing Systems 2, (David Touretzky, ed) Morgan Kaufmann Publishers, 1990, pp 642.

[6] A. Guez, Z. Ahmad, "Accelerated Convergence in the Inverse Kinematics via Multi-Layered Feedforward Networks", Proceedings, IJCNN, Washington, 1989, pp. II-341-347.

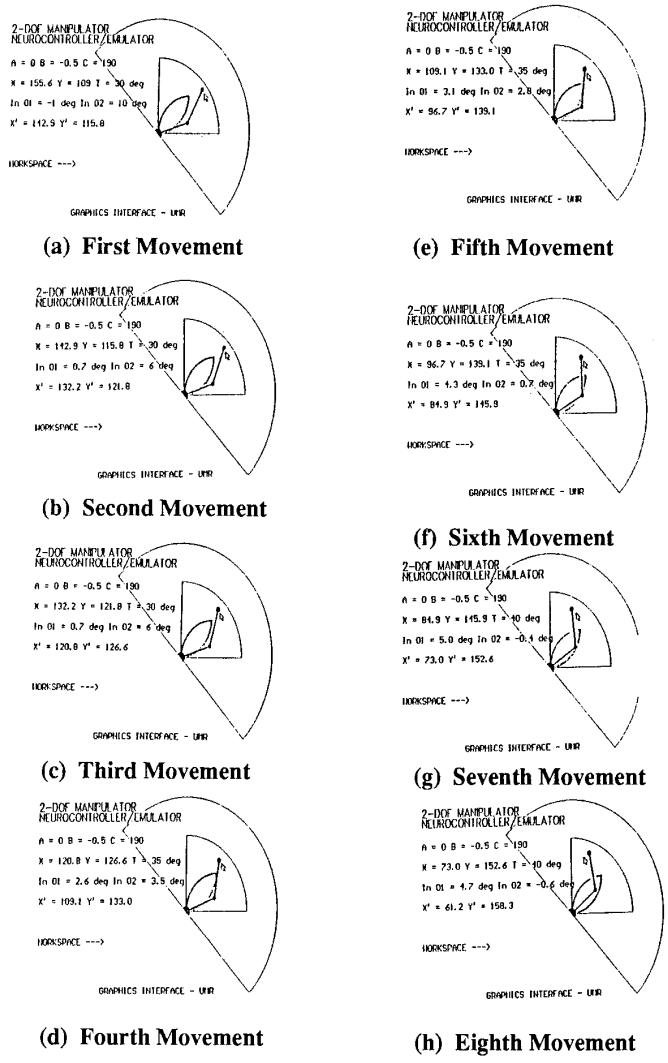
[7] J. Guo, V. Cherkassky, "A Solution to the Inverse Kinematic Problem in Robotics Using Neural Network Processing", Proceedings, IJCNN, Washington, DC, pp. II-299-304.

[8] R. Jacobs, "Increased Rates of Convergence Through Learning Rate Adaptation," Neural Networks, Vol.1 No.3, 1988, pp.295-307.

[9] M. Jordan, "Supervised Learning and Systems with Excess Degrees of Freedom," Proceedings of the 1988 Summer School on Connectionist Models (David Touretzy, ed.) pub. Morgan Kaufman, pp 62-75.

[10] S. Kong, B. Kosko, "Comparison of Fuzzy and Neural Truck Backer-Upper Control Systems," IJCNN Proceedings, San Diego 1990, pp III349-III358.

[11] H. Liu, T. Iberall, G. Bekey, "Neural Network Architecture for Robot Hand Control", IEEE Control Systems Magazine, April, 1989, pp 38-42.



(a) First Movement

(b) Second Movement

(c) Third Movement

(d) Fourth Movement

(e) Fifth Movement

(f) Sixth Movement

(g) Seventh Movement

(h) Eighth Movement

FIGURE 5. Neurocontroller/Emulator Execution

[12] N. Morgan, H. Bourlard, "Generalization and Parameter Estimation in Feedforward Nets: Some Experiments," International Computer Science Institute, TR-89-017, 1989.

[13] D. Nguyen, B. Widrow, "Neural Networks for Self-Learning Control Systems," IEEE Control Systems Magazine, Vol. 10 No. 3, 1990, pp 18 -23.

[14] D. Reilly, L. Cooper, C. Elbaum, "A Neural Model for Category Learning," Biological Cybernetics, Vol. 45, 1982, pp 35-41.

[15] D. Rumelhart, J. McClelland, and the PDP Research Group, PARALLEL DISTRIBUTED PROCESSING: Explorations in the Microstructure of Cognition, Vol. 1: Foundations, Cambridge, MA: MIT Press, 1988.

[16] D. Sobajic, J. Lu, Y. Pao, " Intelligent Control of the INTELLEDEX 6057 Robot Manipulator", Proceedings, IJCNN, San Diego, 1988, pp II-633-640.

[17] F. Tsung, G. Cottrell, "Sequential Adder Using Recurrent Networks," IJCNN Conference Proceedings, Vol.2, June 1989, pp II133-II139.

[18] P. Werbos, "Building and Understanding Adaptive Systems: A Statistical/Numerical Approach to Factory Automation and Brain Research," IEEE Transactions on Systems, Man, and Cybernetics, 1987, pp 7-20.