01 Jan 1992

Hierarchical Neurocontroller Architecture for Robotic Manipulation

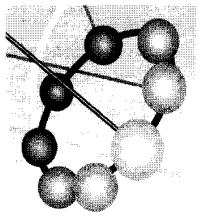
Xavier J. R. Avula
Missouri University of Science and Technology, avula@mst.edu

Luis C. Rabelo

Recommended Citation

X. J. Avula and L. C. Rabelo, "Hierarchical Neurocontroller Architecture for Robotic Manipulation," *IEEE Control Systems Magazine*, Institute of Electrical and Electronics Engineers (IEEE), Jan 1992.
The definitive version is available at https://doi.org/10.1109/37.126851

Hierarchical Neurocontroller Architecture for Robotic Manipulation



Luis C. Rabelo and Xavier J. R. Avula

A hierarchical neurocontroller architecture consisting of two artificial neural network systems for the manipulation of a robotic arm is presented. The higher level neural network system participates in the delineation of the robot arm workspace and coordinates transformation and the motion decision making process. The lower neural network provides the correct sequence of control actions. A straightforward example illustrates the architecture capabilities including speed, adaptability, and computational efficiency.

Neural Networks and Robotic Control

A number of authors have applied neural networks to the engineering problem of robotic control. Guez and Ahmad [1] 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 [2] 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.* [3] investigated the control of a constrained robot manipulator using back-propagation, 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.* [5] 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.

An early version of this paper was presented at the 1991 IEEE International Conference on Robotics and Automation, Sacramento CA, April 7-12, 1991. *Luis C. Rabelo is with the Department of Industrial and Systems Engineering, Ohio University, Athens, OH 45701. Xavier J.R. Avula is with the Department of Mechanical and Aerospace Engineering and Engineering Mechanics, University of Missouri-Rolla, Rolla, MO 65401.*

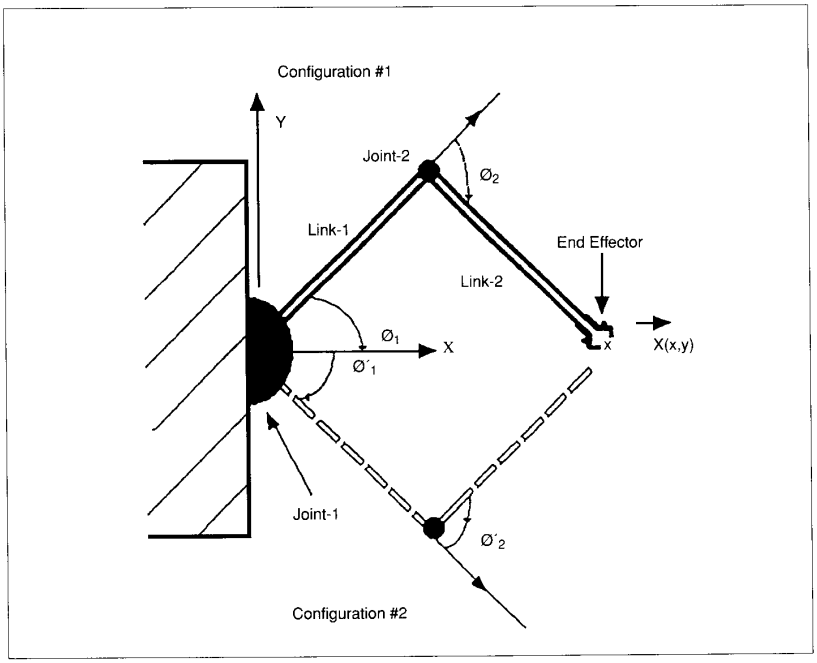


Fig. 1. Modeled robot arm.

The work presented here develops an architecture which can yield a strategy for dynamic decision making that allows the robot end-effector to reach its goal using *a priori* and on-line contextual information. The neural network systems cooperate with the entire architecture to achieve the necessary flexibility to adapt to unforeseen changes in the robot workspace, environmental changes, coordinate transformations, contextual rules (e.g., payload, direction of travel, stiffness), and interactions with other systems. In these 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 evolving systems.

System Model

The computer simulated robotic manipulator model consists of a two dimension version of an arm with two jointed links of equal length. Fig. 1 illustrates the simulated robot arm. 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° 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 (e.g., deformations up to 15% with an increment of temperature of 100°C are possible).

Due to the simplicity of the simulated manipulator, algorithmic relationships yield

0272-1708/92/$03.00©1992IEEE

most of the feedback needed for the supervision of the learning of the different networks.

Neural Networks

Two different neural network systems are presented associated with the prototype of a scheme which uses the integration of neural networks and knowledge-based systems for robotics motion control. These neural network systems participate in the tasks of the motion analysis process at the higher hierarchical level and the process of control-emulation at the lower level.

The neural network system at the higher hierarchical level supports the decision making by providing the motion analysis process with an initial hypothesis. This initial hypothesis is utilized to develop an early motion strategy to achieve the final position. This initiative does not preclude adaptive changes during the course of motion due to unexpected changes. The simplicity of the model utilized permits decomposition of the initial stage into three distinct tasks, each with its own associated neural network arrangement. However, for more complex models other techniques could be utilized such as goal programming, dynamic programming, and neural forward modeling [6],[7]. The first part, consisting of the preliminary motion feasibility analysis, uses a Restricted Coulomb Energy (RCE) network 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 neural networks is employed to map the angular coordinates of the two possible robot configurations that are consistent with the end-effector. Backpropagation is utilized for the coordinates transformation scheme. Special emphasis is placed on finding a reliable approach to obtain an efficient network architecture. Finally, the third network arrangement using backpropagation 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 could use the cooperation 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 are going to 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 Barto *et al.* [8], Jordan [9],[10], Jordan and Rumelhart [6], Kawato [7], Kong and Kosko [11], Nguyen and Widrow [12],[13], and Werbos [14],[15]. 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 high-level planner which manages and coordinates information concerning the task to be performed, and updates the system knowledge. This high-level planner is a computer generated response which uses multiple problem-solving modalities. In this study we have utilized forward modeling because of the availability of data. The forward model is taught by the high-level planner using simulated sensor feedback and implemented in a backpropagation network. Then, the emulator network of the robot arm dynamics is developed and the controller implementation is started. The controller is implemented using a backpropagation network which is 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 these trajectory changes is totally handled by the high-level planner according to sensory information, goals, and optimization factors. In addition, the controller receives input from the emulator at a higher frequency. The methodology which has been successfully used to train backpropagation networks in the motion analysis is utilized to train the emulator-controller neural network system. This methodology is proven to be efficient in the development of the required networks which especially involve large training data sets.

RCE Mapping of a Two Dimensional Robotic Arm Workspace

In the present study, an RCE network was used to accomplish this workspace filter. In an RCE network [16],[17], all examples of a pattern category (e.g., IN, OUT) define a set of points in the feature space that can be characterized as a region (or a set of regions) having some arbitrary shape. This feature of RCE networks makes them appropriate to define and learn complex workspaces (i.e., several degrees of freedom and links of arbitrary geometric shapes). In addition, RCE networks are appropriate for real-time learning of complicated nonlinear class regions, and capable of probability estimations to handle uncertainty.

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 (it is also possible to use polar coordinates, if required) 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 presented, is a good indication of the quality of the categorization (99.9%). The number reached an asymptotic value as training was completed and the network converged.

Mapping with Backpropagation

The neural networks for coordinate transformation, configuration selection, arm emulator, and neurocontroller were developed using the standard backpropagation paradigm. The training process using backpropagation is a difficult problem [18]-[22]. It is necessary 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 (e.g., initial weights), learning parameter values (e.g., learning rates), and to avoid over-training effects (performance degradation due to prolonged training).

In this research the following approaches were utilized:

1. To help to find an appropriate architecture (i.e., number of hidden units) an interactive addition of nodes was performed as proposed by the Dynamic Node Creation (DNC). DNC is a methodology developed at the University of California-San Diego by Ash [18] which adds nodes to the hidden layer(s) of the network during training. A new node is added if the root mean squared (RMS) error curve has flattened out to an unacceptable level. This level depends upon the relationship of the drop in the RMS error over the previous

"flattness window" to the RMS error when the last node was added. When this value falls below a user defined limit, a new node is added. This process is stopped when the desired performance has been achieved.

2. To speed up the convergence behavior, the selection of parameters such as learning rates and the utilization of a momentum factor were utilized. The learning rule utilized consisted of a weight update using momentum (β) with the exception that each weight had its own "adaptive" learning rate parameter (μ) [23]. The "adaptive" learning rate strategy increments the $\mu(s)$ by a small constant if the current partial derivative of the objective function ($E = 1/2\Sigma(\text{Target} - \text{Output})^2$) with respect to the weight (w) and the exponential average of the previous derivatives have the same sign, otherwise μ will be decremented by a proportion to its value. The updating equation of the weights is defined by using w_{ij} as the weight value located between nodes i and j, t is the present iteration, Δw is the weight increment which is equal to the product of the μ and the partial derivative of the objective function with respect to the weight ($\delta E/\delta w_{ij}$),

$$w_{ij}(t) = w_{ij}(t-1) + \Delta w_{ij}(t) + \beta \Delta w_{ij}(t-1).$$

The momentum β is changed dynamically, because each problem has a range of optimal β values to avoid oscillations.

3. To reinforce learning, Combined Subset Training (CST) was utilized [25]. 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. In this research, three different types of data sets were utilized: a training set, a validation set, and a testing set. The training set was utilized for determining the values of the weights. The validation set (a set with unseen data to make on-line tests about network performance) was utilized to avoid over-training effects [21],[22]. The testing set (as expressed by Weigend *et al.* [22]: "It is strictly set apart and never used in training.") was utilized to estimate the expected performance of the network. The criterion utilized in this research to stop training was based on low RMS and minimum output errors (as a function of some reasonable precision desired) for the training and validation data sets, and the network's performance with the testing set.

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 develop a methodology for choosing one which satisfies the motion and contextual constraints. The simplicity of the model allowed us to list the two possible target configurations and to decide an optimum configuration which is compatible with the initial configuration. For models with three or more degrees of freedom (which preclude the listing of possible configurations) the robot arm coordinates transformation and configuration selection should be treated by converting the constraints on solutions to a function to be optimized. This function could be added to the objective training function, and therefore could be part of the supervisory scheme utilized to train the network(s).

The system used to produce this transformation is based on two neural networks, each responsible for one of the possible configurations. The mapping assigns a three dimensional input vector consisting on the X and Y location of the end-effector as well as the temperature, T, (X, Y, T) to a two dimensional vector containing the required shoulder and elbow angular position (θ_1,θ_2).

The final architecture for both neural networks had 15 hidden units. Training sessions starting with one hidden unit and 100 data samples and incrementing nodes using a conservative width for the flatness window, the training epochs were indeed large (i.e., 80 000 epochs). When the final architecture of 15 hidden units was achieved CST was applied up to 800 data samples. It is interesting to note, that the next set of data to be added to the training file was based on the minimum output errors achieved before. Therefore, each data set added as defined by CST concentrated in those points where the network has had some lower performance. However, the effort was automated in its majority in the standard backpropagation simulator using the C programming language. Training trials starting with more hidden nodes initially (if *a priori* knowledge or heuristics specify a lower bound) and combined with learning rate adaptation required considerably less epochs (i.e., 8000 epochs).

For configuration selection, a neural network based on backpropagation 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. This network was developed using the training techniques described above — starting with a training set of 100 data samples and incremented as specified by CST up to 400 data samples.

Arm Emulator

The arm emulator is needed in order to identify the arm dynamic behavior. As expressed by Nguyen and Widrow [12],[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." In addition, using an emulator provides the advantage to extend this approach to more degrees of freedom [6],[9], and learn to control trajectories based on a final position error [6], [7], [9], [10], [12]-[14]. The procedures, in order to develop the emulator, should be encoded in the high-level planner. The high-level planner could examine the arm responses in the cartesian plane with different motor actions. This process will be implemented repetitively until an efficient emulator is developed. Neural networks has several advantages for this "automatic identification process" due to their nonlinearity and learning capabilities.

A neural network is developed using the techniques previously mentioned. This neural network has 5 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. The neural network has two outputs which correspond to the X, Y positions (i.e., next state) after the movement has been performed. The neural network developed has 42 hidden units and initially was trained with 800 data samples. After the final architecture was achieved, the training set was incremented as specified by CST up to 6400 data samples. It is interesting to report that the interactive addition of nodes loses some effectiveness when a network grows above certain number of hidden nodes. Recent research has emphasized in the development of architectures (especially for those which require relatively large sizes of hidden nodes and hidden layers) using pruning techniques based on the second derivatives of the objective function with respect to the weights [20].

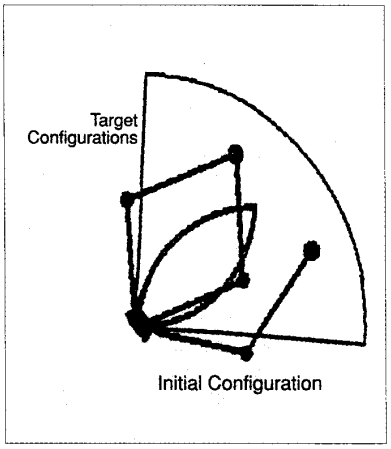


Fig. 2. Motion analysis.

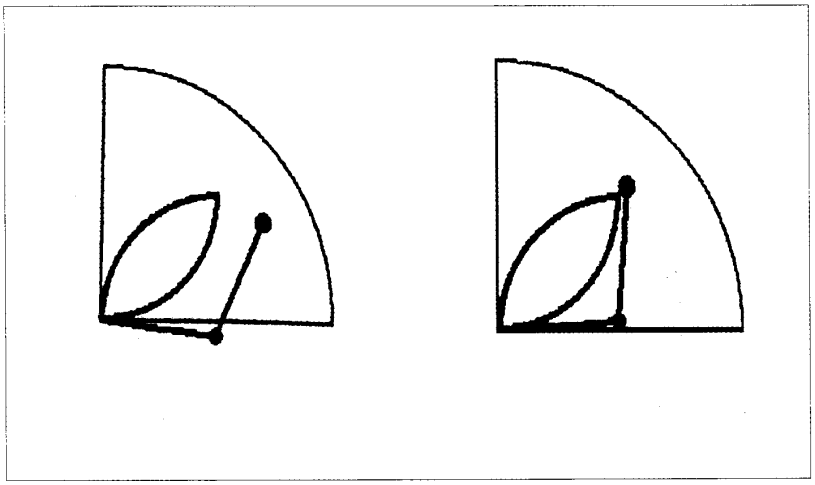


Fig. 3. Neurocontroller/emulator execution. (a) First movement ($X = 150$, $Y = 70$, $T = 25°C$, $\Delta\theta_1 = 2.0°$, $\Delta\theta_2 = 4.8°$, $X = 141.3$, $Y = 78.2$). (b) Fourth movement ($X = 121.7$, $Y = 98.7$, $T = 50°C$, $\Delta\theta_1 = 3.6°$, $\Delta\theta_2 = 3.5°$, $X = 110.9$, $Y = 108.9$).

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 high-level planner which provides the plans. The plans in this application define the type of trajectory desired. The trajectory could be defined as linear or circular. The high-level planner 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 ($Bx + C$) or circle ($Ax^2 + Bx + C$) 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. Then the "new" present state vector is used repeatedly till the targeted position is achieved. The necessary comparison is done by the high-level planner which will decide to send the STOP order ($A = B = C = 0$).

A neural network is developed using backpropagation and the techiques mentioned above to implement the neurocontroller. This neural network has 6 inputs corresponding to X, Y, T, A, B, and C. The outputs correspond to the increments of the shoulder and elbow angles. As it was mentioned above problems with the training sessions were due to the increments of node technique utilized. The network was started training with an 800 data samples. When a reasonable architecture was found (46 hidden units), data samples were incrementally added as specified by CST up to 6400 data samples.

An Example

Here we consider an example to illustrate the integration of the different 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 $X = 150$ cm, $Y = 70$ cm, and a temperature $T = 25°C$ to the final position $X = 80$ cm, $Y = 140$ cm at temperature $T = 60°C$ using a linear trajectory. A linear trajectory is characterized by the general equation $f(x) = Ax^2 + Bx + C$ with $A = 0$. The neural network system developed in this study utilizes the final position coordinates in planning the initial motion strategy. During motion along the prescribed trajectory, the system allows compensation for the linkage dimension changes that would occur due to significant temperature changes in the environment.

The robotic arm operates in the workspace which is confined between the inside and outside curves shown in Fig. 2. After the RCE unit declares that the target position is indeed inside the workspace, the two neural networks developed for coordinate transformation are executed. A selection from the two possible configurations is made by using a backpropagation network which takes into consideration the initial configuration. 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=–1.0, $C = 220$).

The neurocontroller drives the manipulator arm from the initial position to the final position using a quasi-linear trajectory. In spite of temperature variations which produce dimensional changes in the links, the neurocontroller can adapt and drive the manipulator along the planned trajectory. The input vector to the neurocontroller consists of the trajectory parameters A, B, C (here $A = 0$) and the current X, Y, T state and its output defines the angle increments ($\Delta\theta_1$, $\Delta\theta_2$) required to reach the target and maintain the prescribed trajectory. At time $t = 0$ s the current state assumes the initial conditions. The emulator has as inputs the current X, Y, T state and the output of the neurocontroller. The emulator output defines the "new" X, Y position reached by the manipulator. The execution of the neurocontroller/emulator system is illustrated in Fig. 3 which depicts the manipulator arm position at different times and temperature values. The high-level planner monitors the neurocontroller/emulator system in order to decide when to stop the motion execution.

System Capabilities Enhanced

Neural networks have capabilities to learn, to 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 neural networks 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 decision making using contextual information in order to provide an initial strategy,
- provide systems which are adaptable to environmental changes, and
- develop controllers which can create their own knowledge bases about the system dynamics.

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 two degrees of freedom and three dimensional work envelopes are not limiting factors to this architecture due to its concurrent coordination capabilities. However, the needs to improve the accuracy of the system and utilize a more continuous plan definition are recognized. Further research will concentrate in the coordination of several robotic end-effectors using sensor/arm integration and trajectory/configuration optimization as functions of the planning horizon.

References

[1] A. Guez and Z. Ahmad, "Accelerated convergence in the inverse kinematics via multi-layered feedforward networks," *Proc. IJCNN*, Washington, 1989, pp. 341-347.

[2] J. Guo and V. Cherkassky, "A solution to the inverse kinematic problem in robotics using neural network processing," in *Proc. IJCNN*, Washington, DC, pp. 299-304.

[3] D. Sobajic, J. Lu, and Y. Pao, "Intelligent control of the INTELLEDEX 6057 robot manipulator," in *Proc. IJCNN*, San Diego, CA, 1988, pp. 633-640.

[4] D. Bassi and G. Bekey, "High precision position control by cartesian trajectory feedback and connectionist inverse dynamics feedforward," in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, Washington, DC, 1989, pp. 325-332.

[5] H. Liu, T. Iberall, G. Bekey, "Neural network architecture for robot hand control," *IEEE Control Syst. Mag.*, Apr. 1989, pp. 38-42.

[6] M. Jordan and D. Rumelhart, "Forward models: Supervised learning with a distal teacher," Occasional Pap. #40, Center for Cognitive Science, Massachusetts Institute of Technology, Cambridge, MA, 1991.

[7] M. Kawato, "Computational schemes and neural network models for formation and control of multijoint arm trajectory," *Neural Networks for Control*, T. Miller, R. Sutton, and P. Werbos, Eds. M.I.T. Press, 1990, pp. 197-228.

[8] A. Barto, R. Sutton, and C. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Trans. Systems, Man, Cybernetics*, vol. SMC-13, pp 834-846.

[9] M. Jordan, "Supervised learning and systems with excess degrees of freedom," in *Proc. 1988 Summer School on Connectionist Models*. Morgan Kaufman, pp 62-75.

[10] M. Jordan, "Generic constraints on underspecified target trajectories," in *Proc. IJCNN*, vol. 1, June 1989, pp. 217-225.

[11] S. Kong and B. Kosko, "Comparison of fuzzy and neural truck backer-upper control systems," in *Proc. IJCNN*, San Diego, 1990, pp. 349-358.

[12] D. Nguyen and B. Widrow, "The truck backer-upper: An example of self-learning in neural networks," in *Proc. IJCNN*, Washington, DC, 1989, pp. 357-363.

[13] D. Nguyen and B. Widrow, "Neural networks for self-learning control systems," *IEEE Control Syst. Mag.*, vol. 10, no. 3, pp. 18-23, 1990.

[14] P. Werbos, "Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research," *IEEE Trans. Syst., Man, Cybern.*, 1987, vol. 17, pp. 7-20.

[15] P. Werbos, "Consistency of HDP applied to a simple reinforcement learning problem," *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 179-190, 1990.

[16] C. Bachman, L. Cooper, A. Dembo, and O. Zeitouni, "A relaxation model for memory with high storage density," *Proc. Nat. Acad. Sci.*, vol. 21, pp. 2088-3092, 1984.

[17] D. Reilly, L. Cooper, and C. Elbaum, "A neural model for category learning," *Biological Cybern.*, vol. 45, 1982, pp. 35-41.

[18] T. Ash, "Dynamic node creation," Tech. Rep. ICS 8901, University of California-San Diego, 1989.

[19] Y. Chauvin, "Dynamic behavior of constrained back-propagation networks," *Advances in Neural Information Processing Systems 2*, David Touretzky, Ed. Morgan Kaufmann, 1990, p. 642.

[20] Y. Le Cun, J. Denker, and S. Solla, "Optimal brain damage," *Advances in Neural Information Processing Systems 2*, David Touretzky, Ed. Morgan Kaufmann, 1990, pp. 598-605.

[21] N. Morgan and H. Bourlard, "Generalization and parameter estimation in feedforward nets: Some experiments," International Computer Science Institute, TR-89-017, 1989.

[22] A. Weigend, A.D. Rumelhart, and B. Huberman, "Back-propagation, weight elimination and time series prediction," *Connectionist Models: Proc. 1990 Summer School*, D. Touretzky, J. Elman, T. Sejnowski, and G. Hinton, Eds., Morgan Kaufmann, 1990, pp. 105-116.

[23] R. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, no. 3, pp. 295-307, 1988.

[24] D. Rumelhart, J. McClelland, and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1: Foundations, Cambridge, MA: M.I.T. Press, 1988.

[25] F. Tsung and G. Cottrell, "Sequential adder using recurrent networks," *IJCNN Conf. Proc.*, vol. 2, June 1989, pp. 133-139.

Luis Carlos Rabelo obtained a dual bachelor's degree in electrical and mechanical engineering from the Technological University of Panama in 1983, a Master of Science Degree in electrical engineering from Florida Institute of Technology in 1987, a Master of Science Degree in Engineering Management, and a Ph.D. in engineering management from the University of Missouri-Rolla in 1988 and 1990, respectively. He did postdoctoral studies in Nuclear Engineering in 1990-1991. Currently, Dr. Rabelo is an assistant professor in the Department of Industrial and Systems Engineering at Ohio University. His research interests include neural networks, robotics and automation, signal processing, and parallel processing.

Xavier J. R. Avula received the Ph. D. degree in engineering mechanics from the Iowa State University in 1968. He joined the faculty of the University of Missouri-Rolla in 1967 and became professor in 1977. He has authored more than 60 technical publications. He was Co-Editor-in-Chief of the journal *Mathematical and Computer Modelling*, and recently founded the new journal *Modelling and Scientific Computing*. Intermittently, he held visiting positions at Wright-Patterson Air Force Base, IBM, and Politecnico di Torino. His research interests include neural networks, robotics, composite materials, microstructures, gravitational physiology and solid-fluid interaction.