W. Kelly, R. Challoo, et. al, "Neuro-fuzzy Control of a Robotic Arm",
Proceedings of the Artificial Neural Networks In Engineering Conference ,
St. Louis, MO, November 10-13, 1996, pp. 837-842.

NEURO-FUZZY CONTROL OF A ROBOTIC ARM

WALLACE E. KELLY III
Department of Electrical Engineering
Texas A&M University, College Station, Texas 77843-3128

RAJAB CHALLOO, ROBERT MCLAUCHLAN, S. IQBAL OMAR
Intelligent Control Systems Laboratory
Texas A&M University – Kingsville, Kingsville, Texas 78363

ABSTRACT:
This paper first presents a discussion of the reasoning and method for combining neural networks and fuzzy logic. The problem of moving a robotic arm in the presence of an obstacle is discussed. Several neuro-fuzzy controllers are trained using sample data obtained from a human's control of a robotic arm. Their performance is quantified and compared. It is shown that the definition of the fuzzy membership functions plays a significant role in the ability of the neuro-fuzzy controller to learn and generalize. Possible directions for future work are suggested.

NEURO-FUZZY SYSTEMS

Recently, the combination of neural networks and fuzzy logic has received attention. Neural networks bring into this union the ability to learn, but also require an excessive number of iterations for training of complex systems. Fuzzy logic offers a system model based on membership functions and a rule base, but require an explicit stating of the IF/THEN rules.

Several methods for combining neural networks and fuzzy logic have been studied (Khan, 1993) (Lin and Song, 1994) (Nauck, et. al., 1993). In this paper, the authors implement the inference stage of a fuzzy system using a neural network (Challoo, et. al., 1994) (Keller, et. al., 1992). Figure 1 illustrates the system architecture for the described combination of neural networks and fuzzy logic. By replacing the rule base of a fuzzy system with a trainable neural network, complex input-output relationships can be achieved which can not be easily specified by

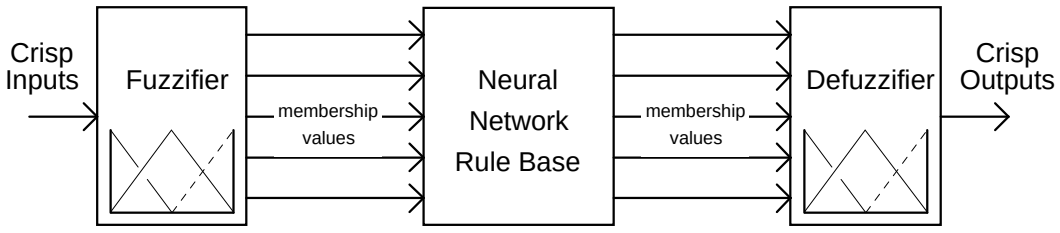

Figure 1. A fuzzy system with neural network rule base

IF/THEN rules. With fuzzification and defuzzification stages augmenting a neural network, significant improvements in the training time, in the ability to generalize, and in the ability to find minimizing weights can be realized. Furthermore, the fuzzy membership functions give the designer more control over the neural network inputs and outputs.

THE ROBOTIC ARM PROBLEM

For robots to become effectively used in a wide range of applications, they must gain the ability to work in unpredictable environments. This paper addresses the problem of planning the trajectory of a three-link robotic arm in the presence of an obstacle.

The arm operates in two dimensions in an environment containing a randomly placed obstacle and goal. The starting position of the arm is arbitrary as well. For the purpose of designing the control system, the positions of the obstacle and goal and the joint angles of the arm are assumed to be available from position feedback sensors in the arm.

The arm is modeled as a three-link planar manipulator, as shown in Figure 2. The model is strictly geometric. That is, the dynamics of moving a finite mass arm are not considered for the purpose of this study. The controller will determine a series of joint angles, $\Theta(t)$, that move the end effector from a given starting position (x_s, y_s) to a desired final position (x_g, y_g) without colliding with the obstacle at (x_o, y_o).

Previous approaches to this problem using fuzzy logic have focused on choosing the joint angles of redundant manipulators given a desired path (Kim and Lee, 1993) (Xu, et. al., 1993) (Wang, et. al., 1994) or specifying criteria for choosing those joint angles using fuzzy rules (Palm, 1992). Unlike these approaches, this work uses a fuzzy controller that *learns the strategy* for moving the arm to the goal position without touching the obstacle. That strategy is learned by observing a human's control of the arm.

The goal must be assumed to lie within the possible reach of the end-effector and the obstacle must be assumed to lie outside the path of the first link. The problem can be summarized as follows:

Given a robotic arm with:

- the current joint angles, $\Theta(0)$,
- an obstacle position, (x_o, y_o), and
- a desired position, or goal, (x_g, y_g);

Find a trajectory, $\Theta(t)$, such that:

- the end-effector reaches the goal,
- the arm does not touch the obstacle, and
- the calculations can be performed in real-time with current hardware.

W. Kelly, R. Challoo, et. al, "Neuro-fuzzy Control of a Robotic Arm",
Proceedings of the Artificial Neural Networks In Engineering Conference ,
St. Louis, MO, November 10-13, 1996, pp. 837-842.

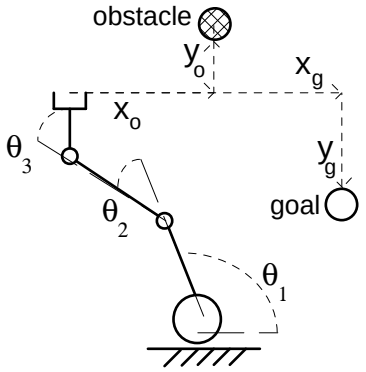

Figure 2. The state representation for the goal-obstacle problem.

STATE REPRESENTATION

One of the primary considerations in the design of the neuro-fuzzy controller is the representation of the state of the system. The ability of the final neuro-fuzzy controller to generalize a solution from training data depends largely on the data representation scheme. The controller must be provided with the joint angles and the locations of the obstacle and goal.

There are actually several ways to provide information about the locations of the goal and obstacle. Providing Cartesian or polar coordinates of the two objects would result in seven, independent inputs. This scheme, however, would require that the controller repeatedly learn the same strategy for similar configurations of the end-effector, goal and obstacle if that configuration happened to occur in a different Cartesian region. If, on the other hand, the location of the goal and obstacle are represented by their relative distance to the end-effector, then strategies can be generalized for similar configurations regardless of where they occur in Cartesian space. Table 1 and Table 2 list the inputs and outputs that were chosen to represent the system.

NEURO-FUZZY SOLUTION

As with most systems involving neural networks, training samples were required which demonstrated the desired input-output relationship. The samples were obtained by cycling through a series of goal and obstacle positions on a simulation of

TABLE 1: INPUTS TO THE NEURO-FUZZY CONTROLLERS

Input	Description
θ_1	joint angle of link1 to the base
θ_2	joint angle of link2 from link1's axis
θ_3	joint angle of link3 from link2's axis
x_o	horizontal distance to the obstacle
y_o	vertical distance to the obstacle
x_g	horizontal distance to the goal
y_g	vertical distance to the goal

TABLE 2: OUTPUTS OF THE NEURO-FUZZY CONTROLLERS

Outputs	Description
$\Delta\theta_1$	a change in the angle between link1 and the base
$\Delta\theta_2$	a small change in the angle between link2 and the link1 axis
$\Delta\theta_3$	a small change in the angle between link3 and the link2 axis

the robotic arm. The simulated arm was manually maneuvered past the obstacle and to the goal. Goal and obstacle positions were taken from the entire reachable region.

As the training samples were recorded, each movement created a new input to the neuro-fuzzy system. Therefore, selecting starting positions of the arm at the two extremes of possible movements generated training samples that extended over the entire state space. Obtaining training samples over an entire range of possible state space is important for neuro-fuzzy design.

The recorded samples were used to train seven neuro-fuzzy controllers to emulate the strategies demonstrated in the human's control of the arm. Testing the controllers consisted of placing the goal and obstacle objects at random locations and assigning a random starting position for the arm. The controller simulation quantified the performance of each neuro-fuzzy controller by determining the percentage error in reaching the goal, detecting collisions, performing all the necessary calculations so that a judgment can be made about the possibility of real-time application.

EXPERIMENTAL RESULTS

The training samples were used to train seven different neuro-fuzzy controllers, named FNN1, FNN2, ... FNN7 for reference. The controllers varied in the definition of their fuzzy membership functions and in the number of hidden nodes.

Figure 3 through Figure 9 show the seven FNN fuzzy membership functions. Notice the change in granularity and also the change in the concentration of the fuzzy values. With the increased granularity in the fuzzy sets, more hidden layer nodes were required. For example, FNN1 partitioned each of the seven inputs into three fuzzy sets yielding 21 input nodes. In this case, 35 hidden layer nodes were used. Figure 10 shows the final RMS training error which indicates which fuzzy membership function definitions were best able to emulate the training samples.

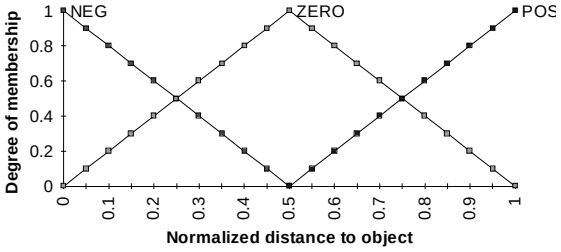

Figure 3. FNN1 - 3 fuzzy sets per input and 35 hidden layer nodes.

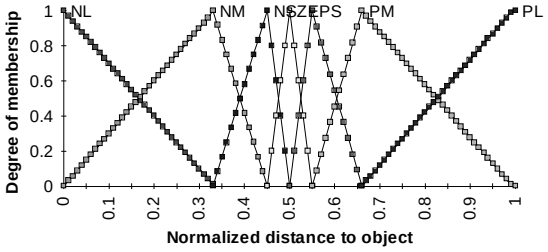

Figure 5. FNN3 - 7 fuzzy sets per input and 65 hidden layer nodes.

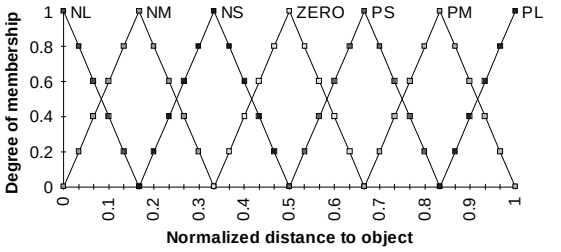

Figure 4. FNN2 - 7 fuzzy sets per input and 65 hidden layer nodes.

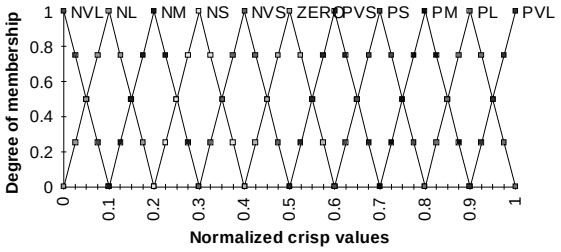

Figure 6. FNN4 - 11 fuzzy sets per input and 85 hidden layer nodes.

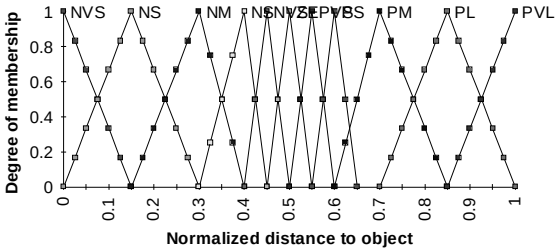

Figure 7. FNN5 - 11 fuzzy sets per input and 85 hidden layer nodes.

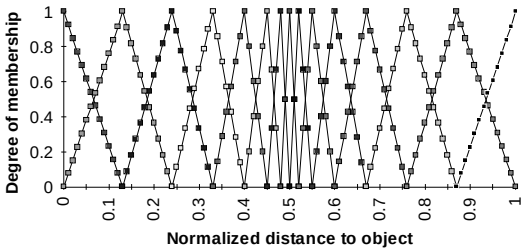

Figure 9. FNN7 - 15 fuzzy sets per input and 100 hidden layer nodes.

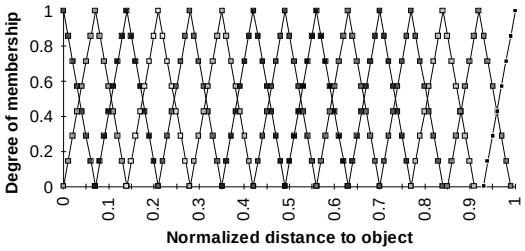

Figure 8. FNN6 - 15 fuzzy sets per input and 100 hidden layer nodes.

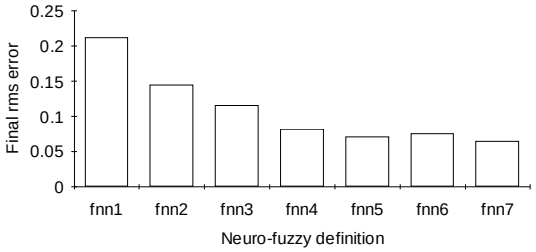

Figure 10. A comparison of the final RMS training errors.

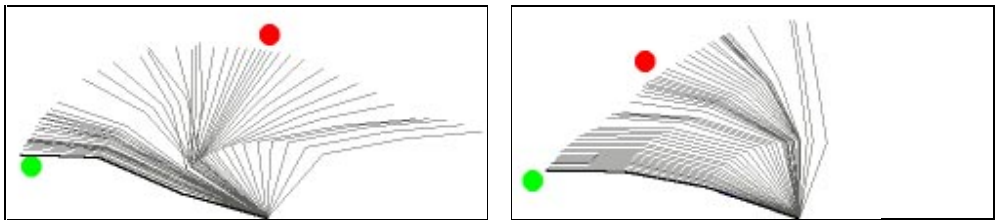

Figure 11. Examples of successful control by the neuro-fuzzy controllers.

Figure 11 shows two screen-shots of successful control by the trained controller. Figure 12 compares the collision and success rate for the seven controllers. The controller was successful if the end-effector touched the goal and the arm never touched the obstacle. A collision occurred if any part of the arm touched the obstacle. The percentage of runs that resulted in the arm attempting to move past a physical constraint, and the percentage of runs that the arm did not reach the goal are not shown. Those two cases account for the remaining percentages.

Finally, many attempts were made to train a back propagation neural network to control the simulated arm. The neural networks without the fuzzification layer were

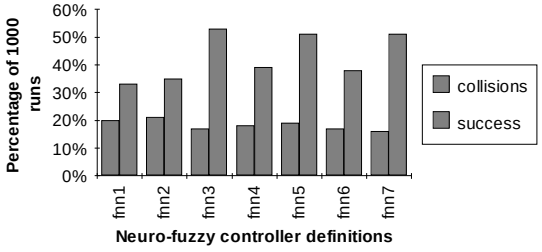

Figure 12. A comparison of the performance of all the controllers.

not able to generalize strategies from the sample data.

CONCLUSIONS

Two conclusions can be drawn from the experimental results. First, the membership functions are an important part of the neuro-fuzzy system. Figure 10 shows that the final training error was reduced 50% by increasing the number of fuzzy values in a fuzzy set. Figure 12 shows that membership functions, defined with an important aspect of the problem in mind, consistently improved performance by more than 25%. Second, the fuzzification stage allowed the neural networks to learn more complex functions than a neural network without fuzzification.

The performance of the neuro-fuzzy controllers is less than perfect. Even the best controller had a collision rate of 17%. A large percentage of the failures resulted from the controller attempting to move past the physical joint constraints. Another situation that decreased performance occurred when the arm started oscillating between two points. A few heuristics in the control program could decrease these problems. The seven FNN controllers of this project were trained to 2150 training samples. Future work will determine the adequacy of the training samples. A study of the relationship of the training samples and the fuzzy membership functions would be particularly helpful.

The use of neuro-fuzzy systems for control has been examined. Fuzzification of a neural network's inputs and outputs will become a standard procedure in neural network applications.

REFERENCES

Challoo, R., Clark, D., McLauchlin, R., and Omar, S. (1994). "A Fuzzy Neural Hybrid System," *Proceedings of the 1994 IEEE International Conference on Neural Networks* , pp. 1654-1657.

Keller, J., Yager, R., and Tahani, H., (1992). "Neural Network Implemention of Fuzzy Logic," *Fuzzy Sets and Systems*, Vol. 45, pp. 1-12.

Khan, E., (1993). "An Elegant Combination of Fuzzy Logic & Neural Nets", *Proceedings of Fuzzy Logic '93*, pp. A223-1 - A223-7.

Kim, S., and Lee, J., (1993). "Inverse Kinematics Solution Based on Fuzzy Logic for Redundant Manipulators", *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 904-910.

Lin, J., and Song, S., (1994). "A Novel Fuzzy Neural Network for the Control of Complex Systems," *Proceedings of the 1994 IEEE International Conference on Neural Networks*, pp. 1668-1673.

Nauck, D., Klawonn, F., and Kruse, R., (1993). "Combining Neural Networks and Fuzzy Controllers" *Fuzzy Logic in Artificial Intelligence (FLAI93)*, ed. Klement, Erich Peter and Slany, Wolfgang, pp. 35-46.

Palm, R., (1992). "Control of a Redundant Manipulator Using Fuzzy Rules," *Fuzzy Sets and Systems*, Vol. 45, pp. 279-298.

Wang, D., and Gu, M., (1994). "Fuzzy Logic Joint Path Generation for Kinematic Redundant Manipulators with Multiple Criteria," *Proceedings of the 1994 IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, pp. 649-656.

Xu, Y., and Nechyba, M., (1993). "Fuzzy Inverse Kinematic Mapping: Rule Generation, Efficiency, and Implemention," *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 911-918.