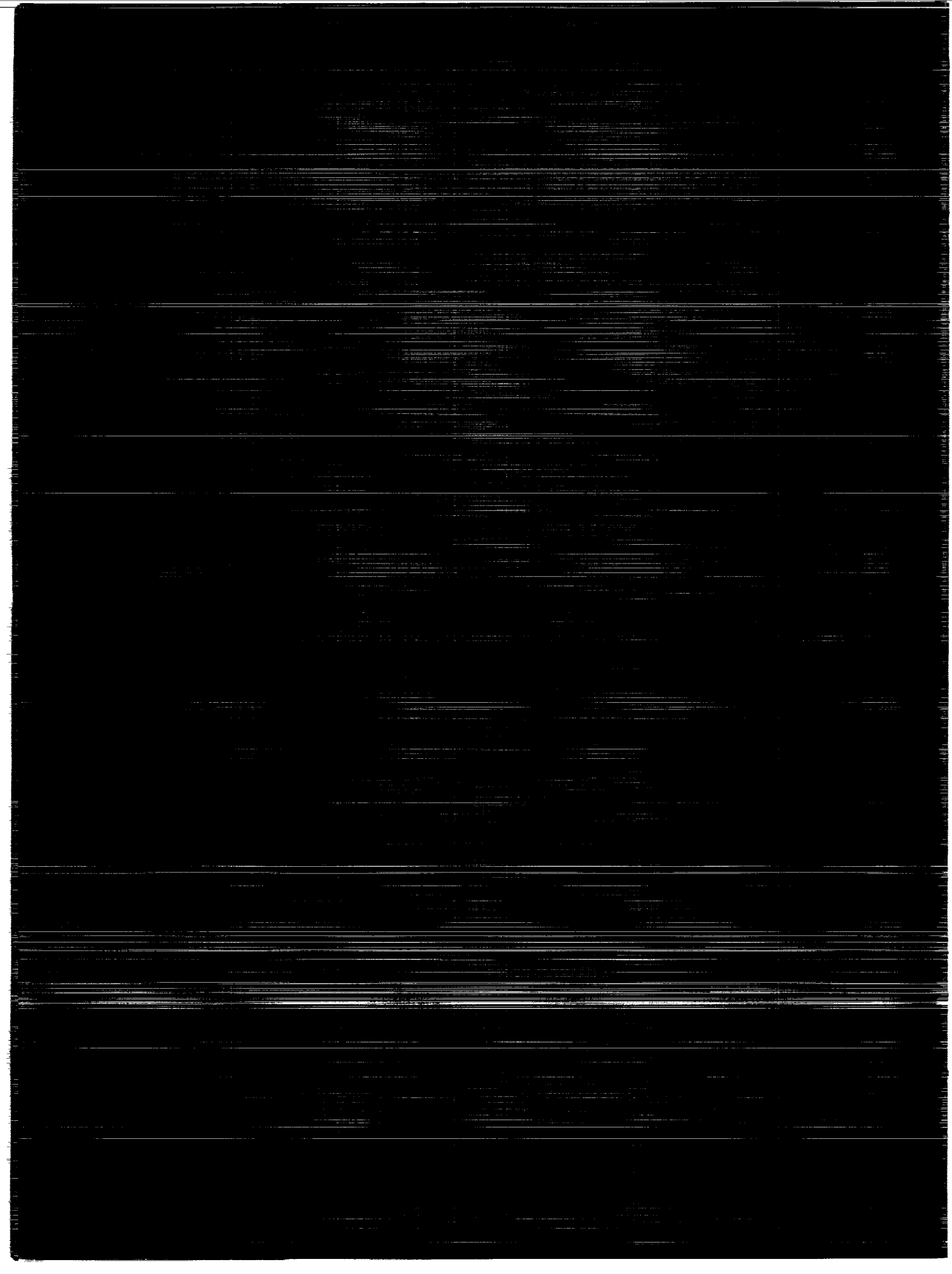
NASA Conference Publication 10061

Proceedings of the Second Joint Technology Workshop on Neural Networks and Fuzzy Logic

Volume II

Proceedings of a workshop held at
Lyndon B. Johnson Space Center
Houston, Texas
April 10 - 13, 1990



NASA Conference Publication 10061

Proceedings of the Second Joint Technology Workshop on Neural Networks and Fuzzy Logic

Volume II

Robert N. Lea and
James Villarreal, Editors
NASA Lyndon B. Johnson Space Center
Houston, Texas

Proceedings of a workshop sponsored by the
National Aeronautics and Space Administration,
Washington, D.C., and cosponsored by
Lyndon B. Johnson Space Center and
the University of Houston, Clear Lake
Houston, Texas
April 10 - 13, 1990

National Aeronautics and
Space Administration

February 1991

Proceedings of the Second Joint Technology Workshop

on Neural Networks and Fuzzy Logic

Program

April 10, 1990 PM

6:00-8:00 Registration

April 11, 1990 AM

7:30-8:00 Registration

8:00-8:30 **Robert T. Savely**, Branch Chief, Software Technology Branch, NASA/Lyndon B. Johnson Space Center, Houston, TX, **Welcoming Remarks**

8:30-9:00 **Jon Erickson**, Division Chief, Automation and Robotics Division, NASA/Lyndon B. Johnson Space Center, Houston, TX, **Overview of Space Station**

9:00-9:30 **Barney Roberts**, Planet Surface System Manager, NASA/Lyndon B. Johnson Space Center, Houston, TX, **Overview of Mars/Lunar Initiative**

9:30-9:45 Break

9:45-10:45 **Bernard Widrow**, Stanford University, **Neural Control Systems**

10:45-11:45 **Bart Kosko**, University of Southern California, Los Angeles, CA, **Fuzzy Sets and Associative Memories**

11:45-1:00 Lunch

April 11, 1990 PM

1:00-1:45 **Hal White**, University of California at San Diego, **Neural Network Representations and Learning of Mappings and their Derivatives**

1:45-2:15 **Masaki Togai**, Togai Infralogic, Inc., Irvine, CA, **Impact of Application of Fuzzy Theory to Industry**

2:15-2:30 Break

2:30-3:00 **Takeshi Yamakawa**, Kyushu Institute of Technology, Iizuka, Fukuoka, Japan, **Time-Sweeping Mode Fuzzy Computer Hardware System --- Forward and Backward Fuzzy Inference Engine ---**

3:00-3:30 **P. Z. Wang**, Institute of Systems Science, National University of Singapore, **The Simplification of Fuzzy Control Algorithm and Hardware Implementation**

3:30-3:45 Break

3:45-5:15 **Lotfi Zadeh**; University of California at Berkeley, moderator of panel discussion on

Fuzzy Sets, Neural Networks, and Intelligent Control

Panel members include:
Bernard Widrow, Stanford University
Bart Kosko, University of Southern California
Masaki Togai, Togai Infralogic Corporation
Takeshi Yamakawa, Kyushu Institute of Technology
P. Z. Wang. National University of Singapore
Hal White, University of California, San Diego
Elie Sanchez, Neural and Fuzzy Systems Institute, Marseilles, France
Paul Werbos, National Science Foundation.

6:00-7:00 Wine and Cheese Reception

7:00-9:00 Banquet and Keynote Speaker

Professor Lotfi Zadeh: The Role of Logic in Human and Machine Intelligence.

April 12, 1990 AM

8:00-8:30 **James Anderson**, Brown University, Providence, RI, **Experiments with Representations in Simple Neural Networks**

8:30-9:00 **James Bezdek**, University of West Florida, **Generalized Self Organizing Clustering Schemes**

9:00-9:15 Break

9:15-9:45 **Hiroyuki Watanabe**, University of North Carolina, Chapel Hill, NC, **A Single Board Fuzzy Inference System**

9:45-10:15 **Isao Hayashi**, Central Research Laboratories, Matsushita Electrical Industrial Co., **The Learning Function of NN-Driven Fuzzy Reasoning under Changes of Reasoning Environment**

10:15-10:30 Break

10:30-11:00 **Kaoru Hirota**, Hosei University, Tokyo, Japan, **A Solution of Inverse Problem of Fuzzy Relational Equation by using Perceptron Model**

11:00-11:45 **Masaki Togai**, Togai Infralogic, Inc., Irvine, CA, **Overview of LIFE (Laboratory for International Fuzzy Engineering) Research**

11:45-1:00 Lunch

April 12, 1990 PM

1:00-1:30 **James Keller**, University of Missouri, Columbia, MO, **Experiments on Neural Network Architectures for Fuzzy Logic**

1:30-2:00 **John Yen**, Texas A&M University, College Station, TX, **Using Fuzzy Logic to Integrate Neural Networks and Knowledge-based Systems**

2:00-2:30 **Hamid Berenji**, Ames Research Center, Palo Alto, CA, **An Architecture for Designing Fuzzy Controllers Using Neural Networks**

2:30-2:45 Break

2:45-3:15 **Rod Taber**, Center of Applied Optics, University of Alabama in Huntsville, Huntsville, Alabama, **Spatiotemporal Pattern Recognition with the Neuron Ring**

3:15-3:45 **Robert Shelton and James Villarreal**, NASA/Lyndon B. Johnson Space Center, Houston, TX, **Spatiotemporal Neural Networks**

3:45-4:15 **Yashvant Jani and Robert N. Lea**, NASA/Lyndon B. Johnson Space Center, Houston, TX, **Fuzzy Logic in Autonomous Spacecraft Operations**

4:15-4:45 **Kumpati (Bob) S. Narendra**, Yale University, New Haven, CT, **Identification and Control of Dynamical Systems using Neural Networks**

4:45-5:15 **Jacob Barhen**, Center for Space Microelectronics Technology, Jet Propulsion Laboratory, Pasadena, CA, **Non-Lipschitzian Dynamics**

April 13, 1990 AM

8:00-8:30 **Dan Greenwood**, Netrologic, San Diego, CA, **Diagnosis and Failure Prediction of the Space Shuttle Main Engine**

8:30-9:00 **Paul Werbos**, National Science Foundation, **Neural Nets for Control and the Link to Fuzzy Logic**

9:00-9:15 Break

9:15-9:45 **C.C. Lee**, University of California at Berkeley, Berkeley, CA, **An Intelligent Control System for Dynamic Processes**

9:45-10:15 **Ronald Yager**, Machine Intelligence Institute, Iona College, New Rochelle, NY, **A Neural Network Based Fuzzy Logic Controller**

10:15-10:45 **Sankar K. Pal**, NASA/Lyndon B. Johnson Space Center, Houston, TX and Indian Statistical Institute, Calcutta, India, **Fuzzy Geometry, Entropy and Image Information**

10:45-11:00 Break

11:00-11:30 **Enrique Ruspini**, Stanford Research Institute, Menlo Park, CA, **The Semantics of Fuzzy Logic**

11:30-12:00 **Robert Dawes**, Martingale Research Corporation, **Identification, Estimation and Control of Dynamical Systems with the Parametric Avalanche Neural Network**

CONTENTS

VOLUME I

CONTENTS

VOLUME II

An ARCHITECTURE FOR DESIGNING FUZZY LOGIC CONTROLLERS USING NEURAL NETWORKS

Hamid R. Berenji
Sterling Federal Systems
Artificial Intelligence Research Branch
NASA Ames Research Center
MS: 244-17, Moffett Field, CA 94035
e-mail: berenji@pluto.arc.nasa.gov

Abstract

In this paper, we describe an architecture for designing fuzzy controllers through a hierarchical process of control rule acquisition and by using special classes of neural network learning techniques. Hierarchical development of the fuzzy control rules is a useful technique which has been used earlier in designing a fuzzy controller with interactive goals [5]. Also, we introduce a new method for learning to refine a fuzzy logic controller. A reinforcement learning technique is used in conjunction with a multi-layer neural network model of a fuzzy controller. The model learns by updating its prediction of the plant's behavior and is related to the Sutton's Temporal Difference (TD) method. The method proposed here has the advantage of using the control knowledge of an experienced operator and fine-tuning it through the process of learning. The approach is applied to a cart-pole balancing system.

1 Introduction

Fuzzy logic controllers have recently experienced a huge commercial success [12,6]. These controllers are usually developed based on the knowledge of human expert operators[4]. However, starting with the Self Organizing Control (SOC) techniques of Mamdani and his students (e.g., [9]), the need for research in developing fuzzy logic controllers which can learn from experience has been realized (e.g., [8]). The learning task may include the identification of the main control parameters (i.e., related to the system identification in conventional and modern control theory) or development and fine-tuning of the fuzzy memberships used in the control rules. In this paper, we concentrate on the latter learning task and develop a model which can learn to adjust the fuzzy memberships of the linguistic labels.

The organization of this paper is as follows. We first discuss the general model of our NeuroFuzzy Controller (NFC) and then we apply this model to the control of a cart-pole balancing system. Finally, we compare this model with other related research works such as the credit assignment in artificial intelligence [10], Barto et. al.'s AHC model [3], and Lee and Berenji's single layer model [8].

2 NFC: A Model for Intelligent Control

Figure 1 illustrates the general model of our intelligent controller. The two main elements in this model are the Action-state Evaluation Network (AEN), which acts as a critic and provides advice to the main controller, and the Action Selection Network (ASN) which includes a fuzzy controller.

2.1 Action-state Evaluation Network (AEN)

The only information received by the AEN is the state of the plant in terms of its state variables and whether a failure has occurred or not. Figure 2 illustrates the structure of an evaluation network including m_h hidden units and n input units from the environment (i.e., x_0, x_1,..., x_n). The triangles represent the calculation-center [1] of the units where the updating equations

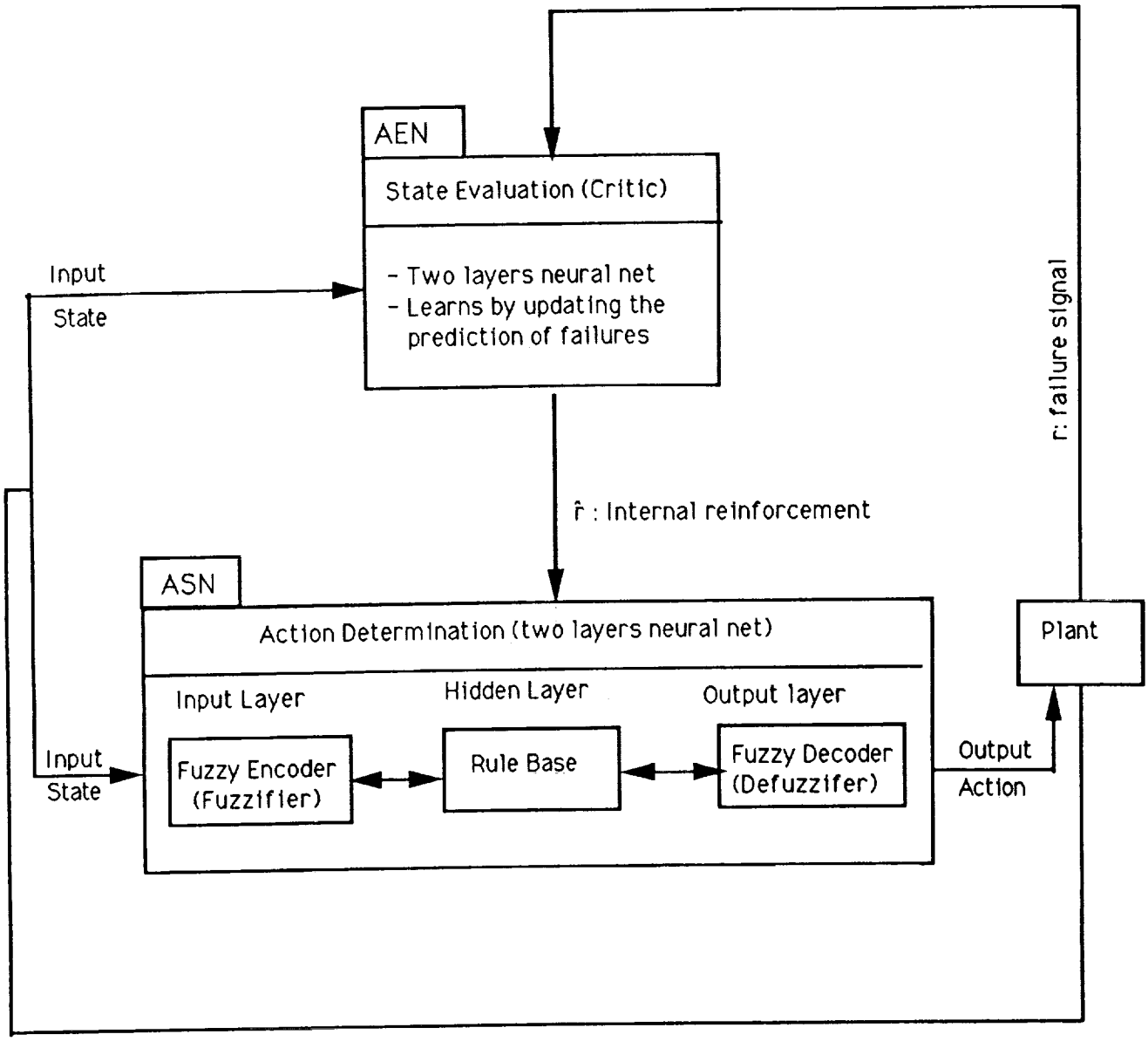
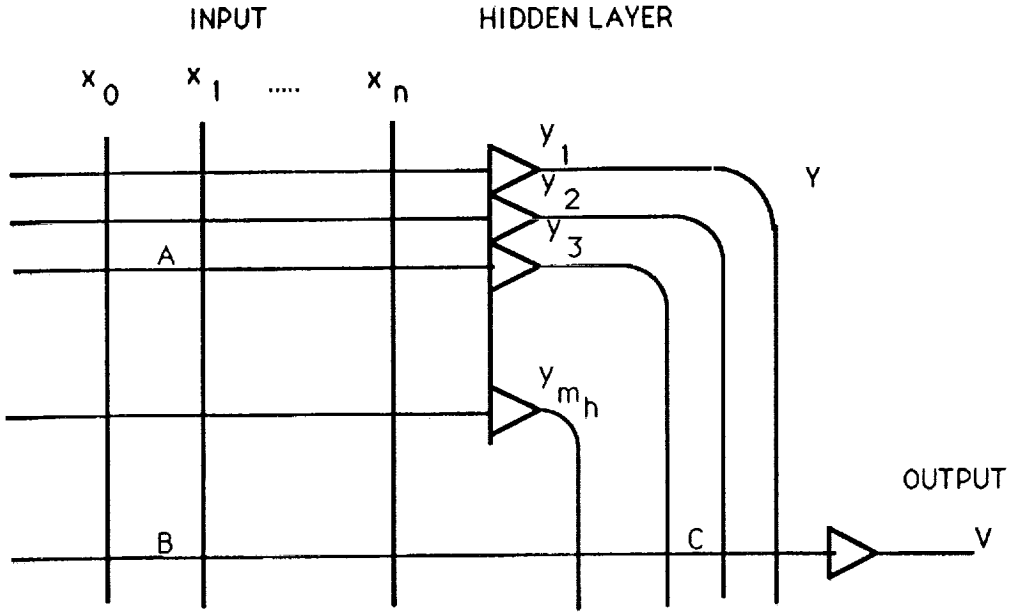


Figure 1: The NFC Model for Intelligent Control



A: Matrix of weights on the arcs connecting the input layer to the hidden layer
B: Matrix of weights on the arcs connecting the input layer to the output layer
C: Matrix of weights on the arcs connecting the hidden layer to the output layer

Figure 2: The Evaluation Network

(to be described bellow) are applied. The input from the environment is provided to all hidden units and output units while an interconnection weight exists at every intersection. Therefore in this network, hidden units receive $n+1$ inputs and have $n+1$ weights each while the output units receive $n+1+m_h$ inputs and have $n+1+m_h$ weights. If A, B, C are the matrices of connection weights, then the output of the evaluation network is:

$$v[t_1, t_2] = \sum_{i=1}^{n} b_i[t_1]x_i[t_2] + \sum_{i=1}^{m_h} c_i[t_1]y_i[t_1, t_2] \tag{1}$$

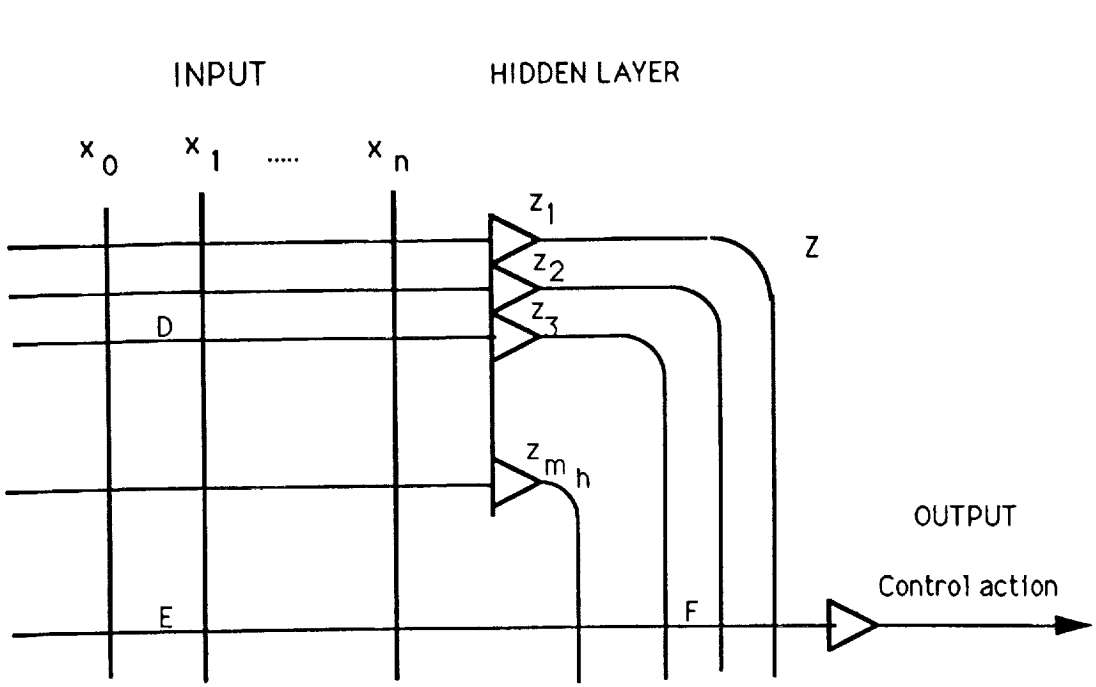
where

$$y_i[t_1, t_2] = g(\sum_{j=1}^{n} a_{ij}[t_1]x_j[t_2]) \tag{2}$$

and

$$g(s) = \frac{1}{1+e^{-s}} \tag{3}$$

In the above equations, double time dependencies are used to avoid instabilities in the updating of weights [2]. This network evaluates the action recommended by the action network as a function of the failure signal and



D: Matrix of weights on the arcs connecting the input layer to the hidden layer
E: Matrix of weights on the arcs connecting the input layer to the output layer
F: Matrix of weights on the arcs connecting the hidden layer to the output layer

Figure 3: The Action Selection Network

the change in state evaluation:

$$\hat{r}[t+1] = \begin{cases} 0 & \text{if state at time } t+1 \text{ is a start state;} \\ r[t+1] - v[t,t] & \text{if state at time t+1 is a failure state;} \\ r[t+1] + \gamma v[t,t+1] - v[t,t] & \text{otherwise} \end{cases} \tag{4}$$

The weights in this network are modified according to the followings:

$$b_i[t+1] = b_i[t] + \beta \hat{r}[t+1] x_i[t] \tag{5}$$

$$c_i[t+1] = c_i[t] + \beta \hat{r}[t+1] y_i[t,t] \tag{6}$$

$$a_{ij}[t+1] = a_{ij}[t] + \beta_h \hat{r}[t+1] y_i[t,t](1 - y_i[t,t]) sgn(c_i[t]) x_j[t] \tag{7}$$

where $0 < \gamma \leq 1$ and $\beta, \beta_h > 0$.

2.2 Action Selection Network (ASN)

The Action Selection Network (ASN) includes a fuzzy controller which consists of a fuzzifier, a rule base and decision making logic, and a defuzzifier all

represented in a network. The design of the rule base for this fuzzy controller follows the algorithm developed in [5] which is based on a hierarchical process considering the interaction of multiple goals.

In this paper, the above fuzzy controller is modeled by a two layered neural network where the input layer includes the fuzzifier whose task is to match the values of the input variables against the labels used in the fuzzy control rules. The hidden layer in this network corresponds to the rules used in the controller and includes the decision making logic. The output layer includes the decoding (defuzzification) process. In the following, a brief explanation on fuzzy logic control is provided. However, for more detailed information, see [4]. The action selector is shown in Figure 3, where the matrices of connection weights are D, E, and F. The individual member of these matrices are labelled d_{ij}, e_i, and f_i. In this network, the hidden nodes represent a fuzzy control rule in the following manner. The inputs to the node are the preconditions of a rule and the output of the node is its conclusion. We assume a Multi Input Single Output (MISO) control system. The output layer combines the conclusion of the individual rules by using the Center Of Area (COA) method [4] which is described below. Let $w(i)$ represent the degree that rule i is satisfied by the input state variables in X which means

$$w(i) = Min\{d_{i1}\mu_{i1}(x_1), d_{i2}\mu_{i2}(x_2), ..., d_{in}\mu_{in}(x_n)\} \tag{8}$$

where $\mu_{i1}(x_1)$ represents the degree of membership of the input x_1 in a fuzzy set representing the label used in the first precondition of the rule i and n is the number of inputs. Then $m(i)$, which represents the result of applying the $w(i)$ on the conclusion of rule i, is calculated from

$$w(i) = \mu_{C_i}(m(i)) \tag{9}$$

where μ_{C_i} represents the monotonic membership function of the label used in the conclusion of rule i. The amount of the control action (i.e., u) is then calculated by using the Center Of Area (COA) method as the following. Assuming discretized membership functions, COA reveals

$$u(t) = \frac{\sum_{i=1}^{m_h} f_i \times m(i) \times w(i)}{\sum_{i=1}^{m_h} w(i) \times f_i} \tag{10}$$

where m_h is the number of nodes in the hidden layer which is equivalent to the number of rules used in the model. We define two more functions here:

$$z_i[t] = g(\sum_{j=1}^{n} d_{ij}[t]x_j[t]) \tag{11}$$

$$p[t] = g(\sum_{i=1}^{n} e_i[t]x_i[t] + \sum_{i=1}^{m_h} f_i[t]z_i[t]) \tag{12}$$

and

$$q[t] = \begin{cases} 1, \text{ with probability } p[t]; \\ 0, \text{ with probability } 1 - p[t] \end{cases} \tag{13}$$

The connection weights are updated according to the followings:

$$e_i[t+1] = e_i[t] + \rho\hat{r}[t+1](q[t] - p[t])x_i[t] \tag{14}$$

$$f_i[t+1] = f_i[t] + \rho\hat{r}[t+1](q[t] - p[t])z_i[t] \tag{15}$$

$$d_{ij}[t+1] = d_ij[t] + \rho_h\hat{r}[t+1]z_i[t](1 - z_i[t])sgn(f_i[t])(q[t] - p[t])x_j[t] \tag{16}$$

where ρ and $\rho_h > 0$.

3 Applying NFC to Cart-Pole Balancing

In this section, we describe the cart-pole balancing problem and apply the NFC model to its control.

3.1 The Cart-Pole balancing problem

In this system a pole is hinged to a motor-driven cart which moves on rail tracks to its right or its left. The pole has only one degree of freedom (rotation about the hinge point). The primary control tasks are to keep the pole vertically balanced and keep the cart within the rail tracks boundaries.

Four state variables are used to describe the system status, and one variable represents the force applied to the cart. These are:

x : horizontal position of the cart on the rail
$\dot{x}$: velocity of the cart
θ : angle of the pole with respect to the vertical line
$\dot{\theta}$: angular velocity of pole
u : force applied to the cart.

We assume that a failure happens when $| \theta | > 12$ degrees or $| x | > 2.4$ meters. Also, we assume that the equations of motion of the cart-pole system are not known to the controller and only a vector describing the cart-pole system's state at each time step is known. In other words, the cart-pole balancing system is treated as a black box by the learning system.

Figure 4 presents the model of NFC as it is applied to this problem. Among the components of this model, we only describe the Action Selection Network here.

3.2 The Action Selection Network

The action network was modeled by defining a multi-layered neural network which receives reinforcements from the evaluation network. This network, as shown in Figure 4, consists of 5 input nodes representing the four state variables and a bias unit, 13 nodes in the hidden layer, and an output node. The nodes in the hidden layer correspond to the fuzzy control rules. For example, node 1 corresponds to the rule:

IF θ is Positive and $\dot{\theta}$ is Positive Then Force is Positive-Large.

As mentioned earlier, the rule base of a fuzzy controller consists of rules which are described using *linguistic variables*. As shown in Figure 5(a) and Figure 5(b), three labels are used here to linguistically define the value of the state variables: Positive (P), Zero (Z), and Negative (N). Seven labels are used to linguistically define the value of force recommended by each control rule: Positive Large (PL), Positive Medium (PM), Positive Small (PS), Zero (ZE), Negative Small (NS), Negative Medium (NM), and Negative Large (NL). The forward calculations in this network is based on fuzzy logic control as described in [5], where nine fuzzy control rules were written for balancing the pole vertically and four control rules were used in positioning the cart at a specific location on the rail tracks. The presence of a connection between

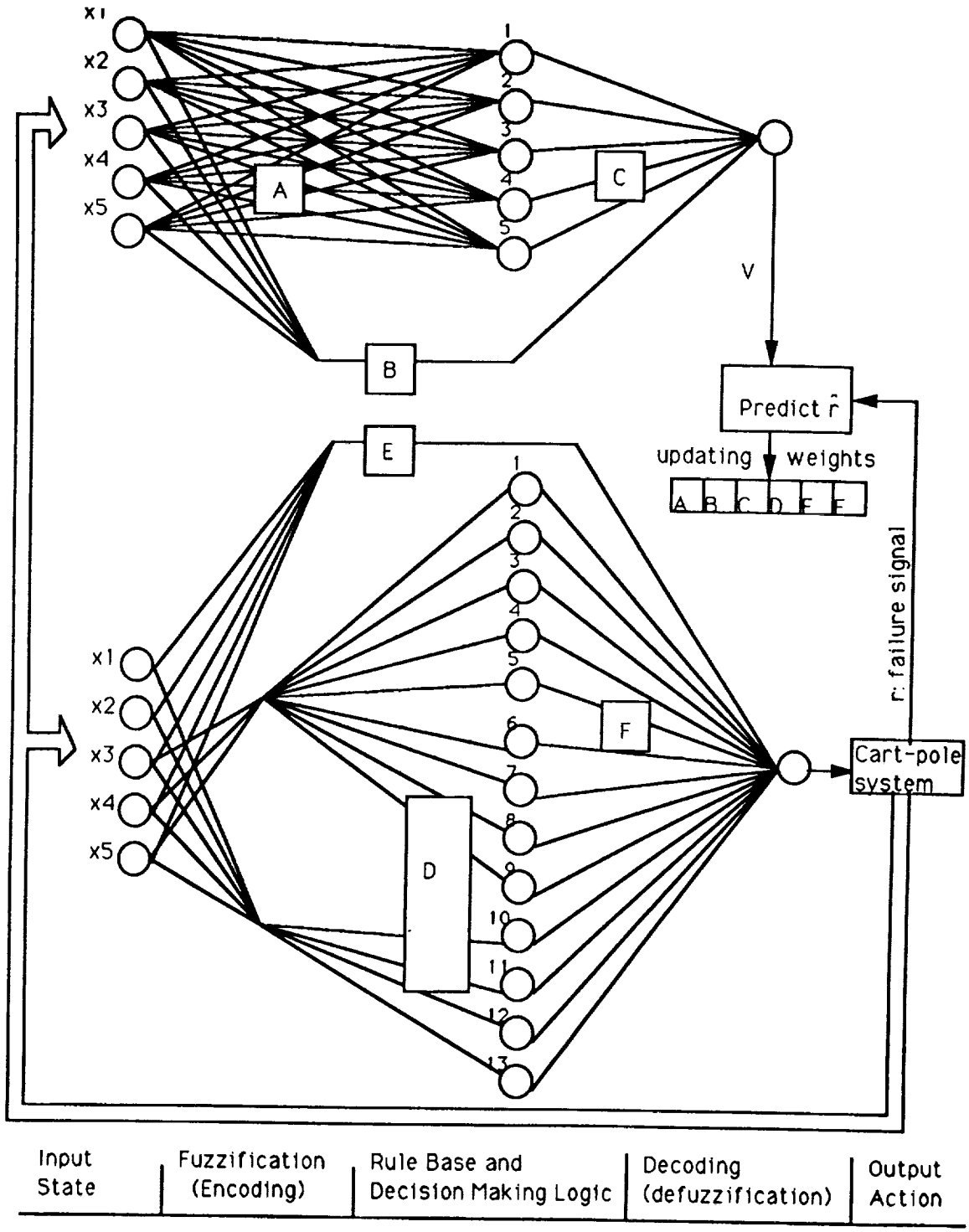


Figure 4: NFC applied to cart pole balancing

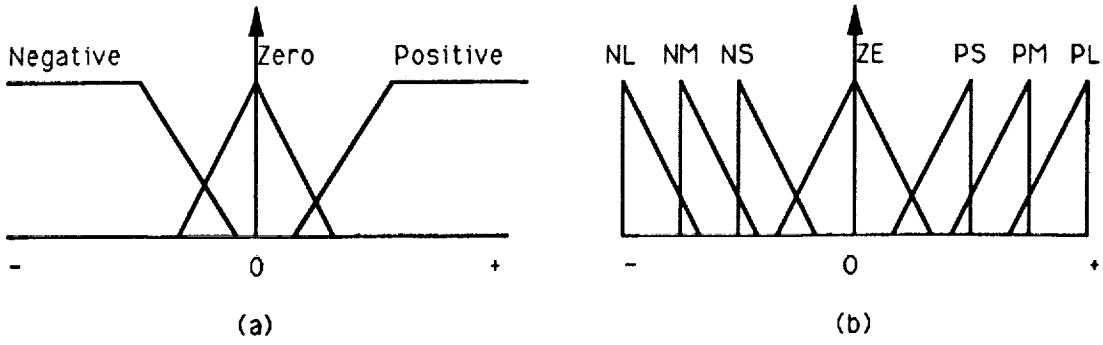


Figure 5: (a)- Three qualitative levels for $\theta, \dot{\theta}, x$, and $\dot{x}$, (b)- Seven qualitative levels for F

an input node j and a node i in the hidden layer indicates that the linguistic value of the input corresponding to node i is used as a precondition in rule i. As shown in Figure 4, the first nine rules, corresponding to the hidden layer nodes 1 to 9, are rules with two preconditions (i.e., θ, and $\dot{\theta}$). The rules 10 through 13 include four preconditions representing the linguistic values of θ, $\dot{\theta}$, x, and $\dot{x}$. In this network, D represents the matrix of connection weights between the input layer and the hidden layer, and F represents a vector of connection weights between the hidden layer and the output node. The amount of force applied to the cart is calculated using the equations (8) to (10) as were given in the last section.

4 Relation to other research

Credit Assignment The evaluation network in our work is similar to the Samuel's early work on credit assignment [10]. The Adaptive Heuristic Critic (AHC) model of Barto et. al. [3] provides a more general approach to credit assignment which learns by updating the predictions of failures. If no failure signal is present, the internal reinforcement provided by AHC is just the difference between the successive predictions of failure. Recently, Sutton [11] has formalized this method as the Temporal Difference methods.

Anderson's .Multi-layer networks We use the same structure as proposed by Anderson [2], however, the action selection network in our model is based on fuzzy logic control. Using the structure of a fuzzy controller, Anderson's approach is extended here to provide for the following attributes in NFC.

- The continuous representation of the output value.
- The inclusion of the human expert operator's control rules in terms of hidden units in the action selection network.

It should be noted that Anderson's goal in [1] was to discover the interesting patterns and strategy learning schemes. Not much effort was spent on making the process learn faster. In our work, although we allow some of the strategy learning to happen automatically, we start from a knowledge base of fuzzy control rules and fine-tune them as learning happens in the neural network.

Single Layer NeuroFuzzy Control Lee and Berenji [8] and Lee [7] have used a single layer neural network which requires the identification of the trace functions for keeping track of the visited states and their evaluations. The generation of these trace function is a difficult task in larger control problems. However, the approach suggested in the current paper does not use trace functions. The neural network representation of the fuzzy control rules in NFC allows faster development and faster learning. Also, in the single layer model, only the generation of the output values were considered. The preconditions of the fuzzy control rules were left untouched. However, in NFC, based on reinforcements received from the environment, both the preconditions and the conclusions of rules can be modified (i.e., fine-tuned).

5 Conclusion

A new model based on the reinforcement learning technique and fuzzy logic control was proposed which is applicable to control problems for which the analytical models of the process are unknown. The NFC model presented here improves the previous models in neurofuzzy control by learning to fine-tune the performance of a fuzzy logic controller.

ACKNOWLEDGEMENT My thanks to Yung-Yaw Chen for many useful discussions. Also, thanks to Charles Anderson for providing me with his software for cart-pole balancing with multi-layer neural networks.

References

[1] C. W. Anderson. *Learning and Problem Solving with Multilayer Connectionist Systems.* PhD thesis, University of Massachusetts, 1986.

[2] C. W. Anderson. *Strategy Learning with Multilayer Connectionist Representation.* Technical Report TR87-509.3, GTE Laboratories Inc., May 1988.

[3] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:834–846, 1983.

[4] R.E. Bellman and L.A. Zadeh. Fuzzy logic controllers. In L.A. Zadeh Yager, R. R., editor, *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, Kluwer Academic Publishers, (to appear).

[5] H.R. Berenji, Y.Y. Chen, C.C. Lee, J.S. Jang, and S. Murugesan. A hierarchical approach to designing approximate reasoning-based controllers for dynamic physical systems. In *Sixth Conference on Uncertainty in Artificial Intelligence*, pages 362–369, 1990.

[6] Y. Kasai and Y. Morimoto. Electronically controlled continuously variable transmission. In *Int. Congress on Transportation Electronics*, Dearborn, Michigan, 1988.

[7] C.C. Lee. Self-learning rule-based controller employing approximate-reasoning and neural-net concepts. *Int. Journal of Intelligent Systems*, 1990.

[8] C.C. Lee and H.R. Berenji. An intelligent controller based on approximate reasoning and reinforcement learning. In *Proc. of IEEE Int. Symposium on Intelligent Control*, Albany, NY, 1989.

[9] T. J. Procyk and E. H. Mamdani. A linguistic self-organizing process controller. *Automatica*, 15(1):15–30, 1979.

[10] A. L. Samuel. *Some Studies in Machine Learning Using the Game of Checkers.* Journal of R & D, IBM, 1959.

[11] R.S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[12] S. Yasunobu and S. Miyamoto. *Automatic Train operation by predictive fuzzy control*, pages 1–18. North-Holland, Amsterdam, 1985.

An Architecture for Designing Fuzzy Controllers using Neural Networks

Hamid R. Berenji

Artificial Intelligence Research Branch

NASA Ames Research Center

Outline

- Rule-based control by fuzzy logic
- Hierarchical fuzzy control
- A hybrid fuzzy logic-neural network controller model
    - Learning in neural networks
        - Supervised learning
            - error back-propagation
        - Unsupervised learning
            - Temporal Difference method
        - Evaluation network
        - Action network
- Conclusion

Approximate Reasoning and Control

Motivations

Human expert controllers perform well using approximate reasoning

Development of a mathematical physical model might not always be possible or might not be computationally feasible

Learning to control a physical system is regarded as one kind of intelligence

Fuzzy Sets and Rule-Based Control

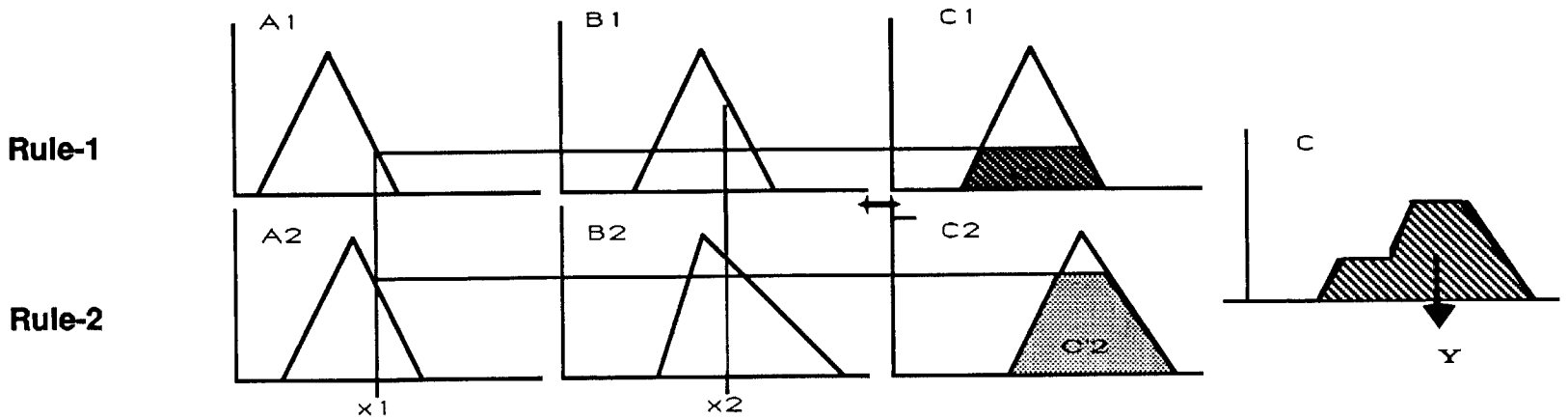
Labeling: Translation of a sensor reading to a label as done by a human expert controller

Example: IF Angular-position is *Positive* and Angular velocity is *Positive* THEN Force is *Positive-Large*

Conflict resolution: is required when more than one rule are triggered by one or more sensor readings.

Example: RULE-1 IF x1 is A1 and x2 is B1 THEN Y is C1

RULE-2 IF x1 is A2 and x2 is B2 THEN Y is C2



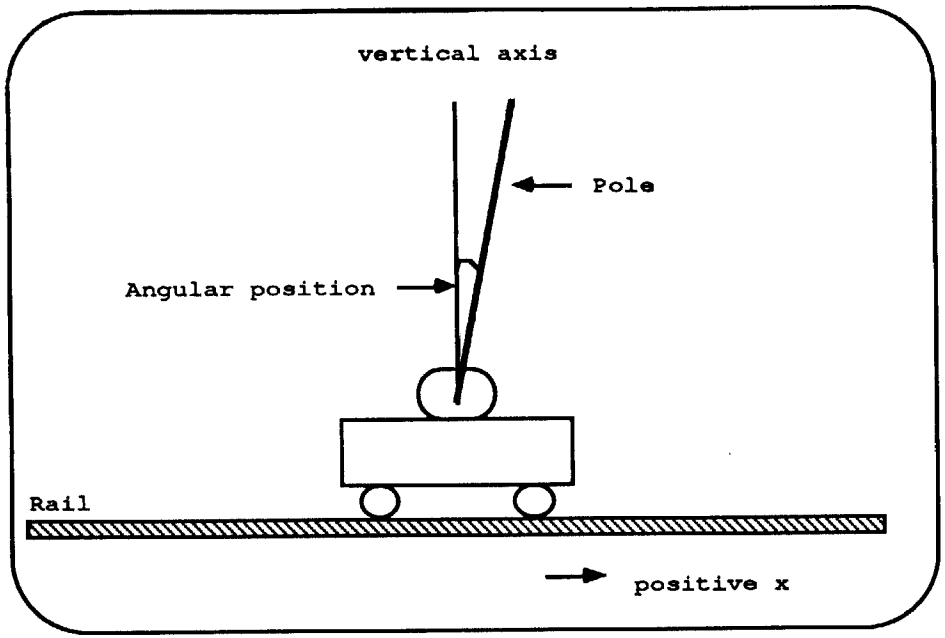
Hierarchical Fuzzy Control

1. Identify the set of goals that the system should achieve and maintain.
2. Assign priorities among goals.
3. Identify the set of input control parameters.
4. Identify the set of linguistic values to describe the values of the input control parameters.
5. Identify the set of linguistic values to describe the values of the output.
6. Acquire the set of rules directly related to the highest priority goal.
7. Acquire the next set of control rules for a lower priority goal combining aspects of approximately achieving the higher level goal e.g.,

   IF goal (i-1) is approximately achieved and U(i) is A THEN Z(i) is C

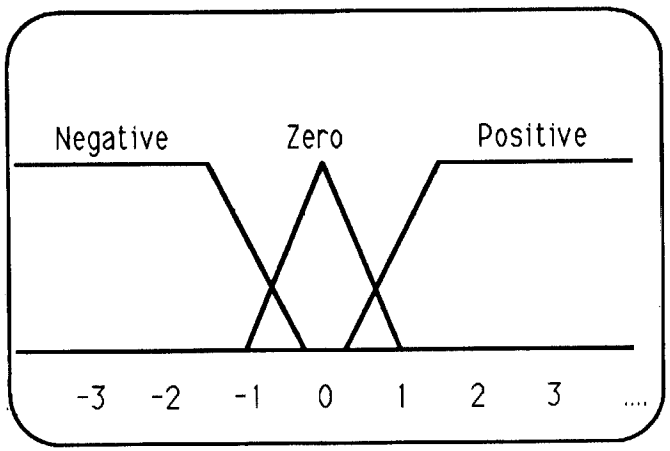
   where Z(i) is the set of ouputs at level (i).

The cart-pole balancing problem

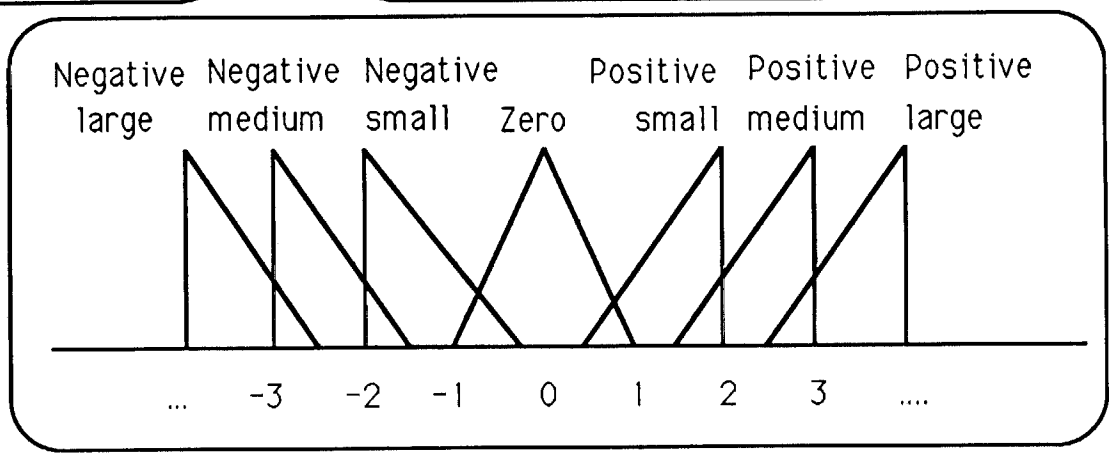


4 input and 1 output variables:

- **Pole Angle**
- **Angular Velocity**
- **Cart Position**
- **Cart Velocity**
- **Force applied to cart**



(a)



(b)

Hiearchical Fuzzy Control (Example:cart-pole balancing)

1. Goals: {position the cart at location x on the track, keep the pole balanced}.
2. Goal priorities:

   Goal 1: Keep the pole balanced

   Goal 2: position the cart at the location x
3. Control parameters: Pole angle, pole velocity, cart position, cart horizontal velocity}.
4. Linguistic values (input): {positive, zero,negative}.
5. Linguistic values (output):

   {positive-large, positive-medium,... negative-large}.
6. Rules for goal one, e.g.,

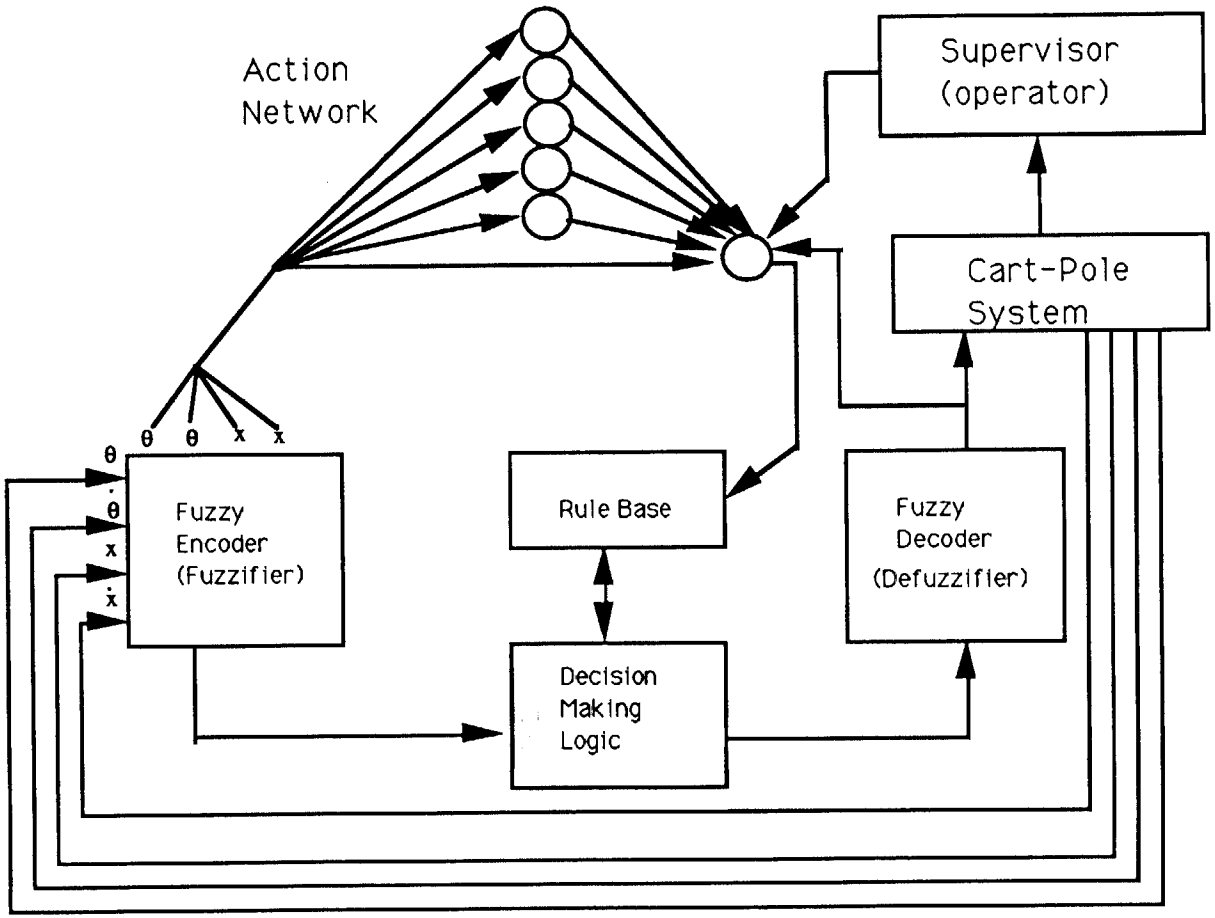
   If pole angle is positive and pole velocity is Zero then F is Positive-small.
7. Rules for lower level goals, e.g.,

   IF (pole angle is Very small and pole velocity is very small) and

   horizontal position is Positive and horizontal speed is Positive,

   THEN F is Positive-medium.

Hybrid Model (Supervised Learning)



Credit Assignment Problem

- A challenging topic of research in AI

- Given the performance (results) of a process, distribute awards or punishments to the individual elements

- In rule-based systems, this means assigning credit or blame to individual rules engaged in problem solving

- Examples: Samuel's checker program, and Michie and Chambers' BOXES system (which learned to balance a pole)

- Performance trace is crucially important in credit assignment

Temporal Difference (TD) Methods

A class of incremental learning procedures specialized for prediction problems. Learning occurs whenever there is a change in prediction over time (e.g., a weatherman's prediction on each day of the week about rain next Saturday).

Related Research:

- Samuel's checker-playing program
- Backpropagation in connectionism
- Holland's bucket brigade

Advantages:

- TD methods appear to learn faster than supervised-learning methods
- Require less memory than supervised learning
- Have many similarities to animal learning internal models of the world (i.e., Pavlovian or classical conditioning)

Disadvantages:

- TD methods can fail!!

TD Algorithms

Experience: Observation-outcome sequence $x_1, x_2, x_3, \ldots x_m, z$

Prediction : $P_1, P_2, P_3, \ldots P_m$ (estimates of z)

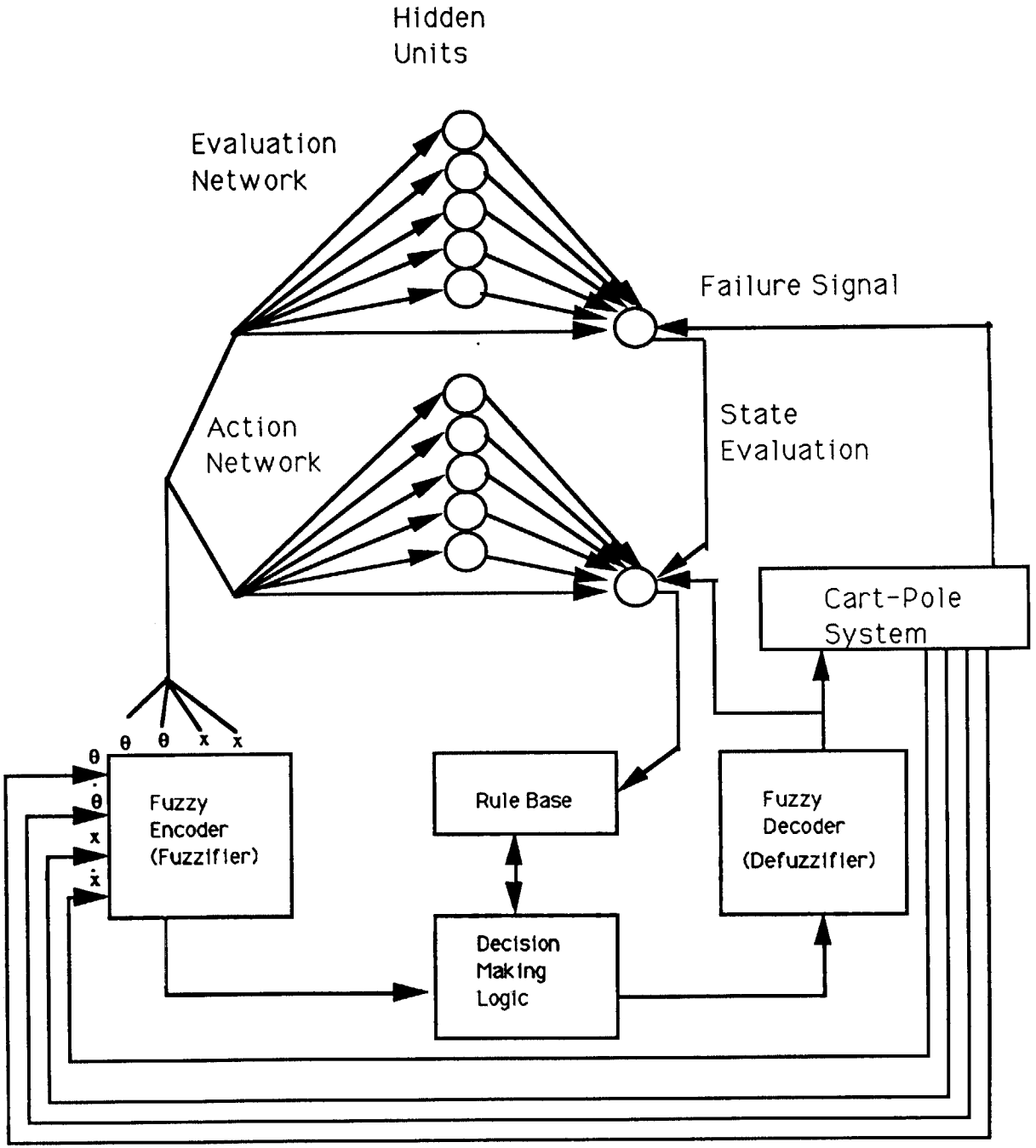
Assumption: P_t is a function of x_t (restrictive, but could be removed)

Weight w: Vector of modifiable parameters

After a complete sequence:

$$w \leftarrow w + \sum_{t=1}^{m} \Delta w_t$$

Hybrid Model (Unsupervised Learning)



Evaluation and Action networks

Evaluation network:

Apportions the blame for the failure among the actions in the sequence leading to a failure.

Action network:

Learns to select actions as a function of states.

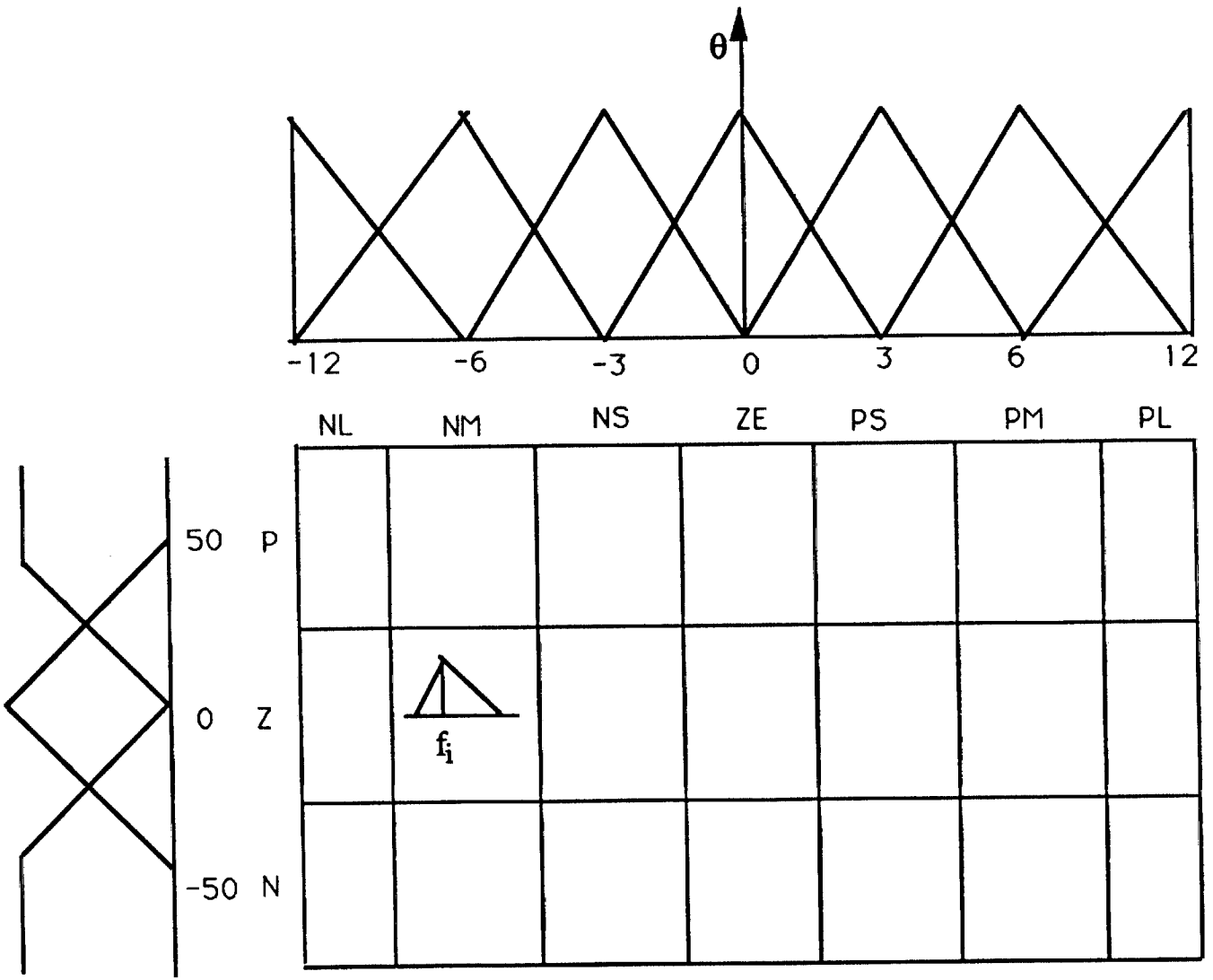
Comparison:
One-layer vs. Two-layer Networks

- With one-layer networks in the hybrid model, only the membership functions of the output control actions could be learned.

- With two-layer networks presented here, the membership functions of the input control parameters (used in the antecedent of rules) could also be learned.

- After forming helpful features, two-layer systems perform better.

Linguistic state description



Conclusion

- Extension to two-layer network allows the learning of the membership functions used in the antecedent of the rules.
- Approximate Reasoning based controllers could become able to learn from experience by using unsupervised learning algorithms such as the Temporal Difference Methods and supervised learning algorithms such as the error backpropagation.

AN OVERVIEW OF THE NEURON RING MODEL

Rod Taber
Center for Applied Optics
University of Alabama in Huntsville
Huntsville, AL 35899

ABSTRACT

The Neuron Ring model employs an avalanche structure with two important distinctions at the neuron level. Each neuron has two memory latches; one traps maximum neuronal activation during pattern presentation, and the other records the time of latch content change. The latches filter short term memory. In the process, they preserve length 1 snapshots of activation history.

The model finds utility in pattern classification. Its synaptic weights are first conditioned with sample spectra. The model then receives a test or unknown signal. The objective is to identify the sample closest to the test signal. Class decision follows complete presentation of the test data. The decision maker relies exclusively on the latch contents.

This paper presents an overview of the Neuron Ring at the seminar level. The appendix contains the information in slide format.

REFERENCES

1. Grossberg, S., "Some Networks that can Learn, Remember, and Reproduce any Number of Complicated Space-Time Patterns I," Stud.Appl.Math, vol. 49, 1970.
2. Hecht-Nielsen, R., "Nearest Matched Filter Classification of Spatiotemporal Patterns," Appl.Optics., vol. 26, pp. 1892-1898, 1987.
3. Taber, R., and Deich, R.O., "Fuzzy Sets and Neural Networks," NASA Proc. First Joint Workshop on Neural Networks and Fuzzy Logic, Houston, Texas, April, 1988.
4. Taber, R., Deich, R.O., Simpson, P.K., and Fagg, A.H., "The Recognition of Orca Calls with a Neural Network,", Int. Workshop on Fuzzy System Applications, Kyushu Institute of Technology, Iizuka, Japan, August, 1988.

5. Taber, R., "The Evolving Neuron and Fuzzy Logic," Approximate Reasoning Tools in Artificial Intelligence, Ed. Verdegay, J.L. and Delgado, M., Verlag TUV Rheinland, Köln, Germany, January, 1990.

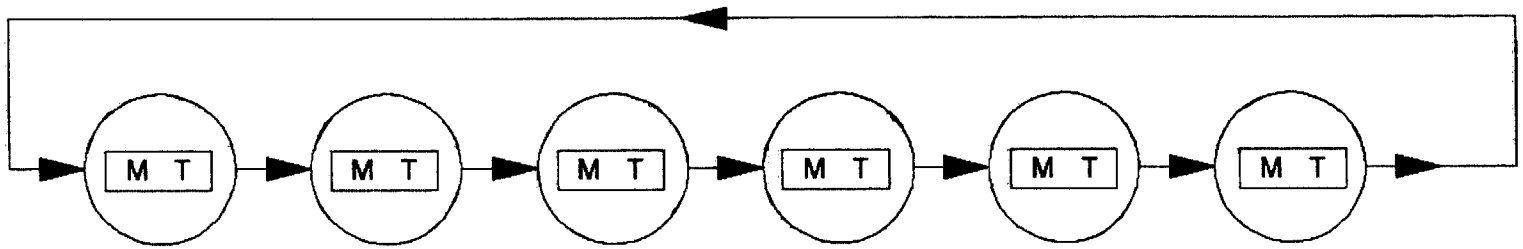
APPENDIX

The next fifteen pages are slides describing the Neuron Ring model.

NEURON RING
SPATIOTEMPORAL PATTERN RECOGNIZER

- EVOLVED FROM GROSSBERG AVALANCHE
- DEVELOPED BY TABER, DEICH 1987
- USES NEW NEURON CALLED DPNL
  - Dot Product Neuron with Latches
- DEFINITION: SPATIOTEMPORAL PATTERN
  - A PATTERN EQUIVALENT TO A SECTION OF A SONG

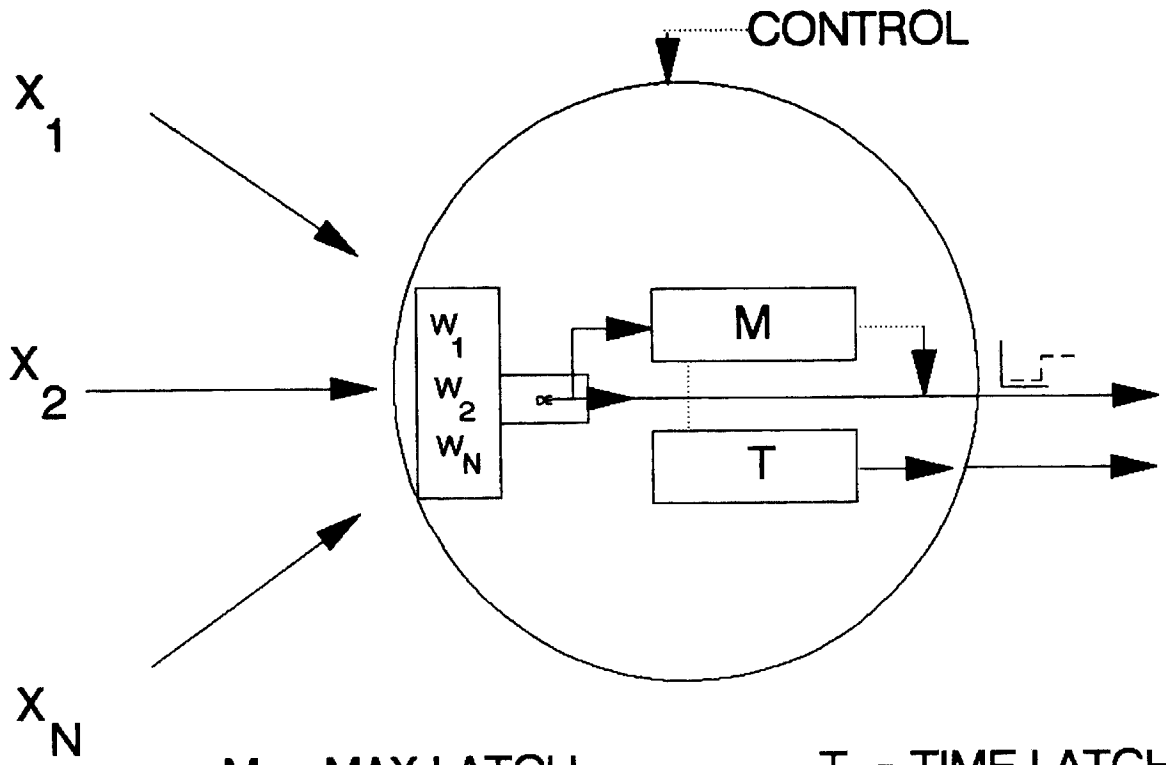
NEURON RING
GROSS ARCHITECTURE



DPNL 1 DPNL 2 ... DPNL N

SINGLE RING FOR A SINGLE SIGNAL

DOT PRODUCT NEURON WITH LATCHES
INTERNALS



M = MAX LATCH T = TIME LATCH

DE = DIFFERENTIAL EQUATION FOR DX/DT

NEURON RING
DIFFERENTIAL EQUATION IN DIFFERENCE FORM

- $X_i^{NEW} = X_i^{OLD} + A(-aX_i^{OLD} + b[I_i^1 + I_i^2])$

= DOT PRODUCT OF WEIGHTS AND ACTIVATION

$[e]^+ = 0$ IF $e \leq \Phi$ ELSE $[e]^+ = e$

$A(e) = e$ IF $e > \Omega$ ELSE $A(e) = ce$

A = ATTACK FUNCTION

b = AMPLIFICATION FACTOR

c Φ Ω = CONSTANTS

NEURON RING
TRAINING

- TRAINING IS FAST
  - IN DIAGRAM WEIGHTS ARE INTERNAL TO DPNL
  - WEIGHT VECTOR = TEMPLATE
  - ACTIVATE CONTROL LINES
  - COPY TEMPLATE TO THE WEIGHT REGISTERS
  - DEACTIVATE CONTROL LINES
  - RING IS READY FOR EXCITATION
- WORKS BEST IF TEMPLATE = CENTROID OF PATTERN CLASS
  - CENTROID CAN BE FOUND VIA AVQ

NEURON RING
FUZZY POST-PROCESSING

- ANALYSIS IS BASED ON [M] AND [T] LATCHES THESE DRIVE CHIP OUTPUT LINES
- DOT PRODUCT IS MEASURE OF SIGNAL-TEMPLATE MATCH
- IF EACH SIGNAL-TEMPLATE PAIR MATCH THEN DONE
- USUALLY, PERFECT MATCH IS NOT POSSIBLE
- NEXT BEST THING IS TO ESTIMATE DEGREE OF MATCH
  - THIS IS DONE VIA FUZZY THEORY

NEURON RING

FUZZY THEORY

- EACH DPNL CONTAINS WEIGHT VECTOR = SIGNAL SECTION
  - DPNL 1 CONTAINS FIRST X MILLISECONDS OF SIGNAL
  - DPNL 2 CONTAINS SECOND X MILLISECONDS
  - DPNL N CONTAINS LAST X MILLISECOND SLICE
- EACH DPNL IS A MINATURE FEATURE DETECTOR
- EACH DPNL RECEIVES,DURING COURSE OF ACTIVATION, ALL OF THE TEST SIGNAL
- THERFORE
  - N DPNLs IN A RING GENERATE N ACTIVATIONS EACH
- AT END OF ACTIVATION
- THIS IS KEY:
- EACH M LATCH CONTAINS A MEASURE OF BEST MATCH

NEURON RING
POSSIBILITY DISTRIBUTION AND SIGMA-COUNT

- **IMAGINE ALL DPNLs FULLY EXCITED AFTER TEST PRESENTATION**
  - **PERFECT = (1,1,1,...1)**
  - **BUT WE HAVE:**
  - **OBSERVED = ($M_1 M_2 M_3 \ldots M_N$)**
  - **WHERE M REAL NUMBER IN (0,1)**
- **COMPUTE POSSIBILITY (OBSERVED IS PERFECT)**
  - **POSS(O IS P) = OBSERVED ∩ PERFECT = PAIRWISE MIN**
  - **THEREFORE POSS(O IS P) = OBSERVED M VECTOR**

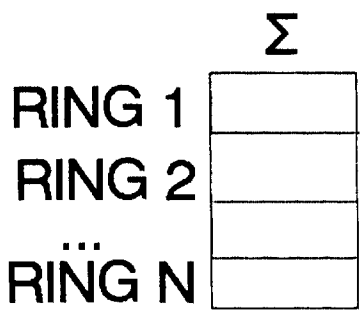
NEXT STEP: COMPUTE ZADEH'S Σ-COUNT = DEGREE OF MATCH

NEURON RING
SIGMA-COUNT = DEGREE OF SIGNAL-TEMPLATE MATCH

- Σ-COUNT = FUZZY SET CARDINALITY = ARITHMETIC ADDITION
  - EXAMPLE:
    - F = {.1 .2 .6}
    - Σ(F) = .1 + .2 + .6 = .9
    - %MATCH = 100(ΣCOUNT\PERFECT)
- EXAMPLE:
  - %MATCH = 100(.9\3) = 30%
- WE WOULD SAY THIS RING MATCHES THE TRAINING TEMPLATE TO DEGREE .3

NEURON RING
EXTENSION TO MULTIPLE SIGNALS

- GENERATE 1 RING PER SIGNAL
  - EXCITE EACH RING WITH SAME TEST SIGNAL
  - FORM TABLE, ONE ROW PER SIGNAL



NEURON RING
MULTIPLE SIGNALS

- FORM VECTOR Q = {Σ Σ ... Σ)
  - FORM R = Q\MAX(Q)
- BAYES DECISION RULE:
  - TEST SIGNAL = SIGNAL I IF Σ_I = MAX(R)
- FIND MAX VALUE IN VECTOR R, INDEX IS THE SIGNAL

(MAX OF R WILL BE 1!)

NEURON RING
NOTES

- ORDERED VECTOR R IS NOT PROBABILITY !
- EACH ELEMENT IN R IS SUPPORT FOR THE HYPOTHESIS: THE SIGNAL IS i
- PROBABILITY WOULD REQUIRE WE HAVE OTHER INFORMATION
  - WE DON'T - JUST ACOUSTIC SIGNATURES

NEURON RING

FURTHER REFINEMENTS

- SAMPLING PHASE TOLERANCE FOR SINGLE RING
- T LATCH WITHIN DPNL CONTAINS TIME THE MAX LATCH CHANGED
  - FORM VECTOR U = $\{T_1 T_2 T_3 \ldots T_N\}$
    - U SHOULD BE IN ASCENDING ORDER
- EXAMPLE U = {1 PM 2 PM 3 PM ...)
- IF NOT, TEST SAMPLE SECTIONS RECEIVED OUT OF ORDER
- SORT U TO DISCOVER ORDER
- SUPPOSE U = {2 PM 1PM 3 PM) THEN INSTEAD OF ABC, WE RECEIVED BAC

NEURON RING

A EUCLIDEAN 2-D METRIC BASED ON M AND T

- WE HAVE THE Σ METRIC ALREADY
- DEVELOP DISTANCE METRIC BY
  - COUNTING # SWAPS NEEDED TO BRING U BACK TO ORDER

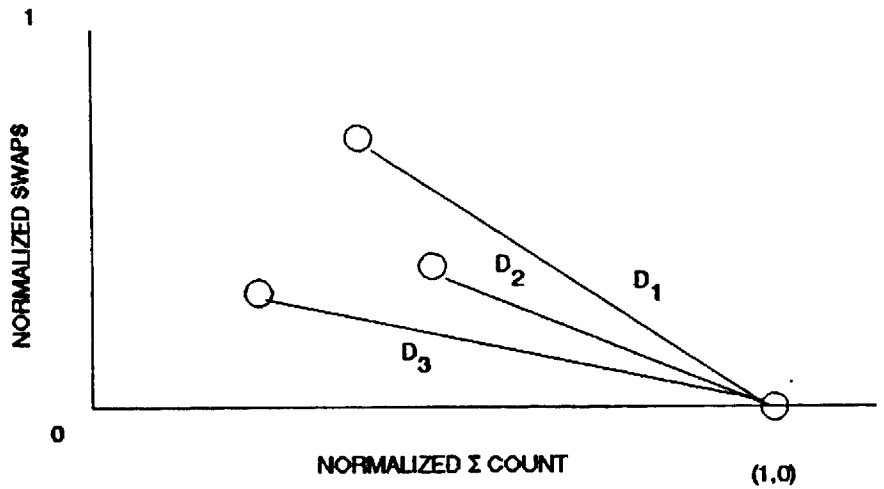
$$D_i = [\,(1-X_i)^2 + Y_i^2\,]^{1\backslash 2}$$

X = NORMALIZED Σ COUNT FOR RING i

Y = NORMALIZED NUMBER OF SWAPS (i.e. #\(N-1) FOR N RINGS)

NEURON RING
2-D DISTANCE METRIC

- IF ALL NEURONS FIRED MAXIMALLY AND IN ORDER OUR METRIC POINT WOULD BE (1,0)
  - 1 = MAXIMUM NORMALIZED Σ
  - 0 = NO SWAPS REQUIRED
- RESULT OF TEST RUN (X,Y)
- $0 \leq X \leq 1, 0 \leq Y \leq 1$
- DISTANCE AWAY FROM PERFECT ...



SIGNALS MAY FALL INTO NATURAL CLUSTERS

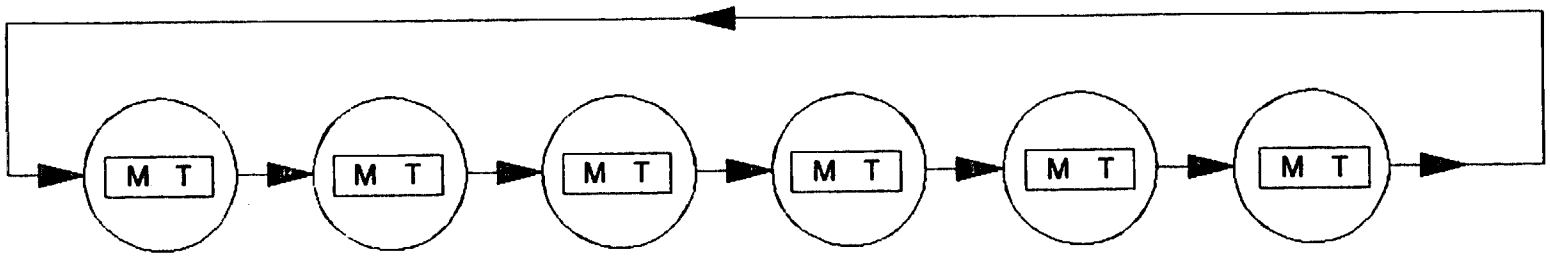
EQUIDISTANT SIGNALS ARE IN THE SAME EQUIVALENCE CLASS

NEURON RING
SPATIOTEMPORAL PATTERN RECOGNIZER

- EVOLVED FROM GROSSBERG AVALANCHE
- DEVELOPED BY TABER, DEICH 1987
- USES NEW NEURON CALLED DPNL
  - Dot Product Neuron with Latches
- DEFINITION: SPATIOTEMPORAL PATTERN
  - A PATTERN EQUIVALENT TO A SECTION OF A SONG

NEURON RING

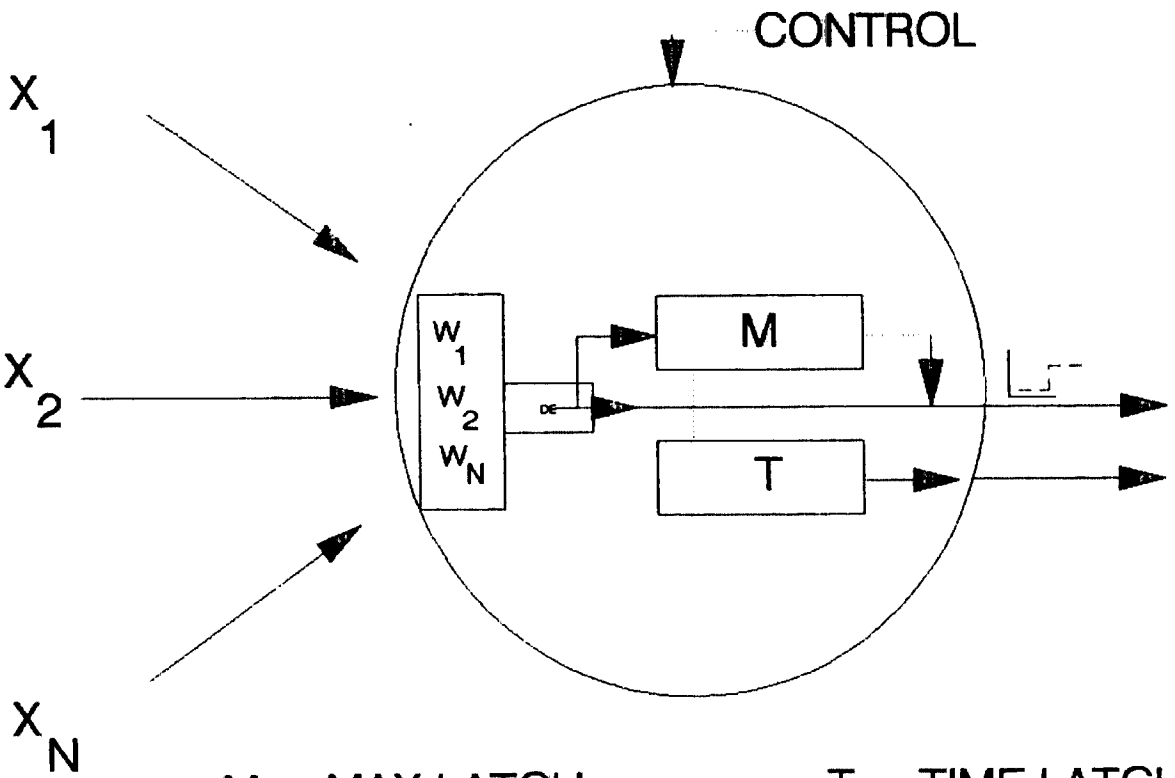
GROSS ARCHITECTURE



DPNL 1 DPNL 2 ... DPNL N

SINGLE RING FOR A SINGLE SIGNAL

DOT PRODUCT NEURON WITH LATCHES
INTERNALS



M = MAX LATCH T = TIME LATCH

DE = DIFFERENTIAL EQUATION FOR DX/DT

NEURON RING
DIFFERENTIAL EQUATION IN DIFFERENCE FORM

- $X_i^{NEW} = X_i^{OLD} + A(-aX_i^{OLD} + b[I_i^1 + I_i^2]^+)$

= DOT PRODUCT OF WEIGHTS AND ACTIVATION

$[e]^+ = 0$ IF $e \leq \Phi$ ELSE $[e]^+ = e$

$A(e) = e$ IF $e > \Omega$ ELSE $A(e) = ce$

A = ATTACK FUNCTION

b = AMPLIFICATION FACTOR

c Φ Ω = CONSTANTS

NEURON RING
TRAINING

- TRAINING IS FAST
  - IN DIAGRAM WEIGHTS ARE INTERNAL TO DPNL
  - WEIGHT VECTOR = TEMPLATE
  - ACTIVATE CONTROL LINES
  - COPY TEMPLATE TO THE WEIGHT REGISTERS
  - DEACTIVATE CONTROL LINES
  - RING IS READY FOR EXCITATION
- WORKS BEST IF TEMPLATE = CENTROID OF PATTERN CLASS
  - CENTROID CAN BE FOUND VIA AVQ

NEURON RING
FUZZY POST-PROCESSING

- ANALYSIS IS BASED ON [M] AND [T] LATCHES THESE DRIVE CHIP OUTPUT LINES
- DOT PRODUCT IS MEASURE OF SIGNAL-TEMPLATE MATCH
- IF EACH SIGNAL-TEMPLATE PAIR MATCH THEN DONE
- USUALLY, PERFECT MATCH IS NOT POSSIBLE
- NEXT BEST THING IS TO ESTIMATE DEGREE OF MATCH
  - THIS IS DONE VIA FUZZY THEORY

NEURON RING

FUZZY THEORY

- EACH DPNL CONTAINS WEIGHT VECTOR = SIGNAL SECTION
  - DPNL 1 CONTAINS FIRST X MILLISECONDS OF SIGNAL
  - DPNL 2 CONTAINS SECOND X MILLISECONDS
  - DPNL N CONTAINS LAST X MILLISECOND SLICE
- EACH DPNL IS A MINATURE FEATURE DETECTOR
- EACH DPNL RECEIVES,DURING COURSE OF ACTIVATION, ALL OF THE TEST SIGNAL
- THERFORE
  - N DPNLs IN A RING GENERATE N ACTIVATIONS EACH
- AT END OF ACTIVATION
- THIS IS KEY:
- EACH M LATCH CONTAINS A MEASURE OF BEST MATCH

NEURON RING
POSSIBILITY DISTRIBUTION AND SIGMA-COUNT

- IMAGINE ALL DPNLs FULLY EXCITED AFTER TEST PRESENTATION
  - PERFECT = (1,1,1,...1)
  - BUT WE HAVE:
  - OBSERVED = ($M_1 M_2 M_3 ... M_N$)
  - WHERE M REAL NUMBER IN (0,1)
- COMPUTE POSSIBILITY (OBSERVED IS PERFECT)
  - POSS(O IS P) = OBSERVED ∩ PERFECT = PAIRWISE MIN
  - THEREFORE <u>POSS(O IS P) = OBSERVED M VECTOR</u>

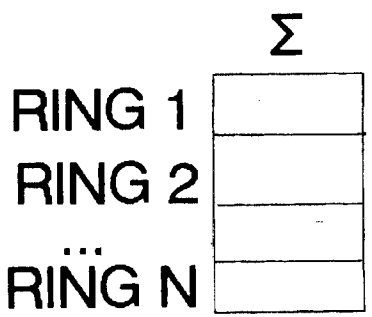
NEXT STEP: COMPUTE ZADEH'S Σ-COUNT = DEGREE OF MATCH

NEURON RING
SIGMA-COUNT = DEGREE OF SIGNAL-TEMPLATE MATCH

- Σ-COUNT = FUZZY SET CARDINALITY = ARITHMETIC ADDITION
  - EXAMPLE:
    - F = {.1 .2 .6}
    - Σ(F) = .1 + .2 + .6 = .9
    - %MATCH = 100(ΣCOUNT\PERFECT)
- EXAMPLE:
  - %MATCH = 100(.9\3) = 30%
- WE WOULD SAY THIS RING MATCHES THE TRAINING TEMPLATE TO DEGREE .3

NEURON RING
EXTENSION TO MULTIPLE SIGNALS

- GENERATE 1 RING PER SIGNAL
  - EXCITE EACH RING WITH SAME TEST SIGNAL
  - FORM TABLE, ONE ROW PER SIGNAL



NEURON RING
MULTIPLE SIGNALS

- FORM VECTOR Q = {Σ Σ ... Σ)
  - FORM R = Q\MAX(Q)
- BAYES DECISION RULE:
  - TEST SIGNAL = SIGNAL I IF Σ_I = MAX(R)
- FIND MAX VALUE IN VECTOR R, INDEX IS THE SIGNAL

(MAX OF R WILL BE 1!)

NEURON RING
NOTES

- ORDERED VECTOR R IS NOT PROBABILITY !
- EACH ELEMENT IN R IS SUPPORT FOR THE HYPOTHESIS: THE SIGNAL IS i
- PROBABILITY WOULD REQUIRE WE HAVE OTHER INFORMATION
  - WE DON'T - JUST ACOUSTIC SIGNATURES

NEURON RING

FURTHER REFINEMENTS

- SAMPLING PHASE TOLERANCE FOR SINGLE RING
- T LATCH WITHIN DPNL CONTAINS TIME THE MAX LATCH CHANGED
  - FORM VECTOR U = $\{T_1 T_2 T_3 \ldots T_N\}$
    - U SHOULD BE IN ASCENDING ORDER
- EXAMPLE U = {1 PM 2 PM 3 PM ...)
- IF NOT, TEST SAMPLE SECTIONS RECEIVED OUT OF ORDER
- SORT U TO DISCOVER ORDER
- SUPPOSE U = {2 PM 1PM 3 PM) THEN INSTEAD OF ABC, WE RECEIVED BAC

NEURON RING

A EUCLIDEAN 2-D METRIC BASED ON M AND T

- WE HAVE THE Σ METRIC ALREADY
- DEVELOP DISTANCE METRIC BY
  - COUNTING # SWAPS NEEDED TO BRING U BACK TO ORDER

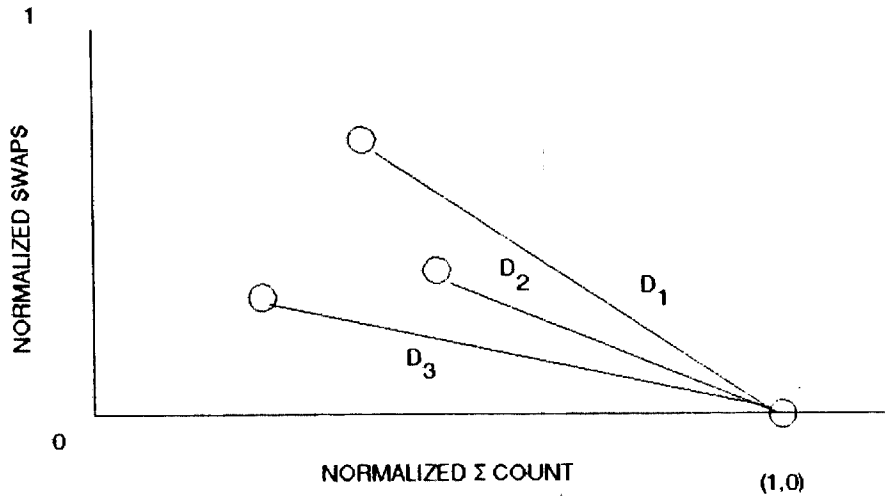
$$D_i = [\,(1-X_i)^2 + Y_i^2\,]^{1\backslash 2}$$

X = NORMALIZED Σ COUNT FOR RING i

Y = NORMALIZED NUMBER OF SWAPS (i.e. #\(N-1) FOR N RINGS)

NEURON RING
2-D DISTANCE METRIC

- IF ALL NEURONS FIRED MAXIMALLY AND IN ORDER OUR METRIC POINT WOULD BE (1,0)
  - 1 = MAXIMUM NORMALIZED Σ
  - 0 = NO SWAPS REQUIRED
- RESULT OF TEST RUN (X,Y)
- $0 \leq X \leq 1, 0 \leq Y \leq 1$
- DISTANCE AWAY FROM PERFECT ...



SIGNALS MAY FALL INTO NATURAL CLUSTERS

EQUIDISTANT SIGNALS ARE IN THE SAME EQUIVALENCE CLASS

N91-20814

A Space-Time Neural Network

James A. Villarreal and Robert O. Shelton

Software Technology Branch/PT4
Information Systems Directorate
Lyndon B. Johnson Space Center
Houston, Texas 77058

ABSTRACT

Neural network algorithms have impressively demonstrated the capability of modelling spatial information. On the other hand, the application of parallel distributed models to processing of temporal data has been severely restricted. This work introduces a novel technique which adds the dimension of time to the well known backpropagation neural network algorithm. The paper cites several reasons why the inclusion of automated spatial and temporal associations are crucial to effective systems modelling. An overview of other works which also model spatiotemporal dynamics is furnished. In addition, a detailed description of the processes necessary to implement the space-time network algorithm is provided. The reader is given several demonstrations which illustrate the capabilities and performance of this new architecture.

INTRODUCTION

Throughout history, the meaning of time has plagued the minds of mankind. The wise Greek philosophers, Socrates, Plato, and Aristotle pondered deeply with what the influence of time had on human knowledge. The English poet, Ben Johnson, wrote "O for an engine to keep back all clocks" giving voice to our ageless lament over the brevity of human life. The great scientist, Einstein, who developed the theory of relativity, believed that space and time cannot be considered separately, but that they depend upon one another.

A need for space-time knowledge capture representation is clearly evident. Human cognitive thought processes involve the use of both space and time. A childhood event is remembered by an occurrence (or space) and its associated place in time. We speak of an event which occurred a specific time ago. Linguistic meanings are expressed in a manner in which proper temporal order plays a crucial role in the conveyance of a concept. Take, for example, the phrases "house cat" and "cat house". Speech production, too, is very order dependent -- subtleties in intonations may change the whole meaning of a concept. The more advanced engineering systems have characteristics which vary over time. For instance, complex machines such as the Space Shuttle Main Engine are abound with sensors, each varying over the life of the machine's operation. A model which is capable of automatically associating spatial information with its appropriate position in time becomes increasingly significant.

Also, microscopic level investigations reveal a need to incorporate time or sequence discovery and adaptation into the modelling framework. It is clearly

evident that information exchange at the neuronal level occurs through a rich interchange of complex signals. Freeman [3] and Baird [1] have done extensive research on the olfactory bulb at anatomical, physiological, and behavioral levels. Their findings have shown that information in biological networks takes the form of space-time neural activity patterns. These dynamic space-time patterns encode past experience, attempt to predict future actions, and are unique to each biological network.

As seen in figure 1, the "classical" neuron has several dendrites which receive information from other neurons. The soma or cell body performs a wide range of functions; it processes information from the dendrites in a manner which is not entirely understood and also maintains the cell's health. The information processed by the neuron is distributed by its axon to other interconnected neurons by the propagation of a spike or action potential. Along each dendrite are thousands of protrusions where neurons exchange information through a process known as synapse. The synaptic cleft releases chemicals known as neurotransmitters. Even at this microscopic level, the relevance for time adaptive neural networks becomes clearly evident. Synaptic clefts take on roles such as neurotransmitter modulators, generators, and filters which cloud the neuron's inner workings and render these ever changing dynamical properties especially difficult to study.

Connectionist architectures have impressively demonstrated several models of capturing spatial knowledge. To accomplish this, the most popular solution has been to distribute a temporal sequence by forcing it into a spatial representation. This approach has worked well in some instances [11]. But there are problems with this approach and it has ultimately prove inadequate.

A REVIEW OF NEURAL NETWORKS

A network is comprised of numerous, independent, highly interconnected processing elements. For backpropagation networks, each element can be characterized as having some *input* connections from other processing elements and some *output* connections to other elements. The basic operation of an element is to compute its *activation value* based upon its inputs and send that value to its output elements. Figure 2 shows a schematic of a processing element. Note that this element has j input connections coming from j input processing elements. Each connection has an associated value called a *weight*. The output of this processing element is a non-linear transform of its summed, continuous-valued inputs by the sigmoid transformation in (1) and (2). Understanding the details of this transformation is not essential here; the interested reader will find an excellent description of such details provided by Rummelhart et. al.[8].

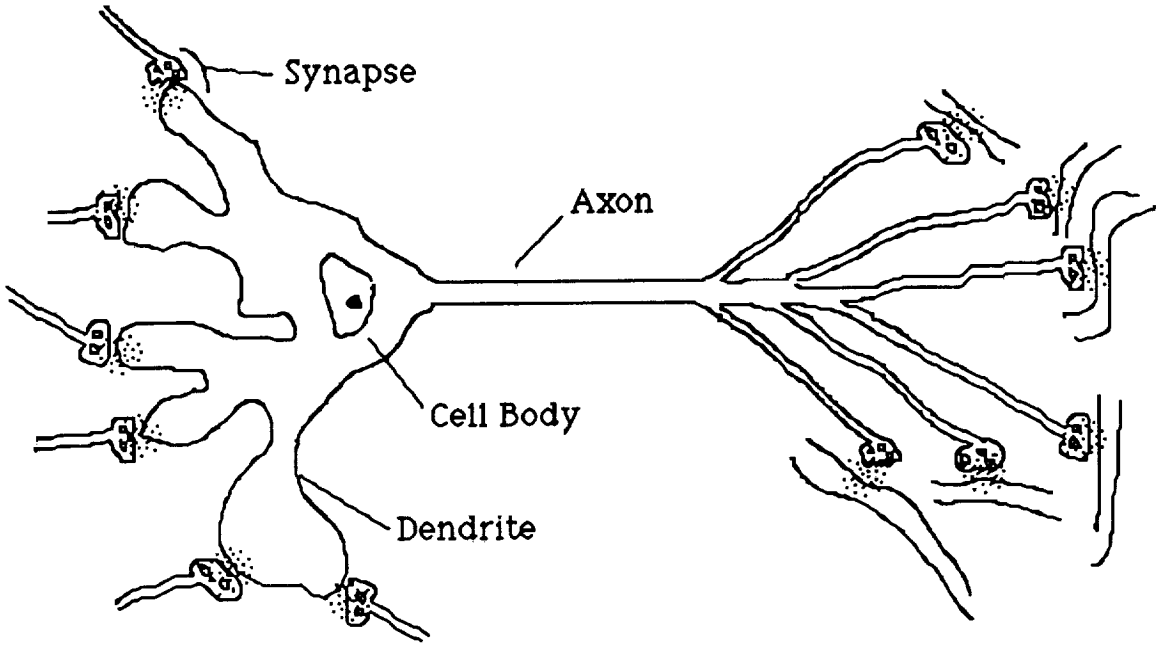


Figure 1 .... Schematized diagram of the classical neuron.

When groups of processing elements are arranged in sequential layers, each layer interconnected with the subsequent layer, the result is a wave of activations propagated from the input processing elements, which have no incoming connections, to the output processing elements. The layers of elements between the inputs and outputs take on intermediate values which perform a mapping from the input representation to the output representation. It is from these intermediate or *hidden* elements that the backpropagation network draws its generalization capability. By forming transformations through such intermediate layers, a backpropagation network can arbitrarily categorize the features of its inputs.

$$E_i = \Sigma w_{ij}\, p_j \tag{1}$$

$$p_i = P(E_i) = \frac{1}{1 + e^{-E_i}} \tag{2}$$

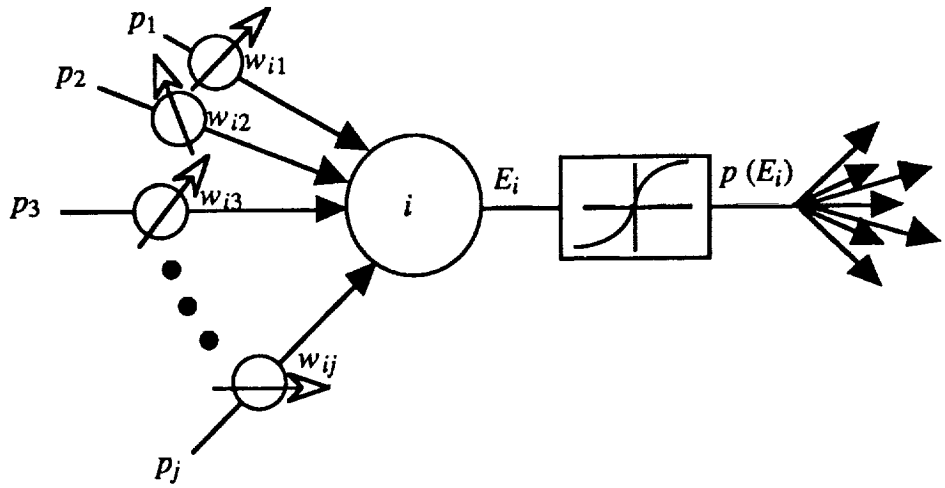


Figure 2 .... Processing element in a backpropagation network.

THE WEIGHTS OF A BACKPROPAGATION NETWORK

The heart of the backpropagation algorithm lies in how the values of its interconnections, or weights, are updated. Initially, the weights in the network are set to some small random number to represent no association between processing elements. Upon being given a set of patterns representing pairs of input/output associations, the network enters what is called a *training* phase. During training, the weights are adjusted according to the learning technique developed by Rumelhart et. al. The training phase is modelled after a behavioristic approach which operates through reinforcement by negative feedback. That is, the network is given an input from some input/output pattern for which it generates an output by propagation. Any discrepancies found when comparing the network's output to the desired output constitute mistakes which are then used to alter the network characteristics. According to Rumelhart's technique, every weight in the network is adjusted to minimize the total mean square errors between the response of the network, $p_{pi,}$ and the desired outputs, t_{pi} ,to a given input pattern. First, the error signal, δi, is determined for the output layer, N:

$$\delta_i^{(N)} = (t_i - p_i^{(N)}) P'(E_i^{(N)}) \qquad (3)$$

The indices *p* and *i* represent the pattern number and the index to a node respectively. The weights are adjusted according to:

$$\Delta w_{ij}^{(n+1)} = \alpha \Delta w_{ij}^{(n)} + \eta \delta_i^{(n+1)} P_j^{(n)} \qquad (4)$$

where $\Delta w_{ij}^{(n)}$ is the error *gradient* of the weight from the *j*th processing element in layer *n* to the *i*th unit in the subsequent layer *(n + 1)*. The parameter α, performs a damping effect through the multi-dimensional error space by relying on the most recent weight adjustment to determine the present

adjustment. The overall effect of this weight adjustment is to perform a gradient descent in the error space; however, note that true gradient descent implies infinitesimally small increments. Since such increments would be impractical, η is used to accelerate the learning process. In general, then, the errors are recursively back propagated through the higher layers according to:

$$\delta_i^{(n)} = \sum_j \delta_j^{(n+1)} w_{ji}^{(n)} P'(E_i^{(n)}) \tag{5}$$

where *P'(E)* is the first derivative of *P(E)*.

OTHER SPATIOTEMPORAL NEURAL NETWORK ARCHITECTURES

Advances in capturing spatial temporal knowledge with neural network architectures have been made by Jordan[4] and Elman[2]. Jordan approaches this problem by partitioning the input layer in a connectionist network into separate plan and state layers. In essence, Jordan's architecture acts as a backpropagation network, except for the specialized processing elements in the state layer, which receive their inputs from the output units, as well as from recurrent connections which allow the state layer elements to "remember" the network's most recent state. In other words, the state units behave as pseudo inputs to the network providing a past-state history. Here, a recurrent connection is one in which it is possible to follow a path from an element back onto itself as shown in figure 3. Recurrent networks of this type allow the element's next state to be not only dependent on external inputs, but also on the state of the network at its most previous time step. For further discussion of this network's operation refer to Jordan. In general, however, this network is trained to reproduce a predetermined set of sequence patterns from a static input pattern. One of the authors (Villarreal), used this network architecture extensively in developing a speech synthesizer. The inputs to the speech synthesis network represented a tri-phoneme combination and the output was partitioned to represent the various vocal tract components necessary to produce speech. I.e., the output layer in the speech synthesis neural network consisted of the coefficients to a time-varying digital filter, a gain element, and a pitch element which excited the filter, and a set of down-counting elements where each count represented a 100 millisecond speech segment. To train a single tri-phone set, the network was first reset by forcing the activation value of the elements in the state layer to zero, then a tri-phone pattern was presented to the network's input and held there during the learning process while the outputs changed to produce the appropriate output characteristics for that particular tri-phone combination. The outputs would represent the transition from one phoneme to another while a smooth transition in pitch, gain, and vocal tract characteristics would take place. The process was then repeated for other tri-phone combinations.

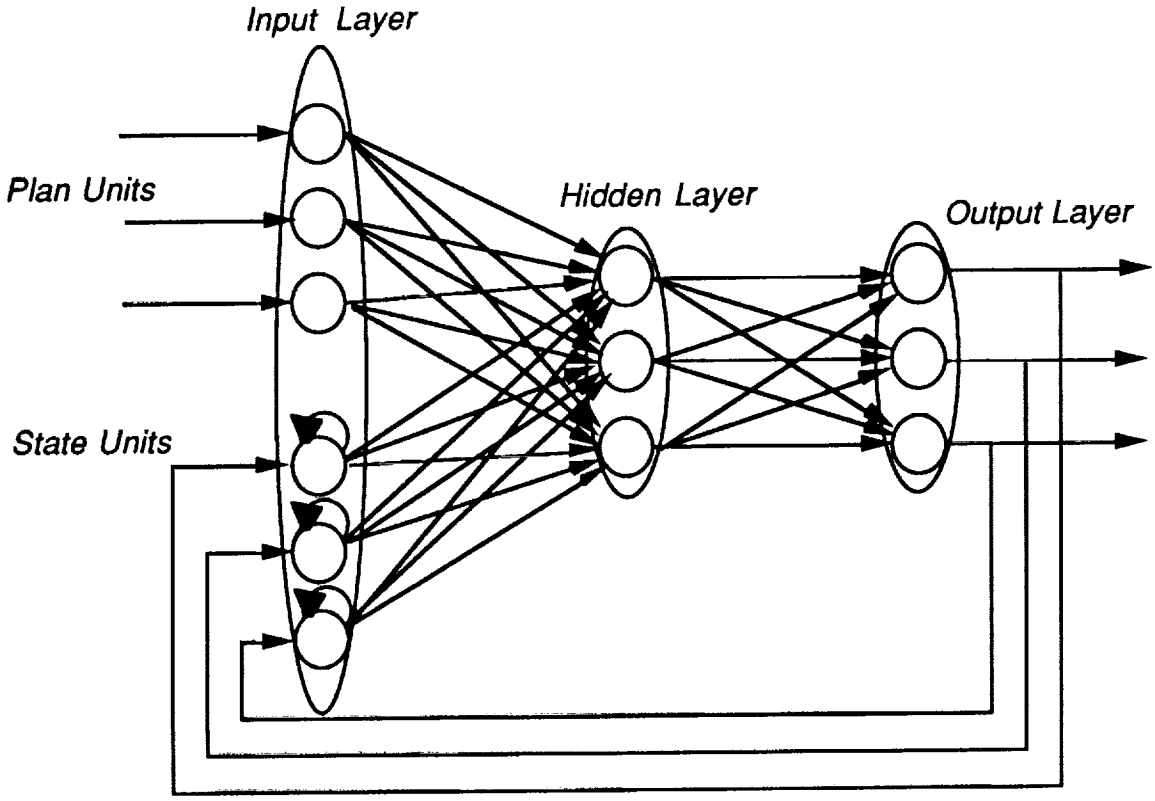


Figure 3 .... The connection scheme for Jordan's network architecture which learns to associate a static input with an output sequence.

Elman modifies Jordan's approach by constructing a separate layer, called the *Context Layer,* which is equal in size to the number of units in the hidden layer (see figure 4). However, the context units receive their input along a one-to-one connection from the hidden units, instead of from the output units as described by Jordan. The process works as follows. Suppose there is a sequential pattern to be processed. Initially, the activation values in the context units are reset to a value midway between the upper and lower bounds of a processing element's activation value, indicating ambiguous or don't care states. A pattern is presented to the network's input, forward propagating the pattern toward the output. At this point, the hidden layer activation levels are transferred one-for-one to elements in the context layer. If desired, error backpropagation learning can now take place by adjusting the weights between output and hidden, hidden and input, and hidden and context layers. The recurrent connections from the hidden to context layers are not allowed to change. At the next time step, the network's previous state is encoded by the activation levels in the context units. Thus, the context layer provides the network with a continuous memory.

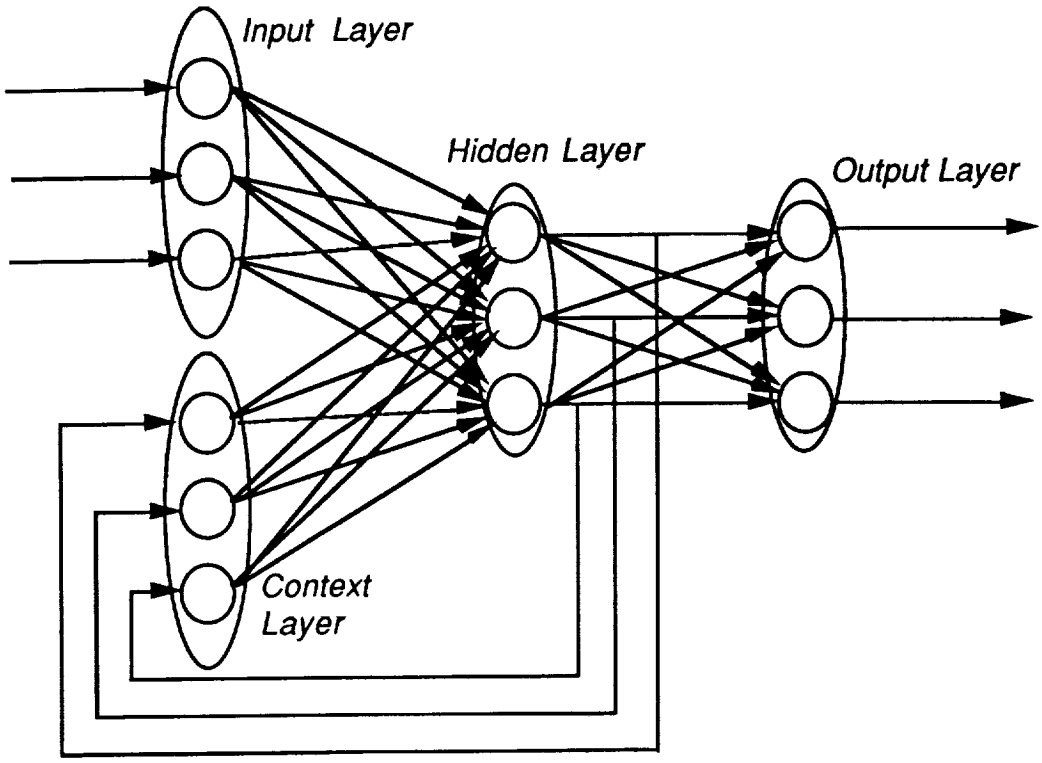


Figure 4 .... With the Elman network, a history of the network's most previous state is stored by transferring the activations in the hidden layer to the pseudo input, context layer. Longer term memories are attainable by adding recurrent connections to the context units.

DESCRIPTION OF THE SPACE-TIME NEURAL NETWORK

Another dimension can be added to the processing element shown in figure 2 by replacing the synaptic weights between two processing elements with an adaptable-adjustable filter. Instead of a single synaptic weight which with the standard backpropagation neural network represented the association between two individual processing elements, there are now several weights representing not only association, but also temporal dependencies. In this case, the synaptic weights are the coefficients to adaptable digital filters. The biological implication of this representation can be seen when one considers that synapses undergo a refractory period -- responding less readily to stimulation after a response. Before proceeding with a description of the space-time network, it is important to introduce digital filter theory and some nomenclature.

DIGITAL FILTER THEORY REVIEW

Linear difference equations are the basis for the theory of digital filters. The general difference equation can be expressed as:

$$y(n) = \sum_{k=0}^{N} b_k x(n-k) + \sum_{m=1}^{M} a_m y(n-m) \quad (6)$$

Where the x and y sequences are the input and output of the filter and a_m's and b_k's are the coefficients of the filter. Sometimes referred to as an s-transform, the well known continuous domain Laplace transform is an extremely powerful tool in control system design because of its capability to model any combination of direct current (DC) or static levels, exponential, or sinusoidal signals and to express those functions algebraically. The s-plane is divided into a damping component (σ) and a harmonic component ($j\omega$) and can mathematically be expressed as

$$s = e^{-(\sigma + j\omega t)} \quad (7)$$

This formulation has several interesting characteristics which should be noted:

- The general Laplace transfer function can be thought of as a rubber sheet on the s-plane. A desirable transfer function is molded by strategically placing a transfer function's roots of the numerator and the denominator in their appropriate positions. In this case, polynomial roots of the numerator are referred to as zeros and "pin" the rubber sheet to the s-plane's ground. On the other hand, polynomial roots of the denominator are referred to as poles and their locations push the rubber sheet upwards -- much like the poles which hold up the tarpaulin in a circus tent. Therefore, zeros null out certain undesirable frequencies and poles can either generate harmonic frequencies (if close enough to the $j\omega$ axis) or allow certain frequencies to pass through the filter.
- Setting the damping coefficient , σ, to zero is effectively similar to taking a cross sectional cut along the $j\omega$ axis. This is the well known Fourier transform.
- A pole on the $j\omega$ axis, signifying no damping, will produce a pure sinusoidal signal. However, a pole which travels onto the left half plane of the s-plane exponentially increases, eventually sending the system into an unstable state.

The discretized form of the Laplace transform has been developed further and is referred to as the z-transform. The notation z^{-1} is used to denote a delay equal to one sampling period. In the s-domain, a delay of T seconds corresponds to e^{-sT} . Therefore, the two variables s and z are related by:

$$z^{-1} = e^{-sT} \quad (8)$$

where T is the sampling period. The mapping between the variables can be further illustrated by referring to figure 5. First notice that the left half plane of the s-plane maps to the area inside a unit circle on the z-plane. In abiding with the Nyquist criterion, sampling at least twice the signal bandwidth, f_s, note that as one traverses from $-f_s/2$ to $+f_s/2$ on the s-plane is equivalent to going from π radians toward 0 radians and back to π radians in a counterclockwise direction

on the z-plane. Furthermore, note that lines in the s-plane map to spirals in the z-plane.

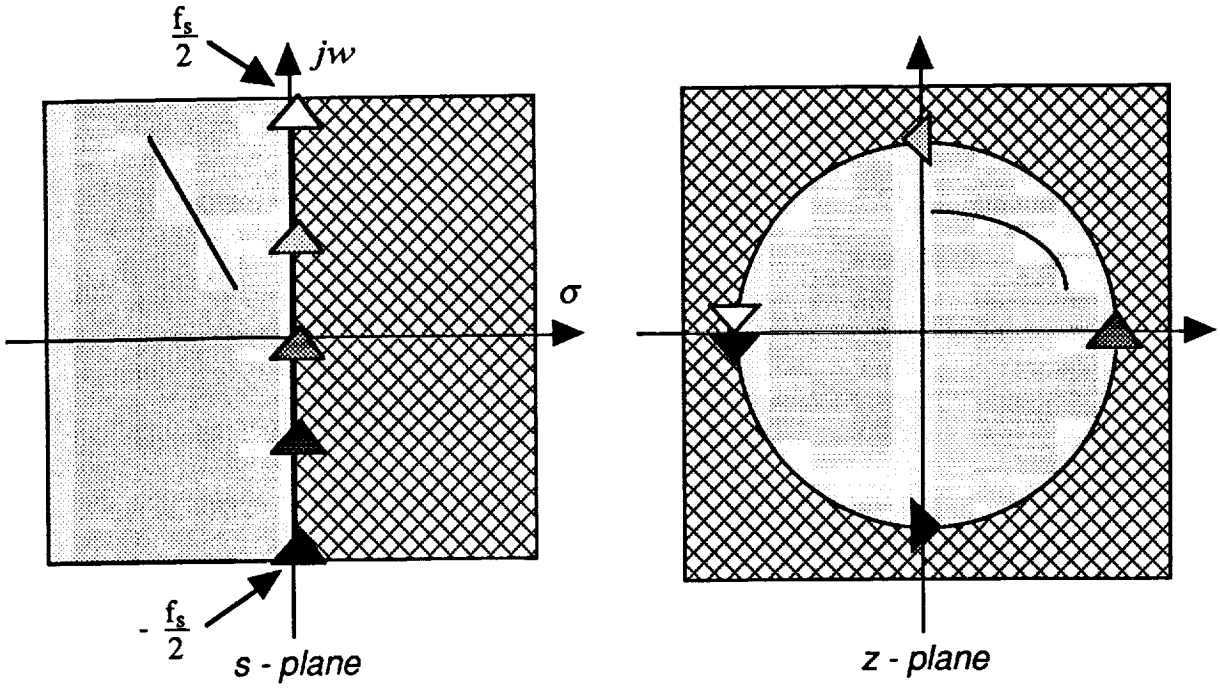


Figure 5 .... Pictorial relationship between the continuous domain s-plane and discrete domain z-plane.

By evaluating the z-transform on both sides of the linear difference equation, we can show that

$$F(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N} b_k z^{-k}}{1 - \sum_{m=1}^{M} a_m z^{-m}} \qquad (9)$$

Digital filters are classified into recursive and nonrecursive types. The nonrecursive type have no feedback or recurrent paths and as such all the a_m terms are zero. Furthermore, digital filters are also classified in terms of their impulse responses. Because nonrecursive filters produce a finite number of responses from a single impulse, such filters are referred to as Finite Impulse Response (FIR) filters. On the other hand, the recursive filters produce an infinite number of responses from an impulse and are therefore referred to as Infinite Impulse Response (IIR) filters. For example, if a unit impulse is clocked through the filter shown in figure 6(a), the sequence

$$b_0, b_1, b_2, ...b_M, 0, 0, 0, 0, 0, ...0, 0, 0$$

will be output. Notice that the filter produces only the coefficients to the filter followed by zeroes. However, if a unit impulse is presented to the filter shown in

figure 6(b), because of the recursive structure, the response is infinite in duration.

FIR and IIR filters each possess unique characteristics which may make one more desirable over another depending on the application. To summarize, the most notable of these characteristics include:

- FIR filters, because of their finite duration are not realizable in the analog domain. IIR filters, on the other hand, have directly corresponding components in the analog world such as resistors, capacitors, and inductive circuits.
- IIR filters cannot be designed to have exact linear phase, whereas FIR filters have this property.
- Because of their recursive elements, IIR filters are orders of magnitude more efficient in realizing sharp cutoff filters than FIR filters.
- Because of their nonrecursiveness, FIR filters are guaranteed stable. This property makes FIR filters much easier to design than FIR filters.

These different properties between FIR and IIR filters must be carefully weighed in selecting the appropriate filter for a particular application.

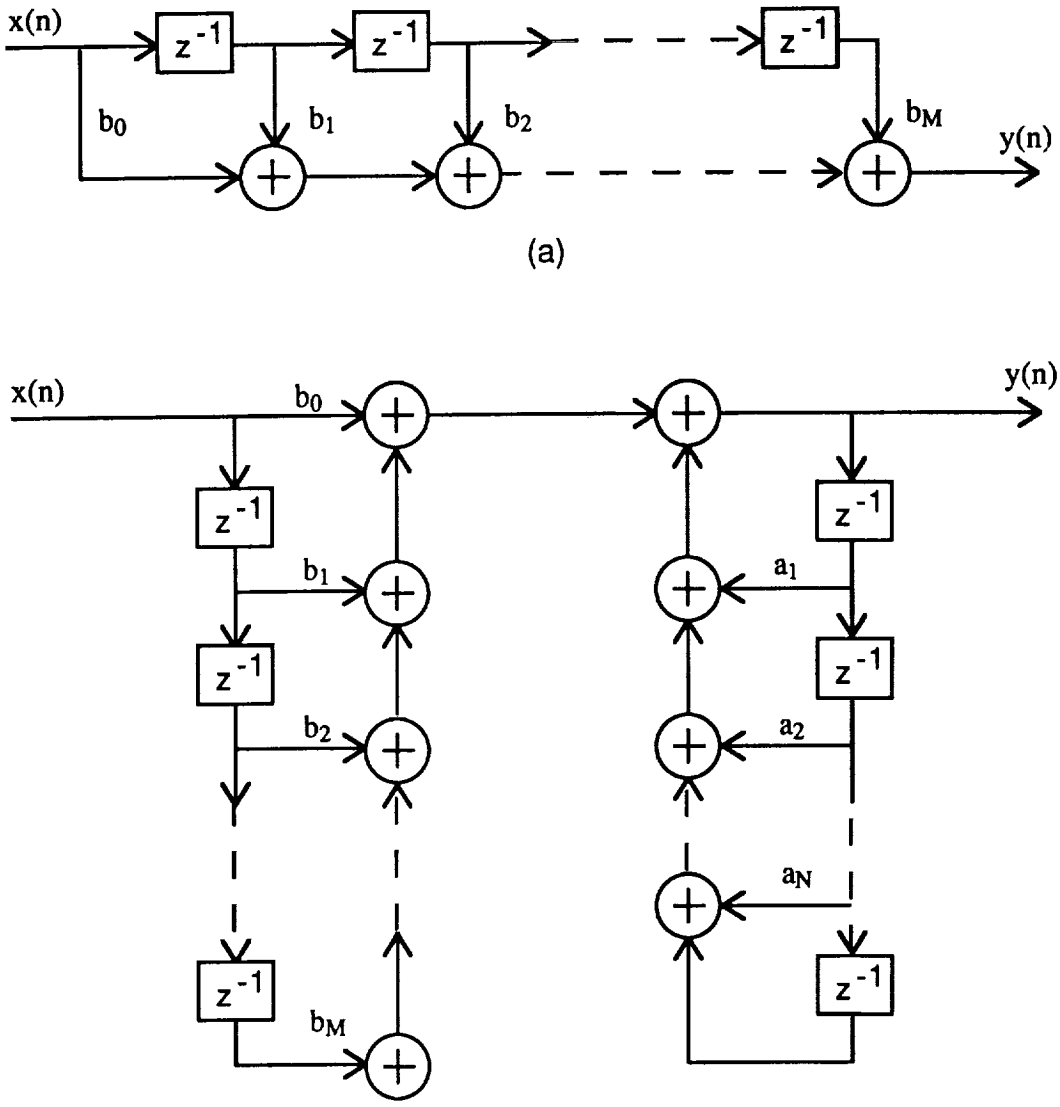


Figure 6 .... (a) Digital network for FIR system; (b) Digital network for IIR system

DESCRIPTION OF THE SPACE-TIME NEURAL NETWORK - CONTINUED

Having introduced digital filter theory, it is now possible to continue with the description of the space-time neural network. What follows is a detailed procedure for constructing and training the space-time neural network. As mentioned earlier, the space-time neural network replaces the weights in the standard backpropagation algorithm with adaptable digital filters. The procedure involves the presentation of a temporal ordered set of pairs of input and output vectors. A network consisting of at least two layers of adaptable digital filters buffered by summing junctions which accumulate the contributions from the subsequent layer is required. A pictorial representation of the space-

time processing element is illustrated in figure 7. In this case, a value, say xj(n), is clocked in to its associated filter, say Hji(n), producing a response yj(n) according to the filter representation

$$y_j(n) = \sum_{m=1}^{M} a_{mj} y_j(n - m) + \sum_{k=0}^{N} b_{kj} x_j(n - k) \tag{10}$$

All remaining inputs are also clocked in and accumulated by the summing junction i:

$$S_i(n) = \sum_{\text{all } j} y_j(n) \tag{11}$$

The contributions from the signals fanning in to the summing junction are then non-linearly transformed by the sigmoid transfer function

$$p_i(S_i(n)) = \frac{1}{1 + e^{-S_i(n)}} \tag{12}$$

This output is then made available to all filter elements connected downstream.

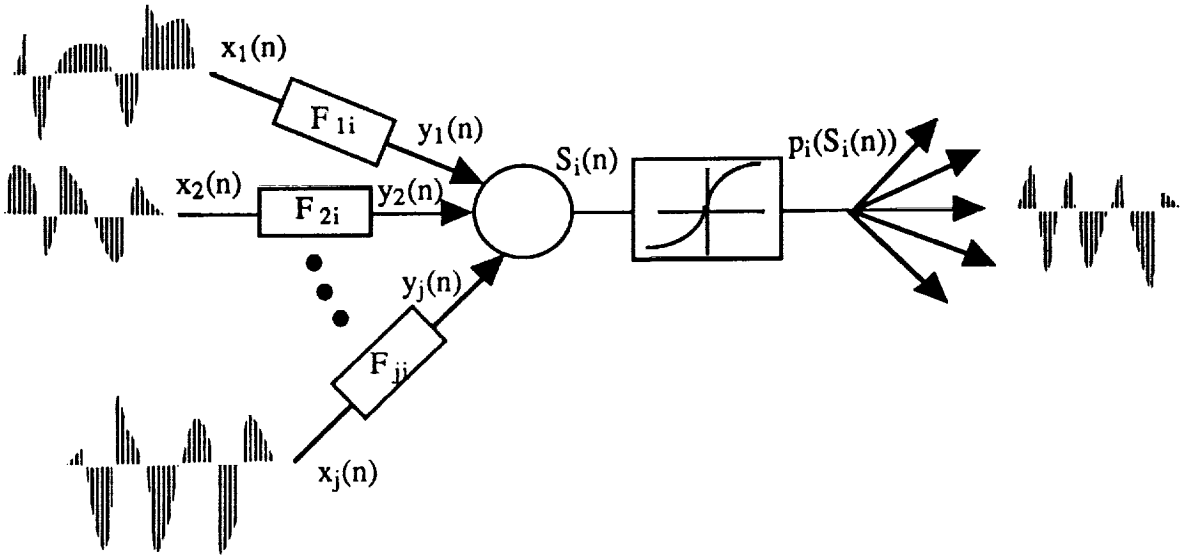


Figure 7 .... A pictorial representation of the Space-Time processing element.

It was earlier discussed that the space-time network is comprised of at least two layers of filter elements fully interconnected and buffered by sigmoid transfer nodes at the intermediate and output. A sigmoid transfer function is not used at the input. Forward propagation involves presenting a separate sequence dependent vector to each input, propagating those signals throughout the intermediate layers as was described earlier until reaching the output processing elements. In adjusting the weighting structure to minimize the error for static networks, such as the standard backpropagation, the solution is straightforward. However, adjusting the weighting structure in a recurrent network is more complex because not only must present contributions be accounted for but contributions from past history must also be considered.

Therefore, the problem is that of specifying the appropriate error signal at each time and thereby the appropriate weight adjustment of each coefficient governing past histories to influence the present set of responses.

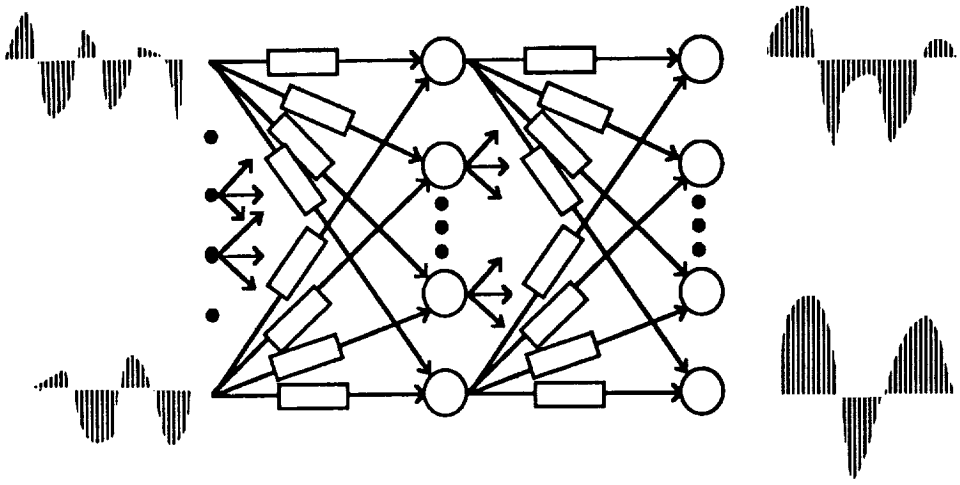


Figure 8 .... A representation of a fully connected network utilizing Space-Time processing elements. This depicts a set of input waveform sequences mapped into an entirely different output waveform sequence.

The procedure for adjusting the weighting structure for the space time network follows. First compute the errors at the output layer for each processing element, *i*,

$$\delta_i = (D_i(k) - A_i(k))\, P'(E_i(k)) \qquad (13)$$

where:

$D_i(k)$	is the *kth* desired response from a given sequence for neuron i at the output layer
$A_i(k)$	is the network's output response at neuron i for the *kth* input sequence pattern
$P'(E_i(k))$	is the first derivative of the sigmoid function for the *ith* output's activation value or $P(E_i(k))(1 - P(E_i(k))$

Now to compute the updates for the coefficients each filter element between the hidden and output layer neurons, a reversal procedure is implemented. Whereas in the forward propagation, input values were clocked into the filter elements, here, backpropagation instead involves the injection of errors into the filter elements according to:

$$\Delta b_{ijk}(n + 1) = \alpha[\eta \Delta b_{ijk}(n) + (1 - \eta)\delta_i X_{ijk}] \qquad (14)$$

where:

$\Delta b_{ijk}(n+1)$	is the update for a zero coefficient, b_k, lying between processing elements i and j
α	is the learning rate
$\Delta b_{ijk}(n)$	is the most recent update for the *kth* zero element between processing elements i and j
η	damps most recent updates
δ_i	is described by (13)
X_{ijk}	contain a history of the activation values for the non-recursive filter elements between neurons i and j, k time steps ago

The recursive components in each filter element are treated the same way and are updated according to:

$$\Delta a_{ijk}(n+1) = \alpha[\eta \Delta a_{ijk}(n) + (1 - \eta)\delta_i Y_{ijk}] \tag{15}$$

where:

$\Delta a_{ijk}(n+1)$	is the update for a zero coefficient, b_k, lying between processing elements i and j
α	is the learning rate
$\Delta a_{ijk}(n)$	is the most recent update for the *kth* zero element between processing elements i and j
η	damps most recent updates
δ_i	is described by (13)
Y_{ijk}	contain a history of the activation values for the non-recursive filter elements between neurons i and j, k time steps ago

For implementation purposes, the present algorithm only considers the accumulation of errors which span the length of the number of zeroes, $^{nz_{ho}}$, between the hidden and output neurons.

$$\delta_{ik} = \sum_{k=0}^{nz_{ho}} P'(E_{ik}) \sum_j \delta_{jk} X_{jik} \tag{16}$$

where:

i	is the index of the hidden neuron
j	ranges over the neuron indices for the output layer
δ_{hi}	is the accumulated error for the ith neuron in the hidden layer

$P'(A_{ik})$ is the first derivative of the sigmoid function for the *kth* history of activation levels for the ith neuron in the hidden layer

$\sum_j \delta_{jk} X_{jik}$ sums the results of injecting the previously computed errors found in equation (13) through the FIR portion of the filter element, X_{jik}, found between the ith neuron in the hidden layer and the jth neuron in the output layer.

Simulations

The first simulation is a variation of the classic XOR problem. The XOR is of interest because it cannot be computed by a simple two-layer network. Ordinarily, the XOR problem is presented as a two bit input combination of (00, 01, 10, 11) producing the output (0, 1, 1, 0).

This problem can be converted into the temporal domain in the following way. The first bit in a sequence XOR'd with the second bit will produce the second bit in an output sequence, the second bit XOR'd with the third will produce the third in an output sequence, and so on.

Input	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	.....
Output	**0**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	.....

In the simulation, the training data consisted of 100 randomly generated inputs and the outputs constructed in the manner described above. A network was constructed which had 1 input, 6 hidden elements, 1 output unit, 5 zero coefficients and 0 pole coefficients in the input to hidden layer, and 5 zero coefficients and 0 pole coefficients in the hidden to output layer. The task of the network was to determine the appropriate output based on the input stream.

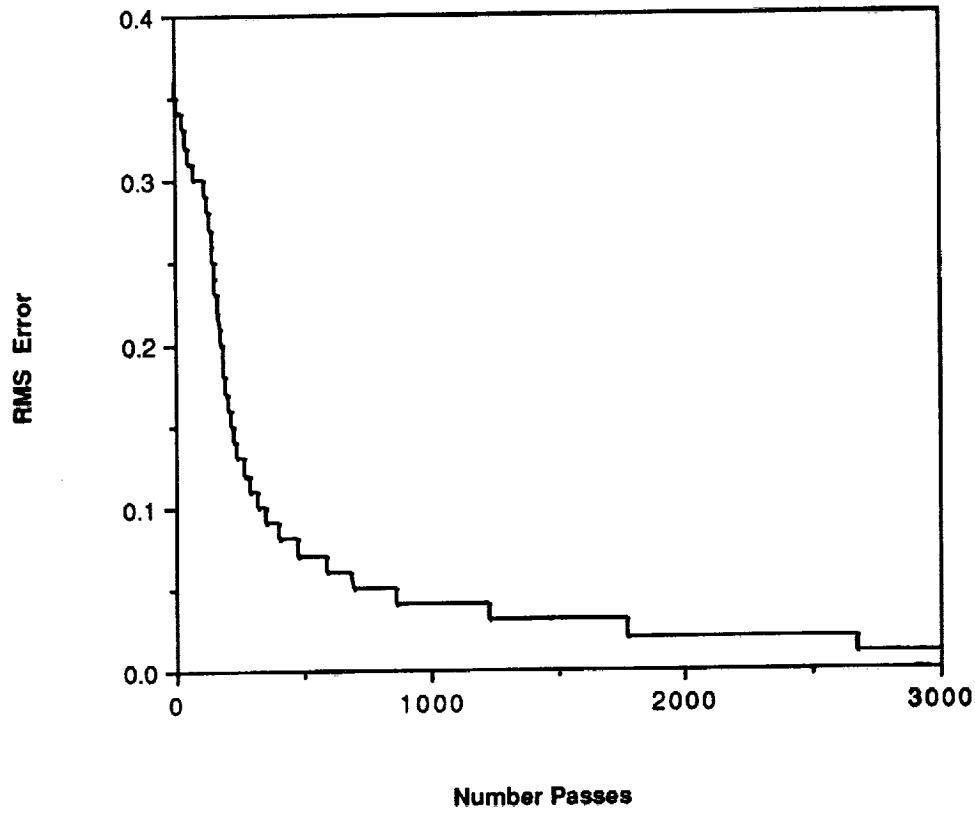


Figure 9 .... Error curve for the temporal XOR problem trained on a 1 input, 5 hidden, 1 output, 5 zeros and 0 poles in input to hidden layer, and 5 zeros and 0 poles in the hidden to output layer.

For the second simulation, a network with 2 input units, 8 hidden units, 8 output units, 5 zeros - 0 poles between input to hidden, and 5 zeros - 0 poles between hidden to output was constructed. One of the authors (Shelton) constructed a problem, called the Time Dependent Associative Memory Test, which would test the network's ability to remember the number of events since the last trigger pattern was presented. The data consisted of 1000 input output pairs where the input bits were randomly constructed and the output appropriately constructed. As an example, consider the first 7 sets of data in the list. Note that a "1" bit sequentially gets added to the output for the input patterns 0 0, 1 0, 1 0, 0 0, 1 0, and 0 1 until the 1 1 pattern is presented which resets the output back to the 1 0 0 0 0 0 0 0 state.

Input	Output
0 0	1 1 0 0 0 0 0 0
1 0	1 1 1 0 0 0 0 0
1 0	1 1 1 1 0 0 0 0
0 0	1 1 1 1 1 0 0 0
1 0	1 1 1 1 1 1 0 0
0 1	1 1 1 1 1 1 1 0
1 1	1 0 0 0 0 0 0 0
1 0	1 1 0 0 0 0 0 0
1 0	1 1 1 0 0 0 0 0
1 1	1 0 0 0 0 0 0 0
1 0	1 1 0 0 0 0 0 0
0 1	1 1 1 0 0 0 0 0
1 1	1 0 0 0 0 0 0 0

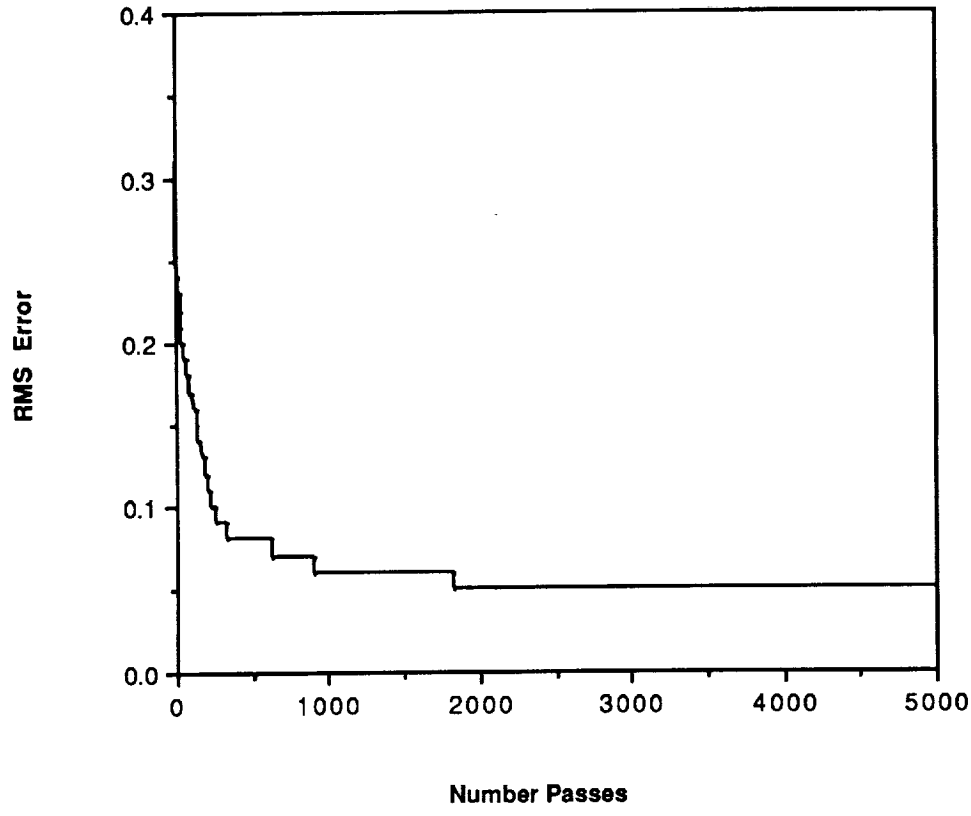


Figure 10 .... Error curve for a 2 input, 8 hidden, 8 output, 5 zero - 0 pole between input to hidden, and 5 zero - 0 pole between hidden to output network operating on the

References

[1] Baird, B., Nonlinear Dynamics of Pattern Formation and Pattern Recognition in the Rabbit Olfactory Bulb, Elsevier Science Publishers B. V., North-Holland Physics Publishing Division, 0167-2789, 1986.

[2] Elman, J. L., Finding Structure in Time, CRL Technical Report 8801, Center for Research in Language, University of California, San Diego, California, 1988.

[3] Freeman, W. J., Why Neural Networks don't yet fly: Inquiry into the Neurodynamics of biological intelligence, IEEE International Conference on Neural Networks, San Diego, California, 1988.

[4] Jordan, M. I., Serial Order: A Parallel Distributed Processing Approach, ICS Report 8604, Institute for Cognitive Science, University of California, San Diego, California, 1986.

[5] Ogata, K., Modern Control Engineering, Prentice-Hall Inc., Englewood Cliffs, NJ, 1970.

[6] Oppenheim, A. V. and Schafer, R. W., *Digital signal processing,* Prentice-Hall Inc., Englewood Cliffs, NJ, 1973.

[7] Rabiner, L. R. and Schafer, R. W., Digital Processing of Speech Signals, Prentice Hall Inc., Englewood Cliffs, NJ, 1978.

[8] Rumelhart, D. E., Hinton, G. E., & Williams, R. J., Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1) (pp. 318-362). Cambridge, Mass.: MIT Press, 1986.

[9] Steiglitz, K., Computer aided design of recursive digital filters, IEEE Transactions on Audio and Electroacoustics 18 (No. 2), pp 123-129, 1970.

[10] Villarreal, J. A. and McIntire, G., Neural Network Based Speech Synthesizer: A Preliminary Report, *Third Conference on Artificial Intelligence for Space Applications,* NASA CP-2492, pp 389-401,1987.

[11] Villarreal, J. A. and Baffes, P., Sunspot Prediction Using Neural Networks, *SOAR'89 - Third Annual Workshop on Automation and Robotics,* 1987.

[12] Yassaie, H., Digital filtering with the IMS A100, IMS A100 Application Note 1, INMOS, 72APP00100, Bristol, UK, 1986.

N91-20815

FUZZY LOGIC IN AUTONOMOUS ORBITAL OPERATIONS

Robert N. Lea
Software Technology Branch/PT4
NASA/Johnson Space Center
Houston, Texas 77058
(713) 483-8085
Nasamail : RLea

Yashvant Jani
LinCom Corporation
Houston, Texas 77058
(713) 488-5700

ABSTRACT

Fuzzy logic can be used advantageously in autonomous orbital operations that require the capability of handling imprecise measurements from sensors. Several applications are underway at the Software Technology Laboratory, NASA/Johnson Space Center to investigate fuzzy logic approaches and develop guidance and control algorithms for autonomous orbital operations. Translational as well as rotational control of a spacecraft have been demonstrated using space shuttle simulations. An approach to a camera tracking system has been developed to support proximity operations and traffic management around the Space Station Freedom. Pattern recognition and object identification algorithms currently under development will become part of this camera system at an appropriate level in the future. A concept to control environment and life support systems for large Lunar based crew quarters is also under development. Investigations in the area of reinforcement learning, utilizing neural networks, combined with a fuzzy logic controller are planned as a joint project with Ames Research Center.

1.0 INTRODUCTION

The current activities of the Software Technology Branch of the Information Technology Division at the NASA Lyndon B. Johnson Space Center are directed towards the development of fuzzy logic [1,2] software capabilities for building expert systems. In particular, the emphasis has been on developing intelligent control systems for space vehicles and robotics. The problem of sensor data monitoring and control of data processing, which includes detection of potential failures in the system and in some cases reconfiguration, is also under investigation. Results of performance tests made on simulated operational scenarios have been very promising. The issues of when, why, and how hardware implementation can be beneficial are also being studied carefully.

There are certain key technology utilization questions to be answered relative to the use of fuzzy logic control over conventional control.

1) Is it possible to create control systems which do not require a high degree of redesign when system configurations change or operating environments differ?

In other words, can adaptivity be achieved through the use of a fuzzy logic based controller in place of a conventional controller ? Experience with the conventional controller development tells us that a typical conventional controller requires significant redesign when there are changes in a) system characteristics, b) system configuration, or c) the environment in which the system is operating.

2) Can a fuzzy controller be used as a high level controller to function in conjunction with classical controllers in a way the human would ?

Specifically, can it be designed to monitor the system, evaluate its performance, and either suggest or force changes to make the system work properly or at least function more efficiently ? A high level controller typically works with abstract parameters which are derived, but not directly measured. It also commands parameters which are not direct controls. There are additional steps between the controller and sensing as well as commanding sides. Such controllers are grouped as intelligent controllers [3] and are not included in the conventional PID controllers group because these controllers perform additional tasks that provide capabilities for self governing or regulation as well as fault tolerance.

3) How easy or difficult is it to design and implement a fuzzy rule base that will control a complex system as opposed to developing a classical control system to do the same problem ?

Fuzzy logic based controllers will be valuable in systems that are highly non-linear and having complex environments that are practically impossible to model. Fuzzy controllers work for a linear system also but probably have less justification in this case, unless the problem is best thought of in a rule-based framework. The Japanese researchers and engineers have demonstrated [4,5,6] the usefulness of fuzzy controllers in the last few years with some impressive applications from a engineering viewpoint, such as the Sendai train controller [7], the air conditioning control system, the camera autofocusing system [8], the gas cooling plant controller [9], the television auto contrast and brightness control, the applications to automobile transmission and braking control, and applications to control of jitter in camera imaging which requires the distinguishing between real motion in the image which is desired and motion of the camera which needs to be filtered out.

4) A particular question of interest to NASA is : where can hardware implementations be utilized advantageously and how easy or difficult is it to transfer fuzzy rule bases to hardware?

In many cases, hardware will be able to take much of the computational burden off the central computing system. Fuzzy processors that perform fuzzy operations and execute fuzzy rulebases have emerged in the computer market [10,11,12,13] and are expected to gain widespread support for inline control of devices. Analog [14,15] as well as digital fuzzy processors are available and can be tailored to specific applications for optimum performance. Space operations can benefit greatly if the speed and power of these fuzzy processors can be utilized to achieve autonomy.

In section 2, a typical mission scenario for autonomous orbital operations is described with activities and tasks involved in carrying out some important steps. The role of fuzzy logic in these operations is discussed in section 3. A short summary of applications of fuzzy logic achieved thus far accomplished in the Software Technology Laboratory is provided in section 4. Current activities that utilize fuzzy logic in orbital operations, future activities, and a summary of our approach are provided in sections 5, 6 and 7 respectively.

2.0 AUTONOMOUS ORBITAL OPERATIONS

A typical rendezvous mission scenario as shown in fig. 1 for satellite servicing [16] requires orbit transfers, rendezvous planning, phasing maneuvers, and guidance and targeting for proximity operations. These tasks are necessarily required to approach and capture a satellite for repair or maintenance or to return it to a space station or the Earth. Repair and maintenance of satellites also requires control of robotic manipulator arms if such repairs are to be performed at the satellite location as opposed to returning it to a permanently manned orbital facility or the Earth. Sometimes a satellite may require only an inspection to determine if there are any problems. In this case, only station keeping or fly-around maneuverers are necessary.

In the problem of rendezvous of two space vehicles it is typically assumed that the target vehicle can maintain a stable orbit during the time required for the rendezvous to take place. Ideally, it will also have a stable attitude (although for vehicles in distress this may not be possible). For severely

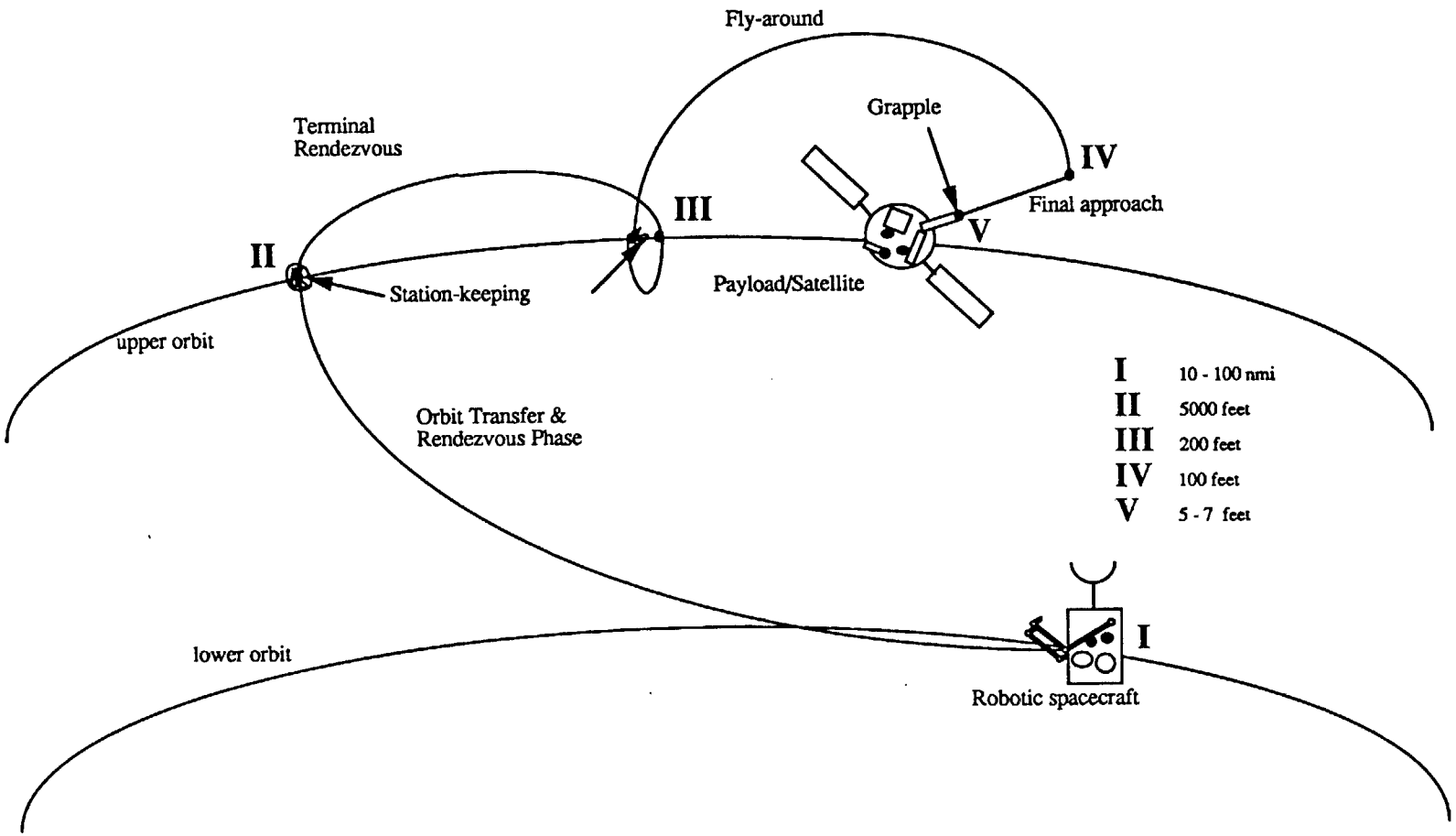


Fig. 1 TYPICAL SATELLITE SERVICING MISSION SCENARIO

distressed vehicles, the actual orbit may also be affected. In either case, the problem of rendezvous and capture may be necessary.

The target vehicle will be assumed to be at the origin of a coordinate system, known as the local vertical local horizontal (LVLH), where positive z is directed from the target to the center of the earth (or, in general to the center of whatever body it is orbiting), positive y is along the negative of the angular momentum vector and positive x completes the right handed coordinate system, as shown in fig. 2. The chasing vehicle will be the only vehicle assumed to be able to intentionally modify its trajectory and attitude in this relative coordinate system. The performance of the tasks above require trajectory control of the active vehicle relative to the target vehicle, including not only relative positions of the two vehicles, but also the attitude of the active vehicle.

Among the rendezvous mission tasks, mission planning based on mission goals and constraints is at the highest level. For example, a scenario for the capture of a satellite will incorporate time requirements, fuel constraints, and lighting and communications requirements based on the best assessment of the current and projected situation. The system will have to be intelligent enough to continually evaluate the status of the rendezvous and learn to adapt to unexpected occurrences through contingency planning or real time tuning of control algorithms. Such a system will require many inputs from a variety of independent sources, i.e. ranging and visual sensors, navigation systems, object recognition systems, human inputs from ground based or space based stations, on-board planning systems, diagnostic systems that report on the health of various systems including individual sensors, and redundancy management systems. Some specific problems are tracking of moving objects with sensors such as cameras, radar, lasers, or star trackers. In the event of multiple objects in the vicinity of the desired target vehicle, it must be possible to recognize the proper one, and for final approach to the vehicle it will be necessary to recognize objects on the target vehicle such as docking ports or grapple fixtures.

The next important task is trajectory control, especially the control of relative position with respect to the target vehicle. This must be performed during the entire time of the rendezvous. In some segments, control has to be very precise, while in other segments, the accuracy requirements are a bit relaxed. Trajectory control requires a continuous knowledge of current state which is typically derived from several sensor measurements. It also requires the information regarding a desired state typically provided by the guidance systems. It should be noted that the information required for the trajectory control is continuously changing with time and is highly dependent on the accuracy of sensor measurements.

Similarly, attitude control is required throughout the mission. A robust attitude control system enhances trajectory control because the execution of desired delta-V is much more accurate. Poor attitude control can definitely result in a mission failure. It should be noted that rotational control has to be very precise during the final approach and docking segments because coupling between rotational changes and the relative distances is significantly high. Again, note that the knowledge regarding current as well as the desired attitude is required and this information changes with time.

Both of the above tasks require processing of sensor data and its synthesis. All measurements must be accurately interpreted and action must be taken accordingly. Since several sensors are used, proper data fusion must be performed and each measurement must be used in its proper context. Otherwise, the probability of mission failure increases very significantly. This data fusion task necessarily includes the monitoring task that must be continuously performed and any deviations from the planned trajectory must be reported immediately.

Once the chaser spacecraft gets close to the target, its approach to the docking port must be carefully maintained with tight control of its translational as well as rotational state. The controller must be very precise and must have a fine tuning capability. At the end of the approach task it must initiate docking and rigidizing procedures, which will use a completely different set of sensors.

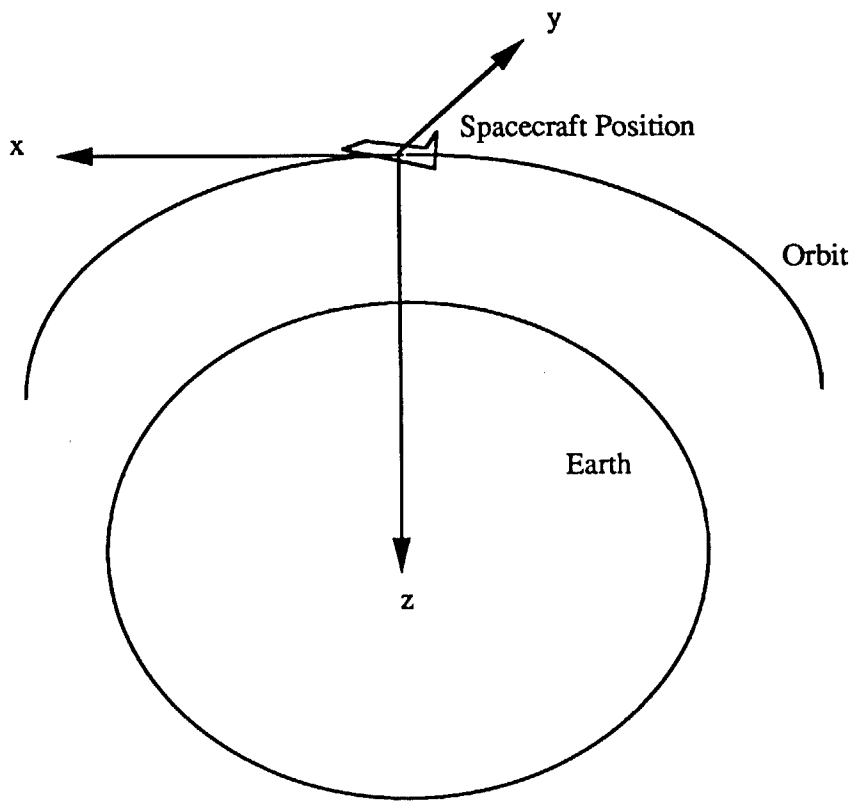


Fig. 2 The Local Vertical Local Horizontal (LVLH) Coordinate Frame

There must be some provision for a recovery procedure in case of a docking failure. When the crew performs these functions, they interpret the measurements according to their training and take action according to the procedures developed in a simulator. These procedures typically include contingency steps in case a docking failure occurs. The autonomous vehicle must have the same capability for mission success.

The vehicle must prepare for return to base with or without the payload. These preparations could be very lengthy or very short depending on what procedure the crew decides to use and how their sequence of actions is organized. In any event, thinking like the crew will definitely help solve the problem of increasing autonomy in rendezvous operations.

3.0 ROLE OF FUZZY LOGIC IN AUTONOMY

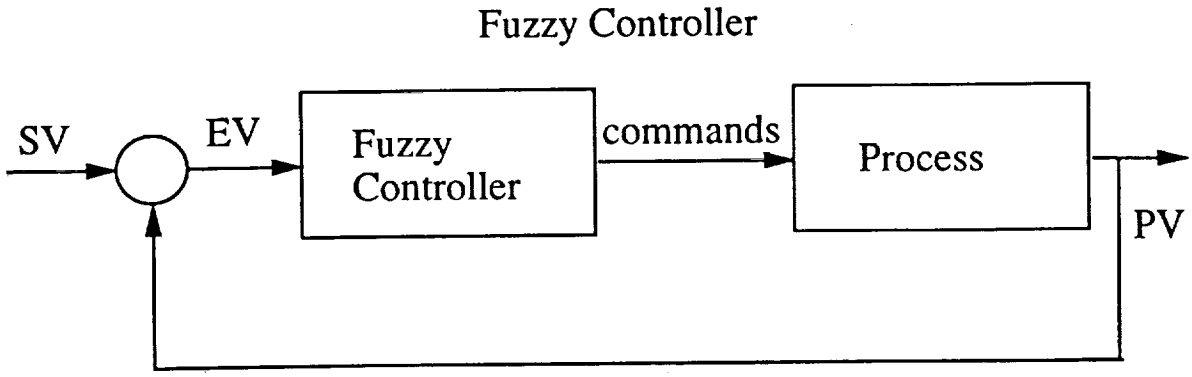
Fuzzy logic will be useful in the proper interpretation of measurements from sensors that are always corrupted by noise and bias. The accuracy of sensors represent a challenge that is not always surmountable. A fuzzy logic framework [1,2] can easily handle imprecise measurements, thus helping the integration process. Also systems may perform incorrectly or at least unexpectedly anomalous for a short time. It is necessary to determine this type of behavior and correctly resolve the situation. Processing of uncertain information using common sense rules and natural language statements is possible in this fuzzy logic framework.

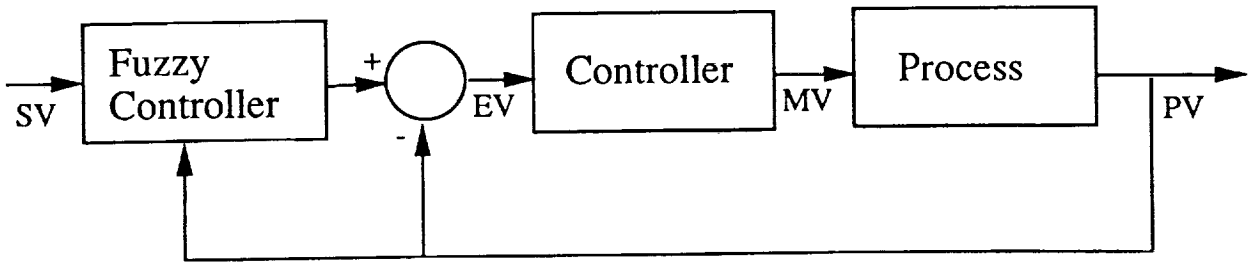
The utilization of sensor data in engineering control systems involves several tasks that historically are done by a man in the decision loop. These include cursory monitoring of data to determine if it should be processed and/or monitoring the output of the system to determine whether the system is performing as expected. All such tasks must be performed based on evaluations of the data according to a set of rules that the human expert has learned, usually from experience. Often, if not most of the time, these rules are not crisp, i.e., there must be some common sense or judgmental type decisions made. Such problems can be addressed by a fuzzy set modelling approach and if done properly can make decisions as well as the expert.

The fuzzy logic approach is simple to understand and easy to implement as a software module. Fuzzy rules provide a framework to implement the human thinking process i.e. the rules reflect the human thought process, such as " If the object is Far_Left then rotate the camera to the left side ". The entire rule base for the controller can be derived in the form of natural language statements as if a human was performing the controlling task. The experiential knowledge of a human controller, the crew in case of space vehicles, can be easily embedded in the software.

Fuzzy logic based controllers can be implemented in several ways as shown in fig. 3. In a strict sense these can be implemented as single controllers with well defined input and output parameters. They can also be implemented as feed forward controllers in conjunction with conventional controllers such that the desired state-value can be altered to provide an appropriate correction. The final command for the process is generated by the conventional controller. An alternative is to implement the fuzzy controller as a tuning system [17] in such a way that the parameters of a PID controller are tuned to better control the process and achieve efficiency. Thus, fuzzy logic controllers offer flexibility and adaptability for the process environment.

Implementation of fuzzy membership functions, rules and related processing is made easy by tools like the TIL Shell [18,19] which has a graphics oriented user interface and fuzzy-C compilers [20] that can generate code for a fuzzy chip or the C code to integrate with other software modules. There are several commercial products available in the industry that allow easy implementation of knowledge base, rulebase and user interfaces. For autonomous operations, it is easier and useful to implement control decisions through knowledge bases and rules so that the heuristics and related experiential knowledge can be used for a particular situation.





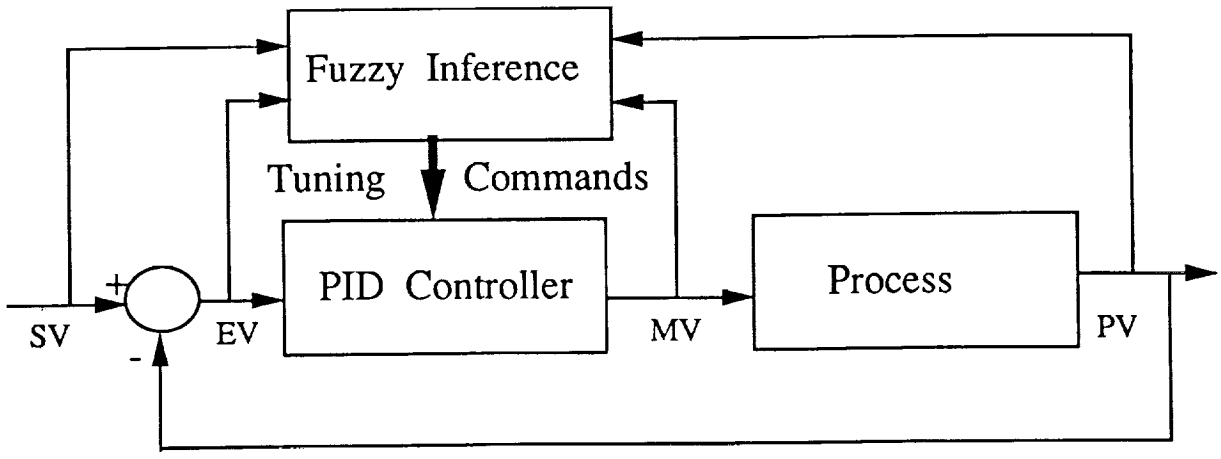


Fig. 3 Fuzzy Logic based Controllers in various configuration

C-2

It is also possible to develop and implement a fuzzy controller in a fuzzy processor, thus, having a fuzzy hardware controller. There are several commercial fuzzy processors [10,11,12,13,14,15] that can process over 30,000 fuzzy rules per second and thus provide a high processing power. These fuzzy processors consume low power with a capability to process general purpose instructions and can be mounted in the back plane of a sensor, for example, a camera. These processors also provide interfaces to hardware as well as the main computer to transfer information and commands. The advanced sensor systems envisioned for space station operations will have such processors embedded as an integral part of the system. Thus, a distributed processing onboard the spacecraft is possible via fuzzy chips.

A camera tracking system [21] described in section 4.4 can be a dedicated sensor with built-in intelligence and speed to perform functions which will normally be performed by the onboard computers. Because of the dedicated nature of a fuzzy chip and its processing power, there is virtually no computational load to the Space Station Freedom (SSF) computers. As a result, the computers will be available for other computing requirements such as complex guidance and navigation schemes. Furthermore, the interfaces between the fuzzy chip and computers will be at a command level requiring reasonably low speed data transfer. There will be no need for a high rate data transfer which could possibly increase costs and decrease reliability.

A significant application of fuzzy logic is in an advisory role in health monitoring and internal reconfiguration of spacecraft subsystems. These processes require a capability to handle uncertain measurements, estimate possibilities of failures and quickly rearrange flow so that autonomous operations are not stopped. Techniques have been developed to update the rule base using reinforcement learning in a given environment and adjust the response or behavior of a controller. These are very important for achieving operational efficiency in space operations.

4.0 PAST ACCOMPLISHMENTS AT JSC

There have been several applications of fuzzy logic to orbital operations at JSC. Sensor data processing control for star tracker navigation evolved during 1985-86 [22,23,24,25,26] and was successfully utilized to analyze space shuttle rendezvous flight data and check the validity of the algorithms. Translational control of a spacecraft based on fuzzy rules [27,28] was developed during 1987-88 and demonstrated [29] during the International Fuzzy Systems Association (IFSA) video teleconference IFSA88 at Iizuka, Japan in 1988. Rotational control of spacecraft attitude has been developed [30] using the phase plane approach and was demonstrated at IFSA89 conference [31] in Seattle, Wa. in 1989. A fuzzy logic based concept for a camera tracking system has been developed and was reported at the 8th International Congress of Cybernetics and Systems in June 1990 [21]. These applications are described in a short summary in this section.

4.1 Sensor Data Processing

In space shuttle rendezvous operations, the star tracker is used to give angle measurements for tracking rendezvous targets when the sun-target-shuttle geometry is such that the target is reflecting light towards the shuttle star tracker and when radar data is not available. When attempting to track a target with the star tracker, a star or debris such as ice crystals (caused by shuttle venting or jet firings) may be acquired. Loss of lock on the true target and acquisition of a false target is possible, especially if the target is dim due to attitude or range, or if the target was tumbling. If a bright false object crosses the target Line Of Sight (LOS) the star tracker might follow the brighter object. From experience, using simulated data, we know the shuttle rendezvous filter processes data under these conditions for a long enough time that the state vector is destroyed.

Under current operational conditions, to guard against any of these problems, a crewman monitors the acquired signal for acceptability prior to allowing the data to be processed, and he monitors the

residuals during data processing to insure that no unusual problems occur. To determine acceptability for processing, the shuttle crewman follows the rule [25].

> If the residual is *less than* the expected error as determined in pre-mission studies, and the *change* in residuals *is less than* 0.05 degrees for *five* consecutive measurements, then allow the filter to process data.

This rule contains deterministic conditions that are actually fuzzy in nature and have been interpreted as fuzzy by the crew at times during actual operations.The general problem considered here is to model the crewman's reasoning and common sense thought process in deciding whether the sensor data is acceptable for use in updating the shuttle-target relative state vector. This involves pre-editing and screening the data, and weighting the relative state vector update.

The Kalman filter editor, as it is designed for the shuttle rendezvous navigation system compares the residual magnitude against a multiple of the expected variance in the residual as derived from the current covariance matrix and the expected sensor error model. Data for which the residual is less than or equal to the expected error is incorporated into the filter state, and data for which the residual exceeds the expected error is not processed by the filter but is displayed to the crew for use in decision processes.

The filter and editor have performed satisfactorily on all rendezvous flights thus far. However, it has been considered essential that the crew be involved in the operations or else erroneous data such as obtained from lock on to stars and debris can be processed by the filter thus corrupting the filter state and necessitating a filter restart. With the current editor design it is not possible to protect against this since a star or debris may be very close to the target LOS.

The crew pre-editing function is to ensure that the true target is acquired prior to data processing. If the object acquired is the true target the residual should be less than the expected error, but more importantly it should stay almost constant. The only variation should be from noise in the sensor and small errors due to propagation of the shuttle and target states. Residual change less than 0.05 degrees is a conservative requirement consistent with the noise and bias in the star tracker.

Star tracker data is useful in maintaining a good relative state vector, but since it gives no range information directly, the state vector is easily corrupted by erroneous data. To guard against processing erroneous data two things were done. First, the pre-editing rule was restated using fuzzy sets which seemed more appropriate than a crisp statement in terms of ordinary Boolean logic. The fuzzy variation of the rule [25] reads as follows.

> If the measurement *residual is small* with respect to that expected value as determined from pre-mission studies and the *residual change is small* with respect to expected propagation errors and noise in the sensor for *several consecutive* measurements then allow the Kalman filter to process data.

Secondly, in order that the measurements be processed in a way consistent with common sense reasoning the decision function for processing data was modeled as a fuzzy set to be used for weighting updates to the state vector. By doing this it is assured that measurements that are close together will be processed similarly. For example, a measurement that slightly passes the residual edit criterion and one that barely fails will be processed similarly, i.e. the one that slightly passes will be allowed to only slightly contribute to the state vector update.

The fuzzy editing criterion was implemented into a simulation version of the shuttle on-board software. Real mission data was processed through this simulation and inputs to the filter were controlled by the fuzzy decision making process defined by the rule rather than the crew and the

current filter editor. The data from this simulation was compared to the results that were obtained during actual missions under ideal conditions as determined by the crew.

For non-nominal flight data collected from the Solar Maximum Mission (SMM) where an Inertial Measurement Unit (IMU) problem caused apparent errors in the star tracker measurements that exceeded the expected error by a factor of about fifty [26], the performance of the fuzzy editing scheme did not differ significantly from the current on-board system with the crew performing their normal pre-editing and monitoring functions. The problem, although actually caused by the redundancy management function, had the net result of an extremely large random bias. This data also had measurements from lock on to stars at the beginning of each star tracker interval. Instead of simulating a break track that would normally be done, it was decided to process the star data in order to test the weighting functions ability to handle problem measurements. As the following data, tabulated in table I, indicates the erroneous data caused no problems [25].

TABLE I. FLIGHT 41 -C IMU SWITCHING ERRORS
(AS COMPARED TO THE ONBOARD SOLUTION)
AFTER PROCESSING 20 MINUTES OF STAR TRACKER DATA

	RANGE	RANGE RATE
Nominal Filter (No Star Protection)	-13600	- 0.5
Fuzzy Editor (Editing when p ^ r = 0)	300	0.31
Fuzzy Editor (Editing when p ^ r < 0.25)	1320	- 0.15
Fuzzy Editor (Editing when p ^ r < 0.5)	1150	- 0.11

The state vector obtained using the fuzzy logic process and the state vector obtained from the actual flight data were then propagated for approximately one hour until radar data was obtained and the two results compared to the radar data to evaluate the filter's performance with the fuzzy editing and weighting rule against the system performance with man in the loop. For this test the p ^ r edit level was set to 0.0. The range and range rate estimates from the onboard navigation system and the system with fuzzy editing and weighting are then compared to the range and range rate measurements from the radar. The deviations from the radar measurements for the two systems are approximately the same. For a radar range of 102695 feet, the range deviations for fuzzy and onboard filters are 1965 and 1835 feet respectively [25]. This fuzzy editing and process control application has thus given very satisfactory results, comparable to that achieved by the crew in the operational system.

4.2 Translational Control of a Spacecraft

Fuzzy sets have been used in developing a trajectory controller for spacecraft applications in proximity operations profiles [27,28]. An automated vehicle controller that interprets the sensor measurements in a manner similar to a human expert has been modeled using fuzzy sets. The control rules were derived from the thinking process used by pilots and were implemented using typical $\pi-$ and s-functions (fig. 4) that can be adjusted for varying degree of fuzziness. Membership function definitions including universe of discourse were based on the targeting equations and control strategy for LOS approach [32]. The control strategy heavily used the experience base for manual operations.

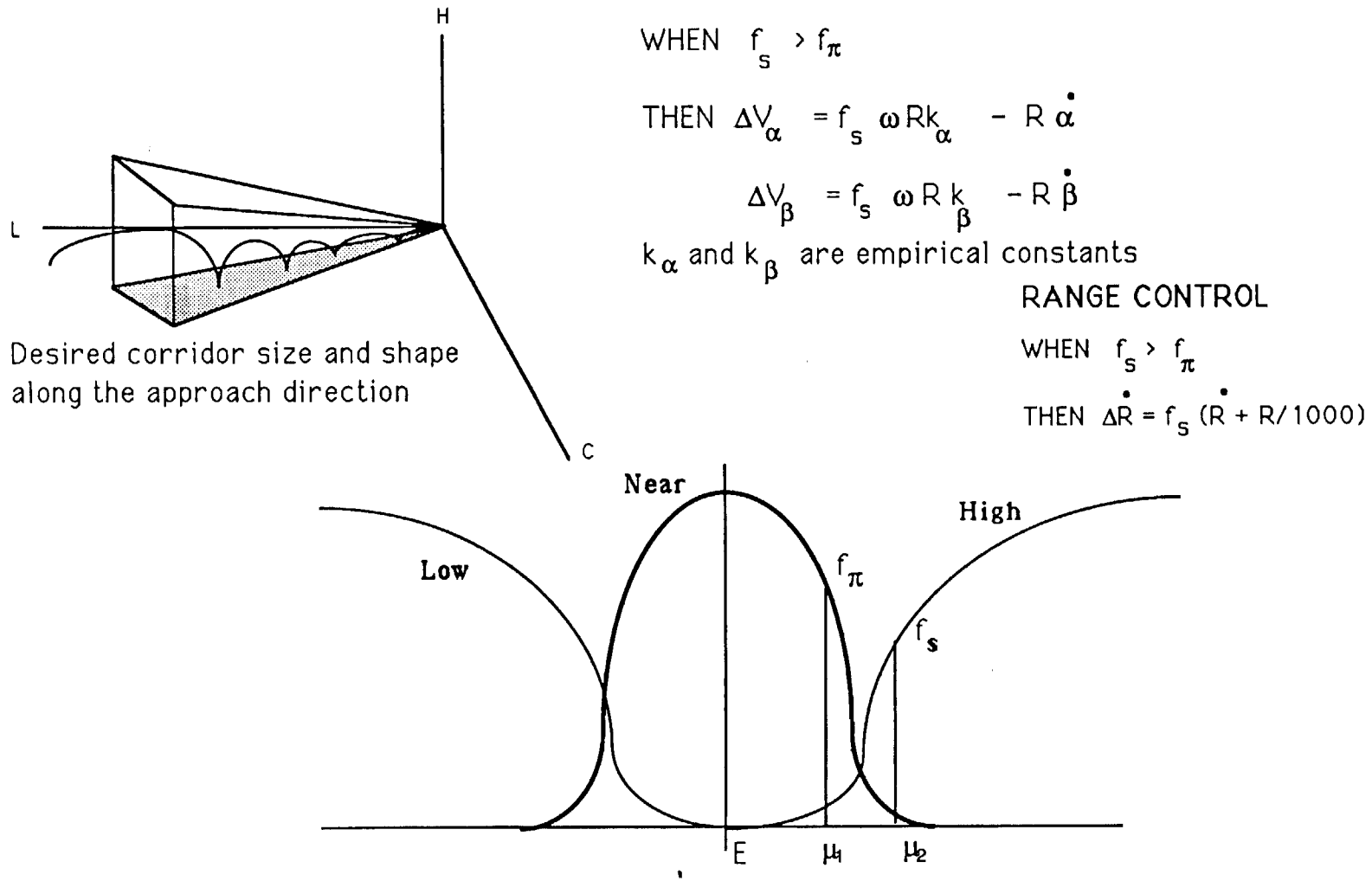


Fig. 4 The Π and S functions used in the control strategy for range parameter

Typical rules used for rendezvous vehicle control and modeled with fuzzy sets are the following :

> If the rendezvous vehicle's orientation with respect to a desired pointing vector to the target vehicle is close to the required orientation then no action is necessary.
>
> If the orientation significantly deviates from the required, then, take appropriate action to correct the problem.

Both in plane and out of plane positions and rates, and range and range rate must be controlled. Fuzzy sets are defined for "somewhat greater than", "somewhat less than" and "approximately equal to" the desired closing rate. They are also defined for "high", "low" and "near" with respect to the desired position (see fig. 4). During some time interval (every two seconds for the Shuttle) the fuzzy sets are evaluated and a determination is made as to whether an action needs to be taken to restore a rate or position to its desired value. If the no change function, such as "approximately equal to" or "near" the desired value, is larger than the corresponding change function, such as "somewhat greater than" or "low" with respect to the desired, then no action is taken. Otherwise an appropriate action is taken to restore the rate or position to the desired. The appropriate action is determined from an estimated action A(u), where u is the current value of the state, required to restore the active vehicle to the desired position from some maximum deviation. This action A(u) is then weighted by the change function S(u), and an action S(u)*A(u) is commanded to the system under control. Furthermore, there are no extreme accuracy requirements for the function A(u). For example, referring to fig. 4, if u_1 is the current value of x, then, $\pi(u_1)>S(u_1)$ and no action is taken. On the other hand, if u_2 is the current value of x, then $S(u_2)>\pi(u_2)$ and an action $S(u_2)*A(u_2)$ is commanded. More than one action can be commanded at a time so long as a constraint of the system under control is not violated.

The fuzzy controller has been implemented into a multi-vehicle dynamical simulator known as the Orbital Operations Simulator (OOS) [33], complete with all environment and sensor models. A small part of this control simulation was demonstrated via tele-video links [29], to the IFSA88 Workshop that was held in Iizuka, Fukuoka, Japan in August 1988. In this simulation, the automated fuzzy controller was used to control the closing rates and relative positions of the shuttle with respect to the SMM satellite. According to the test scenario, the fuzzy controller was required to perform operations including approach to target, fly around and stationkeeping.

Many different scenarios have been run with this automated fuzzy controller to evaluate the performance with respect to flight profiles and delta-V requirements, which is a direct measure of the performance. Comparisons of delta-V requirements for a man-in-the-loop versus the automated controller have shown [27] that the automated controller always uses less delta-V. For a test case involving stationkeeping at 150 feet for 30 minutes, the automated controller required 0.1 ft/sec delta-V whereas 0.54 ft/sec was used in the man-in-the-loop simulation. For v-bar approach from 500 feet to 40 feet within a 25 minute time interval, the automated controller used 2.12 ft/sec vs. 2.99 ft/sec for the man-in-the-loop simulation.

4.3 Rotational Control of Spacecraft

To complement this translational control, it was decided to implement rotational control via fuzzy membership functions and the rules based on the conventional phase plane. It was obvious that such an implementation would provide a direct performance comparison with the conventional control system, thus leading to further insight into understanding the relative merits of fuzzy control systems. Furthermore, an integrated six Degree Of Freedom (DOF) controller can be developed by combining these two control systems.

The rotational control system has been developed on a 386 computer using the fuzzy-C compiler and related software. The 'Phase_Plane' package that contains the membership functions and rules has been implemented in a file called Phase.til [30]. The angle and rate errors, PHI and PHI_DOT, are inputs and the torque is the output for this rotational controller. The input variables have seven membership functions defined over the universe of discourse as shown in fig. 5. The output variable torque has five membership functions as shown in fig. 6. There are 25 rules defined for reducing the PHI and PHI_DOT errors to within their zero (ZO) range (see Table II). These rules are based on the phase plane construct used in the attitude control system.

Table II. Rule base for attitude controller

	phi						
phi_dot	NB	NM	NS	ZO	PS	PM	PB
NB	PM	PM	PS				
NM	PM	PM	PS				
NS	PS	PS	PS				
ZO	PS	PS	ZO	ZO	ZO	NS	NS
PS					NS	NS	NS
PM					NS	NM	NM
PB					NS	NM	NM

Single axis rotational equations were implemented for the pitch axis of the shuttle. The pitch moment of inertia and the positive and negative pitch torques provided by jets were used in this simple simulation to test the fuzzy controller rules. The shuttle jets provide a larger acceleration for positive pitch as compared to the negative pitch. The simulation was set up to provide a constant torque during a cycle time of 80 milliseconds. The pitch attitude and the rate are propagated at this cycle time. When the fuzzy controller asks for a torque greater than 0.5, the constant torque is provided in that direction, otherwise no torque is provided. This simulates the jet-on and off activity at the appropriate time. The fuzzy controller is called at every cycle to evaluate all rules and output the desired torque. Based on this torque the jet is turned on and the rate and angle are propagated. With new values of angle and rate, the angle error and rate error are computed for the next cycle input. Time is also advanced every cycle. Time histories of angle and rate, and the phase plane plot are created for analysis.

Testing for the pitch axis so far has shown very satisfactory results. With several starting states, i.e., initial angle and rate, the system has converged on the commanded value, and manifested a relatively smooth limit cycle around the deadband. The control system response in all cases has been as expected, including overshoot behavior in cases where initial rate error is very large. Tests were performed with some rules turned *off* or deactivated to observe the performance with a limited rulebase. The objective was to reduce the number of rules to a minimum.

Performance of the fuzzy controller with 25 rules was more than adequate for a single axis, and gave us confidence to expand it to three axes case. The phase plane module in the Shuttle Digital

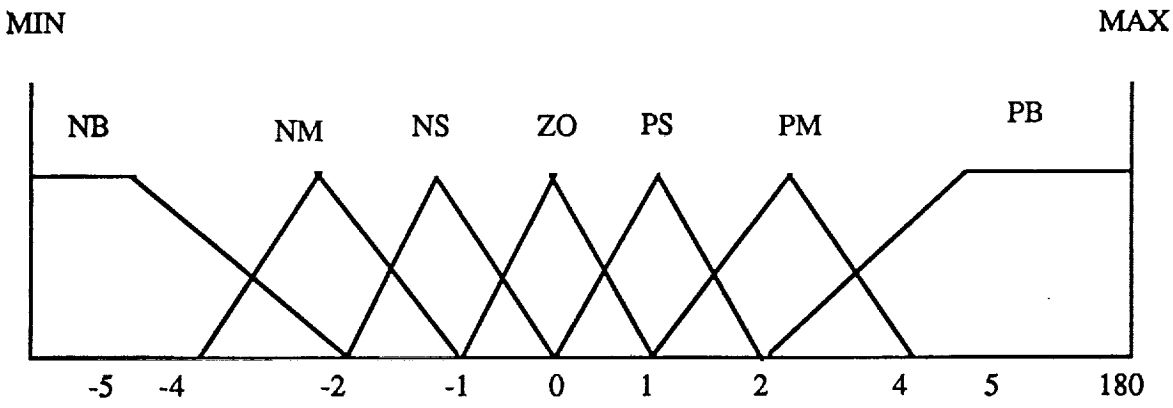


Fig. 5 MEMBERSHIP FUNCTIONS FOR PHI AND PHIDOT

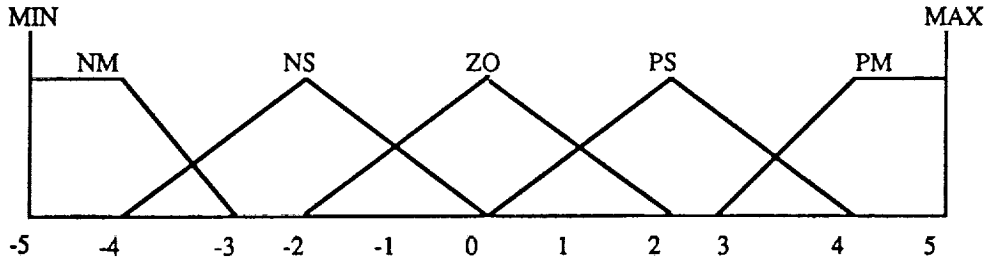


Fig. 6 MEMBERSHIP FUNCTIONS FOR ROTATIONAL ACCELERATION

Auto Pilot (DAP) was replaced by this controller with all other interfaces unchanged. The integration process was completed with only minor modifications to the interfaces. The simulation testing included three axes attitude hold and single axis maneuvers. In a three axes attitude hold case, the fuzzy logic based controller used only 30 % of the fuel used by the DAP. For the case of attitude maneuvers, the fuzzy controller used around 60 % of the fuel used by the DAP. In both cases, the fuzzy controller has shown comparable performance for maintaining attitude and body rates. Detailed testing and analysis is planned to include other maneuver modes and different parameters sets.

4.4 Camera Tracking Control System

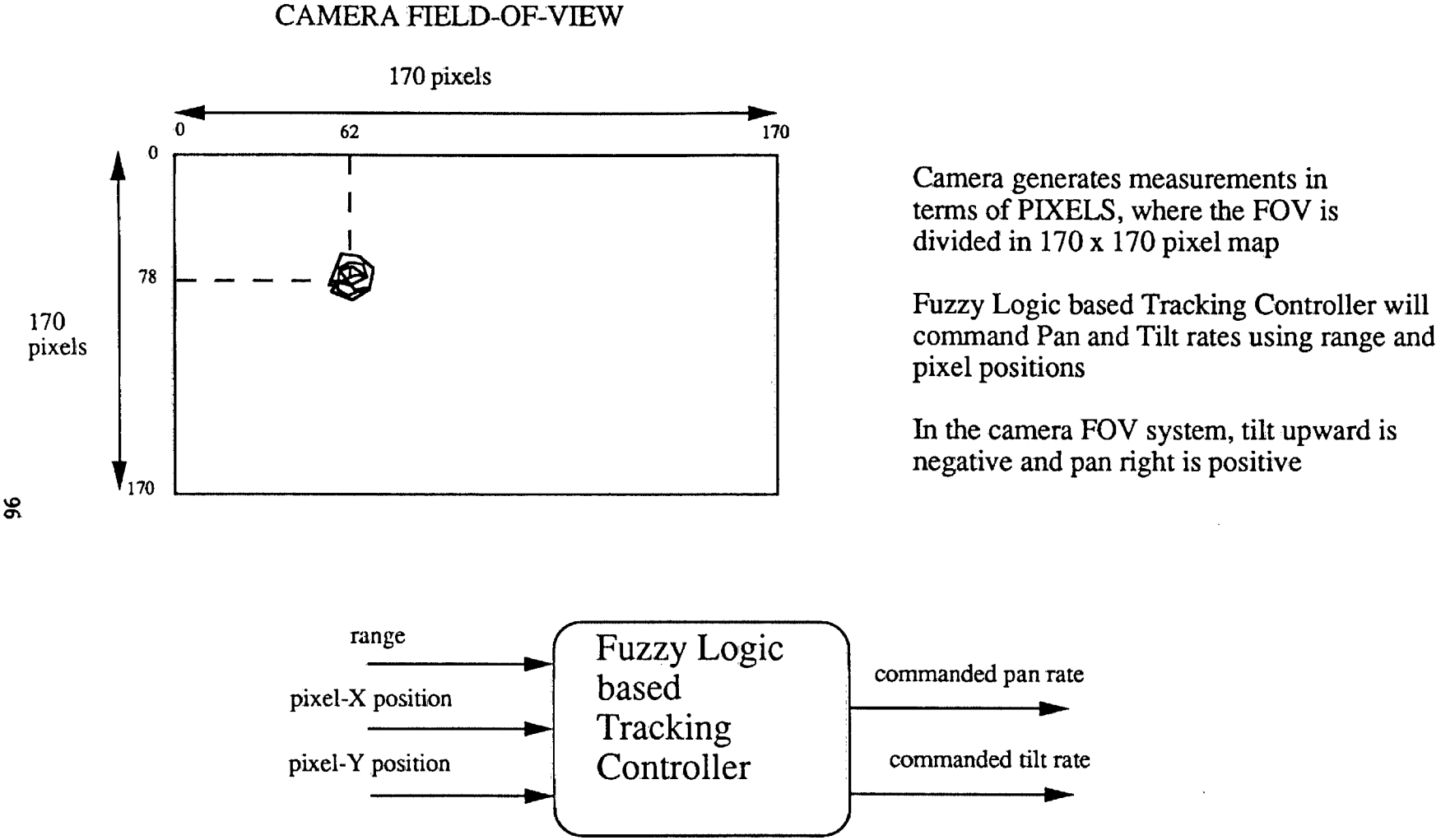
Advanced sensor systems with intelligence and a distributed nature will be required for activities like proximity operations and traffic control around the SSF. There will be several sensors of different types providing various measurements simultaneously as input for processing to such a system. Conceptual development of such a system [21] where several cameras, laser range finders and radar can be used as independent components is in progress within the STL at JSC. The first phase of this development is the camera tracking system based on the fuzzy logic approach that utilizes the object's pixel position as inputs and controls the gimbal drives to keep the object in the Field Of View (FOV) of the camera as shown in fig. 7.

Tracking of an object means aligning the pointing axis of a camera along the object's LOS. The monitoring camera is typically mounted on the pan and tilt gimble drives which are capable of rotating the pointing axis within a certain range. The task of the tracking controller is to command these gimble drives so that the pointing axis of the camera is along the LOS vector which is estimated from the measurements.

For the fuzzy logic based tracking controller, the inputs are range and LOS vector, and the outputs are the commanded pan and tilt rates. The LOS vector is input in terms of pixel position in the camera FOV. When an image is received, it is processed to determine the location of the object in the camera frame which has the vertical, horizontal and pointing vectors as three axes. Usually an image, particularly for complex objects, spans many pixels. Using a suitable technique, the centroid of the image is computed and used as the current location of the object in the viewing plane. This plane is a Cartesian coordinate plane having vertical and horizontal axes. The size of the viewing plane is 170 x 170 pixels, and the origin is at the upper left corner as shown in fig. 7. The range of the object is received from the laser range finder as a measurement. These three parameter values are input to the controller.

Membership functions for the range, horizontal and vertical positions are shown in fig. 8 and the membership functions for the Scale_Factor, Pan and Tilt rates are shown in fig. 9. For simplicity, these functions are triangular shaped over the universe of discourse. The scale_factor parameter is used as an intermediate step and provides the desired flexibility of changing the responsiveness of the fuzzy controller.

The desired image location is the center of the viewing plane, which is at (85,85). If the current location is close to the center, then rotation of the pointing axis is not required. If the location is to the left of center then a left rotation is necessary. Similarly, if the image is down from the horizontal line then a downward rotation is required. These rotations are determined using the position and range measurements and the rule base shown in Table III. First the range measurement is fuzzified and the value of the scale factor is determined based on the scale_factor rules. Necessary defuzzification processing is performed to compute the crisp value of the scale factor. Then, the scale factor and the position measurements are provided to the next set of rules to determine the rate at which the gimble drives should be rotated. There are 30 rules that determine both, pan and tilt rates. Again, the necessary defuzzification processing is performed to compute the crisp values of the pan and tilt rates which can be sent to the gimble drives as command values.



Camera generates measurements in terms of PIXELS, where the FOV is divided in 170 x 170 pixel map

Fuzzy Logic based Tracking Controller will command Pan and Tilt rates using range and pixel positions

In the camera FOV system, tilt upward is negative and pan right is positive

Fig. 7 CONCEPT OF A CAMERA TRACKING SYSTEM

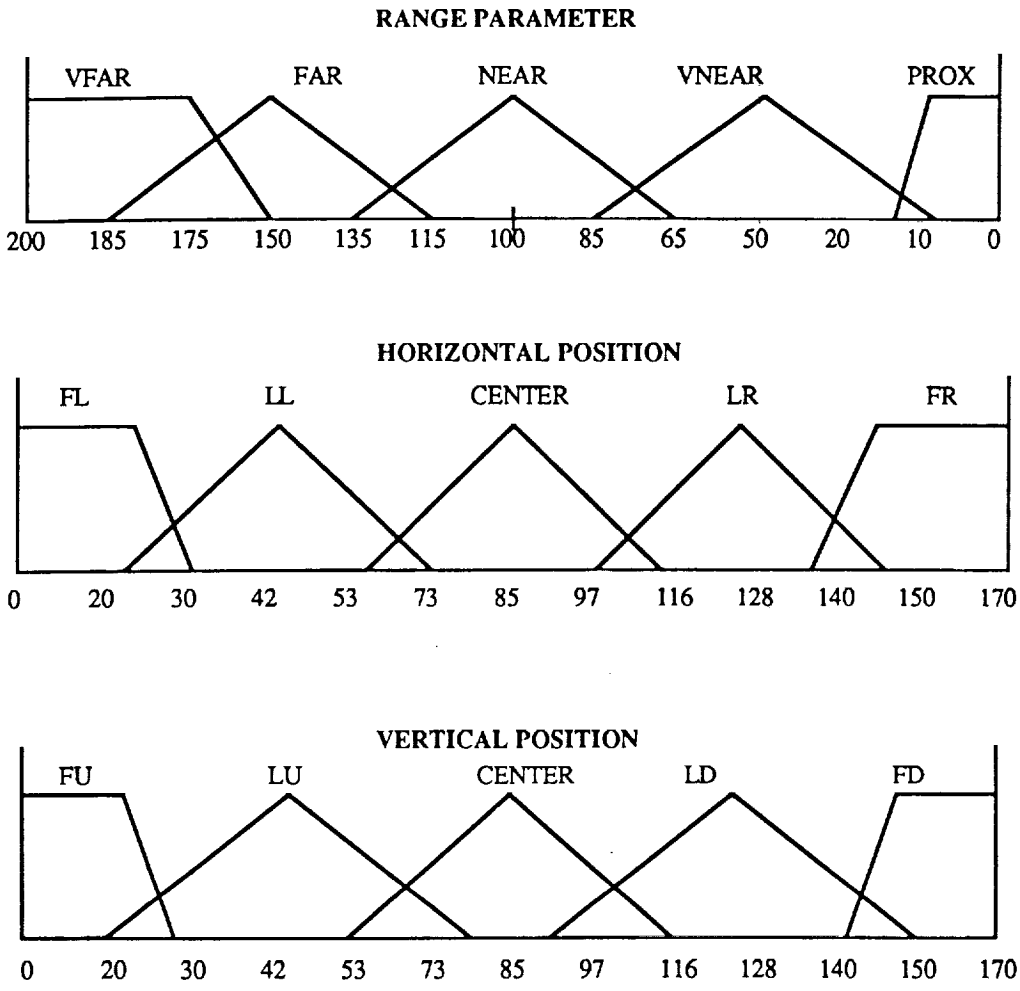


Fig. 8 Membership Functions for Input parameters

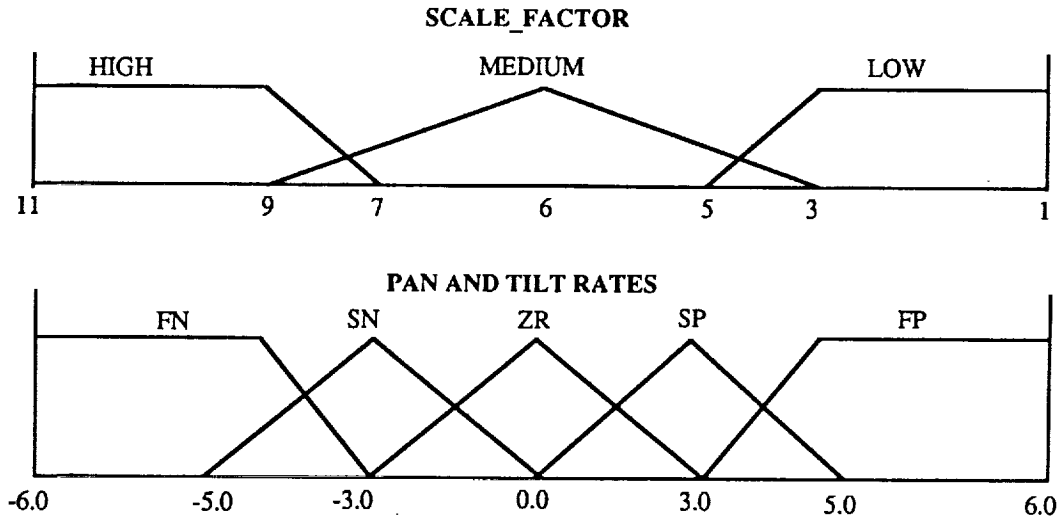


Fig. 9 Membership functions for Scale_Factor and Output parameters for camera tracking system

Table III. Rule base for the tracking task

	Distance Membership Functions				
	VFAR	FAR	NEAR	VNEAR	PROX
Scale_Factor	LOW	LOW	MED	HIGH	HIGH

	Horizontal Position Membership Functions				
	FL	LL	CENTER	LR	FR
LOW	FN	SN	ZR	SP	FP
MED	SN	SN	ZR	SP	SP
HIGH	SN	ZR	ZR	ZR	SP
Scale_Factor	Pan_Rate Membership Functions				

	Vertical Position Membership Functions				
	FD	LD	CENTER	LU	FU
LOW	FP	SP	ZR	SN	FN
MED	SP	SP	ZR	SN	SN
HIGH	SP	ZR	ZR	ZR	SN
Scale_Factor	Tilt_Rate Membership Functions				

Note - Negative Tilt_rate means the pointing axis going upward in FOV

The camera is rotated based on these commands within the limits of its gimble rates and angles. New LOS measurements in the camera FOV are obtained for the next cycle and the processing is repeated. The cycle time is based on the processing time required for the following functions : 1) determining pixel positions, 2) obtaining a range measurement, 3) rotating the gimble drives at a desired rate, and 4) the requirements to track the object within a certain performance envelope. Typical cycle time ranges between 0.1 to 1.0 second.

There are several advantages of our approach that utilizes fuzzy logic in a camera tracking system. This system will be a low power sensor as compared to an active sensor e.g. Radar in the Ku band range, or LADAR using laser frequency. Typically, the active sensor radiates a power pulse towards a target and receives back a reflected pulse. Based on the power transmitted, power received and time between these pulses, parameters like range and range rates are calculated. Since the camera tracking system will not be radiating power, it will be a low power sensor in comparison with active sensors. Since there is already a shortage of power, an important consumable, onboard the SSF, availability of low power sensors is very important for continuous operations. The SSF can afford to keep this type of a sensor working around the clock without having much impact on the power management or other computational load on the main computers.

5.0 CURRENT ACTIVITIES

In this section, we describe the current ongoing activities in the STL in the area of fuzzy logic research. A complete 6 DOF controller is created by combining the translational and rotational controllers. Our integration approach and testing philosophy is described in section 5.1. Our plans for software and hardware testing for the camera tracking system are described in section 5.2. Activity in the area of motion control for Mars rover during sample collection process is described in section 5.3 along with some preliminary results.

5.1 Combined Translational and Rotational Control for Relative Orientation and Distances

The integration approach adopted for combining translational and rotational control systems is simple, straight forward and involves extensive testing [31]. The first step is to implement the previously defined translational rules in the same format using our development environment. This will provide commonality between the code and allow an opportunity for stand-alone testing and optimization of translational rules. The second step is to generate the proper code for the SUN workstation using the fuzzy-C compiler (with appropriate options) and transfer it to the workstation. This step is required only because the development environment is on the 386 computer and the high fidelity simulation is on a SUN workstation. Since the fuzzy-C compiler and associated development environment is portable, there is a plan to develop fuzzy controllers on the SUN workstation and avoid the code transfer. The third step is to develop the test plan that will test all aspects of the 6 DOF controller. The final step is to perform testing and compare the results with the conventional system.

NASA's OOS [33] will be used for testing the 6 DOF controller. It is a high fidelity, multivehicle spacecraft operations simulation that provides 6 DOF equations of motion within an orbital environment including aerodynamic drag. It can be used for engineering analysis as well as real-time operations demonstrations. It provides a framework to integrate and test expert systems and hardware with the software modules commonly known as the onboard flight software. The OOS (fig. 10) executive also provides external interfaces to graphics and expert systems.

The translational fuzzy control system [27] will be used by the autosequencer to generate proper hand controller commands so that the desired range and range rate are maintained during proximity operations. Typically, a shuttle pilot provides these input and controls the relative trajectory. Thus the autosequencer will simulate the crew input via the translational fuzzy control system. The

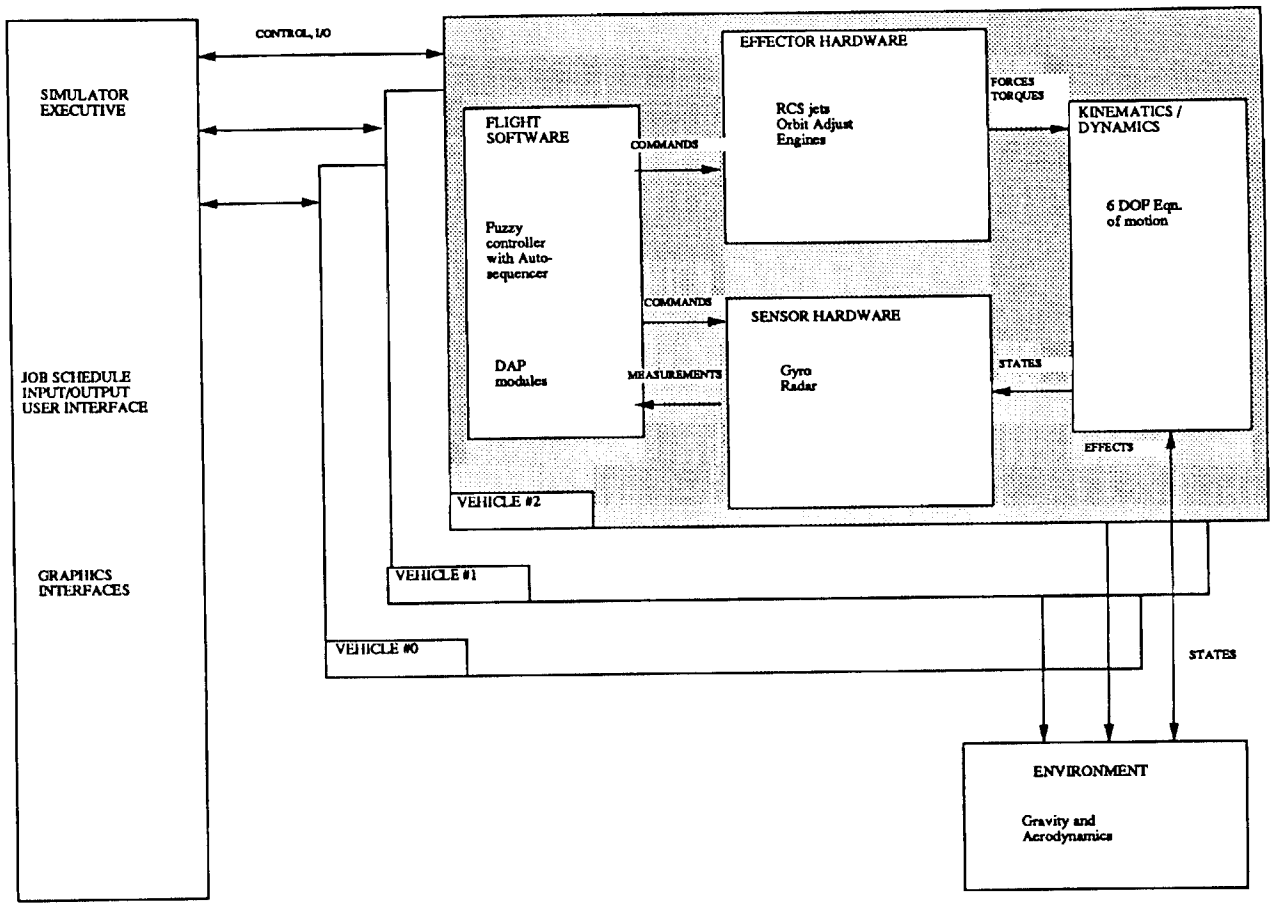


Fig. 10 The Orbital Operations Simulator Architecture

automatic attitude control system of the shuttle called onorbit DAP is implemented in OOS for shuttle on-orbit operations. The rotational fuzzy control system created by replacing the phase plane module will generate commands for jet-select to fire jets for attitude control. Existing interfaces with the Phase Plane module will be maintained intact for the overall integrity of the system. When both fuzzy control systems are used together, it will provide a total 6 DOF controller for proximity operations.

A preliminary test plan has been put together to test the 6 DOF controller. It includes test cases for stationkeeping with a fixed attitude, stationkeeping with attitude changes, LOS approach on V-bar, LOS approach on R-bar, fly around at a constant distance with constant relative attitude, and final approach for docking. Details of these test cases such as initial conditions, commanded attitude maneuvers, etc. are being defined to finalize the test plan.

5.2 Implementation of Fuzzy Controller for a Camera Tracking System using the software and hardware set-up

Activities planned for this year for the camera tracking system include testing of the concept in software as well as hardware simulations. The software testing will be performed in the STL using a 386 based system as well as Sun workstations. The hardware testing will be performed in collaboration with the Engineering Directorate at JSC. It should be emphasized that the software testing will help fine tune the rulebase and the membership functions, while the hardware testing will help to identify all interface problems, real-time performance evaluation, and fine tune the controller in light of actual measurements which will be noisy. Both, software and hardware, testing is required in order to make the system operational and useful.

The tracking controller described earlier in section 4.4 has been implemented using the fuzzy-C development system and necessary software modules in C language have been generated. Its interfaces with the sensor module that provides the measurements and the gimble drive module that accepts the commands have been defined and implemented in C. A top level executive has been designed as shown in fig. 11 with the necessary data flow and the state propagator for a target vehicle. At this time, the Clohessy-Wiltshire equations of motion [34] in the LVLH frame will be used to propagate the target state and to generate the camera measurements. A first order linear gimble drive model has been developed for the pan and tilt servo drives to rotate the pointing axis of the camera. The measurements for the range, horizontal and vertical positions are based on the geometry in an LVLH frame. A detailed test plan will be defined to test the concept for several different scenarios. The fuzzy tracking controller will be tested for the following types of relative trajectories : approach, fly-around, station-keeping, and passing orbits.

The hardware laboratory in the Engineering Directorate has the necessary equipment required for testing: camera, gimble drives, laser range finder and other interface equipment. The camera system will require a digitizer or pixel map generator and interfaces to the computer. The fuzzy controller software developed in the STL will be ported to this computer which will have the necessary hardware interfaces. A test plan that includes real moving targets in the laboratory and various lighting conditions to simulate the orbital environment will be generated and the performance of the fuzzy controller will be analyzed in detail. A study will be performed to determine the responsiveness of the gimble drives with respect to the changing Scale_Factor membership functions.

There is a considerable effort in the STL devoted to the development of algorithms for object identification and pattern recognition. Particularly, the emphasis is given to the algorithms for performing scene analysis and extracting the information from the image using fuzzyiness and related parameters [35]. Results of this effort can be implemented and integrated at various levels in the concept of the camera tracking system to extend the capabilities of the sensor. At what level and how to integrate these algorithms will be investigated as a part of our current activities.

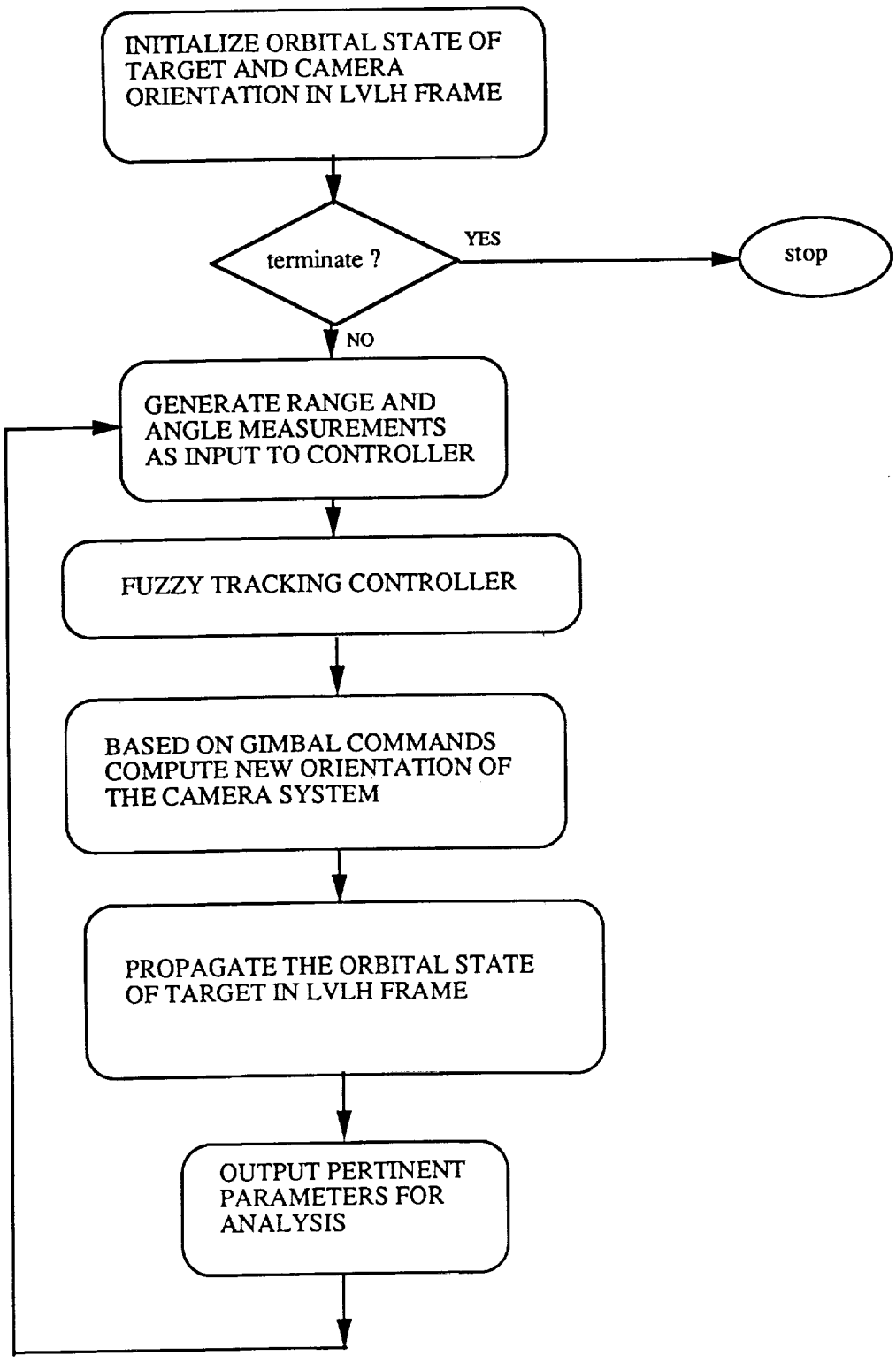


Fig. 11 TESTING OF CAMERA TRACKING FUZZY CONTROLLER IN SIMULATION SOFTWARE

5.3 Trajectory Control for Mars Rover during Sample Collection

While collecting soil samples and surveying the Mars surface, the Mars rover will be moving from one point to another among obstacles which cannot be identified prior to the mission. In order to complete the collection task, the rover must interpret imprecise sensor measurements of obstacle size and distance to determine which obstacles present a hazard and must be avoided and plan a trajectory to avoid these unforeseen obstacles. In addition, since the worst case round trip communications time between Earth and Mars will require 20 minutes, Earth-based tele-robotic control of the Mars rover will be extremely difficult and time consuming and could seriously endanger the success of the mission. Fuzzy trajectory planning and control provides robust real-time control capable of adapting the trajectory profile to avoid unforeseen hazards. The fuzzy logic approach eliminates communications travel time, allows the rover to avoid obstacles which may be unavoidable due to tele-robotic reaction time, and provides adaptable control which will extend the rover performance envelope.

A fuzzy logic approach to trajectory control has been developed [36] which allows the rover to avoid these hazards during the sample collection process. The fuzzy trajectory controller receives the goal or target point from the planner and uses X and Y position errors as well as orientation (Yaw) error in the control system frame and commands the rover in terms of steering angle and velocity. The fuzzy rule-base containing 112 rules for the controller, has been designed to drive the rover towards the X-axis of the control error frame. As the rover approaches this axis, the rover is commanded to the correct orientation error and then slowly drives towards the target point.

The X and Y position error variables were modeled as a shouldered membership set of 5 piece-wise linear functions [19] with a universe of discourse ranging from -100 to 100 meters. The orientation or yaw error variable was modeled as an unshouldered membership set of 7 functions with a universe of discourse ranging from -180 to 180 degrees. The steering variable was modeled as an unshouldered membership set of 5 functions with a universe of discourse ranging from -30 to 30 degrees. Finally, the velocity variable was modeled as an unshouldered membership set of 7 functions with a universe of discourse ranging from -5 to 5 meters/second.

A fuzzy trajectory controller for a Mars rover has been tested on several cases. Preliminary results have shown that the trajectory controller can reach the target position and attitude within 0.0005 meters on the x-error axis, 0.25 meters on the y-error axis, and 0.45 degrees yaw error. It is believed that these accuracies can be reduced by altering the membership function sets for the inputs and outputs. Further testing will facilitate the tailoring of the membership functions to the fuzzy rule set. Our activities in this project have shown that the fuzzy approach provides a control system which can be easily modified and tested.

6.0 FUTURE PROJECTS

In this section, the future activities that are planned for fiscal year (FY) 1991 and beyond are described with an expectation that these activities will be fully funded for new technology development. Activities in the area of traffic management around the SSF utilizing the camera tracking system are described first. Then, the development of reinforcement learning during docking and repair operations is described. Development of a concept for a health monitoring system is described last.

6.1 Application of Camera Tracking System for Traffic Management

Future operations around the SSF will include many vehicles approaching and departing the facility simultaneously. The crew onboard the SSF will have to perform traffic management functions very actively for safety reasons. The camera tracking system can be used effectively during these

operations and can help the crew to efficiently manage traffic around the SSF. During assembly and other Extra-vehicular operations, tracking and monitoring of other objects around the SSF is required for mission success. As part of our future activities we will investigate the applicability of the camera tracking system to the problem of traffic management around the SSF.

As part of our current activities we are planning to implement the fuzzy tracking controller in the hardware laboratory in the Engineering Directorate. The tracking controller will be interfaced with the gimble drives and a pixel map from a camera. It is also possible to interface the output of the camera to the fuzzy processor which can run the fuzzy controller and command the gimble drives. It is planned to purchase suitable fuzzy hardware and perform the necessary testing to prove the concept at the hardware level. We will investigate the performance of fuzzy chips for accuracy, timing and interfaces with a main computer. The use of the concept for several space station applications will be relatively easy and realizable.

The capabilities of the tracking controller can be expanded to perform other functions such as approach toward the object, grapple, object identification, traffic management, and caution and warning to crew. Fast moving objects can be identified easily via prediction of position and thus collision avoidance can also be achieved. Since the system can work as a stand-alone system at the command level and will interrupt the operations flow only if necessary, it can become a node in a distributed sensor system.

6.2 Reinforcement Learning for External Environment during Docking and Repair Operations

A Space Shuttle crew initiates proximity operations procedures and docking maneuvers, when the Orbiter is within 1000 feet of the payload. It is expected that the payload will remain in a stable attitude and in nearly the same orbit during this entire time. Typically, the crew performs an approach known as the v-bar approach, keeping manual control of the Orbiter. Docking maneuvers with the payload are also performed manually. The manual procedures and algorithms used during these tasks by the crew are developed using the real-time Shuttle Mission Simulator Facility on the ground.

During proximity operations, if the procedures require some adjustment, the so-called fine tuning, it is performed real-time, even if it was not learned in the real-time simulation. Real-time adjustments are achieved based upon the current situation (e.g. satellite is not in a stable attitude or its orbit is constantly changing) and goal achievements. Thus the crew constantly learns and updates these procedures and algorithms as their experience base builds-up.

It has been shown that a Fuzzy logic controller can perform the same activities autonomously using sensor measurements as inputs. Fuzzy membership functions and the associated rule base [27,28] have been developed utilizing the same procedures used by the crew during mission operations. A fuzzy reinforcement learning method [37] has been developed at Ames Research Center (ARC) using the inverted pendulum. The fuzzy controller can be combined with the reinforcement learning technique to give it a capability to learn real-time and improve its performance. With this capability, the fuzzy controller can adapt to a new environment and adjust its membership functions and/or rules to appropriately perform the tasks, given enough training instances.

The objectives of this project are to: 1) combine the fuzzy controller developed for the translational motion with the reinforcement learning technique, and 2) demonstrate its performance for the translational control of a spacecraft during proximity and docking operations. This project will be jointly undertaken by two NASA centers: JSC will provide a high fidelity spacecraft simulation, testcases with input and output definitions, and preliminary rules and membership functions for the fuzzy translational controller, while ARC will provide the learning elements with appropriate interfaces to the simulation and updated rule base.

An approach has been developed (as shown in fig. 12) to combine the fuzzy logic controller with reinforcement learning so that a higher level of autonomy for spacecraft operations can be achieved. Such an intelligent controller for a spacecraft is expected to adapt to the surrounding orbital environment and adjust its control strategy. Initial work on this project has been started and a project plan has been put together [38]. Details of the rule base, membership functions, input parameters, output commands and other simulation interfaces are in work.

6.3 Concept Development for Health Monitoring System for Environment and Life Support System for Large Volume Crew Quarters

Continuous monitoring and control of the Environment and Life Support System (ELSS) onboard the SSF is required for the safety of the crew. The preliminary design of the ELSS control system (also known as atmospheric control system) consists of temperature, pressure and composition control which are highly interrelated. The composition control includes control of major cabin atmosphere constituents, oxygen and nitrogen, and the control of humidity and trace contaminants. This preliminary design is based on the following requirements [39].

Relative humidity must be maintained between 25 and 70 % with the constraint that the dew point temperature is always maintained above 59 deg. F. The cabin temperature must be selectable between 64 and 81 deg. F and must be controlled within one deg. accuracy. The cabin atmospheric pressure must be maintained at 14.7 psia within 0.2 psia accuracy. The oxygen partial pressure must be maintained at 2 psia.

The system dynamics model or the *plant* that represents the behavior of the system is non-linear and parameters are highly interrelated. The system equations can be linearized when the volume of the cabin is small and several simplifying assumptions are made. However, the dynamics becomes increasingly complex and non-linear as the volume of the crew quarters increases significantly. In such cases, applying conventional control theory will be very difficult, if not impossible.

In order to properly control the system state, highly accurate information regarding the current state of the system is required. Multiple sensor measurements are required to derive this accurate state information. It should be noted that the accuracy of state information is dependent on sensor accuracy. The sensors will possibly be distributed over the entire volume of the cabin. Thus, the problem can be thought of in two steps: deriving state information based on sensor measurements and controlling the deviations from the desired state. The first step relates to the interpretation of measurements, particularly their accuracy. The second step relates to the control of the state.

A concept of a fuzzy logic based monitoring and diagnosis has been developed to combine several sensor measurements and derive the state information of a non-linear system. The concept can be expanded to maintain a desired state, detect potential component failures and generate immediate advisory messages for corrective actions. As part of our activities in FY91, we will apply this concept to the ELSS of SSF and implement a fuzzy rulebase and membership functions. We will further generate a software demonstration as a proof of the concept and evaluate the suitability of the fuzzy logic based monitoring technique.

7.0 SUMMARY

Applications of fuzzy logic in autonomous orbital operations are described in this paper with past accomplishments at JSC. Current ongoing as well as future activities planned are also described. The main objective of all these activities is to increase autonomy in orbital operations and thus achieve a higher level of operational efficiency desired for future space operations. The approach is to develop modular control that can be upscaled for greater autonomy in an integrated environment. The initial step is to develop a software controller and then to integrate it with hardware at appropriate level. As the activities progress, detailed testing will be performed to check out

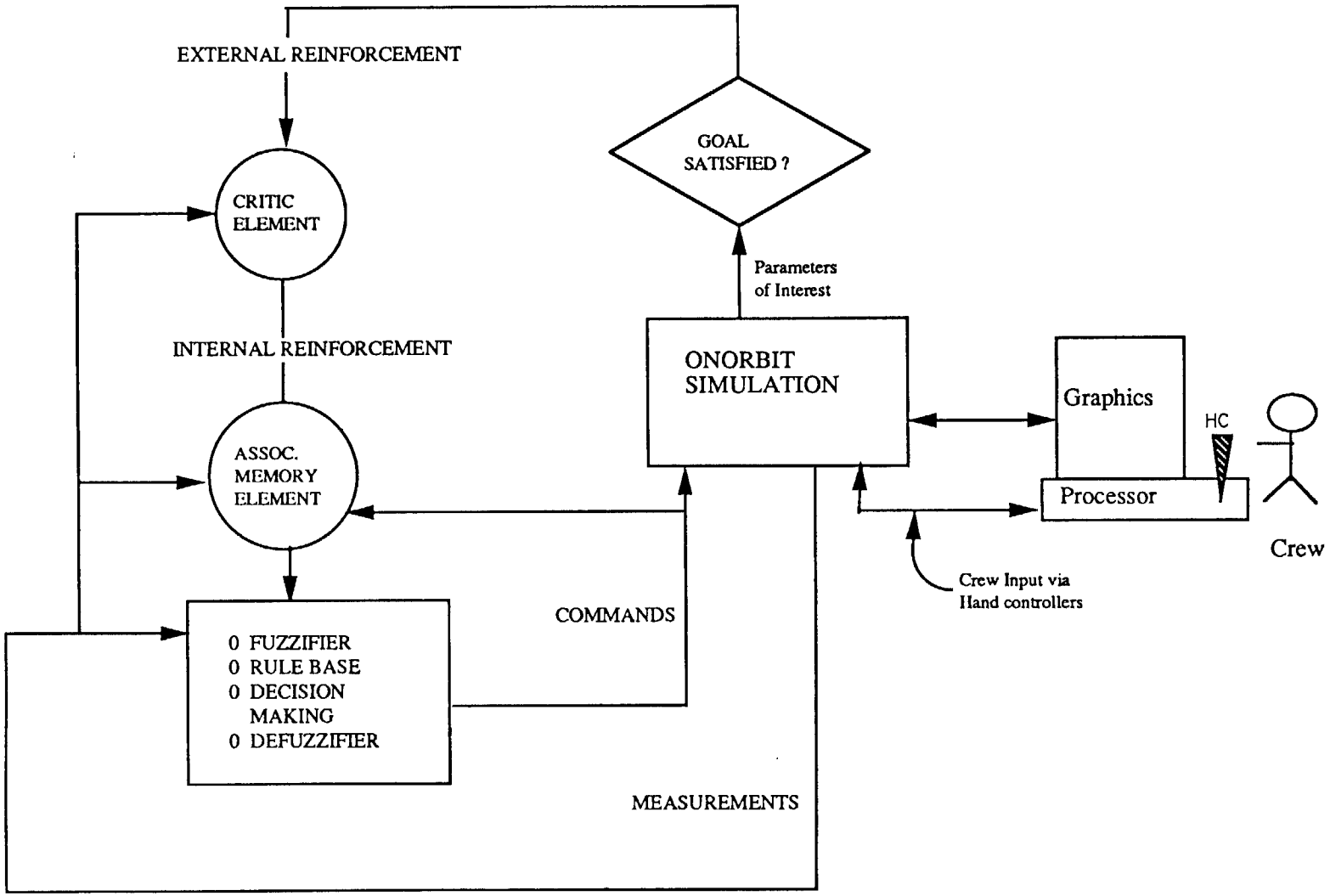


Fig. 12 FUZZY LEARNING SYSTEM FOR DOCKING OPERATIONS

implementation and integration of components. Our preliminary results promise a very successful utilization of fuzzy logic in autonomous orbital operations.

REFERENCES

1. Zadeh, L. : "Fuzzy Sets", Information and Control, vol. 8, pp. 338-353, 1965.

2. Klir G. J. ; and Folger T. A. : Fuzzy sets, Uncertainty, and Information, Prentice Hall, New Jersey, 1988.

3. Antsaklis, P. J.; Passino, K. M.; and Wang, S. J. : An Introduction to Autonomous Control Systems, Proceedings of 5th IEEE International Symposium on Intelligent Control, Volume 1, pp. 21-26, 1990.

4. Rogers, M. ; and Hoshiai, Y. : The Future Looks 'Fuzzy', NEWSWEEK, May 28, 1990.

5. Johnson, R. C. : Clear Leader Emerges : Japan at fuzzy fore, EETimes, Sept. 11, 1989.

6. Armstrong, L. ; and Gross, N. : Why 'Fuzzy Logic' beats black-or-white thinking, Science & Technology section, BUSINESS WEEK, May 21, 1990.

7. Yasunobu, S. ; and Miyamoto, S. : "Automatic Train Operation System by Predictive Control", Industrial Applications of Fuzzy Control, Sugeno, M. (Ed.), 1-18, North-Holland: Amsterdam, 1985.

8. Shingu, T. ; and Nishimori, E. : Fuzzy Based Automatic Focusing System for Compact Camera, Proceeding of IFSA89, pp. 436, 1989.

9. Tobi, T. ; Hanafusa, T. ; Itoh, S. ; and Kashiwagi, N. : Application of Fuzzy Control System to Coke Oven Gas Cooling Plant, Proceedings of IFSA89, pp. 16, 1989.

10. Yamakawa, T. : How Fuzzy Microprocessors Work, Tutorials at the International Workshop on Fuzzy System Applications, pp. 79, 1988.

11. Togai, M. ; and Corder R. J. : A High Speed Fuzzy Processor for Embedded Real-time Applications, Proceedings of SICE, 1989.

12. Corder, R. J. : A High Speed Fuzzy Processor, Proceedings of IFSA89, pp. 379, 1989.

13. Watanabe, H. : VLSI Chip for Fuzzy Logic Inference, Proceedings of the 3rd International Fuzzy Systems Applications Congress, pp. 292, 1989.

14. Tasaka, Y. : Hybrid Bus Type Fuzzy Controller with Analog Fuzzy Chips, Proceedings of IFSA89, pp. 280, 1989.

15. Johnson, R. C. : Fuzzy faithful to get first glimpse of Expert, Technology section of EETimes, pp. 25, June 5, 1989.

16. Lea, R. N. ; and Jani, Y. K. : Automation Issues for Rendezvous and Proximity Operations, Proceedings of the Autonomous Rendezvous and Docking Conference, NASA/Johnson Space Center, Houston, Texas, August 1990.

17. Togai, M. : Commercial Applications of Fuzzy Logic in Japan , Presentation at the workshop on Fuzzy Control Systems and Space Station Applications (sponsored by NASA/Ames Research

Center and McDonnell Douglas Space Systems Company Space Station Division), Huntington Beach, California, November 1990.

18. Perkins, C. ; Teichrow, J. ; and Horstkotte, E. : Fuzzy-C development system : A complete overview, Togai InfraLogic Inc., SOAR-89 conference held at Johnson Space Center, Houston, July 25-27, 1989.

19. Hill, G.; Horsthotte, E.; and Teichrow, J. : TIL Shell User's Manual, release 1.1, Togai InfraLogic Inc., Jan. 1990.

20. Teichrow, J. ; and Horstkotte, E. : Fuzzy-C compiler User's manual, v2.0b, Togai InfraLogic Inc., Irvine, California, April 1989.

21. Lea, R. N., Giarratano, J., Fritz, R. H., and Jani, Y. K. : Fuzzy Logic Control for Camera Tracking System, Proceedings of the 8th International Congress of Cybernetics and Systems, New York, June 1990.

22. Lea, R. N. ; and Goodwin, M., Use of Fuzzy Logic to Determine Navigation Filter Convergence During Space Shuttle Rendezvous, ISA Proc. of Conf. on Robotics and Expert Systems, NASA/Johnson Space Center, Houston, Texas, June 1986.

23. Lea, R. N.: A Fuzzy Set Approach to a Navigation Decision Making Problem, Proceedings of the 1985 Conference on Applied Analysis, University of Houston at Clear Lake, Houston, Texas, 1985

24. Lea, R. N. ; and Giarratano, J. : An Expert System Program Using Fuzzy Logic For Shuttle Rendezvous Sensor Control, proceedings of ROBEXS'86, pp. 327-329, 1986.

25. Lea, R. N. : Applications of fuzzy sets to data monitoring and process control, Proceedings of the ISA88 conference, Vol. 43, part 4, pp. 1495 1988.

26. Lea, R. N.: Applications of Fuzzy Sets in Redundancy Management and Filtering of Sensor Data, Proceedings of the Conference. on Modeling and Simulation, Univ. of Pittsburg, 1987.

27. Lea, R. N. : Automated Space Vehicle Control for Rendezvous Proximity Operations, Telematics and Informatics, vol. 5, no. 3, pp 179-185, 1988.

28. Lea, R. N. : Applications of fuzzy sets to Rule-based Expert System Development, Telematics and Informatics, vol. 6, nos. 3/4, pp 403-406, 1989.

29. Lea, R. N. : Fuzzy Logic Space Vehicle Controller for Rendezvous Proximity Operations, Video Conference Demonstration from Johnson Space Center, International Workshop On Fuzzy Systems Applications (IFSA-88), Iizuka, Fukuoka, Japan, August 20-24 1988.

30. Lea, R. N. ; and Jani, Y. : Spacecraft Attitude Control System Based on Fuzzy Logic Principles, proceedings of ROBEXS'89, 1989.

31. Lea, R. N. ; Togai, M. ; Teichrow, J. ; and Jani, Y. : Fuzzy Logic Approach to Combined Translational and Rotational Control of a Spacecraft in Proximity of the Space Station, Proceedings of the IFSA'89, pp. 23-29 1989.

32. Lineberry, E. C.; Lea, R. N.; and Mitchell, E. V. : Targeting equations and control strategy for proximity operations, Internal note, Mission Planning and Analysis Division, Johnson Space Center, March 1988.

33. Edwards, H. C. ; and Bailey, R. : The Orbital Operations Simulator User's Guide, LinCom corporation, ref. LM85-1001-01, June 87.

34. Clohessy-Wiltshire equations of motion, Shuttle targeting section, Shuttle onboard flight software subsystem requirements, Guidance and Targeting, vol. Sd--xxx--007, Rockwell International, 1981.

35. Pal, S. K. : 'Fuzziness, Image Information and Scene Analysis' in An Introduction to Fuzzy Logic Applications in Intelligent Systems edited by R. R. Yager & L. A. Zadeh, Kluwer Academic Publishers (to appear).

36. Lea, R. N.; Walters, L.; and Jani, Y. K. : A Fuzzy Logic Approach to Mars Rover Trajectory Planning and Control, Proceedings of the 1st International Symposium on Measurement and Control in Robotics, Session D.3, pp. D.3.1.1, June 1990.

37. Lee, C. C. ; and Berenji, H. R. : An Intelligent Controller Based On Approximate Reasoning And Reinforcement Learning, Proceedings of IEEE International Symposium on Intelligent Control, Albeny, NY 1989.

38. Lea, R. N., Jani, Y. K., and Berenji, H. : Fuzzy Logic Controller with Reinforcement Learning for Proximity Operations and Docking, Presentation at the 5th IEEE International Symposium on Intelligent Control, vol. 2, pp. 903, 1990.

39. Mankamyer, M. : Atmospheric Control Systems, Presentation at the workshop on Fuzzy Control Systems and Space Station Applications (sponsored by NASA/Ames Research Center and McDonnell Douglas Space Systems Company Space Station Division), Huntington Beach, California, November 1990.

Identification and Control of Dynamical Systems Using Neural Networks

(Paper not provided by publication date.)

Non-Lipschitzian Dynamics

(Paper not provided by publication date.)

Space Shuttle Main Engine Fault Detection Using Neural Networks

NETROLOGIC

Thomas Bishop
Dan Greenwood
Kenneth Shew
Fareed Stevenson

5080 Shoreham Place, Suite 201
San Diego, California 92122
(619) 587-0970

ABSTRACT

A method for on-line SSME anomaly detection and fault typing using a feedforward neural network is described. The method involves the computation of features representing time-variance of SSME sensor parameters, using historical test case data. The network is trained, using backpropagation, to recognize a set of fault cases. The network is then able to diagnose new fault cases correctly. An essential element of the training technique is the inclusion of randomly generated data along with the real data, in order to span the entire input space of potential non-nominal data.

1. Introduction

NETROLOGIC has devised a new system that uses neural networks for on-line detection of fault conditions in the Space Shuttle Main Engine (SSME). In order to recognize danger signs early enough to shut down the rocket engine and minimize damage resulting from unforeseen malfunctions, an SSME fault detection system needs to be faster and more accurate than existing systems. Even with the current failure response systems which utilize automatic redlining, redundant sensor and controller voting logic, and human monitoring, post test analysis shows the emergence of anomalous engine behavior well before a shutdown sequence is initiated. Neural networks can provide improved test-stand SSME fault detection with natural extensions to in-flight monitoring.

A fast SSME diagnostic method is essential since a large number of simultaneous sensor measurements (over 200 are available) are input to a test shutdown decision module at a high sampling rate. Sensor data fusion and evaluation are complicated issues since clues to engine performance may involve subtle combinations of sensor measurements varying through time. There is a high cost associated with unnecessary shut-downs (false alarms) as well as missed detections (failure to detect an impending catastrophe).

A detection system should not alter the current engine or control system and should utilize all existing data. Since the SSME's major components are line replaceable units, ideally a fault detection system should be independent of engine-to-engine performance variation and of older engine failure signatures.

Neural networks can contribute to an effective solution since they are

1) fast, especially if implemented on parallel hardware;

2) capable of discovering subtle patterns of input data without being explicitly taught what combinations are significant;

3) capable of generalizing based on previously learned examples; and

4) robust --- relatively insensitive to noisy data.

2. Data Source and Description

We used the well-known backpropagation to train our three layer feedforward network with training examples from sensor data from actual SSME test cases (see Figure 1), conducted between 1981 and 1989. Most of the data resulted from recordings of cases in which faulty engine performance occurred. We restricted our attention to time periods after the SSME reached full power, since steady-state fault diagnosis is a sufficiently difficult and important problem, and the use of data from periods of

transient SSME operation would introduce considerable complications. We will investigate the application of neural nets to failure detection during the transient phase in the near future. Neural nets can recognize distinctive time series such as temperature transients, and will be useful for rocket engine transient analysis.

The six fault cases that we have used represent failures of various types, caused by malfunctions in different hardware components, such as a fuel leak in the main combustion chamber outlet neck in one case, and a cracked liquid oxygen post in another. Although this provides a variety of data for training and testing, it also means that there is not enough fault data to generalize about any particular failure type.

In each of the fault cases we observed that there was a relatively long period during which the SSME functioned normally prior to malfunctioning, consequently, there was an abundance of nominal sensor data. However, there was a very limited amount of fault data in three cases, because the interval between the fault-declare time and the time of the last sensor measurements was very short (as short as 0.2 seconds).

The fault-declare time for each of the fault cases was based on an analysis of failure investigation reports which showed the time when sensors started to indicate signs of problems or faulty performance. We determined the time when a fault-detection system should have been able to declare that something was wrong enough to warrant shutting down the SSME. Sensor samples taken before the fault-declare time are considered nominal, and samples taken after that time are considered fault data.

We only used a subset of the total number of different sensor measurements, referred to as Parameter Identifiers (PIDs). These PIDs were sampled 25 times per second. We selected twelve PIDs (see Figure 2) for use in our current study. Selection of this subset of data was based on two factors:

1) Availability for all cases under investigation. Different test cases were inconsistent in which sensors were installed and functioning. Since a fundamental objective is to combine data from different test cases, and generalize to other cases, data must have the same format for all cases. Therefore we only chose a PID if it was available for nearly all of the cases used in our study. However, this is not an absolute restriction: if a particular PID is missing from a particular test case, it is possible to use null values for that PID in that case. In fact, it is essential that our method should accommodate missing, faulty, or "dead" sensors.

2) Significance for diagnosis. Analysis of fault case profiles shows that, for a given case, some sensors show strong early symptoms of faulty operation, while other sensors appear to have less value for diagnosis. Naturally we chose PIDs which were significant in the cases under investigation.

3. Pre-Processing of Data

The inputs to the network were derived from PID values. Each sample fed into the network corresponded to a particular point in time. However, the input values were not simply the raw values for each PID at that time. The nature of the variation in PID values over time may be more indicative of faulty performance than the value of the PIDs at any isolated moment. For example, in case 901-331, fault symptoms included an increasing HPOT discharge temperature concurrent with a decreasing MCC pressure. Therefore, for each point in time, three features were calculated for each PID, which take into account the medium, long, or short-term history of that PID leading up to that time. These features are described in Figure 3.

Thus, the total number of simultaneous inputs to the network for each point in time was three times the number of PIDs. We have used twelve PIDs and 36 input units. In future studies, more features will be computed for each sample, to provide more detailed input of time-variation of PIDs, or to explicitly input features which code relationships between other features. In theory, the network is capable of performing any computation on the inputs, so such compound features would be superfluous. In practice, however, it might prove to be useful to input such features explicitly in order to encourage the network to learn in a way that will lead to better generalization. The three features currently used are minimal, yet appear to be sufficient for the tasks attempted so far.

4. Network Architecture

We used a feedforward neural network model consisting of a layer of input units, plus one or more layers of hidden units, plus a layer of output units. Units are analogous to neurons. The connections between them are analogous to synapses. In the feedforward model, each of the input units is connected to each of the hidden units, and each of the hidden units is connected to each of the output units. Each of the connections is characterized by a weight, which is the strength of the connection. In the basic operation of the network, connections are one-way, going from inputs to outputs (hence the name feedforward). Each unit attains a level of activation by taking the weighted sum of its inputs. It then produces its own output, which is a function of its activation. We have used the logistic function given by $f(x) = 1 / (1 + \exp(-x))$.

Feedforward networks can be trained to associate arbitrary input patterns with arbitrary output patterns, and they have the ability to categorize and generalize, so that similar inputs are mapped to similar outputs, and new input patterns (different from those on which the network has been trained) will be mapped to outputs based on their similarity to training patterns. Training is accomplished by the generalized delta rule (backpropagation of error). After each input sample is fed forward through the network, the output is compared with the desired output. The weights are then adjusted iteratively to reduce any discrepancies (for a detailed description of backpropagation, please see [6])

The choice of how many hidden layers to use, and how many units to have in each layer, is dictated by two opposing factors. On the one hand, it is generally easier for a network to perform an exact mapping from a set of inputs to a desired set of outputs, if there are more hidden units. On the other hand, if there are too many hidden units, the network is liable to "over-learn" the training data, and may be less successful at generalizing to new data. We have found that a single hidden layer of three to six units is sufficient for the network mappings we have attempted so far.

5. Assignment of roles to output units

The output of the network represents its evaluation of the input data. The activations of the output units are all floating-point numbers, which take on values anywhere between zero and one. We currently use three output units, each of which represents a different diagnosis category. The three categories are:

1) Nominal

2) Fault (of a type previously witnessed)

3) Deviant (anything that departs from nominal).

For each output unit, activation levels near 1.0 mean "yes", and levels near 0.0 mean "no". Intermediate levels of activation may be regarded as the degree of confidence in that diagnosis.

The first priority of an SSME fault detection method must be to decide when to shut down the engine to minimize damage leading to a potential catastrophe. To the extent that this is a yes-or-no decision, we only need to know whether or not the engine's performance is nominal. This may be described as anomaly detection. Beyond this, however, it may be necessary to distinguish between different failure types. This will be true if different shut-down or safety procedures are employed depending on failure type. Also, if the neural network forms a part of a larger fault detection system, it may be of value for the network to report what failure type it perceives, thus providing a more useful input to the rest of the system.

Fault detection should involve the notification of a failure, the isolation of the type of failure, and the estimation of the severity. The detection of a failure which would warrant a shutdown sequence was emphasized, the isolation and estimation functions were secondary. Further study for isolation and estimation will also be pursued, however, a system which emphasizes detection during testing would alleviate some of the complexity or computational burden associated with pursuing all three goals of fault detection simultaneously.

Under the constraint of limited fault data, and keeping in mind the primary

importance of shut-down decision making, we focused on anomaly detection rather than fault-typing, and employed only a single output unit for the "fault" category. In the future, when more fault data (real or simulated) becomes available, our method may be extended with no fundamental changes to incorporate more output units for individual failure types.

Using only historical nominal and fault data, the network can be trained to distinguish nominal and fault data that it is trained on, but when we ask it to generalize to new cases (cases that have not been used for training), the results may be disappointing. Unless a new case is very similar to one of the training cases, this new fault data will not resemble the old fault data any more than it resembles the old nominal data. In our experience, the network output "nominal" for all samples in the new faault cases, both before and after the fault-declare time. Evidently the problem was that the fault data in the training cases were too limited, involving only particular PIDs with specific time profiles. A network trained to recognize a particular small set of fault cases cannot be expected to recognize a new fault case, which is likely to involve different PIDs indicating degraded performance with completely new behavior.

In order to train a network to distinguish nominal data from all possible non-nominal data, we needed a source of non-nominal data. Fault data from real fault cases were insufficient for this purpose since, even if we used all the fault data currently available, it would still not span the entire space of potential non-nominal data. Therefore, we experimented with using random data evenly distributed throughout the total input space of the network. We called these data "deviant." The network was given a combination of nominal, fault, and deviant data, and trained to recognize each type. The extra task of recognizing deviant data forced the network to learn the boundaries of the nominal data.

6. Training Method and Initial Results

Our usual method was to train a network on data from several SSME test cases shuffled together with randomly generated "deviant" data, test the network on the training cases, and also test on new cases. In three of the cases there were very low proportions of fault data. Therefore, in order to train the network on a balanced set of samples, the fault samples in those cases were duplicated a hundred times in the training data file before it was shuffled.

When we trained and verified the network on actual fault cases, we found that the network was capable of learning the training data with very high accuracy. It would output "nominal" when fed nominal data, and "fault" when fed fault data. When learning was not quite perfect, the incorrect outputs always occurred for data immediately before or after the fault-declare time. This showed that the transition period around the fault-declare time was the most difficult to learn, as it should be if the network was using criteria involving the continuous progression of PID values through time.

The only case which presented some difficulty was case 249. It is not clear from post-test analysis what fault-declare time is appropriate for this case. Proposed times range from as early as 320 seconds to as late as 405 seconds after start-up. When we used an early declare time and combined case 249 with other cases for training, the net had difficulty reconciling this with the other cases used during training. Apparently, the data in the middle period of 249 is too similar to other data which is nominal, so that it could only be learned as a fault through overlearning, that is, by paying too much attention to distinguishing details with no relevance to fault symptoms.

Our initial results with generalizing to new cases were very promising. The network was able to diagnose new fault cases correctly without training. As expected for these cases, none of the data was evaluated as faulty. Data before the fault-declare time was classified by the network as nominal, and data after the fault-declare time was classified as deviant. The fault-declare times for untrained fault cases determined by the networks have been remarkably consistent with the fault-declare times established on the basis of expert post-test analysis. In case 249, mentioned above, a network (which had been trained on cases 259, 331, 436, and random data) diagnosed the data as deviant after 331 seconds; our proposed fault-declare times ranged between 320 and 405 seconds. The same network, when tested on case 340, output strongly deviant after 283 seconds. Our fault-declare times ranged between 280.3 and 290 seconds.

7. Other Failure Detection Systems

A typical tradeoff consideration for failure detection is detection performance versus filter behavior under normal conditions. A design specific to certain failures may provide failure isolation at the expense of performance in detecting nominal data. Certain detection filters take into account such a tradeoff. Under normal or nominal conditions, the bandwidths of the Kalman filters used in detection filters will be increased to be sensitive to the failure isolation designs, yet this increase makes the system more susceptible to sensor noise. With the incorporation of the deviant output, neural nets do not have to be trained to detect specific failures and detection performance will not be hindered under normal conditions. Normal operation should not degrade, since neural nets can be insensitive to sensor noise.

Another failure detection system involves voting schemes. Such schemes can efficiently rule out faulty sensors and are very useful for false alarms, but often pay the price of hardware redundancy for a reliable means of failure detection. Failures such as thermal effects and power failures can also affect the "like" sensors utilized by voting systems in the same way. Since failure detection involves voting between these like sensors, a problem which affects all the sensors will not be detected.

Multiple hypothesis filter-detectors can be too complex for a practical failure detection system [8], [9]. Multiple hypothesis filter-detectors are considered to yield the best performance in the widest class of field for detection, isolation, and estimation, but the complexity can be of major concern. These filters involve the computation of

probabilities of all the types of failures under consideration, which may require much time and storage capabilities. Neural nets, on the other hand, are not considered very complex in terms of what the network or implementor has to do. Storage and time considerations are not a problem with neural nets either. When implemented in massive parallelism or by an accelerator board, neural nets are able to respond quickly. Very little computational overhead exists since nets require only two matrix multiplication and two activation applications. The matrices involved in the computation to determine the output are the interconnection matrix between the input and hidden layer and the interconnection matrix between the hidden and output layer. Since only two layers are needed for a successful neural network, only two activation applications are required also. Moreover, neural nets should be able to perform well for SSME fault detection. Some other failure sensitive filters can also become oblivious to new sensor outputs by learning the data too well. In these cases, the Kalman filter and the precomputed covariance utilized become too small and, therefore, oblivious to new data.

Innovations-based detection systems, such as the generalized likelihood ratio (GLR) test, can be sensitive to modeling errors [5], [9]. The GLR test may provide fast failure recovery, but it is imperative for a good estimation of failure parameters that the model is accurate. Neural nets are not considered very complex and the creation of accurate models is not difficult.

The key issues to be addressed in discussing the merits of one system compared to another are complexity in implementation, performance with respect to false alarms and delays in detection, and robustness, such as modeling errors and sensitivity concerns. Our initial results indicate that neural nets do very well in resolving these issues in comparison with other methods.

8. Acknowledgement

We gratefully acknowledge Claudia Meyer and Larry Cooper of NASA Lewis, for their assistance in obtaining data and explaining the subtleties of interpreting it, Dr. Chris Bowman of Ball Corporation for helping configure the neural network we used, and James Villareal of NASA Johnson Space Center for his programmatic support.

Figure 1: SSME (Space Shuttle Main Engine) TEST CASES

Six Fault Cases

Case 901-331 *July 15, 1981*
LOX Post Fractures, Erosion-MCC
Time 152 - 233.48; Fault-Declare 232.3
2010 nominal, 28 fault, 2038 total samples

Case 902-249 *September 21, 1981*
Power Transfer Failure, Turbine Blades
Time 261.96 - 450.56; Fault-Declare 320
1451 nominal, 3265 fault, 4716 total samples

Case 901-340 *October 15, 1981*
Turn Around Duct Cracked/Torn
Time 201.96 - 300; Fault-Declare 280.6
1966 nominal, 486 fault, 2452 total samples

Case 901-364 *April 7, 1982*
Hot Gas Intrusion to Rotor Cooling
Time 131.96 - 230; Fault-Declare 210
1951 nominal, 501 fault, 2452 total samples

Case 901-436 *February 14, 1984*
Coolant Liner Buckle
Time 551.96 - 611.08; Fault-Declare 610.55
1471 nominal, 8 fault, 1479 total samples

Case 750-259 *March 27, 1985*
MCC Outlet Manifold Neck, Fuel Leak
Time 41.96 - 101.50; Fault-Declare 101.3
1485 nominal, 4 fault, 1489 total samples

Two Nominal Cases

Case 902-457 *November 1988*
Time 100 - 250
3751 nominal samples

Case 902-463 *February 1989*
Time 101.96 - 238.16
3405 nominal samples

Figure 2:

PIDs (Parameter ID's) for SSME (Space Shuttle Main Engine)

18 (566) MCC CLNT DS T
Main Combustion Chamber Coolant Discharge Temperature B

24 (371) MCC FU INJ PR (MCC HG IN PR)
Main Combustion Chamber Hot Gas Injector Pressure A

40 OPOV ACT POS
Oxidizer-Preburner Oxidizer Valve Actuator Position A

42 FPOV ACT POS
Fuel Preburner Oxidizer Valve Actuator Position A

52 (459) HPFP DS PR
High Pressure Fuel Pump Discharge Pressure A

63 MCC PC
Main Combustion Chamber Pressure Average

209 (302) LPOP DS PR
High Pressure Oxidizer Pump Inlet Pressure A

231 (663) HPFT DS T1 A
High Pressure Fuel Turbine Discharge Temperature A

232 (664) HPFT DS T1 B
High Pressure Fuel Turbine Discharge Temperature B

233 HPOT DS T1
High Pressure Oxidizer Turbine Discharge Temperature A

234 HPOT DS T2
High Pressure Oxidizer Turbine Discharge Temperature B

261 (764) HPFP SPEED
High Pressure Fuel Turbopump Shaft Speed

These are all CADS sensor measurements taken 25 times per second. Numbers in parentheses are corresponding facility measurements.

Figure 3: Features computed for each PID for each sample

(1) **(AVG2(t) - AVG1(t)) / s**

(2) **(AVG2(t) - AVG2(t0)) / s**

(3) **(X(t) - AVG1(t - .08)) / s**

Where,

AVG2(t) is the mean value of the PID for the 2 seconds (50 samples) leading up to time t.

AVG1(t) is the mean value of the PID for the 0.08 seconds (3 samples) leading up to time t.

s is the standard deviation of the PID value.

t0 is time soon after SSME reaches steady-state operation.

X(t) is the value of the PID at time t.

These three features are intended to encode the essential history of each PID value, providing sufficient information for the neural network to perform fault diagnosis. They represent the degree of change (positive or negative) over medium, long, and short periods of time.

The time t0 is used to calculate a base average value for each PID, to provide an unchanging reference point for measuring the long-term change in the PID value. We have simply used the first 2 seconds of data in the time-slice used for each test case to compute AVG2(t0).

In order to make all of the network inputs fall within the same range, all three features are scaled according to the standard deviation of the PID. The standard deviation does not depend on the particular test case; for each PID, a standard deviation is calculated on the basis of all available test cases combined.

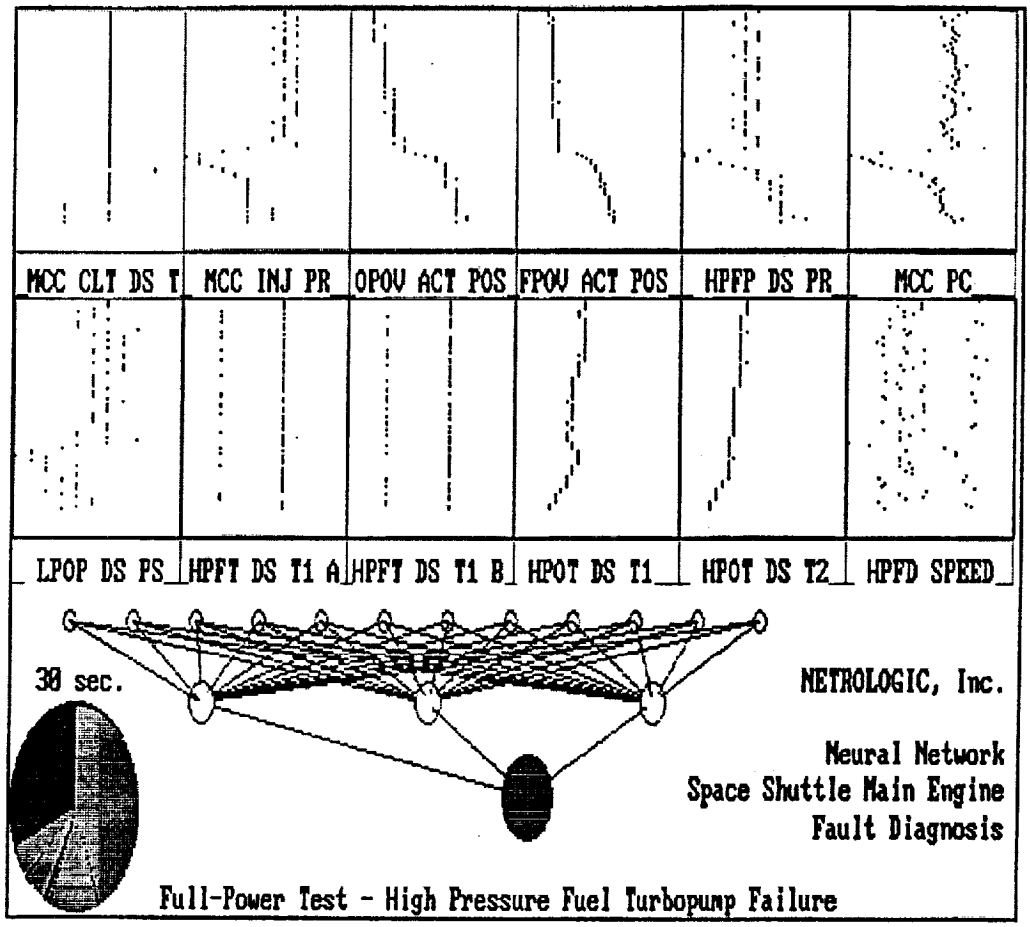


Figure 4: Conceptual Diagram

This is a computer-screen image of our demonstration program. The windows at the top of the picture are graphs of the twelve PID values varying with time. The schematic diagram conceptually portrays the neural network units and connections. Twelve inputs, three hidden units, and a single output unit are shown (note that our current approach actually employs 36 input, 6 hidden and 3 output units).

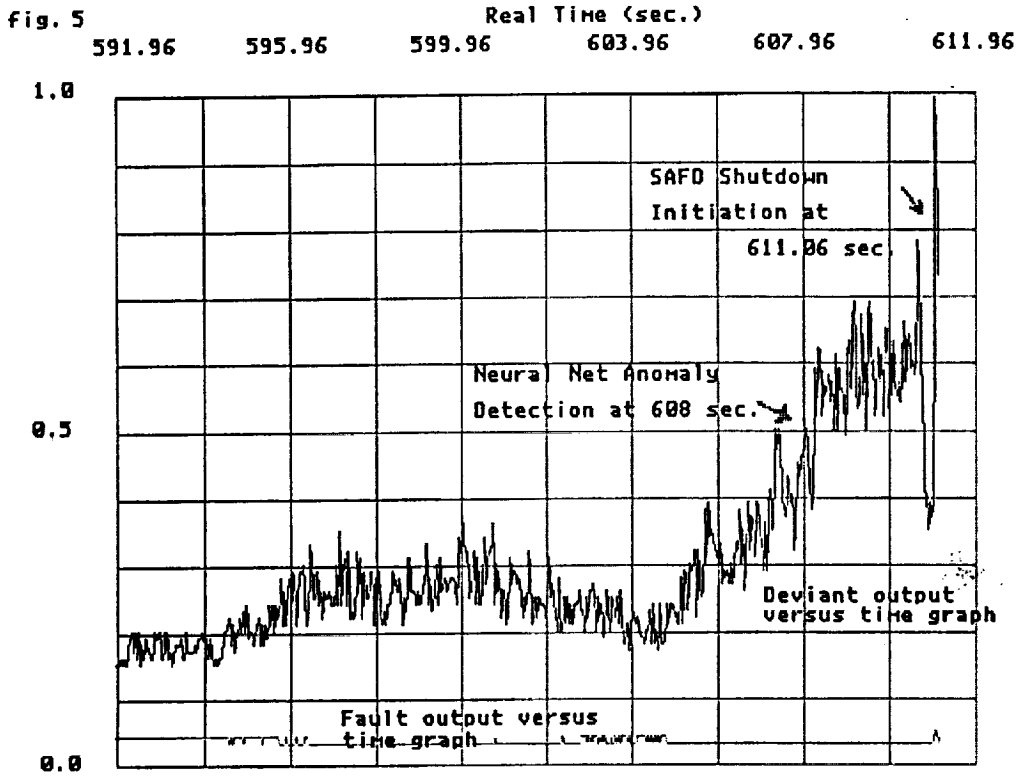


Figure 5: Graph of Neural Network Output

This shows the results of training the neural net on a case where the primary and secondary faceplates burned causing a problem in the main combustion chamber (901-331), a case where cracks were found in the high pressure fuel turbopump (901-340), and a case where a hotgas intrusion to rotor cooling occurred from a breach in a kaiser helmet (901-364). After training, the network was tested on case 901-436, where the high pressure fuel turbopump was massively damaged. The graph shows that the neural net provided earlier fault detection than that of the SAFD results provided in the "Failure Control Techniques Report For The SSME," by Rocketdyne. The graph of the third output unit, which indicates nominal data, is not shown. The nominal output is simply the reflection of the deviant output around the horizontal axis labelled 0.5.

References

[1] Cikanek, Harry A.: "Space Shuttle Main Engine Monitoring Experience And Advanced Monitoring Systems Development."

[2] Cikanek, Harry A.: "SSME Failure Detection." Proceedings of the American Control Conference, June 19-21, 1985, NASA MSFC.

[3] Dietz, Kiech, and Ali: "Neural Network Models Applied to Real-Time Fault Diagnosis." Journal of Neural Network Computing, Vol. 1 No. 1, 1989.

[4] Kelley, Glover, Teeter, and Tischer: "Diagnostic Needs of the Space Shuttle Main Engine." SAE Technical Paper Series, Oct. 15-18, 1984.

[5] Kerr, T.H.: "The Controversy Over Use of SPRT and GLR Techniques and Other Loose-Ends in Failure Detection." American Control Conference, Proceedings, Vol. 3, pp.966-977, 1983.

[6] McClelland, J.L., and Rumelhart, D.E., "Learning Internal Representations by Error Propagation," pp. 318-364, in *Parallel Distributed Processing*, Vol. I., The MIT Press, Cambridge, MA, 1987.

[7] Mehra and Peschon: "An Innovations Approach to Fault Detection and Diagnosis in Dynamic Systems." Automatica, Vol.7, pp. 637-640, 1971.

[8] Taniguchi, M.H.: "Literature Review Results." Failure Control Techniques For The SSME, Rocketdyne, Phase I report. April, 1987.

[9] Willsky, Alan S.: " A Survey of Design Methods For Failure Detection in Dynamic Systems." Automatica, Vol. 12, pp. 601-611, 1976.

Space Shuttle Main Engine Fault Detection Using Neural Networks

NETROLOGIC

Neural Networks and Fuzzy Logic Workshop
NASA, Johnson Space Center
April 11-13, 1990

GOALS AND NATURE OF PROBLEM

- USE TRAINABLE PATTERN CLASSIFIERS FOR SPACE SHUTTLE MAIN ENGINE ANOMALY DETECTION
- PROVIDE EARLIER AND MORE ACCURATE ON-LINE ANOMALY DETECTION
  (PREVIOUS DETECTION SYSTEMS - REDLINES, HUMAN MONITORING - MISSED EARLY SIGNS OF ENGINE FAILURE)
- IMPROVE TEST STAND MONITORING, EXTEND TO IN-FLIGHT MONITORING
- SHUTDOWN DECISION MODULE MUST INTEGRATE AND EVALUATE LARGE NUMBER OF SIMULTANEOUS SENSOR MEASUREMENTS AT HIGH RATE
- HIGH PENALTY FOR
  - FAILURE TO DETECT IMPENDING CATASTROPHE
    (TEST-STAND DAMAGE AS HIGH AS $26 MILLION FOR A SINGLE FAILURE; FAILURE IN FLIGHT, IF IT EVER OCCURS, MAY CAUSE LOSS OF HUMAN LIFE)
  - UNNECESSARY SHUT-DOWN (FALSE ALARM)
    (COSTS THOUSANDS OF DOLLARS ON TEST STAND; IN FLIGHT, EMERGENCY LANDING WITH ENGINE SHUT DOWN UNNECESSARILY MAY ENDANGER LIFE)

- AS SHUTTLE ENGINE FIRING IN PROGRESS, "RAW" INPUT TO ANOMALY DETECTION SYSTEM IS SEQUENCE OF VECTORS

$$P(\tau_i) \quad i = 0, 1, \ldots, s-1$$

(s = # SAMPLES TAKEN SO FAR)

- TIME STARTS FROM LAUNCH: $\tau_0 = 0$

- SAMPLES TAKEN AT REGULAR RATE

  (TYPICAL SAMPLING RATE 25 PER SECOND, OR ONE SAMPLE EVERY 0.04 SECONDS)

- FOR EACH POINT IN TIME τ, EACH COMPONENT OF $P(\tau)$ IS THE VALUE OF A PARTICULAR SENSOR MEASUREMENT

$$P(\tau) = (P_1(\tau), P_2(\tau), \ldots, P_N(\tau))$$

(N = # SENSORS EMPLOYED)

- SENSORS $P_1, P_2, \ldots, P_N$ REFERRED TO BY PARAMETER IDENTIFICATION NUMBERS, OR "PIDS"

- OVER 200 PIDS AVAILABLE

- TEST FIRING DATA NOT CONSISTENT:
  FOR MOST TEST FIRINGS, SOME PIDS NOT PRESENT OR NOT VALID
  (SENSORS NOT BUILT INTO EARLY VERSIONS OF ENGINES OR FAILED SENSORS)

- CRITERIA FOR INITIAL CHOICE OF PIDS
  - SUBSET OF PIDS USED IN ROCKETDYNE'S SAFD ALOGORITHM
  - SIGNIFICANT FOR DIAGNOSIS IN ANOMALOUS FIRINGS UNDER INVESTIGATION
  - AVAILABLE FOR MOST TEST FIRINGS UNDER INVESTIGATION
    (DESIRABLE FOR GENERALIZING FROM ONE FIRING TO ANOTHER, BUT NOT ABSOLUTE REQUIREMENT - MISSING OR FAILED SENSORS MUST BE TAKEN INTO ACCOUNT ANYWAY)

- METHOD ALLOWS FOR USING MORE PIDS IN FUTURE

TWELVE PIDS USED IN CURRENT STUDY

P_{18} = MCC CLNT DS T
(MAIN COMBUSTION CHAMBER COOLANT DISCHARGE TEMPERATURE B)

P_{24} = MCC FU INJ PR
(MAIN COMBUSTION CHAMBER HOT GAS INJECTOR PRESSURE A)

P_{40} = OPOV ACT POS
(OXIDIZER-PREBURNER OXIDIZER VALVE ACTUATOR POSITION A)

P_{42} = FPOV ACT POS
(FUEL PREBURNER OXIDIZER VALVE ACTUATOR POSITION A)

P_{52} = HPFP DS PR
(HIGH PRESSURE FUEL PUMP DISCHARGE PRESSURE A)

P_{63} = MCC PC
(MAIN COMBUSTION CHAMBER PRESSURE AVERAGE)

P_{209} = LPOP DS PR
(HIGH PRESSURE OXIDIZER PUMP INLET PRESSURE A)

P_{231} = HPFT DS T1 A
(HIGH PRESSURE FUEL TURBINE DISCHARGE TEMPERATURE A)

P_{232} = HPFT DS T1 B
(HIGH PRESSURE FUEL TURBINE DISCHARGE TEMPERATURE B)

P_{233} = HPOT DS T1
(HIGH PRESSURE OXIDIZER TURBINE DISCHARGE TEMPERATURE A)

P_{234} = HPOT DS T2
(HIGH PRESSURE OXIDIZER TURBINE DISCHARGE TEMPERATURE B)

P_{261} = HPFP SPEED
(HIGH PRESSURE FUEL TURBOPUMP SHAFT SPEED)

- TEST FIRING MAY LAST OVER TEN MINUTES, SO NUMBER OF SAMPLES s MAY REACH TENS OF THOUSANDS

- VECTORS $P(\tau_i)$, $i = 0, 2, \ldots, s-1$ FORM s x N MATRIX (N = # PIDS)

- THIS POTENTIALLY HUGE MATRIX MUST BE EVALUATED QUICKLY (PREFERABLY BEFORE NEXT SAMPLE) PROVIDING STRONG MOTIVATION FOR EXTRACTING MANAGEABLE (AND CONSTANT) NUMBER OF FEATURES FROM MATRIX, USING FAST CLASSIFICATION ALGORITHMS AND MACHINERY, ESPECIALLY PARALLEL PROCESSING

- IDEALLY, SSME PERFECTLY UNDERSTOOD, HEALTH STATUS DETERMINED FROM SENSOR MEASUREMENTS BY APPLICATION OF THEORETICALLY DEDUCED RULES

- BUT SSME IS COMPLICATED, ITS BEHAVIOR NOT ENTIRELY PREDICTABLE

- MAIN RESOURCES FOR CREATING DIAGNOSTIC SYSTEM ARE
  - EXPERT KNOWLEDGE (MUCH OF THIS IN FAILURE INVESTIGATION SUMMARIES)
  - DATA ACCUMULATED FROM PREVIOUS NOMINAL & ANOMALOUS SSME FIRINGS

- USE TRAINABLE PATTERN CLASSIFICATION SOFTWARE TO LEARN TO CLASSIFY TRAINING DATA, ATTEMPT TO GENERALIZE CORRECTLY TO NOVEL DATA

- NEURAL NETWORKS OFFER
  - SPEED, ESPECIALLY IF IMPLEMENTED ON PARALLEL HARDWARE
  - AUTOMATIC LEARNING OF SUBTLE FEATURES IN LARGE QUANTITIES OF DATA
  - CAPABILITY OF GENERALIZING BASED ON PREVIOUSLY LEARNED EXAMPLES

SSME TEST FIRING DATA EMPLOYED FOR CLASSIFIER TRAINING AND TESTING

(FIRINGS CONDUCTED ON GROUND BETWEEN 1981 AND 1989)

- TWO NOMINAL FIRINGS (902-457, 902-463)
- SIX ANOMALOUS FIRINGS REPRESENTING VARIOUS FAILURE TYPES
  - (901-331) CRACKED LIQUID OXYGEN POST
  - (902-249) POWER TRANSFER FAILURE, TURBINE BLADES
  - (901-340) TURN AROUND DUCT CRACKED/TORN
  - (901-364) HOT GAS INTRUSION TO ROTOR COOLING
  - (901-436) HIGH PRESSURE FUEL TURBOPUMP COOLANT LINER BUCKLE
  - (750-259) FUEL LEAK IN MAIN COMBUSTION CHAMBER OUTLET NECK

(MORE TEST FIRINGS TO BE ADDED IN FUTURE)

FAULT-DECLARE TIMES BASED ON FAILURE INVESTIGATION REPORTS FOR EACH FIRING, PLUS AS OUR OWN ANALYSIS OF SENSOR DATA

- FAULT-DECLARE TIME IS TIME WHEN SENSORS FIRST SHOW SYMPTOMS OF FAULTY ENGINE PERFORMANCE, SO THAT AN ANOMALY DETECTION SYSTEM IDEALLY SHOULD HAVE BEEN ABLE TO INITIATE SSME SHUT-DOWN

- FOR NETWORK TRAINING, SENSOR SAMPLES TAKEN BEFORE FAULT-DECLARE TIME CONSIDERED NOMINAL DATA, SAMPLES TAKEN AFTER THAT TIME CONSIDERED ANOMALOUS DATA (HOWEVER SOME SAMPLES MAY BE LEFT OUT OF THE TRAINING SET IF IN DOUBT WHETHER TO CONSIDER ANOMALOUS)

- WHEN TESTING NETWORK PERFORMANCE, FAULT-DECLARE TIMES USED FOR COMPARISON

ATTENTION INITIALLY RESTRICTED TO PERIODS OF STEADY-STATE OPERATION

EXPLANATION FOR NON-ROCKET EXPERTS: THE SSME OPERATES AT VARIOUS POWER (THRUST) LEVELS, MEASURED BY THE MAIN COMBUSTION CHAMBER PRESSURE, P_{63}. NORMALLY A FIRING HAS A SCHEDULED SEQUENCE OF POWER LEVELS. PERIODS DURING WHICH THE POWER LEVEL IS HELD APPROXIMATELY CONSTANT ARE CALLED "STEADY-STATE", AND MAY LAST A FEW SECONDS OR A FEW MINUTES. IN BETWEEN THE STEADY-STATE PERIODS ARE INTERVALS OF THROTTLING, KNOWN AS "TRANSIENTS". TRANSIENTS USUALLY LAST ONLY A FEW SECONDS.

- MOST MAJOR FAILURES OCCURRED DURING STEADY-STATE
- TAILORING METHOD TO STEADY-STATE DATA ALLOWS USEFUL ASSUMPTIONS:
  - SENSOR VALUES NOT EXPECTED TO CHANGE SIGNIFICANTLY (ALTHOUGH IN PRACTICE THEY CHANGE CONSIDERABLY)
  - UNCHANGING VALUES CAN BE CONSIDERED NOMINAL
  - SAME CRITERIA FOR ENGINE HEALTH SHOULD APPLY REGARDLESS OF AMOUNT OF TIME ELAPSED IN STEADY-STATE PERIOD
- TRANSIENT ANOMALY DETECTION INHERENTLY MORE DIFFICULT:
  - SENSOR DATA CHANGE IN COMPLICATED WAYS
  - PATTERNS OF CHANGE MAY DEPEND ON EXACT NATURE OF TRANSIENT (START & FINISH POWER LEVELS, RATE OF THROTTLING, ETC)
- NOT APPROPRIATE TO GENERALIZE ACCROSS SAMPLES TAKEN AT DIFFERENT TIMES DURING TRANSIENTS
- IN FUTURE, MOST TECHNIQUES WE EMPLOY FOR STEADY-STATE COULD BE EXTENDED TO APPLY TO TRANSIENT ANOMALY DETECTION (RECURRENT NEURAL NETWORKS PARTICULARLY PROMISING)

- AT EACH TIME T_i, MOST RECENT SAMPLE $P(T_i)$ IS KEY DATA FOR DIAGNOSIS
- SAMPLES $P(T_j)$, $j < i$, ALSO PROVIDE IMPORTANT INFORMATION
  - FOR DETECTING SIGNIFICANT CHANGES OR RECOGNIZABLE "FAULT SIGNATURES" IN THE GRAPHS OF PID VALUES AS FUNCTIONS OF TIME
  - FOR MEASURING DURATIONS OR COUNTING REPETITIONS OF POSSIBLY ANOMALOUS CONDITIONS
  - FOR COMPUTING MOVING AVERAGES, TO SMOOTH OUT "NOISE"
  - IN ORDER TO CONSTRUCT AN ANOMALY DETECTION SYSTEM WHICH IS GENERAL ENOUGH TO WORK ON VARIOUS ENGINES AT VARIOUS POWER LEVELS, IT MAY BE DESIRABLE TO USE DATA FROM ONE TIME INTERVAL IN A GIVEN FIRING AS A POINT OF REFERENCE FOR EVALUATING DATA FROM LATER TIME INTERVALS IN THE SAME FIRING
- PRE-PROCESSING OF PID VALUES: CALCULATION OF FEATURES
  - CONSOLIDATE RAW DATA FROM HUGE s x N MATRIX

    (s = # SAMPLES $P(T_i)$, $i = 0, \ldots, s-1$)

    (N = # PIDS IN EACH SAMPLE)
  - ENCODE ESSENTIAL TIME INFORMATION
  - COMPOUND FEATURES MAY ALSO BE FORMED FROM PIDS BY CALCULATING DIFFERENCES BETWEEN PIDS, AVERAGES OF PIDS, SPECIAL FORMULAS TO COMBINE REDUNDANT PIDS, ETC
    (SOME OF THE PIDS ARE IN FACT ALREADY COMBINATIONS OF THIS TYPE, BUT WE HAVE NOT CREATED ANY NEW FEATURES IN THIS WAY)
  - SCALE AND TRANSLATE FEATURES SO
    - ALL CENTERED AROUND SAME VALUE (E.G. ZERO)
    - ALL VARY WITHIN SAME APPROXIMATE RANGE (E.G. BY SCALING ACCORDING TO STANDARD DEVIATIONS)

WE CURRENTLY CALCULATE TWO FEATURES FOR EACH PID

- RECENT CHANGE

$$\frac{Avg1(t)-Avg2(t)}{\sigma}$$

- LONG-TERM SMOOTHED CHANGE

$$\frac{Avg2(t)-Avg2(t_s)}{\sigma}$$

doesn't really take about ~

WHERE

$Avg1(t)$ = MEAN PID VALUE FOR 0.12 SECONDS (3 SAMPLES)

$Avg2(t)$ = MEAN PID VALUE FOR 2 SECONDS (50 SAMPLES)

(AVERAGES CALCULATED OVER TIME INTERVAL ENDING AT TIME t)

σ = STANDARD DEVIATION OF PID VALUE

(MEASURED OVER ALL STEADY-STATE DATA FROM ALL AVAILABLE FIRINGS)

t_s = TIME 3 SECONDS AFTER START OF CURRENT STEADY-STATE INTERVAL

- THESE FEATURES RESEMBLE CALCULATIONS USED IN ROCKETDYNE'S SAFD ALGORITHM

- RESULT OF PRE-PROCESSING IS D-DIMENSIONAL FEATURE VECTOR

$$X(\tau_i) = (X_1(\tau_i), X_2(\tau_i), \ldots, X_d(\tau_i))$$

WHICH IS FUNCTION OF PID SAMPLES $P(\tau_i)$, $i = 0, 1, \ldots, s$

- FEATURE VECTORS X HAVE FOLLOWING PROPERTY: THE ORIGIN OF D-DIMENSIONAL FEATURE SPACE

$$0 = (0, 0, \ldots, 0)$$

WHERE ALL D FEATURES ARE ZERO, IS "MOST NOMINAL" OF ALL POSSIBLE SAMPLES, SINCE IT INDICATES ALL SENSORS REMAINING AT CONSTANT LEVEL DURING STEADY-STATE OPERATION

- NON-ZERO VALUES OF FEATURES INDICATE DEVIATIONS FROM CONSTANT VALUE

- TWELVE PIDS, WITH TWO FEATURES EACH, YIELD TWENTY-FOUR INPUTS TO PATTERN CLASSIFICATION SOFTWARE

NEURAL NETWORK ARCHITECTURE:

THREE LAYER FEEDFORWARD NETWORK TRAINED BY BACKPROPAGATION

- BIOLOGICAL ANALOGY: UNIT = NEURON, CONNECTION = SYNAPSE
- LAYER OF INPUT UNITS
  (ONE FOR EACH FEATURE = 24 INPUT UNITS IN CURRENT MODEL)
- LAYER OF HIDDEN UNITS
  (8 - 12 UNITS IN A SINGLE LAYER FOUND TO BE SUFFICIENT SO FAR)
- LAYER OF OUTPUT UNITS
  (ONE FOR NOMINAL-VS-ANOMALOUS DIAGNOSIS, OTHERS FOR FAULT TYPING)
- EACH INPUT UNIT CONNECTS TO EACH HIDDEN UNIT, AND EACH HIDDEN UNIT CONNECTS TO EACH OUTPUT UNIT
- CONNECTIONS BETWEEN UNITS CHARACTERIZED BY WEIGHTS
  (CONNECTION STRENGTHS): EXCITATORY OR INHIBITORY
- CAPABLE OF PERFORMING ANY MAPPING FROM INPUTS TO OUTPUTS
- TRAINING ACCOMPLISHED BY BACKPROPAGATION OF ERROR
  (WEIGHTS CHANGED AFTER EACH TRAINING PASS ACCORDING TO GENERALIZED DELTA RULE)

- NOTE: CHOICE OF HOW MANY HIDDEN UNITS DETERMINED BY
  - NOT ENOUGH HIDDEN UNITS: IMPOSSIBLE FOR NETWORK TO PERFORM DESIRED MAPPING ON TRAINING DATA
  - TOO MANY HIDDEN UNITS: NETWORK MAY OVER-SPECIALIZE ON IDIOSYNCRACIES OF TRAINING DATA, FAILING TO FIND MORE GENERAL FEATURES DISTINGUISHING DATA CATEGORIES

NETWORK OUTPUT = CLASSIFICATION OF INPUT DATA

- SINCE FIRST PRIORITY OF DIAGNOSTIC SYSTEM IS SHUT-DOWN DECISION MAKING, ESSENTIAL CLASSIFIER OUTPUT HAS ONLY TWO VALUES:
  - ANOMALOUS (RECOMMEND SHUTTING DOWN ENGINE) OR
  - NOMINAL (RECOMMEND PROCEEDING AS USUAL)
- MORE COMPLEX FORMS OF EVALUATION MAY PROVIDE
  - DESCRIPTION OF ANOMALY, WHETHER OF KNOWN FAILURE TYPE
  - WHICH ENGINE PARTS ARE INVOLVED
  - ESTIMATE OF SEVERITY
  - SIMILARITY TO DATA FROM PREVIOUS FAILURES
  - DEGREE OF CONFIDENCE IN DIAGNOSIS
- ANOMALY DETECTION VS. FAULT TYPING
  - FAULT TYPING REQUIRED IF SHUT-DOWN PROCEDURES DEPEND ON FAILURE TYPE, OR NETWORK FORMS PART OF LARGER DIAGNOSTIC SYSTEM (WHICH CALLS FOR MORE SPECIFIC DIAGNOSIS BY NETWORK)
  - WE HAVE EXPERIMENTED WITH FAULT-TYPING, TREATING EACH ANOMALOUS TEST FIRING IN TRAINING SET AS REPRESENTING ONE FAULT TYPE
- CURRENT NETWORK CONFIGURATION HAS
  - AN OUTPUT UNIT TRAINED TO FIRE LOW IF NOMINAL AND HIGH IF ANOMALOUS
  - ADDITIONAL OUTPUT UNITS FOR EACH FAULT TYPE (I.E., ONE FOR EACH ANOMALOUS TEST FIRING IN TRAINING SET)
  - THUS WHEN TRAINING ON DATA INCLUDING FIVE ANOMALOUS FIRINGS, WE EMPLOY SIX OUTPUT UNITS IN FEEDFORWARD NETWORK

AVAILABLE NOMINAL AND ANOMALOUS DATA CURRENTLY VERY LIMITED

- ONLY A HANDFUL OF TEST FIRINGS TO USE FOR TRAINING
  (MORE NOMINAL DATA CAN EVENTUALLY BE OBTAINED FROM NASA, BUT ANOMALOUS FIRINGS ARE RARE -- FORTUNATELY!)
- EACH FIRING PROVIDES MANY DATA SAMPLES. HOWEVER SAMPLES FROM A GIVEN FIRING TEND TO LIE ON A TRAJECTORY, EACH SAMPLE BEING CLOSE TO PREVIOUS SAMPLE

- IMPOSSIBLE FOR THIS LIMITED QUANTITY OF DATA TO COME CLOSE TO SPANNING ENTIRE 24-DIMENSIONAL POTENTIAL INPUT SPACE
  (IN 24-DIMENSIONAL SPACE MOST POINTS ARE VERY FAR APART. THE NUMBER OF QUADRANTS IN 24-SPACE IS 2^{24} = 16,777,216)

- GENERALIZATION TO NEW DATA REQUIRES BOTH INTERPOLATION AND EXTRAPOLATION
  - COMPLETE DECISION BOUNDARY BETWEEN NOMINAL AND ANOMALOUS REGIONS CANNOT BE UNIQUELY DETERMINED FROM ANY FINITE AMOUNT OF TRAINING DATA
  - NETWORK MUST BE TRAINED APPROPRIATE RESPONSE TO UNPRECEDENTED INPUT DATA
  - UNLESS NEW ANOMALOUS FIRING VERY SIMILAR TO ONE OF TRAINING FIRINGS, NEW ANOMALOUS DATA WILL NOT RESEMBLE OLD ANOMALOUS DATA ANY MORE THAN IT RESEMBLES OLD NOMINAL DATA
  - NEED TO MAKE ASSUMPTIONS ABOUT SHAPE OF NOMINAL REGION TO BE MAPPED OUT BY ANOMALY DETECTION SYSTEM, IMPOSE THESE ASSUMPTIONS ON TRAINABLE CLASSIFIER
  - A BASIC ASSUMPTION WILL LEAD TO DETECTION OF NEW FAULT TYPES: ANY NEW DATA SUFFICIENTLY DIFFERENT FROM ALL PREVIOUSLY ENCOUNTERED NOMINAL DATA TO BE CONSIDERED ANOMALOUS
  - TO FORCE FEEDFORWARD NEURAL NETWORK TO CATEGORIZE NEW DATA IN ACCORDANCE THIS ASSUMPTION, IT HAS BEEN FOUND ADVANTAGEOUS TO ADD I M I T A T I O N NOMINAL AND ANOMALOUS TRAINING DATA TO TRAINING DATA FROM ACTUAL SSME FIRINGS

- GENERATE IMITATION DATA RANDOMLY DISTRIBUTED THROUGHOUT SUITABLE PART OF INPUT SPACE
  - IMITATION ANOMALOUS DATA EITHER RANDOMLY DISTRIBUTED (WHICH PLACES IT GENERALLY FAR OUT IN INPUT SPACE) OR WITH VALUES OF SOME COMPONENTS NEAR KNOWN FAULT READINGS)
  - IMITATION NOMINAL DATA WITHIN EXPECTED RANGES OF NOMINAL FEATURES (CURRENTLY LIMITED EXPERIENCE WITH ADDING GENERATED NOMINAL DATA)
- COMBINE RANDOM DATA WITH GENUINE NOMINAL AND ANOMALOUS DATA FOR TRAINING
- TRAIN NETWORK TO CATEGORIZE GENERATED ANOMALOUS DATA AS ANOMALOUS, GENERATED NOMINAL DATA AS NOMINAL
  - TASK OF RECOGNIZING GENERATED DATA FORCES NETWORK TO LEARN BOUNDARIES OF EXPECTED NOMINAL REGION

SOME FINDINGS AT INTERMEDIATE STAGE IN OUR RESEARCH

- NEURAL NETWORK CLASSIFIER IS ALWAYS CAPABLE OF LEARNING TRAINING DATA WITH VIRTUALLY 100% ACCURACY, OUTPUTTING "NOMINAL" WHEN FED NOMINAL DATA, AND "ANOMALOUS" WHEN FED ANOMALOUS DATA

- GENERALIZING TO NEW (UN-TRAINED) ANOMALOUS FIRINGS HAS BEEN SYSTEMATICALLY UNDERTAKEN ACCORDING TO SINGLE HOLD-OUT PRINCIPLE:

  - TRAIN NETWORK ON ALL TRAINING DATA (TO INCLUDE GENUINE DATA FROM NOMINAL AND ANOMALOUS TEST FIRINGS AS WELL AS SOME IMITATION ANOMALOUS DATA), EXCEPT FOR DATA FROM ONE TEST FIRING DELIBERATELY WITHELD

  - TEST SAME NETWORK ON DATA FROM FIRING WHICH WAS WITHELD FROM TRAINING

- NETWORK HAS DEMONSTRATED ABILITY TO CORRECTLY CLASSIFY THIS DATA THAT IS NEW TO IT AS NOMINAL UP UNTIL FAULT-DECLARE TIME, AND ANOMALOUS THEREAFTER

- POSITIVE RESULT OF GENERALIZATION IS CONTINGENT ON TRAINING WITH RANDOM IMITATION ANOMALOUS DATA (OTHERWISE NEW DATA IS ALWAYS CLASSIFIED AS NOMINAL)

- FAULT-TYPING (ACTIVATIONS OF ADDITIONAL OUTPUT UNITS) IS LEARNED CORRECTLY FOR TRAINING DATA, BUT NEW DATA IS NEVER CLASSIFIED AS BELONGING TO ANY PREVIOUS FAULT-TYPE

- NOW WHEN GENERALIZATION IS NOT SUCCESSFUL, CHIEF PROBLEM IS FALSE ALARMS (CLASSIFICATION OF NEW NOMINAL DATA AS ANOMALOUS)

- AN APPROACH HAS BEEN FOUND FOR RECOGNIZING WHEN A FALSE-ALARM IS DEPENDENT ON FEATURES CORRESPONDING TO SINGLE PID, AND IMMEDIATELY DETERMINING WHICH PID IS RESPONSIBLE:

  - MULTIPLE COPIES (ONE FOR EACH PID) OF EACH FEATURE VECTOR ARE SEPERATELY FED THROUGH NETWORK

  - EACH COPY IS ALTERED BY HAVING FEATURES CORRESPONDING TO ONE OF PIDS REPLACED WITH ZEROS (REMEMBER THAT FOR CURRENT FEATURES, ZERO MEANS NO-CHANGE, AND NON-ZERO INDICATES DEVIATION FROM CONSTANT STEADY-STATE VALUE)

  - NETWORK OUTPUTS FOR EACH COPY SHOW WHAT CLASSIFICATIONS WOULD BE IF EACH PID IN TURN INDICATED NO CHANGE

  - ZEROING OUT PID RESPONSIBLE FOR FALSE ALARM RESULTS IN CORRECT CLASSIFICATION AS NOMINAL UP UNTIL FAULT-DECLARE TIME, AND ANOMALOUS THEREAFTER

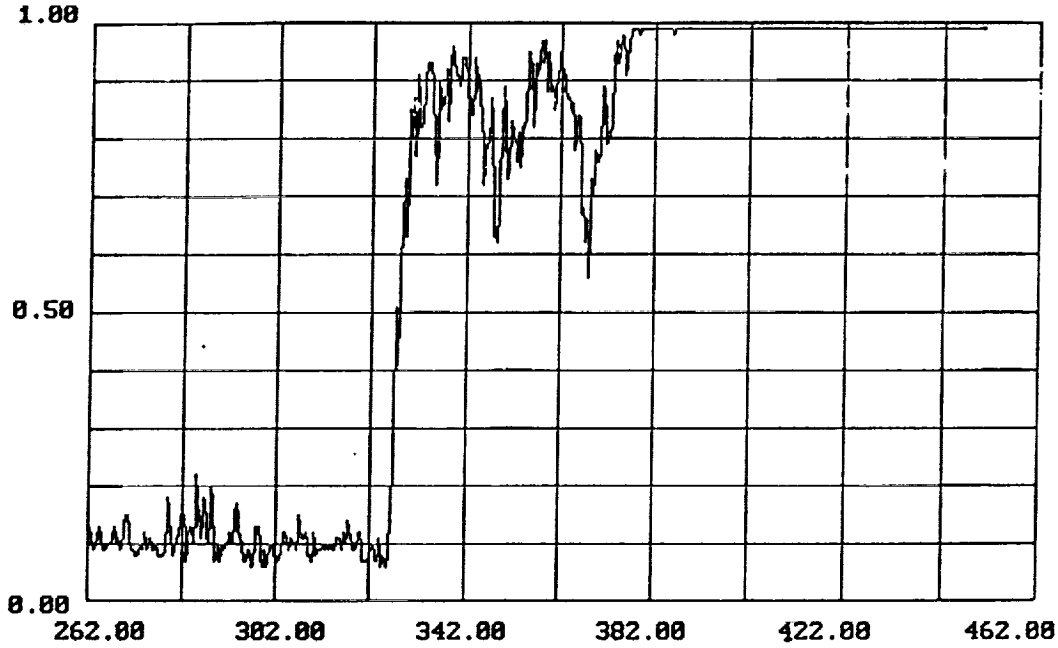
  - SUCH RESULTS SUGGEST POSSIBILITY OF INCORPORATING VOTING SCHEME INTO MAKING CLASSIFIER OUTPUT MORE ROBUST WITH RESPECT TO FALSE ALARMS CAUSED BY ANY SINGLE FEATURE, IF IT IS FOUND APPROPRIATE TO REQUIRE MORE THAN ONE PID TO MANIFEST SYMPTOMS BEFORE MAKING AN ANOMALOUS CLASSIFICATION, OR SIMPLY AS AID TO ISOLATING POSSIBLE SENSOR FAILURES

WORK IN PROGRESS

- FURTHER TRAINING AND TESTING OF FEEDFORWARD NEURAL NETWORKS, EMPLOYING SEVERAL NEW KINDS OF SIMULATED OR MODIFIED SUPPLEMENTARY TRAINING DATA:
  - GENERATE SIMULATED / MODIFIED DATA DYNAMICALLY DURING TRAINING, RATHER THAN PUTTING INTO TRAINING DATA FILE AND USING REPEATEDLY (MUCH MORE EVEN COVERAGE OF FEATURE SPACE)
  - RESTRICT RANDOM SIMULATED ANOMALOUS DATA TO STAY OUTSIDE OF REGIONS ASSUMED TO BE NOMINAL (REQUIRE MINIMUM LENGTH FOR ANOMALOUS FEATURE VECTORS, ETC --- MAY DECREASE FALSE-ALARMS)
  - USE RANDOMLY GENERATED NOMINAL DATA CLOSE TO ORIGIN (JUSTIFICATION: NO GENUINE ANOMALOUS FEATURE VECTORS HAVE BEEN OBSERVED WITHIN A CERTAIN RADIUS OF ORIGIN, BUT FALSE ALARMS HAVE OCCURRED THERE)
  - MODIFY GENUINE NOMINAL FEATURE VECTORS BY REPLACING SOME COMPONENTS WITH ZERO VALUES (TO PREVENT FALSE ALARMS DUE TO MISSING SENSORS, AND TO FILL OUT NOMINAL REGION IN ACCORDANCE WITH ASSUMPTION THAT IN STEADY-STATE CONTEXT, UNCHANGING SENSOR VALUE SHOULD NOT CAUSE FEATURE VECTOR TO BE REGARDED AS ANOMALOUS)
  - MODIFY GENUINE ANOMALOUS FEATURE VECTORS IN SAME WAY (TO MAKE ANOMALY DETECTION MORE ROBUST, NOT DEPENDENT ON ANY SINGLE PID, TO GUARANTEE DETECTION EVEN USING TESTING METHOD SUGGESTED ABOVE IN WHICH APPARENT ANOMALY DUE TO ONLY ONE PID MAY NOT BE ENOUGH TO WARRANT ENGINE SHUT-DOWN)

- EXPERIMENTING WITH VARIATIONS IN TRAINING TECHNIQUE AND NETWORK ARCHITECTURE, ESPECIALLY RECURRENT NETWORKS:
  - RECURRENT NETWORKS DESIGNED TO CLASSIFY TIME SERIES DATA
  - ACTIVATIONS OF HIDDEN UNITS FEED BACK TO RETAIN MEMORY FOR CLASSIFYING SUBSEQUENT INPUTS IN TIME SERIES CONTEXT
  - AUTOMATICALLY LEARNED INTERNAL FEATURES OF RECURRENT NETS MAY BE USEFUL ADDITION OR ALTERNATIVE TO OUR EXPLICITLY COMPUTED CHANGE-MEASURING FEATURES

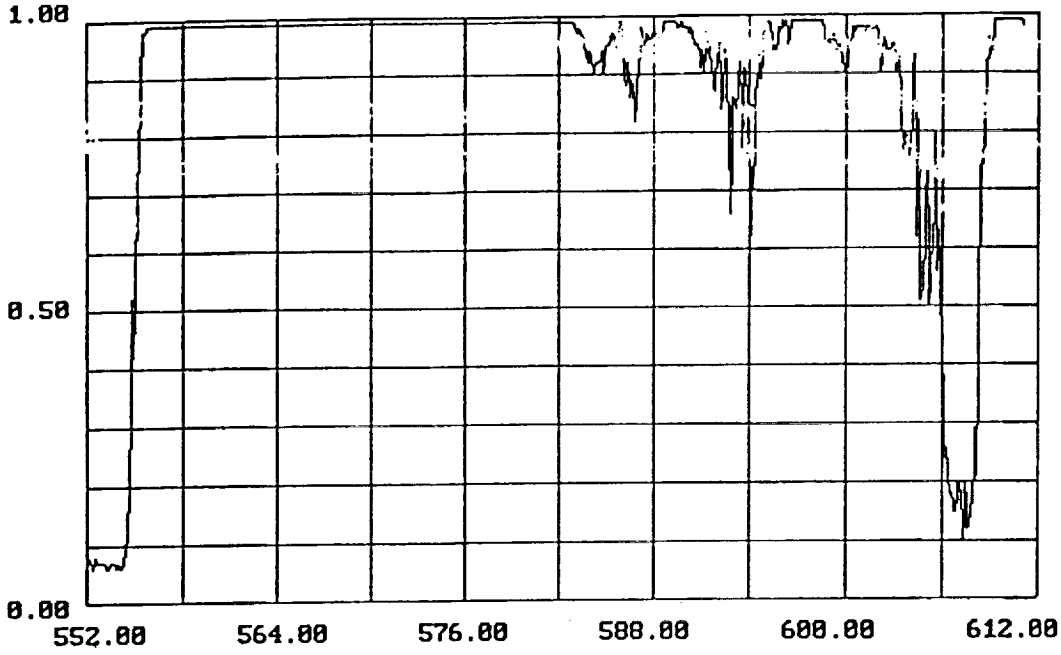
- USING SOME GEOMETRICAL PERSPECTIVES ON THE PROBLEM, EXPERIMENTING WITH PLAUSIBLE ALTERNATIVE METHODS FOR EXTRAPOLATING FROM TRAINING DATA TO DETERMINE BOUNDARIES OF NOMINAL REGION IN 24-DIMENSIONAL VECTOR SPACE:
  - LENGTHS OF FEATURE VECTORS (I.E. DISTANCE FROM ORIGIN) FOUND TO BE GOOD INDICATORS OF TRANSITIONS FROM NOMINAL TO ANOMALOUS DATA
  - NOMINAL REGION COULD BE CHARACTERIZED BY ESTABLISHING MAXIMUM LENGTH FOR NOMINAL FEATURE VECTORS IN ANY GIVEN DIRECTION
  - DETERMINE THESE MAXIMUM LENGTHS FOR TRAINING DATA, GENERALIZE TO NOVEL DATA BY VARIATION ON NEAREST NEIGHBOR PRINCIPLE, DEFINING NEARNESS ACCORDING TO ANGLES BETWEEN VECTORS
  - INITIAL IMPLEMENTION OF THIS APPROACH USES SEQUENTIAL ALGORITHMS, COULD BE IMPLEMENTED IN PARALLEL (ALONG SIMILAR LINES AS THE PROBABILISTIC NEURAL NETWORK, WHICH ALSO RESEMBLES NEAREST NEIGHBOR CLASSIFIER)



Graph of Neural Network Output for Novel Data

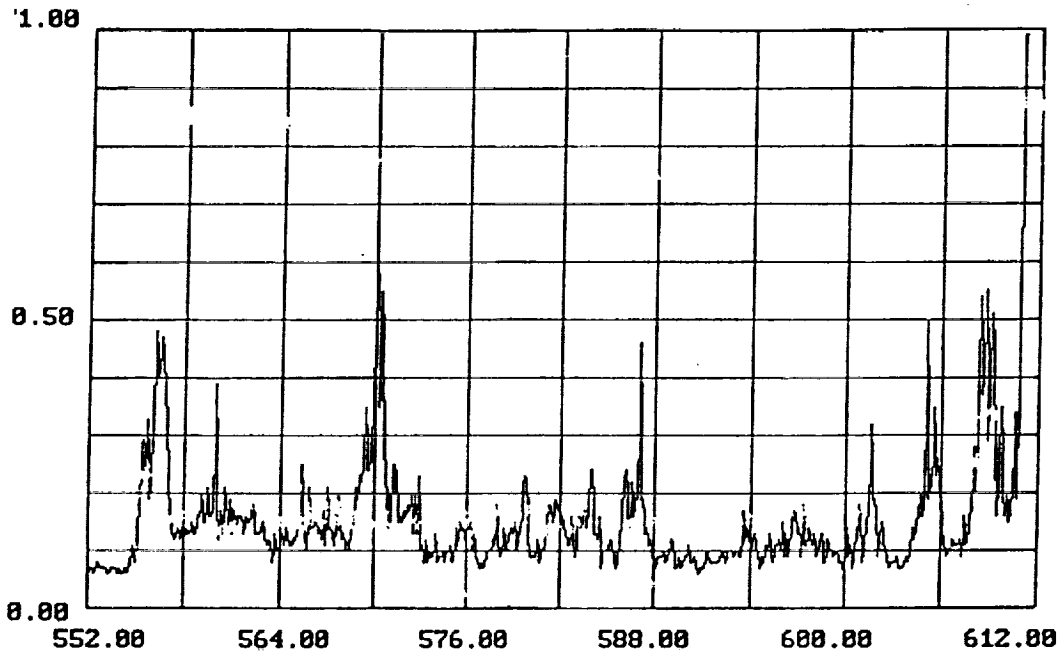
A neural network was trained on data from all test firings except 901-249, plus randomly generated anomalous data. The graph shows the activation of the nominal-versus-anomalous output unit when the network was tested on firing 901-249.

The network clearly begins to detect an anomaly around 328 seconds, a few seconds after symptoms began to occur according to Failure Investigation Summary. The SSME was not actually shut down until 450.58 seconds, after massive damage had occurred.



Example of a "False-Alarm" in Generalization to Novel Data

Network was trained by holding out only the anomalous firing 901-436, and tested on that firing. The actual fault did not occur until 610 seconds, and early warning as early as shown on this graph does not appear to be realistic. Therefore this must be regarded as a false alarm.



Result of zeroing out features for PID 24
in the same "False Alarm" case

The time at which the graph of the "deviant" output unit finally goes above .6 is now precisely the fault-declare time determined by analysis for the novel anomalous firing 901-436. (PID 24, and the two features calculated were in fact out of range for the training firings.)

N91-20817

NEUROCONTROL AND FUZZY LOGIC: CONNECTIONS AND DESIGNS

by Paul J. Werbos
Room 1151, National Science Foundation
Washington, D.C. 20550
(202)-357-9618

ABSTRACT

Artificial neural networks (ANNs) and fuzzy logic are complementary technologies. ANNs extract information from systems to be learned or controlled, while fuzzy techniques mainly use verbal information from experts. Ideally, both sources of information should be combined. For example, one can learn rules in a hybrid fashion, and then calibrate them for better whole-system performance. ANNs offer universal approximation theorems, pedagogical advantages, very high-throughput hardware, and links to neurophysiology. Neurocontrol -- the use of ANNs to directly control motors or actuators, etc. -- uses five generalized designs, related to control theory, which can work on fuzzy logic systems as well as ANNs. These designs can: copy what experts do instead of what they say; learn to track trajectories; generalize adaptive control; maximize performance or minimize cost over time, even in noisy environments. Design tradeoffs and future directions are discussed throughout.

It will begin by discussing the simpler, more common application of ANNs -- to learning a mapping from a vector $\underline{X}$ to a vector $\underline{Y}$. Then it will discuss neurocontrol, and the central importance of neurocontrol to understanding intelligence.

This paper will mainly discuss neurocontrol -- the use of neural networks (artificial or neural) to directly control motors, actuators, muscles or other kinds of overt physical action. It will also discuss the relation between artificial neural networks (ANNs) and fuzzy logic, and how best to combine them.

ANNs and Fuzzy Logic in General

Neurocontrol is still a small part of the greater neural network community. Most people use ANNs for applications like pattern recognition, diagnostics, risk analysis, and so on. They mostly use ANNs to learn static mappings from an "input vector," $\underline{X}$, to a "target vector," $\underline{Y}$. For example, $\underline{X}$ might represent the pixels which make up an image, while $\underline{Y}$ might represent a classification of that vector. Given a training set made up of pairs of $\underline{X}$ and $\underline{Y}$, the network can "learn" the mapping, by adjusting its weights so as to perform well on the training set.

This kind of learning is called "supervised learning." There are many forms of supervised learning used by different researchers, but the most popular is basic backpropagation[1]. Basic backpropagation is simply a unique implementation of least squares estimation. In basic backpropagation, one uses a special, efficient technique to calculate the derivatives of square error with respect to all the weights or parameters in an ANN; then, one adjusts the weights in proportion to these derivatives, iteratively, until the derivatives go to zero. The components of $\underline{X}$ and $\underline{Y}$ may be 1's and 0's, or they may be continuous variables in some finite range.

Fuzzy logic is also used, at times, to infer well-defined mappings. For example, if $\underline{X}$ is a set of data characterizing the state of a factory, and $\underline{Y}$ represents the presence or absence of various breakdowns in the factory, then fuzzy rules and fuzzy inference may be used to decide on the likelihood that one of the breakdowns may be present, as a function of $\underline{X}$.

Which method is better to use, when?

The simplest answer to this question is as follows: since ANNs extract knowledge from training databases, and fuzzy logic extracts rules from human experts, we should simply decide who we trust more, in the particular application. (When in doubt, we can try both and try for an evaluation after the fact.) In principle, empirical data represents the real bottom line while expert judgment is only a secondary source; however, when the empirical data is too limited to allow us to learn complex relations, expert judgment may be all we have.

In many applications, there are some parts of the problem for which we have adequate data, and others for which we do not. In that case, the practical approach is to divide the problem up, and use ANNs for part and fuzzy logic for another part. For example, there may be an intermediate proposition R which has an important influence on Y; we may build a neural net to map from X to R, and a fuzzy logic system to map X and R into Y, or vice-versa. Amano et al[2], for example, have built a speech recognition system in which ANNs detect the features, and a fuzzy logic system goes on to perform the classification. Many people building diagnostic systems have taken similar approaches[3].

In the current literature, many people are using fuzzy logic as a kind of organizing framework, to help them subdivide a mapping from X to Y into simpler partial mappings. Each one of the simple mappings is associated with a fuzzy "rule" or "membership function." ANNs or neural network learning rules are used to actually learn all of these mappings. There are a large number of papers on this approach, reviewed in [4]. Because these are typically very simple mappings -- with only one or two layers of neurons -- we can choose from a wide variety of neural network methods to learn the mappings; however, since the ANNs only minimize error in learning the individual rules, there is no guarantee that they will minimize error in making the overall inference from X to Y. This approach also requires the availability of data in the training set for all of the intermediate variables (little R) used in the partial mappings. Strictly speaking, this approach is a

special case of the previous paragraph; in the general case, some rules can be learned while others come from experts.

Many people in fuzzy logic might say that fuzzy logic is more than just rules and inference. There is also such a thing as fuzzy learning. In fact, much of the neural network literature on learning (like backpropagation[1]) applies directly to any well-behaved nonlinear network. It can be applied directly to the inference structures used in fuzzy logic. We could easily get into a situation where fuzzy logic people and neural network people use the exact same mathematical recipe for how to adapt a particular network, and use different names for the same thing. Personally, I would prefer to focus on the generalized mathematical learning rules, so that we can speak a more universal language, and avoid distinctions without a difference.

There are some problems which cannot be easily subdivided into expert-based parts and learning-based parts. For example, there are theories of international conflict which involve a rich structure, containing a large number of parameters known with varying degrees of confidence; it is important to expose the entire structure to the discipline of historical testing ("backcasting" and "calibration"). In situations like that, the best procedure is to combine fuzzy logic and learning. (In Bayesian terms, one would regard this as a convolution of prior and posterior knowledge, to determine the correct conditional probabilities, conditional upon all available information.) For example, we can use fuzzy logic and interviews with experts to derive an initial structure, and estimates of uncertainty. Then, one can use generalized backpropagation directly to adjust the weights (or uncertainty levels or other parameters) in that network. We can even use backpropagation to minimize an error measure like:

$$E = \sum_i (Y_i - \hat{Y}_i)^2 + \sum_j C_j (W_j - W_j^{(0)})^2 , \qquad (1)$$

where C_j is the prior degree of certainty about parameter W_j, and $W_j^{(0)}$ is the prior estimate of the

parameter. This kind of convolution approach could also be applied, of course, to the learning of independent rules or membership functions, as described in [4]. In a recent meeting to discuss long-term strategic planning issues, I suggested a two-stage approach: (1) build up an initial inference system or model using conventional techniques, which adapt individual rules or equations; (2) then -- after assessing degrees of certainty -- adjust all of the weights in a "calibration" phase, using backpropagation to make sure that the overall structure adequately fits the overall structure in historical data.

So far as I know, the idea of applying backpropagation to a fuzzy logic network was first published in 1988[5]. Matsuba of Hitachi, in unpublished work, first proposed the use of equation 1. Backpropagation is important in this application, because it can adapt multilayer structures.

Backpropagation cannot be used to adapt the weights in a more conventional, Boolean-logic network. However, since fuzzy logic rules are differentiable, fuzzy logic and backpropagation are more compatible. Strictly speaking, it is not necessary that a function be everywhere differentiable to use backpropagation; it is enough that it be continuous and be differentiable almost everywhere. Still, one might expect better results from using backpropagation with modified fuzzy logics, which avoid rigid sharp corners like those of the minimization operator.

One reason for liking fuzzy logic, after all, is that it can do a better job than Boolean logic in representing what actually exists in the mind of a human expert. This being so, modified fuzzy logics -- which are even smoother -- may be even better. Fu[6] has gotten good results applying backpropagation ~~fuzzy logic~~ to simple fuzzy logic structures (using special rules to handle the corner points), while Hsu et al[7] have proposed a modified logic. Presumably the fuzzy logic literature itself includes many examples of smooth, modified fuzzy logics. Among the obvious possibilities are: (1) to use simple ANNs themselves in knowledge representation; (2) to use functional forms similar to those used by economists, in production functions and cost functions, with parameters to reflect the

importance, the complementarity and the substitutability of different inputs.

Fuzzy logic has the advantage that it can be applied in a flexible way, using a different inference structure for each case in the training set. This inference structure may contain logic loops, which go beyond the capability of what ANN people call "feedforward" networks. The inference structure may be a "simultaneously recurrent" network. Nevertheless, backpropagation can be used on such inference structures (using the memory-saving methods in [8]) to calculate the derivatives of error with respect to every parameter, at a cost less than the cost of invoking the inference structure a single time. Thus one can use backpropagation here as well. Hybrid systems like this may be too expensive to justify for unique applications, but they make considerable sense in generalized software systems.

When complex inference is required, in fuzzy logic as in conventional logic, the design of an inference engine can be very tricky. Neurocontrol systems may be used, in essence, as inference engines. In fact, I would argue that this is precisely how the human brain does inference -- that the true "deep structure" of language is a collection of neural nets which learn, through experience, how to perform more and more effective inference (in a nonBoolean environment). Inference may be more difficult than other forms of control problem; however, there are parallels between neurocontrol systems and existing inference engines which suggest some real possibilities here.

Stinchcombe and White have proven (IJCNN 1989) that conventional ANNs can represent essentially any well-behaved nonlinear mapping. However, in applications of ANNs, many researchers have begun to encounter the limitations of any static mapping. In recognizing dynamic patterns[1], like speech or moving targets, or in real-world diagnostics[9], it is often necessary to add memory of the past. As one adds such memory, it becomes more and more important to build up robust dynamic models of the system to be analyzed or controlled. Neural networks can do this[10], in part by adapting intermediate features and developing representations which an expert might not

have thought of.

Neurocontrol in General

In 1988, neurocontrol was just beginning a major period of growth. At that time, NSF sponsored a workshop on neurocontrol at the University of New Hampshire, chaired by W. Thomas Miller[11], who brought together a small, mixed group of neural network people, control theorists and experts in substantive application areas. In the very early part of that workshop, a few people echoed the old arguments about who is better -- control theorists or neural networkers. Within a very short time, however, it became apparent that this issue was utterly meaningless. It was meaningless because it revolved about a distinction without a difference. The reason for this is illustrated in Figure 1.

****

INSERT FIGURE 1 (VENN DIAGRAM)

****

Figure 1 is a Venn diagram, telling us that neurocontrol is a subset both of neural network research and of control theory. In the course of the workshop, it became apparent that the existing work in neurocontrol could be reduced to five fundamental design strategies, each of which occurred over and over again, with variations, in numerous papers. (Individual papers tend to highlight their unique aspects, of course.) All five turned out to be generic approaches which could be applied to any large, sparse network of differentiable functions or to an even larger class of networks. One may call these "functional networks," as opposed to neural networks. All five methods could be fully understood as generic methods within control theory. By remembering that neurocontrol is a subset of both disciplines, we are in a position to draw upon both disciplines in developing more advanced designs and applications.

This situation is particularly important to fuzzy logicians, because the inference structures of fuzzy logic are themselves functional networks. In this paper, I will present numerous boxes labelled as "neural networks," but every such box could just as easily be filled in with a fuzzy inference structure varying over time. In other words, every one of the five "neurocontrol" methods can also be applied directly to fuzzy learning as well. In practice, one would often want to fill in different boxes with different things -- perhaps an ANN for one box, a hybrid neural/fuzzy map (as described in the previous section) for another, and a conventional fixed algorithm for a third. This kind of mixing and matching is quite straightforward, once one understands the basic principles.

Why should we be interested at all in the special case where the functional network is built up from the traditional kinds of artificial neurons? Why should we be interested in functional forms close to the conventional form used in ANNs[1]:

$$x_i \;=\; s\Big(\sum W_{ij} x_j\Big)\,, \qquad (2)$$

where:

$$s(z) \;=\; \frac{1}{1+e^{-z}} \quad ? \qquad (3)$$

(Here, x_i represents the "output" or "activation" of a model neuron, while W_{ij} represents a "weight" or "parameter" or "connection strength" or "synapse strength.")

There are at least four reasons for paying attention to the special case represented by neural networks: (1) the universal mapping theorems of White and Gallant and others; (2) the availability of special purpose computer hardware; (3) the pedagogical value of the special case; and (4) the link to the brain.

The theorems of White and others have excited great interest in the control community, because they show that conventional ANNs do something very similar to what Taylor series do -- provide a basis for approximating an arbitrary nonlinear function. As with Taylor series, the nonlinearity

is very simple, offering a hope of workable practical tools.

The availability of special purpose computer hardware is a decisive factor in favor of ANNs. There are many cases where a task can be done equally well using conventional sequential methods or neural nets, and where both approaches involve a similar degree of computational complexity. (For example, there are cases where an ANN can simply be trained to mimic the input-output behavior of an existing algorithm.) In such cases, ANNs may have a decisive advantage in real-world implementation, because of the hardware.

Intel, for example, recently produced a neural net chip -- now publicly available -- under encouragement from the U.S. Navy at China Lake (with some NSF support acknowledged in the documentation). David Andes of China Lake has stated that one handful of these chips has more computational power than all of the Crays in the world put together. This is critical in applications where it is acceptable to add on a few extra chips, but not to haul along a Cray. Other companies -- such as Syntonics in the U.S. and Oxford Computing in England -- have also come up with impressive chips. Users without the technical knowledge (or clients) to wire up chips have reported that the neural board by Vision Harvest, Inc. (which includes a special-purpose chip) offers some of the same advantages. More and more products of this sort may be expected, especially if the optical approach reaches maturity.

Fuzzy logic chips have also been developed. However, because of the complexity of fuzzy logic, as normally practiced, these chips cannot take advantage of parallel distributed architecture as much as neural chips do. At the recent conference in Houston on neural nets and fuzzy logic, the Japanese developer of one of the leading fuzzy chips stated unequivocally that one could expect far more computational throughput from a neural chip than from a fuzzy chip.

Harold Szu of the Naval Research Laboratories has often argued that digital parallel computers constitute the real "fifth generation" of computers, as far beyond current PCs as the PCs are beyond

the old LSI mainframes. In a similar vein, he argues that fixed-function, analog distributed hardware -- either VLSI or optical -- represents a sixth generation. The NSF program in neuroengineering got its start when people like Carver Mead[12] -- often viewed as the father of all VLSI -- and people like Psaltis and Farhat and Caulfield (famous in optical computing) argued that this sixth generation could achieve a thousand-fold or million-fold improvement in throughput over even the fifth generation. The challenge was to find a way to use this hardware in a truly general-purpose way. That is the goal which led to the neuroengineering program at NSF. Some engineers would simply define an ANN as a general-purpose system capable (in principle) of efficient implementation in such hardware.

A third reason for being interested in neural networks as such is their pedagogical value. The importance of this should not be underestimated. For example, when I first published backpropagation as a generalized method for use with any functional network, it received relatively little attention, in part because the mathematics were unfamiliar and difficult. Later, when several authors (including myself) presented it as a method for use with simplified ANNs -- with interesting interpretations, with nice flow charts using circles and lines, and with easy-to-use software packages (exploiting the simplicity which comes from giving the user no choice of functional form) -- the method became much better known[13]. Even now, for many people, it is easier to learn how to use a new design in the ANN special case, and then generalize this knowledge, than it is to start with the purest, most general mathematics. The explosion of interest in neural networks has also been very useful in motivating a new generation of graduate students, with diverse backgrounds, to learn the relevant mathematics. The effort to attract graduate students from diverse and nontraditional backgrounds -- especially women and minorities -- is now a major national priority, because of the changing composition of the young adult population in the United States.

A fourth reason for being interested in neurocontrol is the desire to be explicit about the link

to the human brain. This link can be useful in both directions -- from engineering to biology, and from biology to engineering.

The output of the human brain as a whole system is the control over muscles (and other actuators), as illustrated in Figure 2. Therefore the function of the brain as a whole system is

****

INSERT FIGURE 2 (BRAIN)

****

control, over time, so as to influence the physical environment in a desired direction. Control is not part of what goes on in the brain; it is the function of the whole system. Even though lots of pattern recognition and reasoning and so on occur within the brain, they are best understood as subsystems or phenomena within a neurocontroller. To understand the subsystems and phenomena, it is most important to understand their function within the larger system. In short, a better understanding of neurocontrol will be crucial, in the long-term, to a real understanding of what happens in the brain. (For a more concrete discussion of this, see [10].) Because the mathematics involved are general mathematics, they should be applicable to chips, to neurons, and to any other substrate we are capable of imagining to sustain intelligence.

The brain is living proof that it is possible to build an analog, distributed controller which is capable of effective planning (long-term optimization) under conditions of noise, qualitative uncertainty, nonlinearity, and millions of variables to be controlled at once, all with a very low incidence of falling down or instability. Control at such a high level necessarily includes pattern recognition and systems identification as subsystems. Table 1 compares the five major design strategies now used in neurocontrol against the four most challenging capabilities of the brain of engineering importance.

****

INSERT TABLE 1 (Matrix of capabilities versus methods)

****

Table 1 was developed two years ago[11], but it still applies to all the recent research which I am aware of (except that a very few clever researchers like Narendra have developed interesting ways to combine some of these approaches). Supervised control is the strategy of building a neural network which imitates a pre-existing control system; this is like expert systems, except that we copy what a person says instead of what he does, and can operate at higher speed. Direct inverse control builds neural nets which can follow a trajectory specified by a user or a higher-level system. Neural adaptive control does what conventional adaptive control does, but it uses neural networks for the sake of nonlinearity and robustness; for example, an ANN may learn how to track an external Reference Model (as in conventional MRAC design). Backpropagating utility and adaptive critics are two techniques for optimal control over time -- to maximize utility or performance, or to minimize cost, over time. All five will be discussed in more detail in later sections.

Table 1 does suggest that we are now on a well-defined path to duplicating the most important capabilities of the human brain. However, the human brain is more than just a set of cells and learning rules. It is also a very large mass of cells. For the next few years, it may be better to think of ANNs as artificial mice (at best) rather than artificial humans. Mice are magnificent at some very difficult control and even planning tasks, but they are not very good at calculus (or is it that they don't pay attention?). Artificial humans are certainly possible, in my view, but there are many reasons to move ahead one step at a time. Personally, I find myself most interested in the last group of methods, because of its importance to understanding true intelligence; however, there are many engineering applications where it pays to use a simpler approach, and the brain itself may be a hybrid of many approaches.

Areas of Application

Four major areas have been discussed at length [10,11] for possible applications of neurocontrol:

- o Vehicles and structures
- o Robots and manufacturing (especially of chemicals)
- o Teleoperation and aid to the disabled
- o Communications, computation and general-purpose modeling (e.g. economics)

This paper cannot describe all these areas in depth, but a few words may be in order.

In vehicles and structures, the aerospace industry has been a leader in applying these concepts. Unfortunately, the most exciting applications remain proprietary. NSF has been mainly interested in sponsoring high-risk applications which in turn serve as risk-reducers in high-risk projects of economic importance. Risk reduction comes from providing an alternative, back-up approach to solving very difficult problems which conventional techniques may or may not be adequate to solve. The National Aerospace Plane is a prime example. The goal is not to replace humans in space, but to improve the economics required to make the human settlement of space a realistic possibility. In October of 1990, NSF and McDonnell-Douglas are planning to jointly sponsor a technical workshop on Aerospace Applications of Neurocontrol, which will hopefully serve to advance this area. Barhen of the Jet Propulsion Laboratory has discussed a possible $15 million per year initiative on neural networks from NASA, with a control component. Ideally, there should be NSF/NASA cooperation here, so as to stimulate the development and testing of the most advanced forms of neurocontrol.

The chemical industry has also been quite active. Major sessions have been held at the American Control Conference and at the annual meetings of the chemical societies on this topic. The Chemical Reaction Processes program at NSF is also planning a workshop in October, focusing on neurocontrol, and laying the groundwork for expanded activity. The Bioengineering and Aid to

the Disabled program has recently held a broad workshop, to prepare for its approved initiative in this general area.

All of these new activities were motivated by interests expressed in the engineering community itself. There are many cases where industry or industry-oriented researchers are coping with fundamental issues which mainstream academics are barely beginning to address.

Supervised Control and Conventional Fuzzy Control

In the usual expert systems approach, a control strategy is developed by asking a human expert how to control something. Supervised control is essentially the ANN equivalent of that approach.

In supervised control, the first task is to build up a training set -- a database -- which consists of sensor inputs ($\underline{X}$) and desired actions ($\underline{u}$). Once this training set is available, there are many neural network designs and learning rules (like basic backpropagation) which can learn the mapping from $\underline{X}$ to $\underline{u}$.

Usually, the training set is built up by asking a human expert to perform the desired task, and recording what the human sees ($\underline{X}$) and what the human does ($\underline{u}$). There are many variations of this, of course, depending on the task to be performed. (Sometimes the input to the human, $\underline{X}$, comes from electronic sensors, which are easily monitored; at other times, it may be necessary to develop an instrumented version of the task, using teleoperation technology, as a prelude to building the database.) The goal is essentially to "clone" a human expert.

Supervised control has two other applications besides cloning a human expert. First, it can generate a controller which is faster than the expert. For example, a human might be asked to fly a slowed-down simulated version of a new aircraft. The ANN could then be implemented on a neural net chip, which allows it to operate at a higher speed -- higher than what a human could keep up with. Second, it can be used to create a compact, fast version of an existing automated

controller, developed from expert systems or control theory, which was too expensive or too slow to use in real-time, on-board applications. Supervised control is similar, in a way, to the old "pendant" system used to train robots; however, unlike the pendant system, it learns how to respond to different situations, based on different sensor input.

When should we use supervised control, with ANNs (or other networks), and when should we use fuzzy knowledge-based control?

Knowledge-based control is like following what a person says, while supervised control is like copying what the person does. Parents of small children may remember the famous plea: "Do what I say, not what I do." Knowledge-based systems obey this injunction. Supervised controllers do not.

There are many tasks where it is not good enough to ask people what they do, and follow those rules. For example, if someone asked you how to ride a bicycle, and coded those rules up into a fuzzy controller, the controller would probably fall down a lot. Your system would be like a child, who just started riding a bicycle, based on rules he learned from his mother. The problem is that your knowledge of how to ride a bicycle is stored "in your wrists," in your cerebellum and in other parts of your brain which you can't download directly into words. A supervised controller can imitate what you do, and thereby achieve a more mature, complete and stable level of performance. (This may be one reason why children have evolved to be so imitative, whether their parents like it or not.) Other forms of ANN control can go further, and learn to do better than the human expert; however, it may be best to initialize them by copying the human expert, as a starting point, in applications where one can afford to do so.

The example here does not tell us that neurocontrol should be preferred over fuzzy logic in all cases. As with the problem of learning a mapping, discussed above, the theoretical optimum is to combine knowledge-based approaches and ANN approaches. As a practical matter, the theoretical optimum is often unnecessary and too expensive to implement. However, there are tasks which are

too difficult to do in any other way.

As an example, consider the problem of learning how to do touch-typing. Even a human being cannot learn to do touch-typing simply by hunting and pecking, and gradually increasing speed. In a technical sense, we would say that the problem of touch-typing is fraught with "local minima," such that even the very best neural network -- the human brain -- can get stuck in a suboptimal pattern of behavior. To learn touch typing, one begins with a teacher, who explicitly conveys rules using words. Then one fine-tunes the behavior, using neural learning. Then one learns additional rules. Only after one has initialized the system properly -- by learning all the rules -- can one rely solely on practice to improve the skill. Morita et al[14] have shown how a two-stage approach -- knowledge-based control followed by backpropagation-based learning -- can improve performance, in certain supervised control problems. There are other ways to deal with local minima, but they complement the use of symbolic reasoning, rather than compete with it.

Advanced practitioners of supervised control no longer think of it as a simple matter of mapping $\underline{X}(t)$, at time t, onto $\underline{u}(t)$. Instead, they use past information as well to predict $\underline{u}(t)$. They think of supervised control as an exercise in "modeling the human operator." The best way to do this is by using neural nets designed for robust modeling, or "system identification," over time. There is a hierarchy of such ANN designs, the most robust of which has yet to be applied to supervised control[10].

Supervised control with an ANN was first performed by Widrow[15]. Kawato, in conversation, has stated that Fuji has widely demonstrated working robots based on supervised control. Many other applications have been published.

Direct Inverse Control

Direct inverse control is a highly specialized method, used to make a plant (like a robot arm) follow a desired trajectory, a trajectory specified by a human being or by a higher-order planning system. The underlying idea is illustrated in Figure 3.

****

INSERT FIGURE 3 (Direct Inverse Control)

****

Let us suppose, for example, that we had a simple robot arm, controlled by two joints. One joint controls the angle θ_1, and the other determines θ_2. Our goal is to move the robot hand to a point in two-dimensional space, with coordinates X_1 and X_2. We know that X_1 and X_2 are functions of θ_1 and θ_2. Our job, here, is to go backwards -- for given (desired) X_1 and X_2, we want to calculate the θ_1 and θ_2 which move the hand to that point. If the original mapping from $\underline{\theta}$ to $\underline{X}$ were invertible (i.e., if a unique solution always exists for θ_1 and θ_2), then we can try to learn this inverse mapping directly.

To do this, we simply wiggle the robot arm about for awhile, to get examples of θ_1, θ_2, and the resulting X_1 and X_2. Then we adapt a neural network to input X_1 and X_2 and output θ_1 and θ_2. To use the system, we plug in the desired X_1 and X_2 as input.

Miller[11] has used direct inverse control to achieve great accuracy (error less than 0.1%) in controlling an actual, physical Puma robot. Morita[14] has used direct inverse control with a fuzzy network, but with an ANN learning rule, and claims that this is better than supervised control for the same problem.

In direct inverse control, as in supervised control, it works better to think of the mapping problem in a dynamic context[10], to get better results. This may explain why Miller has gotten better accuracy than many other researchers using this method. (For example, some authors report positioning errors of 4% of the work space. Miller's method may be like getting 4% error in

reducing the remaining gap between the desired position and the actual position; as that gap is reduced from one time step to the next, it should go to zero quite rapidly.)

Direct inverse control does not work when the original map from $\underline{\theta}$ to $\underline{X}$ is not invertible. For example, if the degrees of freedom of the control variables (like $\underline{T}$) are more or less than the degrees of freedom of the observable (like $\underline{X}$), there is a problem. Eckmiller[16] has found a way to break the tie, in cases where there are excess control variables; however, methods of this sort do not fully exploit the value of additional motors in achieving other desirable goals such as smooth motion and low energy consumption.

Kawato's "cascade method" (in [10]) and Jordan [17] describe more general ways of following trajectories, which do achieve these other goals, by rephrasing the problem as one of optimal control. They define a cost function as the error in trajectory following, plus a term for jerkiness or torque change. Then they adapt a neural network to minimize this cost function. To do this, they use the backpropagation of utility -- a different ANN design, to be discussed later on.

Neural Adaptive Control

Neural adaptive control tries to do what conventional adaptive control does, using ANNs instead of the usual linear mappings. Because there are many tools used in conventional adaptive control, this is a complex subject [10,18-20].

One common tool in adaptive control is Model Reference Adaptive Control, where a controller tries to make a system follow specifications laid down in a Reference Model. In the conference on neural networks and fuzzy logic in Houston this year, Narendra described a straightforward way to do this with ANNs. One can simply define a cost function to equal the gap between the output of the reference model and the actual trajectory, and then minimize this cost function exactly as Jordan and Kawato did -- by backpropagating utility. In actuality, one does not have to use the

backpropagation of utility to minimize this cost function; one could also use adaptive critic methods here[10].

In adaptive control, the goal is often to cope with slowly varying hidden parameters. There are two different ways of doing this with ANNs, which are complementary. One is by real-time learning -- where an ANN, like a biological neural network, adapts its weights in real time in response to experience. Another is by adapting memory units which are capable of estimating the hidden parameters. Even without real-time learning, it is possible to train an ANN offline so that it will be adaptive in real-time, because of this memory[10]. Ideally, one would want to combine both kinds of adaptation, but there is a price to be paid in so doing. The main price is that backpropagation through time must be replaced by adaptive critics[10] both in control and in system identification; the tradeoffs involved will be discussed in the next section.

In conventional, linear adaptive control it is often possible to prove stability algebraically in advance by specifying a Liapunov function [18]. In nonlinear adaptive control, it is far more difficult[20]. In actuality, however, the "Critic" networks to be discussed below function very much like Liapunov functions (especially in the BAC design). For many complex, nonlinear problems, it may be necessary to adapt a Liapunov function after the fact, and verify its properties after the fact, rather than specify it in advance.

Backpropagating Utility and Adaptive Critics

General Concepts

Backpropagating utility and adaptive critics are two general-purpose designs for optimal control, using neural networks. In both cases, the user specifies a utility function or performance index to be maximized, or a cost function to be minimized. In both cases, these designs will always have more than one ANN component. Different components are adapted by different learning rules, aimed at minimizing or maximizing different things.

There will always be an Action network, which inputs current state information (and perhaps other information), and outputs the actual vector of controls, $\underline{u}(t)$. The utility function itself can also be thought of as a network (the Utility network), even though it is not adapted. (Some earlier papers talked about "reinforcement learning," which is logically a special case of utility maximization[10,11].) In most cases, there will also be a Model network, which inputs a current description of reality, $\underline{R}(t)$, and the action vector $\underline{u}(t)$; it outputs a forecast of $\underline{R}(t+1)$ and of $\underline{X}(t+1)$, the vector of sensor inputs at time $t+1$. (In some cases, the Model network can be a stochastic network, which outputs simulated values rather than forecasts.) Finally, in the case of Critic designs, there will be a Critic network, which inputs $\underline{R}(t)$ and possibly $\underline{u}(t)$, and outputs something like an estimate of the sum of future utility across all future times.

The real challenge in maximizing utility over time lies in the problem of linking present action to future payoffs, across all future time periods. There are really only two ways to address this problem, in the general case. One is to take a proposed Action network, and explicitly work out its future consequences, for every future time period. This is exactly what the calculus of variations does, in conventional control theory, and it is also what the backpropagation of utility does. The backpropagation of utility is equivalent to the calculus of variations, but -- because derivatives are calculated efficiently through large sparse nonlinear structures -- one may hope for less expensive implementation. A second approach is to adapt a network which predicts the optimal future payoff (over all future times) starting from a given value for $\underline{R}(t+1)$, and to use that network as the basis for choosing $\underline{u}(t)$. This requires that we approximate the payoff function, J^0, of dynamic programming. This is the Adaptive Critic approach.

Backpropagating Utility

The backpropagation of utility through time is illustrated in Figure 4.

****

INSERT FIGURE 4 (Backpropagating Utility)

****

In the backpropagation of utility, we must start with a Model network which has already been adapted, and a Utility network which has already been specified. Our goal is to adapt the weights in the Action network. (In practice, of course, we can adapt both the Action net and Model net concurrently; however, when we adapt the Action net, we treat the Model net as if it were fixed.) To do this, we start from the initial conditions, $\underline{X}(0)$, and use the initial weights in the Action network to predict $\underline{X}(t)$ at all future times t. Then we use generalized backpropagation to calculate the derivatives of total utility, across all future time, with respect to all of the weights in the Action network. This involves backwards calculations, following the dashed lines in Figure 4. Then we adjust the weights in the Action network in response to these derivatives, and start all over again. We iterate until we are satisfied. The mechanics are described in more detail in [1], but Figure 4 really tells the whole story.

The backpropagation of utility was first proposed in 1974[21]. By 1988, there were four working examples. There was the truck-backer-upper of Nguyen and Widrow, and the "cascade" robot arm controller of Kawato, both published in [10]. There was Jordan's robot arm controller[17], and my own official DOE model of the natural gas industry[22]. Recently, Narendra and Hwang have reported success with this method.

The backpropagation of utility is a very straightforward and exact method. Unfortunately, there have been few reported successes this past year. This may be due in part to a lack of straightforward tutorials (though [1] and [22] should help). The biggest problem in practical applications may be the difficulty of adapting a good Model network. In some applications, it may be good enough to build a Model network which inputs $\underline{X}(t)$ and $\underline{u}(t)$, which uses $\underline{X}(t+1)$ as its

target, and contains time-lagged memory units (as described in [1]) to complete the state vector description; however, in some applications, it is crucial to go beyond this, and insert special "sticky" neurons -- designed to represent slowly-varying hidden parameters -- and elements of robust estimation [10].

The biggest limitation of backpropagating utility is the need for a forecasting model, which cannot be a true stochastic model. In fuzzy logic, this is not so bad, because the variable being forecasted may itself be a measure of likelihood or probability. In some applications, however -- like stock market portfolio optimization -- a more explicit treatment of probabilities and scenarios may be important. There are tricks which can be used to represent noise, even when backpropagating utility, but they are somewhat ad hoc and inefficient[10].

Another problem in backpropagating utility is the need to learn in an offline mode. The calculations backwards through time require this. Various authors have devised ways to do backpropagation through time in a time-forwards direction [e.g.23], but those techniques are either very approximate or do not scale well with large problems or both; in any case, Narendra[19] has questioned the stability of such methods. Nevertheless, even if we backpropagate utility in an offline mode, we can still develop a network which adapts in real-time to changes in slowly-varying parameters; we can "learn offline to be adaptive online." [10]. This should be very attractive in many applications, because true real-time learning is more difficult.

Adaptive Critics

Adaptive critic methods, by contrast, do permit true real-time learning and stochastic models, but only at a price: they lack the exactness and simplicity of backpropagating utility. One reason for their lack of simplicity is the wide variety of designs available -- from simple 2-Net structures, which work well on small problems, through to complex hybrids, which hopefully encompass what

goes on in the human brain[10,11].

Adaptive critic methods may be defined, in broad terms, as methods which attempt to approximate dynamic programming as first described in [24]. Dynamic programming is the only exact and efficient method available to control actions or movements over time, so as to maximize a utility function in a noisy, nonlinear environment, without making highly specialized assumptions about the nature of that environment. Figure 5 illustrates the trick used by dynamic programming to solve this very difficult problem.

****

Figure 5 (inputs and outputs of dynamic programming)

****

Dynamic programming requires as its input a utility function U and a model of the external environment, F. Dynamic programming produces, as its major output, another function, J, which I like to call a secondary or strategic utility function. The key insight in dynamic programming is that you can maximize the function U, in the long-term, over time, simply by maximizing this function J in the immediate future. After you know the function J and the model F, it is then a simple problem in function maximization to pick the actions which maximize J. The notation here is taken from Raiffa[25], whose books on decision analysis may be viewed as a highly practical and intuitive introduction to the ideas underlying dynamic programming.

Unfortunately, we cannot use dynamic programming exactly on complicated problems, because the calculations become hopelessly complex. (Bayesian inference sometimes entails similar complexities.) However, it is possible to approximate these calculations by using a model or network to estimate the J function or its derivatives (or something quite close to the J function, like the J' function of [26] and [27].) Adaptive critic methods may be defined more precisely as methods which take this approach.

If this kind of design were truly fundamental to human intelligence, as I would claim, one might expect to find it reflected in a wide variety of fields. In fact, notions like U and J do reappear in a wide variety of fields, as illustrated in Table 2 (taken from [28]):

****

Table 2 (Examples of J and U)

****

Please note that the last entry in Table 2, the entry for Lagrange multipliers, corresponds to the derivative of J, rather than the value of J itself. In economic theory, the prices of goods are supposed to reflect the change in overall utility which would result from changing your level of consumption of a particular good. Likewise, in Freudian psychology, the notion of emotional charge associated with a particular object corresponds more to the derivatives of J; in fact, the original inspiration for backpropagation[29] came from Freud's theory that emotional charge is passed backwards from object to object, with a strength proportionate to the usual forwards association between the two objects[30]. The Backpropagated Adaptive Critic (BAC) design reflects that theory very closely. The word "pleasure" in Table 2 should not be interpreted in a narrow way; for example, it could include such things as parental pleasure in experiencing happy children.

In order to build an adaptive critic controller, we need to specify two things: (1) how to adapt the Action network in response to the Critic; (2) how to adapt the Critic network.

The most popular adaptive critic design by far is the 2-network arrangement of Barto, Sutton and Anderson[31], illustrated in Figure 6. In this design, there is no need for a model of the process to be controlled. The estimate of J is treated as a gross reward or

****

Figure 6. The 2-Net Design of Barto, Sutton and Anderson

****

punishment signal. This design has worked well on a wide variety of real-world problems, including robotics[32], autonomous vehicles and fuzzy logic systems. Williams, in [20], has reported some interesting new results on convergence. Unfortunately, this approach becomes very slow as the number of control variables or state variables grows to 10 or 100. The reason for this is very straightforward: knowing J is not enough to tell us which actions were responsible for success or failure, and it does not tell us whether we need more or less of any component of the action vector. This design is like telling a student that he or she did "well" or "poorly" on an exam, without pinpointing which answers were right or wrong; it is a lot harder for a student to improve performance when he or she has no specific idea of what to work on.

Fortunately, there are alternative designs which can overcome this problem. Note that it is critical to modify both the Action network and the Critic network, to permit learning at an acceptable speed when the number of variables is large (as in the human brain). There are also some other tricks which can help, discussed by myself, by Barto, and by Sutton [10,11,20].

To speed up learning in the Action network, for large problems, there are now two major alternatives: (1) the Backpropagated Adaptive Critic (BAC), shown in Figure 7;
(2) the Action-Dependent Adaptive Critic (ADAC), shown in Figure 8.

****

Insert Figures 7 and 8: BAC and ADAC (as adapting Action net)

****

The BAC design is closer to dynamic programming than is the 2-net design, because there is a more explicit attempt to pick $\underline{u}(t)$ so as to maximize $J(t+1)$, based on the use of generalized backpropagation to calculate the derivatives of $J(t+1)$ with respect to the components of $\underline{u}(t)$. The dashed lines in Figure 7 represent the calculation of derivatives. (Usually we adapt the weights in the action network in proportion to these derivatives, rather than adapting $\underline{u}(t)$ itself.) The cost of

BAC is that we need to develop a Model network, as we do when backpropagating utility. The adaptation of a good dynamic model can be a challenging task at times[10].

ADAC [26,27] avoids the need for an explicit model, but the Critic network in Figure 8 would have to represent the combination of the Critic and Model in Figure 7. Jordan, in conversation, has stated that he adapted an action-dependent Critic network in 1989, based on an independent paper by Watkins on "Q learning" (discussed in [20]), but found the resulting Critic network to be rather complex. In an ideal world, one would want to combine both approaches, so as to combine the modularity and cleanliness of BAC with the model-independent robustness of ADAC; however, BAC may be good enough by itself in many applications. Jameson has reported some preliminary results with BAC[33], and other aerospace-oriented researchers may have dealt with larger applications; however, more work is needed. Whatever the details, the adaptation of Action network in large-scale problems is clearly central to the future of this discipline and of our ability to understand organic intelligence.

In adapting the Critic networks, few people have gone beyond simple, scalar methods which are more or less equivalent[34] and which have severe scaling problems. There are two alternatives which should scale much better: (1) Dual Heuristic Programming (DHP), which outputs estimates of the derivatives of J; (2) Globalized DHP (GDHP), which outputs an estimate of J (or its components), but which adapts the Critic by minimizing error in the implied derivatives as well as the estimate of J. These methods were first proposed in the 1970s [23,24], but are described in more modern language in [10] and [11]. Both methods require the existence of a Model network. Hutchinson of BehavHeuristics has claimed real-world commercial success in applying such methods, but many of the details are proprietary.

Example of a Hybrid System

In 1988, a friend of mine asked how I would use these methods to assist in some very complex social decision problems, well beyond the scope of this paper. Given the nature of his application, I recommended a very conservative approach for the time being. As a first stage, I would obtain a conventional sort of modeling system, capable of storing and analyzing time-series data, and capable of manipulating forecasting models built up from any of three methodologies: (1) econometric-style equations; (2) fuzzy logic; (3) ANNs. I would look for a linkage capability, so that models of specific sectors (built up from different methodologies and often revised) could be combined together to yield composite streams of forecasts. Then I would build a general purpose "dual compiler." The dual compiler would input a sectoral model (in text form or parsed into a tree), and output a "dual subroutine" (like those in [1]), so as to facilitate the use of generalized backpropagation. Then I would implement a whole set of tools using backpropagation.

Tool number one would be a simple sensitivity analysis tool. The user would type in a utility function or target function. The tool would then calculate the derivatives of utility with respect to all of the inputs -- initial values, policy variables, and parameters -- which affected the original forecast, in one quick sweep through the process. It would report back the ten or the hundred most important inputs. (There is a scaling problem here in deciding which input is most important; the user could be given a choice, for example, between looking for the biggest derivatives, the biggest elasticities, or the biggest derivatives weighted by some other variables.) The user could go on to make plans to change these inputs, so as to increase utility, or he could first evaluate in detail whether he believes that the inputs are really important. (Tests of this sort can in fact be very useful in pinpointing weaknesses of an integrated modeling system[8], or real-world uncertainties which require more analysis). The cost of a comprehensive sensitivity analysis is the key issue here; using more conventional tools, one must often wait a long time and spend a lot of money to get even a partial sensitivity analysis, and the results are usually out of date.

Tool number two would help in reassessing the importance of the key inputs. For any given input, it would use the intermediate information generated by backpropagation (as in [8]) to identify the path of connections which really made that input important. It could even display this information as a kind of tree or flow chart. This would be similar in purpose to the inference sequences printed out as "explanations" by many expert systems.

Tool number three would be an extended version of tools one or two. Instead of first derivatives, it would provide information based on low-cost second derivatives (as described in [23], based on calculations like those in [5,11]). For example, the sensitivity of utility to dollars spent in 1992 may be a key measure of policy effectiveness; it may be useful to see how that measure, in turn, would be changed by other factors (such as diminishing returns or complementary variables). At the optimum, the first derivative of utility with respect to any policy variable will be zero; the derivatives of that derivative give information about why the policy variable should be set at a particular level.

Tool number four would be a full-fledged version of backpropagating utility. The user could flag certain variables or parameters as policy variables, and the computer would be asked to suggest an optimal improvement upon current plans, so as to maximize utility. The resulting suggestion may be a local minimum, but it should at least be better than the starting plans.

Tool number five would be a model calibration tool, based on the backpropagation of error, and robust estimation concepts like those of [10]. At a minimum, this would be a relatively quick and objective way to calibrate a model as a whole system to fit the past; it could replace the rather elaborate and ad hoc "tweaking" which usually goes into most complex models in the real world for calibration purposes. Tool number six would go back and identify how the resulting parameter estimates or rules were influenced by different cases in the input dataset; this would provide an integrated, nonlinear version of the highly respected linear diagnostic tools developed by Belsley,

Kuh and Welsch[35].

These six tools are the most obvious needed tools, exploiting backpropagation, but a host of other tools are possible involving estimation diagnostics, decision diagnostics and convergence tools. Also, there is no need to develop the six tools in the order of my discussion.

In principle, one can even build a strategic assessment or stochastic planning tool, based on adaptive critic methods but permitting user-specified assessment models, as described in [28].

To bring all these tools together in a general-purpose modeling package, capable of running on desktop workstations, would not be a trivial task. However, there are important applications, and some work has begun in this direction. All of these tools aim at effective two-way man-machine communication, so as to exploit the capabilities of both forms of intelligence.

Conclusions

Neurocontrol and fuzzy logic are complementary, rather than competitive, technologies. There are numerous ways of combining the two technologies. Which combination is best depends very heavily on the particular application; there is always a tradeoff between "general syntheses" -- which combine everything but require the expense of implementing everything -- and direct, simple designs tuned to particular concrete problems. Given the natural human tendency towards inertia, it is critical to be aware of a wide variety of options, and to ask "Why not?" when considering new approaches. Even within neurocontrol, there is a wide variety of designs available, ranging from simple off-the-shelf technologies (easily applied to fuzzy logic networks) through to areas where fundamental research is still needed and vital to our understanding of real intelligence.

REFERENCES

1. P.Werbos, Backpropagation through time: What it does and how to do it, Proceedings of the IEEE, October 1990.

2. A.Amano et al, On the use of neural networks and fuzzy logic in speech recognition. In Proceedings of the International Joint Conference On Neural Networks (IJCNN). New York: IEEE, June 1989.

3. J. Schreinemakers and D. Touretzky, Interfacing a neural network with a rule-based reasoner for diagnosing mastitis. IJCNN Proceedings. Hillsdale, NJ: Erlbaum, January 1990.

4. H. Takagi, Fusion technology of fuzzy theory and neural networks. In Proceedings of Fuzzy Logic and Neural Networks, Izzuka, Japan, 1990.

5. P.Werbos, Backpropagation: past and future. In Proceedings of the Second International Conference on Neural Networks. New York: IEEE, 1988. A transcript of the actual talk, with slides, is available from the author, and covers some additional topics.

6. L.Fu, Backpropagation in neural networks with fuzzy conjunction units. IJCNN Proceedings. New York: IEEE, June 1990.

7. L. Hsu et al, Fuzzy logic in connectionist expert systems. IJCNN Proceedings. Hillsdale, NJ: Erlbaum, January 1990.

8. P.Werbos, Generalization of Backpropagation, Neural Networks, October 1988. When simultaneous recurrence is not present, the calculations are much simpler, as in [22].

9. P. Werbos, Making diagnostics work in the real world: a few tricks. In A. Maren (ed.), Handbook of Neural Comp. Appl. Academic Press, 1990.

10. P.Werbos. Neurocontrol and related techniques. In A. Maren (ed.), Handbook of Neural Comp. Appl.. Academic Press, 1990.

11. W.T.Miller, Sutton and Werbos (eds), Neural Networks for Control. Cambridge, Mass.: MIT

Press, 1990.

12. C. Mead, Analog VLSI and Neural Systems. Reading, Mass.: Addison-Wesley, 1989.

13. P.Werbos, Links between artificial neural networks (ANN) and statistical pattern recognition. In I.Sethi and Jain (eds), Artificial Neural Networks and Statistical Pattern Recognition: Old and New Connections. Elsevier, forthcoming.

14. Morita et al, Fuzzy knowledge model of neural network type. IJCNN Proceedings. Hillsdale, NJ: Erlbaum, 1990.

15. B.Widrow and Smith, Pattern-recognizing control systems. In 1963 Computer and Information Sciences (COINS) Symposium Proceedings. Washington, D.C.: Spartan.

16. R. Eckmiller et al, Neural kinematics net for a redundant robot arm. In IJCNN Proceedings. New York: IEEE, June 1989.

17. M. Jordan, Generic constraints on underspecified target trajectories. In IJCNN Proceedings. New York: IEEE, June 1989.

18. K. Narendra and Annaswamy, Stable Adaptive Systems. Englewood, NJ: Prentice-Hall, 1989.

19. K. Narendra and Parthasarathy, Identification and control of dynamical systems using neural networks,IEEE Trans. Neural Networks, Vol. 1, No. 1 (March 1990).

20. K. Narendra (ed), Proceedings of the Sixth Yale Workshop on Adaptive and Learning Control. New Haven, Conn.: K. Narendra, Yale, 1990.

21. P. Werbos, Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph.D. thesis to Harvard U. Committee on Applied Mathematics, November 1974.

22. P. Werbos, Maximizing long-term gas industry profits in two minutes in Lotus using neural network methods,IEEE Trans. SMC, March/April 1989.

23. P. Werbos, Applications of advances in nonlinear sensitivity analysis. In R. Drenick ans Kozin (eds), Systems Modeling and Optimization: Proceedings of the International Federation for

Information Processing. New York: Springer-Verlag, 1982.

24. P. Werbos, Advanced forecasting methods for global crisis warning and models of intelligence,General Systems Yearbook, 1977 issue.

25. H.Raiffa,Decision Analysis: Introductory Lectures on Making Choices Under Uncertainty. Reading, Mass.: Addison-Wesley, 1968.

26. P.Werbos,Neural networks for control and system identification. In IEEE CDC Proceedings. New York: IEEE, 1989.

27. G.Lukes, B. Thompson and P.Werbos, Expectation driven learning with an associative memory. In IJCNN Proceedings (Washington). Hillsdale, NJ: Erlbaum, 1990.

28. P.Werbos, Generalized information requirements of intelligent decision-making systems. In SUGI-11 Proceedings. Cary, NC: SAS Institute. A revised version, available from the author, is somewhat easier to read, and elaborates more on connections to humanistic psychology.

29. P.Werbos, Elements of intelligence, Cybernetica (Namur), No. 3, 1968.

30. D.Yankelovitch and Barrett, Ego and Instinct: The Psychoanalytic View of Human Nature-Revised. New York: Vintage, 1971. This is an unusually clear presentation, though my own work would suggest it is too pessimistic in its conclusions.

31. A.Barto, Sutton and Anderson, Neuron-like adaptive elements that can solve difficult learning control problems, IEEE Trans. SMC. SMC-13, p. 834-846.

32. J. Franklin, Reinforcement of robot motor skills through reinforcement learning. In IEEE/CDC Proceedings. New York: IEEE, 1988.

33. J.Jameson, A neurocontroller based on model feedback and the Adaptive Heuristic Critic. In IJCNN Proceedings (San Diego). New York: IEEE, June 1990.

34. P. Werbos, Consistency of HDP applied to a simple reinforcement learning problem, Neural Networks, Vol. 3, p.179-189, March 1990.

35. D.Belsley, Kuh and Welsch,Regression Diagnostics: Identifying Influential Data and Sources of Collinearity. Wiley, 1980.

LIST OF FIGURES, TABLES AND CAPTIONS

Figures

Tables

FIGURE 1

WHAT IS NEUROCONTROL?

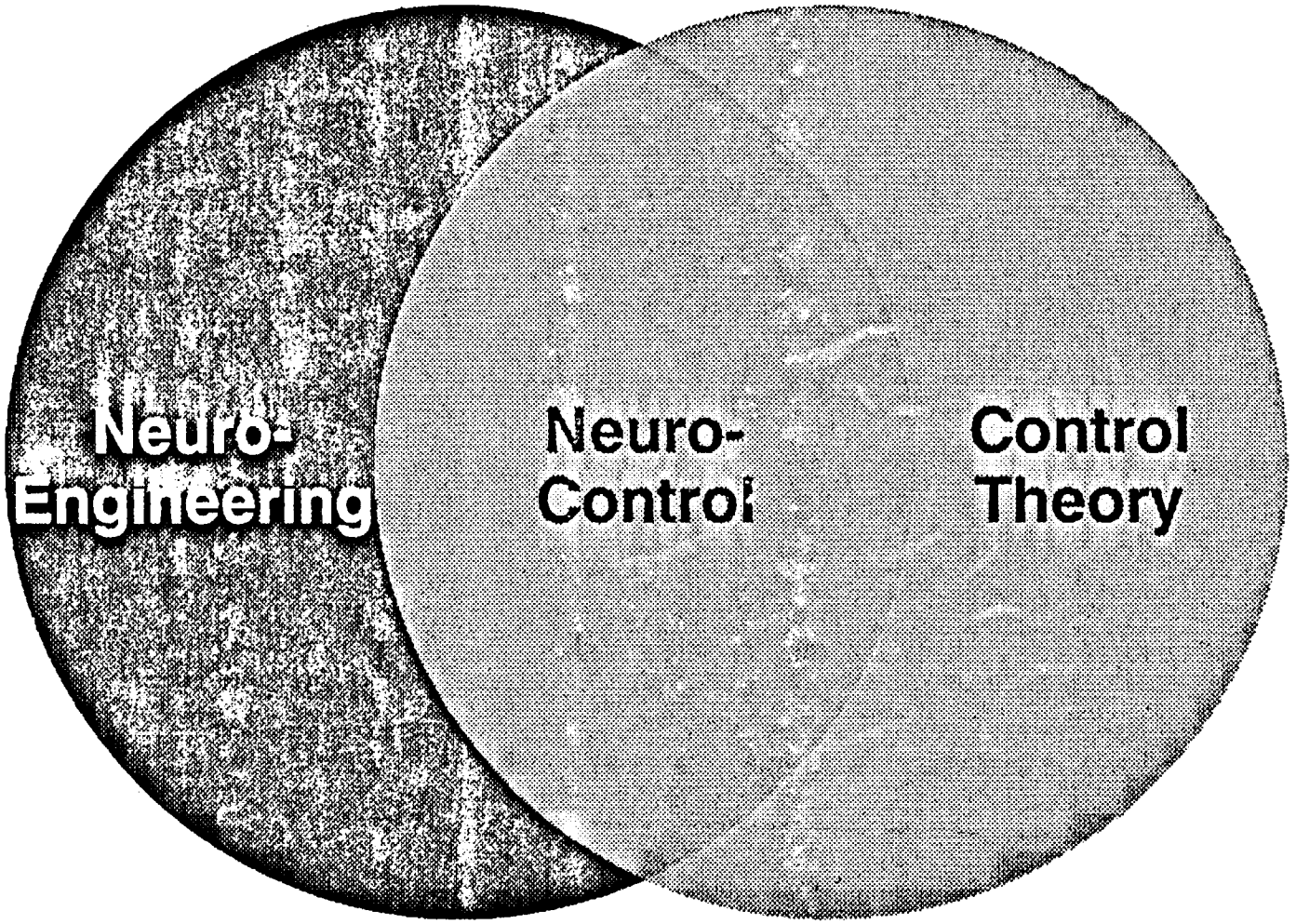


Figure 2

CAN WE DESIGN AND UNDERSTAND INTELLIGENCE?

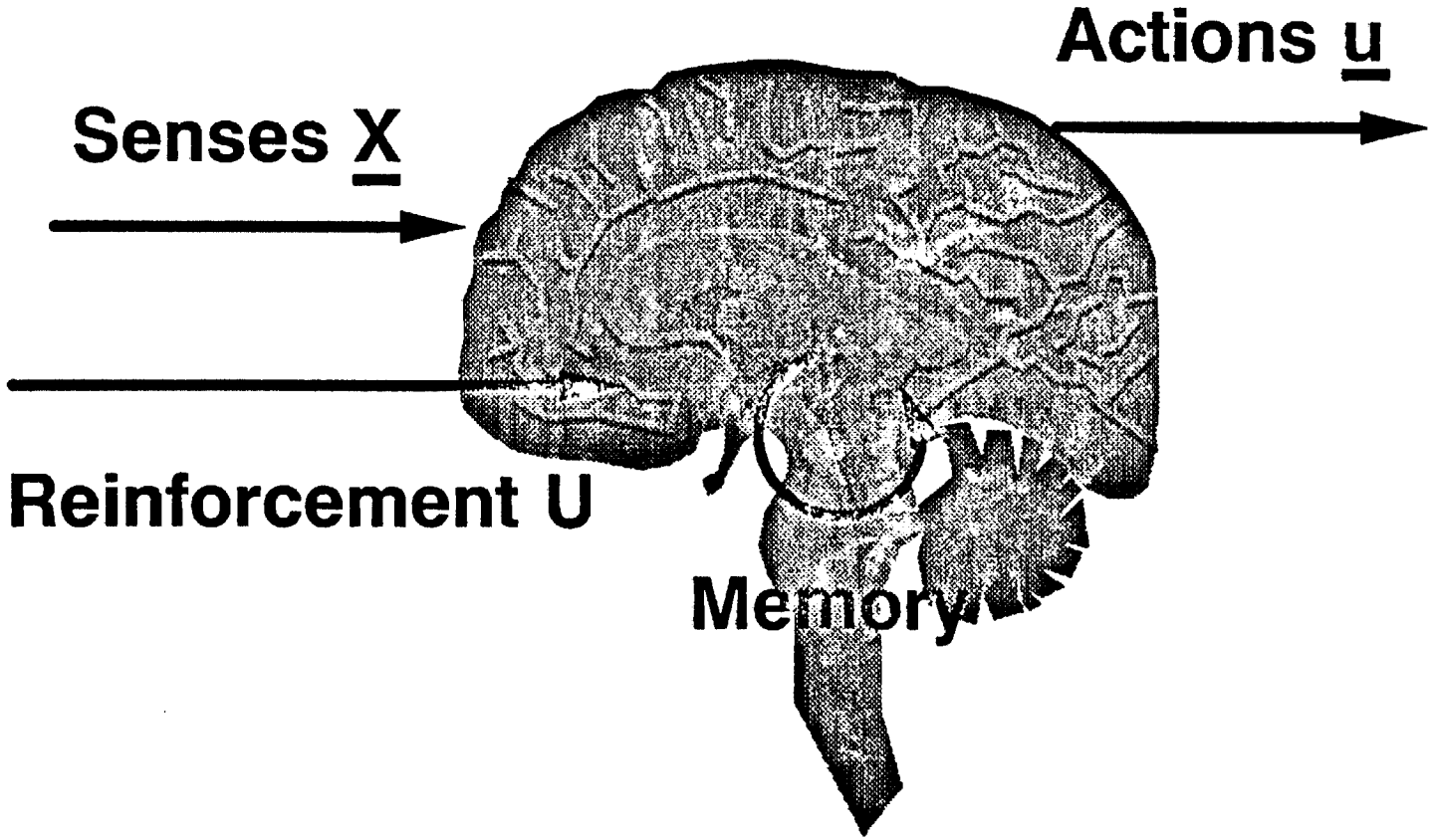
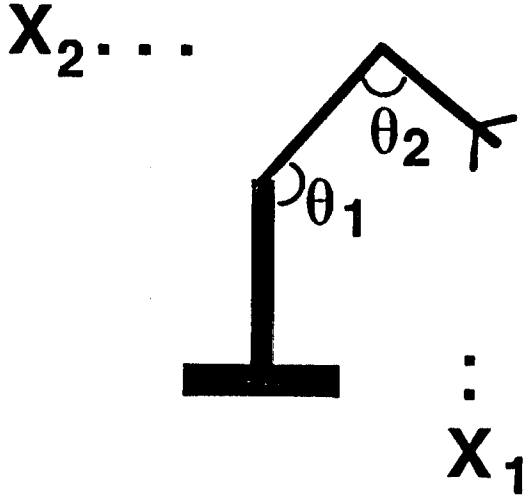


Figure 3

BASIC INVERSE CONTROL



$$X_1, X_2 = f(\theta_1, \theta_2)$$

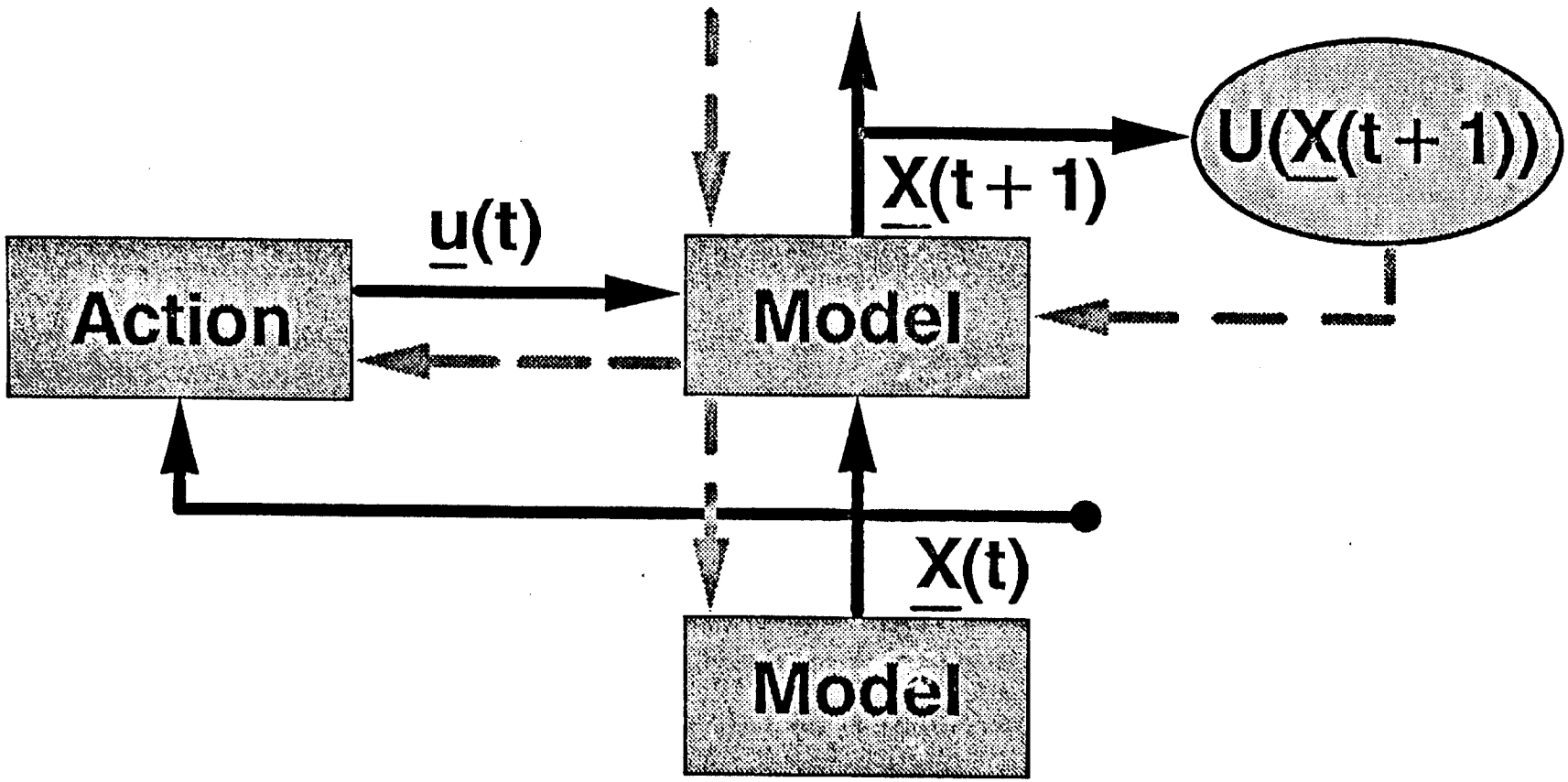
$$\theta_1, \theta_2 = f^{-1}(X_1, X_2)$$

Input X_1, X_2; Target θ_1, θ_2

~~Miller: X (t-1), Low Error~~

~~Jordan, Kawato Not Basic~~

Figure 4



~~BTT: Werbos, Jordan, Widrow, Kawato~~

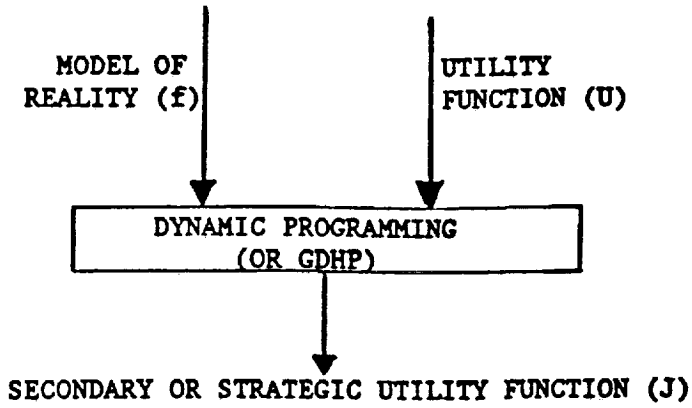
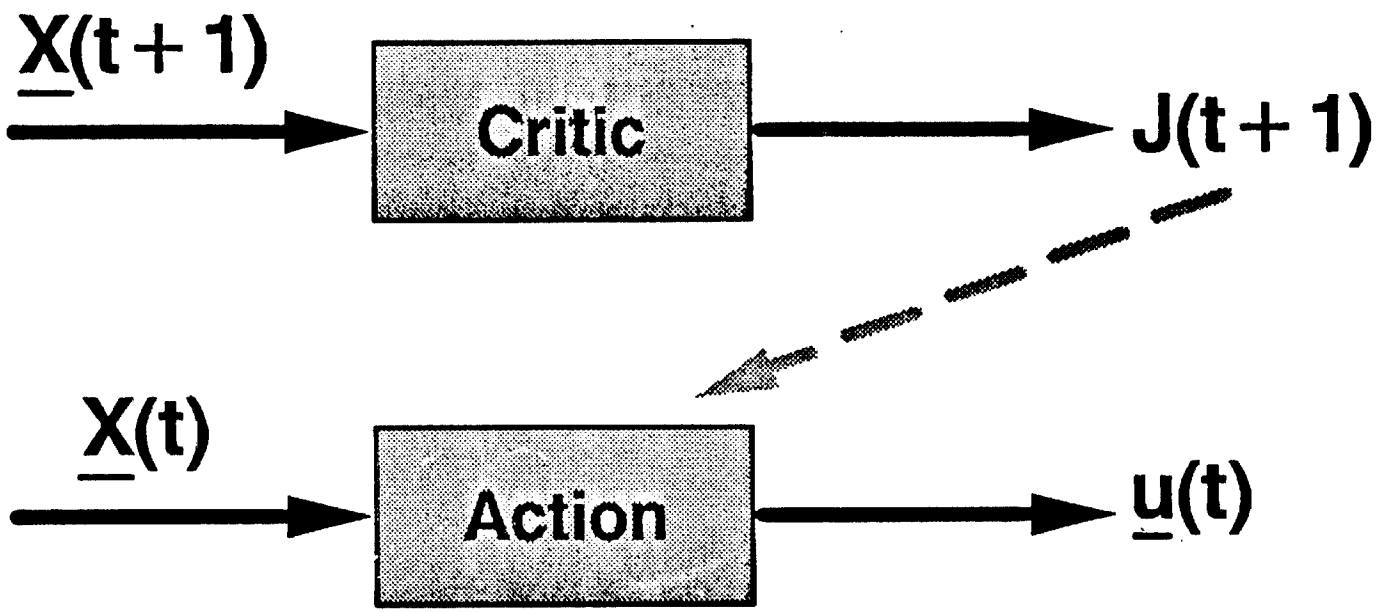
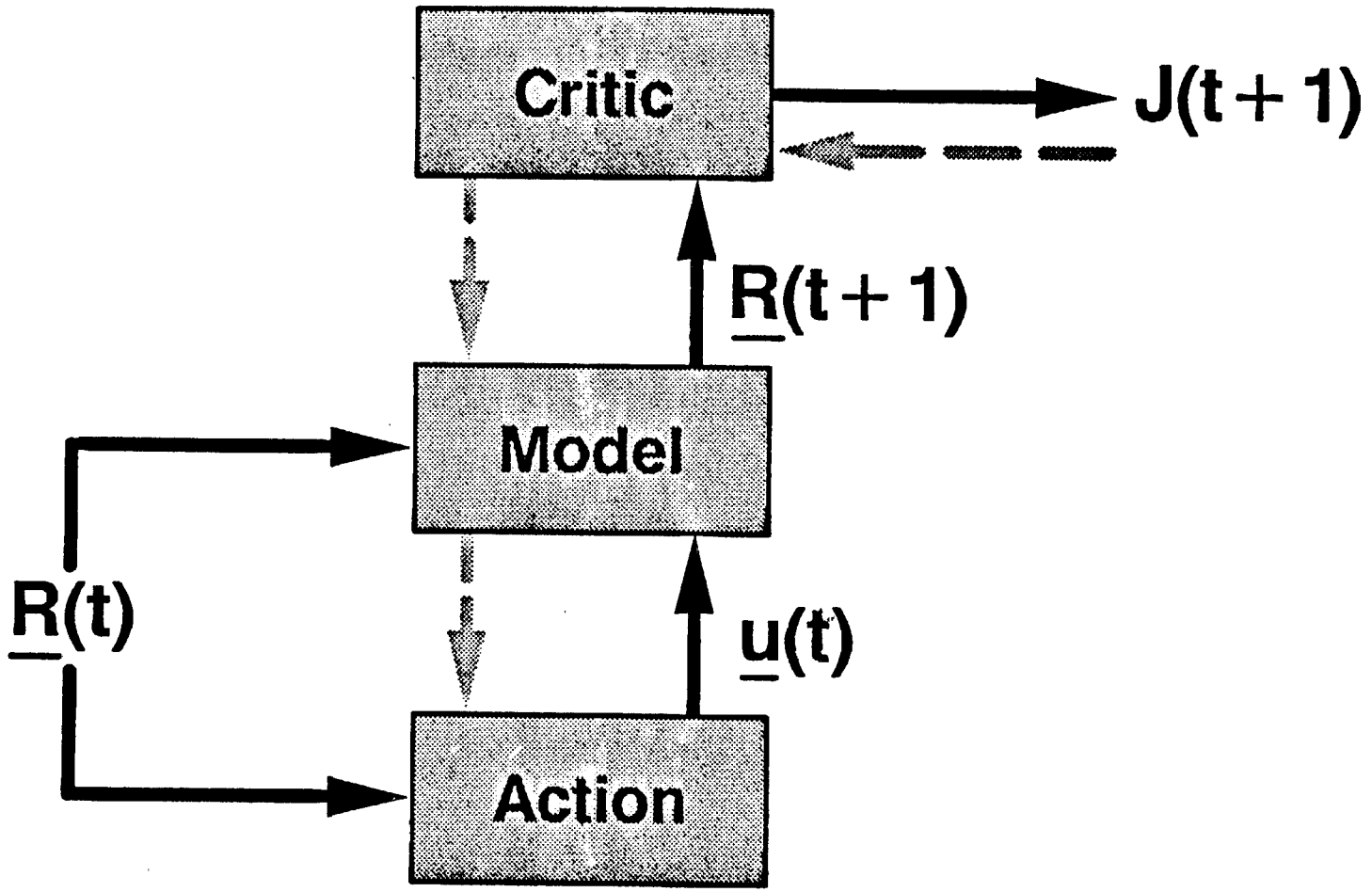


Figure 5

Figure 6



2-Net Adaptive Critic



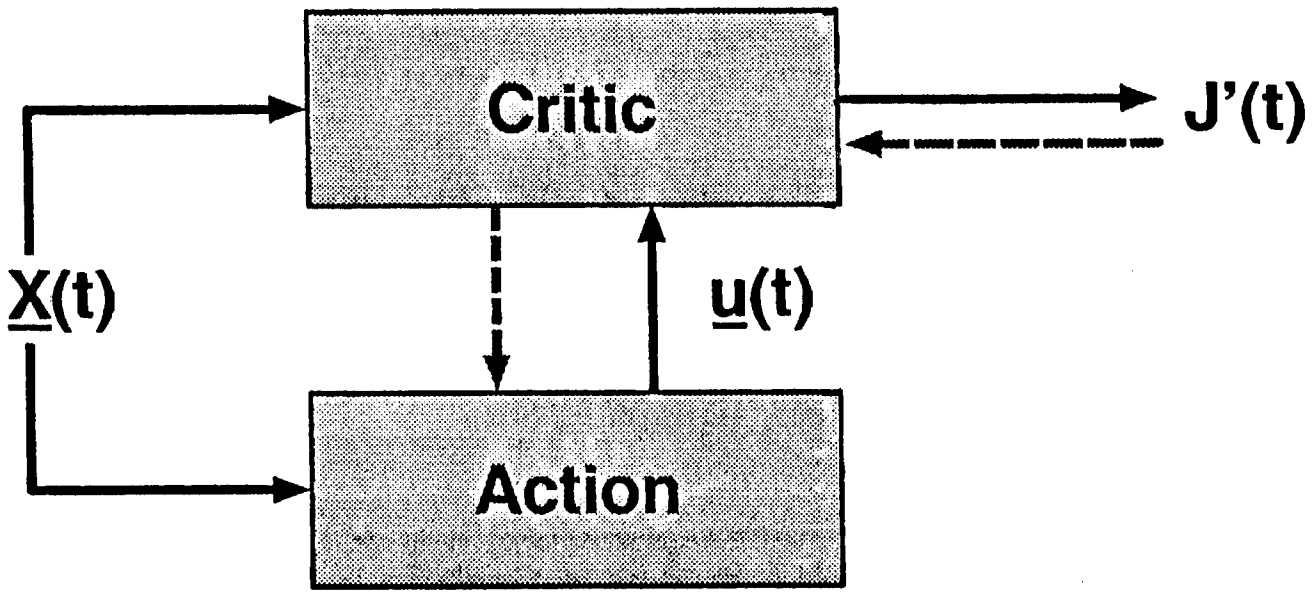
Backpropagated Adaptive Critic

TABLE 1

	Many Motors	Noise	Long-Term Optimization, Planning	Real-Time Learning
Supervised Control	X	X		X
Direct Inverse Control	(X)	X		X
Neural Adaptive Control	X	?		?
Backpropagating Utility	X		X	
Adaptive Critics				
2-Net		X	X	X
BAC+DHP,etc.	X	X	X	X

Figure 8

ADAC



Action Dependent Adaptive Critic [2,6]

TABLE 2

Examples of Strategic Utility (J)

Domain	Basic Utility (U)	Strategic Utility (J)
Chess	Win/Lose	Queen = 9 points, etc.
Business Theory	Current Profit Cash Flow	Present Value of Strategic Assets (Performance Measures)
Human Thought	Pleasure/Pain Hunger	Hope/Fear Reaction to Job Loss
Behavioral Psychology	Primary Reinforcement	Secondary Reinforcement
Artificial Intelligence	Utility Function	Static Position Evaluator (Simon) Evaluation Function (Hayes-Roth)
Government Finance	National Values, Long-Term Goals	Cost/Benefit Measures
Physics	Lagrangian	Action Function
Economics	Current Value of Product to You	Market Price or Shadow Price ("Lagrange multipliers")

Actually, the items in the last row of this table refer to the derivatives of the two kinds of utility. They refer to the change in overall utility which would result from a small change in your level of consumption of a particular good. Likewise, the benefit of increasing a particular measure of environmental quality is really just the derivative of J with respect to that measure. In Freudian psychology, the emotional charge associated with a particular object is really just the derivative of J with respect to the corresponding variable. (All of these examples could be discussed at greater length; for example, I would contend that human "pleasure" includes parental pleasure in experiencing happy children, but this issue goes well beyond the theory of intelligence as such.)

Derivatives are vital to the operation of any intelligent system. For example, when SAS estimates the parameters of a nonlinear model, it starts from the initial guesses which you provide for these parameters. It adjusts these guesses upwards or downwards, so as to minimize forecasting error over historical data. In order to do this efficiently, it has to know whether a small positive adjustment of any parameter would increase error, decrease error, or leave the error unchanged; in other words it has to know the derivative of error with respect to every parameter.

INTELLIGENT CONTROL BASED ON FUZZY LOGIC AND NEURAL NET THEORY

Chuen-Chien Lee

Department of Electrical Engineering and Computer Sciences
University of California
Berkeley, CA 94720

ABSTRACT

In the conception and design of intelligent systems, one promising direction involves the use of fuzzy logic and neural network theory to enhance such systems' capability to learn from experience and adapt to changes in an environment of uncertainty and imprecision. This paper explores an intelligent control scheme by integrating these multi-disciplinary techniques. A self-learning system is proposed as an intelligent controller for dynamical processes, employing a control policy which evolves and improves automatically. One key component of the intelligent system is a fuzzy logic-based system which emulates human decision-making behavior. Another key component is cognitive neural models derived from animal learning theory, which stimulate memory association and learning behavior. It is shown that the system can solve a fairly difficult control learning problem. Simulation results demonstrate that improved learning performance can be achieved in relation to previously described systems employing bang-bang control. The proposed system is relatively insensitive to variations in the parameters of the system environment.

I. INTRODUCTION

During the past several years, a highly promising direction in the design of intelligent systems has emerged. More specifically, the direction in question involves the use of fuzzy logic and neural network theory to enhance the ability of intelligent systems that can learn from experience and adapt to changes in an environment of uncertainty and imprecision. This paper provides a brief introduction on a fuzzy logic-based system [16][17] and cognitive neural models [18][19], and explores an intelligent control system by integrating these multi-disciplinary techniques. The approach described here may be viewed as a step in the development of a better understanding of how to combine a fuzzy logic-based system with a neural network to achieve a significant learning / adaptive capability.

This research was supported in part by NASA Grant NCC-2-275 and Micro Grant 88-094.

A. Why Fuzzy Logic Control?

There are many complex industrial processes which cannot be satisfactorily controlled by conventional methods due to modeling difficulties and unavailability of quantitative data regarding input-output relations. And yet, skilled human operators can control such systems quite successfully without having any quantitative models in mind. Furthermore, the operation of many man-machine systems requires the use of rules of thumb, intuition, and heuristics. All of these features are uncertain and imprecise and cannot be addressed adequately by conventional methods. As the increasing complexity and nonlinearity of control systems render conventional methods less effective, a rule-based system based on fuzzy logic becomes an increasingly attractive alternative.

In fact, during the past several years, rule-based controllers based on fuzzy logic [16][17] have emerged as one of the most active and fruitful areas for research in the application of fuzzy set theory [34]. Among the representative applications of fuzzy logic-based controllers are the subway system in the city of Sendai [33], container ship crane control [32], elevator control [4][30], nuclear reactor control [2][11], automobile transmission control [23], air conditioners [22], anti-lock break systems [24] and human-quality robot eyes [5]. Experience shows that a rule-based controller using fuzzy logic make it possible to emulate and even surpass the decision-making ability of a skilled human operator.

Although there is an extensive literature describing various fuzzy logic-based controllers using approximate reasoning, the acquisition of the

rule base in such controllers is not as yet well understood. In past applications, fuzzy decision rules are either obtained from verbal expressions or observations of human operator control actions. Since domain experts and skilled operators do not structure their decision making in any formal way, the process of transferring expert knowledge into a usable knowledge base is tedious and unsystematic. Our research aims at the development of a better understanding of such problems, with a view to enhancing the potential of fuzzy logic-based controllers, which can operate effectively in an environment of uncertainty and imprecision.

One direction that is beginning to be explored is that of the conception and design of fuzzy systems which have the capability to learn from experience. In this context, a combination of techniques drawn from both fuzzy logic and neural network theory may provide a powerful tool for the design of intelligent systems which can emulate the decision-making ability of a skilled human operator and the ability to learn and adapt to changes in an environment of uncertainty and imprecision.

B. Why Cognitive Neural Models?

The theory of animal learning is inferred from observed behavior and constitutes carefully testified postulates regarding elemental processes of learning. Recent research into animal learning can be separated into two categories: the behavioral and neural substrates of learning, namely, the psychological and physiological levels of learning. One way to bridge such a gap is to postulate neural analogies of behavioral modification paradigms. Hebb's postulate [9] for synaptic plasticity was the first trial as a neural analogy of associative learning, which attempted to bridge psychology and neurophysiology. The theory of adaptive networks originated with [9] and continues to be influenced by plausible neural analogies of behavioral conditioning [6][12] [7][28][26][29][13][27][14][8][15].

Contemporary artificial neural networks are frequently referred to as connectionist models, parallel distributed processing (PDP) models, and adaptive / self-organizing networks. Basically, it is a complex system of neuron-like processing units that operate asynchronously but in parallel and whose function is determined by the network topology of connectivity. Artificial neural networks provide a new computational structure, a plausible approach for information processing because of its adaptivity / learning as well as massive parallelism.

Although new learning algorithms and VLSI technologies have recently provided strong impetus to neural network research, many problems still exist. Among them, the comprehensibility of neural networks, theoretical parsimony / enormous cost, and limited empirical successes are some of the major issues underlying the limitations of current neural networks. The learning behavior of such networks is difficult to understand, and the role of generic elements and subnetworks is unclear. Furthermore, most of these networks lack a theoretical foundation. The time and effort required to develop neural network architectures (network topology) and training is very high. Research has been directed in the main at "modeling applications", while relatively few "fielded applications" have emerged [3]. Most of such applications are restricted to pattern recognition, categorization, and realizations of associative memory. They are still toy research problems at the proof-of-concept stage. Among the few exceptions, the Adaptive Channel Equalizer (developed by Bernard Widrow) is perhaps the most commercially successful of all neural network applications to date. It is a single-neuron device used now in virtually all long-distance telephone systems to stabilize voice signals [3].

Klopf [13] has postulated that, "An intelligent system will have to build on a foundation that amounts to a highly detailed, immense microscopic knowledge base, a knowledge base that can be interfaced effectively with higher functional levels." From this perspective, a neural substrate could develop into the microscopic knowledge base. The macroscopic capabilities of intelligence could then be built on top of this. Given the limitations of current neural networks, a plausible scheme is to incorporate capabilities previously found on the macroscopic, network level into the microscopic, neuronal (single-neuron) level.

In this connection, we introduce cognitive single-neuron models that coincide with existing animal learning theory. Each proposed model provides a basis for understanding and explaining Pavlovian conditioning [25][20] and instrumental conditioning [20], respectively, which are the best understood animal learning processes. In particular, one model, an associative critic neuron, captures the predictive nature of Pavlovian conditioning, which is essential to the theory of adaptive / learning systems. Another model, an associative learning neu-

ron, possesses the associative nature of instrumental conditioning, which stores in memory the temporal relationship between input and output.

C. Outline

The problem of learning via credit assignment [4] is described in Section II. The statement of the pole-balancing problem follows. This problem may be viewed as a canonical example of dynamic control. Some concepts from earlier related work are given in Section III. They serve as a basis for comparison of previous and proposed approaches. The proposed intelligent system is presented in Section IV. Here, a fuzzy logic-based controller is introduced, and a learning system with cognitive neural models is proposed. Computer simulation results are described in Section V. The paper closes with a concluding remark in Section VI.

II. A CASE STUDY: THE POLE BALANCING PROBLEM

In machine learning, the problem of learning to control physical dynamical systems has been, and remains, a challenging goal. In this context, the credit-assignment problem is often encountered in adaptive problem-solving systems, and is especially acute when evaluative feedback is delayed or infrequent. Basically, the credit-assignment problem, is to determine a strategy for assigning positive credit (reward) to desirable actions and negative credit (punishment) to undesirable actions in a way that would lead to the achievement of a specific goal. In what follows, we describe an approach to the building of an intelligent rule-based system that can learn to control a dynamical system without prior knowledge of its input-output relations.

Our approach focuses on a paradigmatic control problem - the pole-balancing problem - which has been the object of several studies in the literatures of control and neural networks. The pole balancing system is described as follows. A rigid pole is hinged to a cart, which is free to move on a one-dimensional track. The pole can rotate in the vertical plane of the track and the controller can apply an impulsive force of bounded magnitude to the cart at discrete time intervals. By balancing the pole, we mean that the pole never deviates by more than, say, 12 degrees, from the vertical. The equations of motion of the cart-pole system are not known to the controller, which implies that the cart-pole system is treated as a black box. What is known is a vector describing the cart-pole system's state at every time step. If the pole falls, it receives a failure signal. After a failure signal has been received, the system is reset to its initial state and a new attempt is made. On the basis of this evaluative feedback, the controller must develop its own control strategy and learn to balance the pole for as long as possible. Since a failure signal usually occurs only after a long sequence of individual control decisions, the sparsity of this signal makes the credit-assignment problem nontrivial.

III. PREVIOUS RELATED WORK

There are two noteworthy previous studies which have addressed the pole-balancing problem. The first is that of Michie and Chambers [21] in 1968. They constructed a program called BOXES that learned to balance the pole by applying two opposite constant forces. The second study is that of Barto, Sutton, and Anderson [1] in 1983, which used neuronlike adaptive elements to solve the same problem by using two constant forces. In general, both approaches can handle the credit-assignment problem that we mentioned. In both, the state space is partitioned into several non-overlapping regions and no symbolic reasoning techniques are employed. Both are limited to only two control actions: pushing the cart left or right with a force of fixed magnitude. The problem is thus one of bang-bang control.

In contrast to these approaches, we attempt to solve the problem through the use of symbolic problem-solving techniques, employing a fuzzy rule-based controller with approximate reasoning. Furthermore, a continuous control scheme is employed, namely, the controller can apply a force with a magnitude within [-10,+10] newtons. In this way, better performance of the controlled system may be achieved but the complexity of the problem is increased substantially. An overlapping partition of the state space forms a linguistic space. The overlapping partition enhances the speed of learning and robustness. We will have more to say about these issues at a later point.

IV. THE INTELLIGENT CONTROL SYSTEM

Experience shows that a fuzzy logic-based system using approximate reasoning [16][17] make it possible to emulate and even surpass the decision-making ability of a skilled human operator. And, neural network theory [3] provide a new computational structure, a plausible approach for infor-

mation processing because of its adaptivity / learning as well as massive parallelism. In this connection, We developed an intelligent control scheme by integrating human decision-making and animal learning behavior employing fuzzy logic and neural network theory.

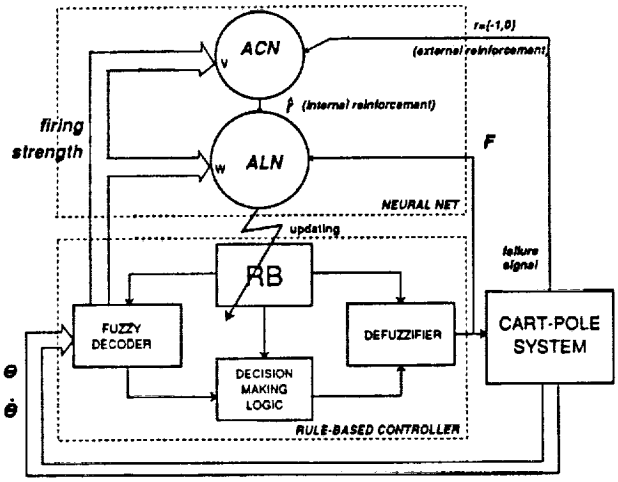


Fig. 1. Schematic representation of the intelligent system.

As shown in Figure 1, one key component of the intelligent system is a fuzzy logic-based controller which emulates human decision-making behavior. Another key component is a neural net. The net is composed by two cognitive neural models, an associative critic neuron (ACN) and an associative learning neuron (ALN), derived from animal learning theory, which stimulate memory association and learning behavior.

As a key component of the intelligent controller, the fuzzy logic-based system provides a linguistic description of control strategy. It is composed by a rule base, a fuzzy decoder, decision-making logic, and a defuzzifier. In general, the rule base describes control strategy which has the form of a collection of fuzzy control rules. For example, *if the angle of the pole is positive large and the angular velocity is positive large, then the applied force is positive large.* These are implemented and manipulated using fuzzy set theory [34] and are to be learnt by the proposed neural net. The fuzzy decoder inspects the incoming system state and fires the rules in parallel. A set of firing strength (x_i) is then generated and serves as input for the decision-making logic and neural net. The decision-making logic, the inference engine of the system, emulates human decision-making behavior based on the principles of approximate reasoning [35]. The defuzzifier takes a fuzzy control decision from the decision-making logic and determines a non-fuzzy control action (F).

The learning capability of the intelligent systems is provided by the associative critic neuron (ACN) and associative learning neuron (ALN). More specifically, the ACN is derived by using Pavlovian conditioning theory [25][20]. It captures the predictive nature of Pavlovian conditioning and has to do with criticism ($\hat{r}$) from the environment (r) associated with the system state (x_i). The ACN derives from the instrumental conditioning theory [20]. It is an associative memory system, which remembers the temporal relationships between input (x_i) and output (F), and associates each fuzzy control rule with an appropriate fuzzy control action (F_i).

A. Fuzzy Logic Control

In recent years, rule-based controllers employing approximate reasoning have emerged as one of the most active areas of research in the applications of fuzzy set theory. Such reasoning [35] plays an essential role in the remarkable human ability to make rational decisions in an environment of uncertainty and imprecision. In essence, approximate reasoning is the process or processes by which a possibly imprecise conclusion is deduced from a collection of imprecise premises. By employing the techniques of fuzzy set theory [34], approximate reasoning (with precise reasoning viewed as a limiting case) can be studied in a formal way.

The concept of a fuzzy set may be viewed as an extension of an ordinary (crisp) set. In a fuzzy set, an element can be a member of the set with a degree of membership varying between 0 and 1. Thus, a fuzzy set F in a universe $U = \{u_i, i=1, ..., n\}$ is defined by its membership function $\mu_F : U \rightarrow [0,1]$. If the $\mu_F(u_i)$ are 0 or 1, the fuzzy set is an ordinary set. As a special case, a fuzzy singleton is a fuzzy set containing just one element with degree 1.

A concept which plays an important role in the applications of the theory of fuzzy sets is that of a *linguistic variables*. To illustrate, if *speed* is interpreted as a linguistic variable, that is, a variable whose values are linguistic labels of fuzzy sets, then the values of *speed* might be

$$T(speed) = \{slow, moderate, fast,$$

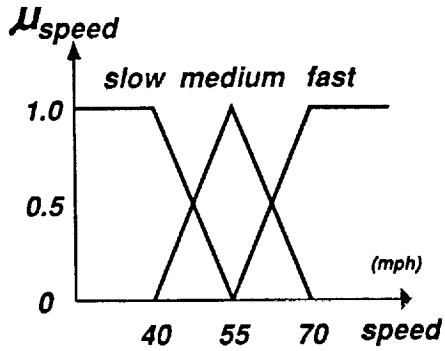


Fig. 2. Diagrammatic representation of various linguistic values of speed.

very slow, more or less fast, $\cdots$ }.

In a particular context, *slow* may be interpreted as, say, "a speed below about 40 mph", *moderate* as "a speed close to 55 mph" and *fast* as "a speed above about 70 mph". Figure 2 shows this interpretation in the framework of fuzzy sets.

The set-theoretic operations on fuzzy sets are defined via their membership functions. More specifically, let A and B be two fuzzy sets in U with membership functions μ_A and μ_B, respectively. The membership function $\mu_{A\cup B}$ of the *union* $A\cup B$ is defined pointwise for all $u\in U$ by

$$\mu_{A\cup B}(u) = \max\{\mu_A(u),\mu_B(u)\}.$$

Dually, the membership function $\mu_{A\cap B}$ of the *intersection* $A\cap B$ is defined pointwise for all $u\in U$ by

$$\mu_{A\cap B}(u) = \min\{\mu_A(u),\mu_B(u)\}.$$

If $A_1, \ldots, A_n$ are fuzzy sets in $U_1, \ldots, U_n$, respectively, the *Cartesian product* of $A_1, \ldots, A_n$ is a fuzzy set in the product space $U_1\times\cdots\times U_n$ with the membership function

$$\mu_{A_1\times\cdots\times A_n}(u_1,u_2,\cdots,u_n) = \min\{\mu_{A_1}(u_1),\cdots,\mu_{A_n}(u_n)\}.$$

Assume that the fuzzy sets A, A', B, and B' are the linguistic values of x and y. An example of approximate reasoning involving x and y is the following:

premise 1 : x *is* A',

premise 2 : *if* x *is* A *then* y *is* B,

consequent: y *is* B'.

For instance:

premise 1 : *the speed of a car is very high*,

premise 2 : *if the speed of a car is high then the probability of an accident is high*,

consequent: *the probability of an accident is very high.*

This type of fuzzy inference is based on the compositional rule of inference for approximate reasoning suggested by Zadeh [35].

A rule-based controller consists of a set of fuzzy control rules which are processed through the use of approximate reasoning. For simplicity, suppose that we have the two rules:

R_1 : *if* x *is* A_1 *and* y *is* B_1 *then* z *is* C_1,

or

R_2 : *if* x *is* A_2 *and* y *is* B_2 *then* z *is* C_2.

Approximate reasoning, given (x is A') and (y is B'), computes the degree of partial match between the user-supplied facts and the knowledge rule base as follows.

The degrees of match of (A_i and A) and (B_i and B) are given respectively by

$$\alpha_i = \max_u \min\{\mu_{A_i}(u), \mu_A(u)\},$$

$$\beta_i = \max_v \min\{\mu_{B_i}(v), \mu_B(v)\}.$$

The firing strength of the i^{th} rule is given by

$$x_i = \min\{\alpha_i, \beta_i\}.$$

Hence, the i^{th} rule recommends a control decision as follows:

$$\mu_{C_i'}(w) = \min\{x_i, \mu_{C_i}(w)\}.$$

The consequences of multiple rules can be combined by a conflict-resolution process which treats the sentence connective *or* as a union operator. The combined consequence is then given by

$$\mu_C(w) = \max\{\mu_{C_1'}, \mu_{C_2'}\}.$$

The combination of consequences is illustrated in Figure 3.

In on-line processes, the states of a control system are essential to a control decision (action). The underlying data are usually obtained from sensors and are crisp. It may be necessary to convert these data into the form of fuzzy sets [16]. In practice, however, crisp data are frequently treated as

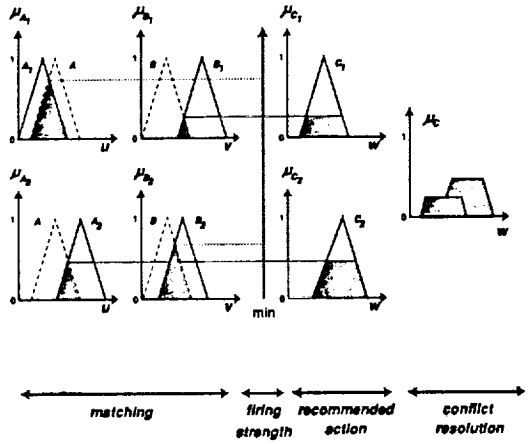


Fig. 3. Diagrammatic representation of approximate reasoning using fuzzy input.

fuzzy singletons. In this case, the corresponding inference mechanism is shown in Figure 4.

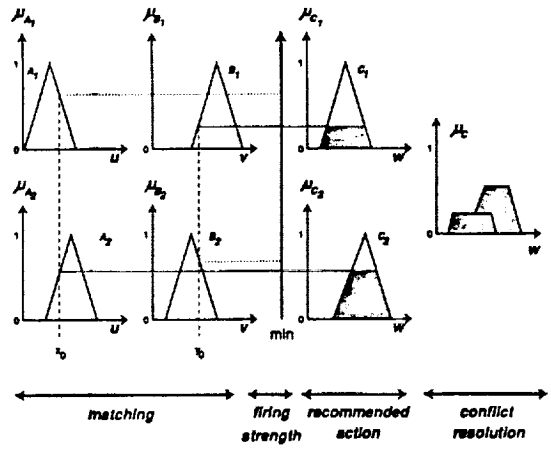


Fig. 4. Diagrammatic representation of approximate reasoning using crisp input.

Furthermore, in on-line control, the inference process should lead to a non-fuzzy control action. This necessitates the use of a *defuzzifier*. A defuzzifier can be implemented by using *max criterion, mean of maximum* or *center of area* algorithms [17]. The defuzzifier used here is employing the *center of area* algorithm.

In what follows, the fuzzy control rules are assumed to be of the form

$$R_i : \textit{if } x \textit{ is } A_i \textit{ and } y \textit{ is } B_i \textit{ then } z \textit{ is } C_i,$$
$$i = 1, 2, \ldots, n \ ,$$

where x, y, and z are linguistic variables representing the angle of the pole with respect to the vertical axis, angular velocity of the pole, and applied force, respectively; A_i, B_i, and C_i are the linguistic values (fuzzy sets) of the linguistic variables x, y, and z in their respective universes of discourse, [-12,+12] degrees, R, and [-10, +10] newtons. The definitions of linguistic values A_i and B_i are shown in Figure 5 (a) and (b). The problem is to learn the linguistic values C_i, which take the form of triangles, defined on the control force universe [-10,+10] newtons. The conception of fuzzification is performed as shown in Figure 5 (c). The location of the vertex of such a triangle is to be learned, while the coordinates of the base are functions of the vertex location value, say in the extreme case, +/-2 newtons away from that vertex.

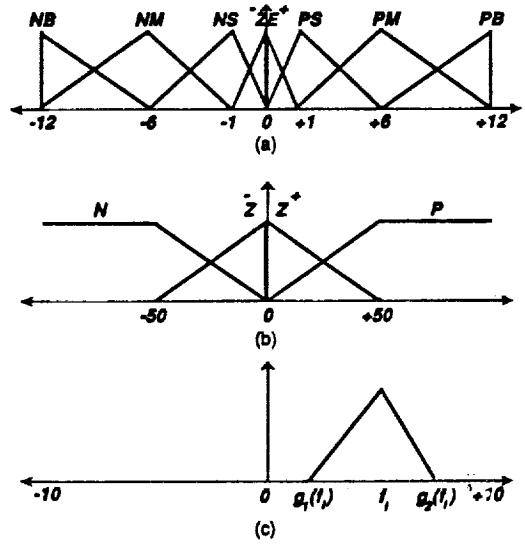


Fig. 5. (a) Linguistic values of angle, (b) angular velocity, and (c) applied force.

To summarize the ideas thus far discussed, the conception of a 2-D linguistic state space is formed. The x axis is θ with seven linguistic values; the y axis is $\dot{\theta}$ with three linguistic values. Thus, 8×4 fuzzy control rules are involved. Each fuzzy control rule corresponds to a fuzzy cell. The premise of a fuzzy control rule determines the cell's coordinates in the linguistic state space. The consequent of the rule is taken to be the content of the cell, which is to be learned by the proposed neurons, the ALN and ACN. Once a system input is sensed, the cells are fired in parallel. The fuzzy decoder takes the current state of the cart-pole system as an input and

has n outputs (firing strengths) going to the ALN and ACN. Each output of the fuzzy decoder corresponds to a fuzzy cell. The activity of the output is the firing strength. The firing strength serves as an input to both the ALN and ACN, and is also used to compute the recommended control action in each rule (cell).

B. Learning with a Neural Net

As has been mentioned in Section II, the principal difficulty in the learning process is that the training information (failure signal) is very sparse. Many of the previously employed neural networks such as the Adaline, perceptrons, and Hopfield nets, are effective for the solution of supervised pattern classification problems. In contrast, our network consists of the ACN and ALN which perform unsupervised learning. The ACN has to do with the criticism from the environment associated with the system state. The ALN takes the criticism and associates n fuzzy control actions with n fuzzy cells (the consequents of n fuzzy control rules). Since the ACN predicts the criticism at every time step, the ALN can continuously update itself before the failure signal occurs. This is the basis for the solution of the credit-assignment problem.

1. ACN

The ACN is derived from Pavlovian conditioning theory [25][20]. The best known example of Pavlovian conditioning comes from Pavlov's research on the conditioned reflex of salivation by dogs. Prior to conditioning, when a dog hears the sound of a bell, it pricks its ears. Then, when the food is presented to it, it salivates. If this sequence of events is repeated, the dog soon starts to salivate in reaction to the sound of the bell. In effect, the dog has been "conditioned" to react to the bell. As can be seen, the sound of a bell can be used to predict the occurrence of salivation before the presence of food. This predictive relationship between food and the sound of a bell has important implications. Thus, the ACN captures this predictive nature of the Pavlovian conditioning.

The correspondence between Pavlovian conditioning and the behavior of our system is as follows. Food corresponds to the evaluative feedback (failure signal). The salivation by reflex is equivalent to an *external reinforcement r(t)* with the value -1.0 if failure signal occurs, otherwise 0.0. The sound of a bell relates to the i^{th} fired fuzzy cell (fuzzy control rule) with firing strength x_i. The salivation resulting from the bell's sound is the *predictive reinforcement* $v_i(t)$ of the i^{th} fuzzy cell. It is worth noting that, in the extreme, the i^{th} rule with firing strength either 1.0 or 0.0 is the exact case of presence or absence of a bell's sound in the conditioning of a Pavlov dog. In other words, our ACN operates in a continuous mode, which treats Pavlovian conditioning as a special case. In effect, the ACN attempts to predict the reinforcement $v_i(t)$ that can eventually be obtained from the environment by choosing a control action for that fuzzy cell.

As an extension of single-input/single-output analogy, multiple inputs in the ACN necessitate an output which is a weighted sum of the predictive reinforcements of all fired fuzzy cells. The weighted sum *p(t)* is the *total reinforcement* of all fired fuzzy cells at time t. Furthermore, an internal reinforcement $\hat{r}(t)$, the criticism, is generated as a temporal difference of the total predictive reinforcements.

As shown in Figure 1, the ACN has an external reinforcement input, $r(t)$, from the cart-pole system, n inputs, $x_i(t)$, $i=1, ..., n$, from corresponding fuzzy cells, and an output, $\hat{r}(t)$, as internal reinforcement signal (criticism) for the ALN and itself. The total reinforcement at time t is given by

$$p(t) = G(\sum_{i=1}^{n} v_i(t)x_i(t)),$$

where G could be a sigmoid-shaped function, identity function, mean of maximum algorithm or center of area algorithm. The associative learning rule for the i^{th} fuzzy cell is in part characterized by a local memory trace $\bar{x}_i(t)$ and the internal reinforcement $\hat{r}(t)$. The predictive reinforcement $v_i(t)$ of the i^{th} fuzzy cell (fuzzy control rule, fuzzy system state) is updated by

$$v_i(t+1) = v_i(t) + \beta\hat{r}(t)\bar{x}_i(t),$$

where β is a positive learning-rate parameter. The local memory trace is defined by

$$\bar{x}_i(t+1) = \lambda\bar{x}_i(t) + (1-\lambda)|x_i(t)v_i(t)|,$$

where λ, $0 \le \lambda < 1$, is a trace-delay parameter. The trace takes the form of an exponential curve. It is strengthened by the degree of firing strength of the i^{th} fuzzy cell (fuzzy control rule) together with its current weight, and weakened if the rule is not fired. The trace thus keeps track of how long ago the i^{th} fuzzy control rule fired and also how often it was

fired. The internal reinforcement is calculated as

$$\hat{r}(t) = r(t) + \gamma p(t) - p(t-1),$$

where γ, $0 \le \gamma < 1$, is a discount-rate parameter. The internal reinforcement serves as criticism, depending on a relative difference of $p(t)$ and $p(t-1)$. If the pole does not fall and $\gamma p(t) > p(t-1)$, then $r(t)=0$ and $\hat{r}(t)>0$, a reward is given. If the pole does not fall and $\gamma p(t) < p(t-1)$, then $r(t)=0$ and $\hat{r}(t)<0$, and a punishment is effected. The discount factor γ implies a bias for the condition in which $p(t)$ equals $p(t-1)$. More specifically, once the pole does not fall and keeps in the same state, a reward is given through the use of a discount factor. On the other hand, if the pole falls, then $p(t)=0$, $r(t)=-1$ and $\hat{r}(t)<0$, and a punishment is issued. If $p(t-1)$ fully predicts the occurrence of the failure, there is no punishment. As shown, a negative feedback mechanism is implicitly incorporated into the internal reinforcement.

The proposed ACN model might be viewed as an extension of the Sutton-Barto model [18]. More specifically, in the context of animal learning phenomena, a sigmoid-shaped acquisition curve is observed. This is not simulated in the Sutton-Barto model. In our model, it can be achieved by making a change in the associative strength proportional to the current associative strength [18]. It has been demonstrated by computer simulation that the ACN accounts for many phenomena observed in Pavlovian conditioning, such as a sigmoid-shaped acquisition curve, inter-stimulus interval effects, trace conditioning, and delay conditioning. A more detailed discussion of this aspect of our model is described elsewhere [18].

2. ALN

The ALN is derived from the instrumental conditioning theory [20]. A simple example is teaching a dog to perform a trick. During training, if the dog does well, it is given a reward. If not, it is punished. After training, the dog has learned a new trick. The association of the dog's response and reinforcement has in effect been "conditioned". The correspondence between this conditioning and the ALN is as follows. A dog corresponds to the i^{th} fuzzy control rule with firing strength x_i. The response of the dog relates to the control force $(w_i f_i)$ of the i^{th} rule. The reinforcement as reward/punishment is equivalent to the internal reinforcement from the ACN. The ALN does the following: the i^{th} fuzzy control rule can produce correct control force of the i^{th} rule under the internal reinforcement from the ACN. In effect, the ALN is a content-addressable memory system which associates each fuzzy control rule with an appropriate fuzzy control action.

As shown in Figure 1, the ALN has an internal reinforcement input, $\hat{r}(t)$, from the ACN, n inputs, $x_i(t)$, $i=1, ..., n$, from the fuzzy decoder, a control action input, $F(t)$, from the defuzzifer, and n associative weights w_i $i=1, ..., n$, as outputs for the rule base. Each associative weight $w_i(t)$ is transformed -- by using the concepts of dynamical normalization and fuzzification -- into a fuzzy set having the form of a triangle as described in the previous section. Symbolically,

$$F_i(t) = fuzzifier\ (f_i(t)),$$

where $f_i(t)$ is the location of the vertex of the triangle. It is given by

$$f_i(t) = H(w_i(t) + noise(t)), \quad i = 1, ..., n,$$

where H is a dynamic sigmoid function which may be viewed as a dynamic normalization function and provides a continuous output within the range [-10,+10]. For the purpose of computer simulation, the following function is used:

$$H(x,t) = \begin{cases} \dfrac{10x}{T(t)+x} & x>0, \\ \varepsilon & x=0, \\ \dfrac{10x}{T(t)-x} & x<0. \end{cases}$$

where $T(t) = k_1 \max_i |w_i(t)|$ is an offset-tuning parameter which determines the slope of the sigmoid-shaped curve; and k_1 is a constant. The associative learning rule for each $w_i(t)$ is

$$w_i(t+1) = w_i(t) + \bar{\alpha}(t)\hat{r}(t)e_i(t),$$

$$\bar{\alpha}(t) = \frac{\alpha k_2}{k_2 + t},$$

where $\bar{\alpha}$ is a dynamical positive learning-rate parameter with a initial value α and k_2 is a weight-freeze parameter. The weight-freeze parameter determines the decreasing rate of the dynamical learning rate $\bar{\alpha}$. $\hat{r}(t)$ is the criticism from the ACN. The associativity trace, $e_i(t)$, is given by

$$e_i(t+1) = \delta e_i(t) + (1-\delta)F(t)x_i(t),$$

where δ, $0 \le \delta < 1$, is another trace-decay parameter. The associativity trace takes the form of an

exponential and it remembers for how long and how often a fuzzy control rule has fired as well as what control action was taken at that time.

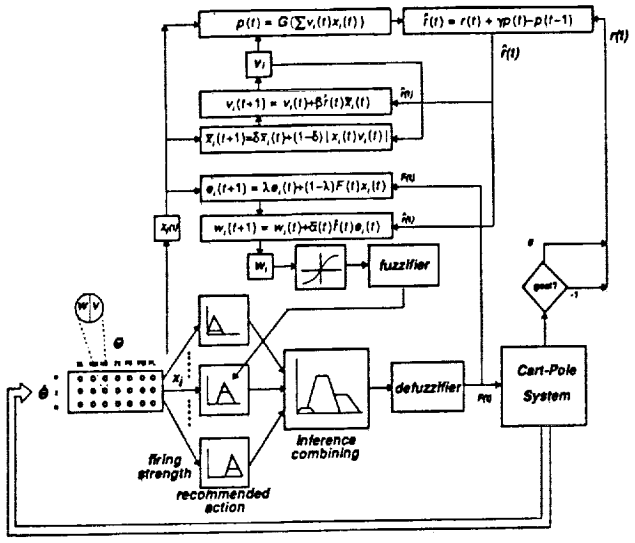


Fig. 6. Signal flow of the intelligent control system.

Figure 6 illustrates the signal flow of our proposed controller during a learning process. In principle, once a system state is sensed, the set of fuzzy control rules is fired in parallel. A set of firing strengths (x_i) is then generated and serves as input to both ALN and ACN. The information about the system state is then fed into the two neuronlike elements by the set of firing strengths. The firing strength together with the predictive reinforcement (v_i, or desirability) of the i^{th} fuzzy rule generates the local memory trace ($\overline{x_i}$, desirability trace) of the i^{th} fuzzy rule. The total reinforcement, p, or equivalently, the desirability of all fired fuzzy cells, is computed based on the firing strength and the reinforcement (desirability) of each fuzzy rule. A non-fuzzy control action, F, is determined after the processes of inference combining and defuzzification. The control action, F, together with the firing strength, x_i, of each rule contributes the associativity trace, e_i, of each rule. After applying the control action to the plant, a goal evaluation, r, is made, which takes binary values. Based on the yes-no evaluation, the criticism, $\hat{r}$, which is a more informative evaluation, is generated. It plays an important role in the solution of the credit-assignment problem. The weights (v_i, w_i) in learning rules are thus updated on the basis of the criticism and their own local memory trace, ($\overline{x_i}$, e_i). A fuzzy control force in each rule is generated from the w_i by the use of dynamic normalization and fuzzification.

V. SIMULATION RESULTS

We implemented our system on a Sun workstation. For comparison purposes, we also implemented Barto's system [1] for solving the same problem. The mass of the cart and initial pole were 1.0 kg and 0.1 kg, respectively. The length of the pole was 1.0 meter. The parameter values used in our simulation were: α=1000, β=0.5, γ=0.95, δ=0.9, λ=0.8, ε=0.1, k_1=0.15, and k_2=2500. A run was called "success" whenever the number of steps before failure was greater than 60,000. The external reinforcement $r(t)$ was -1 when the failure signal occurred, otherwise, it was 0. Every trial began with the same initial cart-pole states, θ=0, $\dot{\theta}$=0, x=0, $\dot{x}$=0, and ended with a failure signal when $|\theta|$>12 degrees. All memory traces, x_i and e_i, were set to zero. All the weights, w_i; were set to zero, and a lower bound v_i (=-0.0001) was set to all the weights. In testing the performance of the system, the simulator was run by applying the Adams predictor-corrector method with a time step of 20 ms [19].

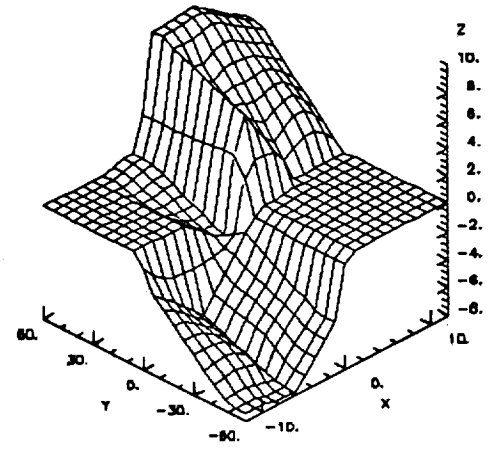


Fig. 7. Learned control surface based on the proposed intelligent system with COA defuzzifier.

A. Learning / Training

The proposed controller and Barto's system are capable of learning to balance the pole. However, experiments show that our system has a better learning performance [19]. The proposed controller learns to balance the pole by 6 trials with COA

defuzzifier. Figure 7 illustrates the learned control surfaces based on our intelligent system employing defuzzifier *center of area*(COA). The performance of Barto system, in average, took 27 trials to balance the pole [19].

Additional observations were made on the state trajectory of the angle of the pole with respect to the vertical axis. We observed the data after the systems learned their own control strategy. The data showed that, in every case, our controller could keep the angle within a smaller region compared with Barto's. Figure 8 illustrates one set of these data from our system and Barto's, respectively.

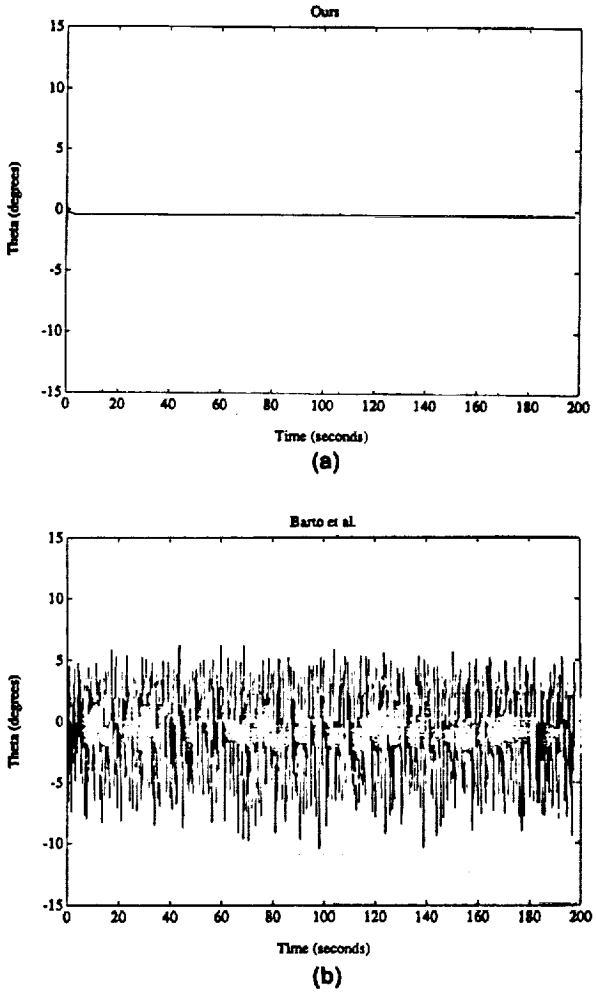


Fig. 8. (a) State performance of the pole angle based on the proposed controller. (b) State performance of the pole angle based on Barto's system.

B. Adaptation

Adaptation is intended to adjust to unforeseen changes in environmental conditions using prior knowledge. Training involves constructing a knowledge base of an application domain (e.g. a pole-balancing task) with little a priori domain knowledge. The capability of learning to solve new tasks by modifying previous learned knowledge (adaptation) is compared with that of starting from scratch (training). Extensive simulation studies of such schemes have been carried out. They show that the proposed controller tolerates a wide range of uncertainty as well as a lack of system information, e.g., parameter changes in the length and mass of the pole, changes of failure criteria, and a slanted cart-pole system.

The adaptation experiments were based on pre-learned knowledge by employing the same parameter settings as that in the last section. The length and mass of the pole were 0.1kg and 1.0m, the angle constraint for failure evaluation was -/+12°, and the initial value of the angle of the pole with respect to the vertical axis is 0.0°. The system took 6 trials to learn the task.

In the first set of experiments, the system is required to adapt to changes in the length and mass of the pole. Six experiments were performed. The first two were to increase the original mass of the pole by a factor of 10 and 20, respectively. The third and fourth ones were to change the original length of the pole by a factor of 2 and 1/2, respectively. The last two were to replace the original pole by two shorter poles. The length and mass of the first pole were reduced to 2/3 of the original values, while the second one is 1/4. Without pre-training, the system took 10, 15, 5, 11, 8 and 6 trials to learn these tasks. However, with the pre-trained knowledge, the system successfully completed these tasks without any further trials. The result shows the robustness of the proposed intelligent system.

In the second set, we added a more severe constraint on the angle of the pole for failure evaluation. The angle constraints were changed from +/-12° to +/-6°, +/-3°, and to +/-1°, respectively. The system needed 4 and 6 trials to learn the first two tasks with no initial knowledge, but it failed in the last task since a finer partition of input space is required. While with pre-training, the system adapted to all tasks without further trials.

In the third set, the system was required to adapt to the changes in the length and mass of the

pole (by a factor of 1/2) and angle constraint (+/−3°). The training took 6 trials, while adaptation can handle the new task well.

Finally, the cart-pole system was lifted at the right end in such a way that the base of the system and the surface of the table formed an angle of 12°. The system took 10 trials to balance the pole. However, the system with the trained knowledge needed no further trials to complete the new task.

VI. CONCLUDING REMARK

In this article, we have proposed a symbolic problem-solving approach to a class of learning control problems. More specifically, we have attempted to develop an intelligent control scheme by integrating human decision-making with a fuzzy logic-based system and animal learning behavior with cognitive neural models. The proposed intelligent control system learns and improves its rule base for better control strategy from experience and adapts to changes in an environment of uncertainty and imprecision. In this way, we avoid an ad-hoc rule-tuning process which is usually inefficient and lacking in consistency. It has been shown that the proposed intelligent system has a better performance of learning speed and system behavior in relation to previous approaches. Furthermore, the system is quite robust. The controller is relatively insensitive to variations in the parameters of the system environment, e.g., in the context of pole-balancing, changes in the length and mass of the pole, failure criteria, and slanting the base of the cart-pole system. In addition, the controller can be primed with pre-trained control knowledge which minimizes rapid changes during adaptation.

The approach described in this paper may be viewed as a step in the development of a better understanding of how to combine a fuzzy logic based system with a neural network to achieve a significant learning capability. We plan to address various aspects of this important issue in subsequent papers.

ACKNOWLEDGMENT

I am greatly indebted to Professor Lotfi A. Zadeh of the University of California, Berkeley for his encouragement of this research. The assistance of Professor Zadeh is gratefully acknowledged.

REFERENCES

1. A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems", *IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-13, no. 5*, pp. 834-846, 1983.
2. J. A. Bernard, "Use of a rule-based system for process control", *IEEE Control Systems Magazine, vol. 8, no. 5*, pp. 3-13, Oct. 1988.
3. *DARPA Neural Network Study*, Fairfax: AFCEA International Press, Dec. 1988.
4. Fujitec, "FLEX-8800 series elevator group control system," *Fujitec Co., Ltd*, Osaka, Japan, 1988.
5. Fujitsu, "Fujitsu Announces Creation of Human-Quality Robot Eye," *The Japan Times* Jan. 10, 1990.
6. S. Grossberg, "Some networks that can learn, remember, and reproduce any number of complicated space-time patterns," *Journal of Mathematics and Mechanics vol. 19*, pp. 53-91, 1969.
7. S. Grossberg, "Classical and instrumental learning by neural networks," in *Progress in Theoretical Biology vol. 3*, R. Rosen and F. Snell, Eds., New York: Academic Press, pp. 51-141, 1974.
8. S. Grossberg and N. A. Schmajuk, "Neural Dynamics of attentionally-modulated Pavlovian conditioning: Conditioned reinforcement, inhibition, and opponent processing," *Psychobiology*, pp. 195-240, 1987.
9. D. O. Hebb, *The Organization of Behavior*, New York: Wiley, 1949.
10. Y. Kasai and Y. Morimoto, "Electronically controlled continuously variable transmission," *Proc. Int. Congress on Transportation Electronics*, Dearborn, Michigan, Oct. 1988.
11. M. Kinoshita, and T. Fukuzaki, T. Satoh, and M. Miyake, "An automatic operation method for control rods in BWR plants," *Proc. Specialists' Meeting on In-core Instrumentation and Reactor Core Assessment*, Cadarache, France, 1988.
12. A. H. Klopf, "Brain function and adaptive systems - A heterostatic theory," *Aire Force Cambridge Research Laboratories Special Report AFCRL-72-0164*, 1972.
13. A. H. Klopf, *The Hedonistic Neuron*, Washington: Hemisphere, 1982.
14. A. H. Klopf, "A drive-reinforcement model of single neuron function: An alternative to the Hebbian neuronal model," in *Neural Networks*

for Computing, Snowbird, Ut., 1986 J. S. Denker, Ed., New York: American Institute of Physics, 1986, pp. 265-270.

15. A. H. Klopf, "Drive-reinforcement learning: A real time learning mechanism for unsupervised learning," in *IEEE 1st International Conference on Neural Networks*, San Diego, June, 1987.
16. C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller - Part I," *IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-20, no. 2*, 1990.
17. C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller - Part II," *IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-20, no. 2*, 1990.
18. C. C. Lee, "Modeling behavioral substrates of associative learning and memory: adaptive neural models," *IEEE Trans. on Systems, Man, and Cybernetics*, to appear.
19. C. C. Lee, "A Self-Learning Rule-Based Controller Employing Approximate Reasoning and Neural Network Concepts," *International Journal of Intelligent Systems, vol. 5, no. 3*, 1990.
20. N. J. Mackintosh, *The Psychology of Animal Learning*, New York: Academic Press, 1974.
21. D. Michie and R. A. Chambers, "Boxes: an experiment in adaptive control," *in Machine Intelligence 2*, E. Dale and D. Michie, Eds., Oliver and Boyd, Edinburgh, 1968, pp. 137-152.
22. Mitsubishi, "World's First Fuzzy Logic Controlled Commercial Air Conditioner Begins Volume Shipping," *NEWS - Togai InfraLogic Inc.* Dec. 19, 1989.
23. Nissan, "New auto systems use fuzzy logic," *New York Times*, July 1, 1989.
24. Nissan, "Nissan Patents 'Fuzzy Logic' ABS, Gearbox," *Automotive Electronic News*, July 31, 1989.
25. I. P. Pavlov, *Conditioned reflexes*, London: Oxford University Press, 1927.
26. J. M. Pearce and G. Hall, "A model for Pavlovian learning: variations in the effectiveness of conditioned but not of unconditioned stimuli," *Psychological Review vol. 87, no. 6*, pp. 532-552, 1980.
27. J. M. Pearce, H. Kaye, and G. Hall, "Predictive accuracy and stimulus associability: Development of a Model for Pavlovian learning," in *Quantitative Analyses of Behavior III*, M. L. Commons, R. J. Herrnstein, and A. R. Wagner, Eds., New York: Harper and Row, Publishers, pp. 241-255, 1982.
28. R. A. Rescorla and A. R. Wagner, "A theory of Pavlovian conditioning: variations in the effectiveness of reinforcement and nonreinforcement," in *Classical Conditioning II: Current Research and Theory*, A. H. Black and W. F. Prokasy, Eds., New York: Appleton-Century-Crofts, pp. 64-99, 1972.
29. R. S. Sutton and A. G. Barto, "Toward a modern theory of adaptive networks: expectation and prediction," *Psychological Review vol. 88, no. 2*, pp. 135-170, 1981.
30. Toshiba, "'Command-AI' Elevator Systems," *Nikkei Industry News*, Sept. 2, 1989.
31. A. R. Wagner, "Expectancies and the priming of STM," in *Cognitive Processes in Animal Behavior* S. H. Hulse, H. Fowle, and W. K. Honig, Eds., New York: Halsted Press, pp. 177-209, 1978.
32. S. Yasunobu and G. Hasegawa, "Evaluation of an automatic container crane operation system based on predictive fuzzy control," *Control Theory and Advanced Technology, vol. 2, no. 3*, pp. 419-432, 1986.
33. S. Yasunobu and S. Miyamoto, "Automatic train operation by predictive fuzzy control," in *Industrial Applications of Fuzzy Control*, M. Sugeno, ed., Amsterdam: North Holland, 1985.
34. L. A. Zadeh, "Fuzzy sets," *Information and Control, vol. 8*, pp. 338-353, 1965.
35. L. A. Zadeh, "Outline of a new approach to the analysis complex systems and decision processes," *IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-3*, pp. 28-44, 1973.

An Neural Network Based Fuzzy Logic Controller

(Paper not provided by publication date.)

FUZZY GEOMETRY, ENTROPY AND IMAGE INFORMATION

Sankar K. Pal*
Software Technology Branch/PT4
National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas 77058, U.S.A.

INTRODUCTION

A gray tone picture possesses some ambiguity within the pixels due to the possible multivalued levels of brightness. The incertitude in an image may arise from grayness ambiguity or spatial (geometrical) ambiguity or both. Grayness ambiguity means "indefiniteness" in deciding a pixel as white or black. Spatial ambiguity refers to "indefiniteness" in shape and geometry of a region e.g., where is the boundary or edge of a region? or is this contour "sharp"?

When the regions in a image are ill-defined (fuzzy), it is natural and also appropriate to avoid committing ourselves to a specific (hard) decision e.g., segmentation/thresholding and skeletonization by allowing the segments or skeletons or contours, to be fuzzy subsets of the image. Similarly, for describing and interpreting ill-defined structural information in a pattern (when the pattern indeterminary is due to inherent vagueness rather than randomness), it is natural to define primitives and relation among them using labels of fuzzy set.

The present article provides various uncertainty measures arising from grayness ambiguity and spatial ambiguity in an image, and their possible applications as image information measures. The first part of the article consists of definitions of an image in the light of fuzzy set theory, and of information measures (arising from fuzziness) and tools relevant for processing/ analysis e.g., fuzzy geometrical properties, correlation, bound functions and entropy measures. The second part provides formulation of algorithms along with management of uncertainties (ambiguities) for segmentation and object extraction, and edge detection. The output obtained here is both fuzzy and nonfuzzy. Ambiguity in evaluation and assessment of membership function has also been described here.

* Dr. Pal is on leave from the post of Professor in the Electronics and Communication Sciences Unit, Indian Statistical Institute, Calcutta 700035, India.

IMAGE DEFINITION

An image X of size MxN and L levels can be considered as an array of fuzzy singletons, each having a value of membership denoting its degree of brightness relative to some brightness level ℓ, $\ell = 0, 1, 2, \ldots L-1$. In the notation of fuzzy sets, we may therefore write

$$X = \{\mu_X(x_{mn}) = \mu_{mn}/x_{mn};\ m = 1, 2 \ldots M;\ n = 1, 2, \ldots N\} \quad (1)$$

where $\mu_X(x_{mn})$ or μ_{mn}/x_{mn}, $(0 \le \mu_{mn} \le 1)$

denotes the grade of possessing some property μ_{mn} (e.g., brightness, edginess, smoothness) by the (m,n)th pixel intensity x_{mn}. In other words, a fuzzy subset of an image X is a mapping μ from X into [0, 1]. For any point $p \in X$, $\mu(p)$ is called the degree of membership of p in μ.

One may use either global or local information of an image in defining a membership function characterizing some property. For example, brightness or darkness property can be defined only in terms of gray value of a pixel x_{mn} whereas, edginess, darkness or textural property need the neighborhood information of a pixel to define their membership functions. Similarly, positional or co-ordinate information is necessary, in addition to gray level and neighborhood information to characterize a dynamic property of an image.

Again, the aforesaid information can be used in a number of ways (in their various functional forms), depending on individuals opinion and/or the problem to his hand, to define a requisite membership function for an image property.

MEASURES OF FUZZINESS AND IMAGE INFORMATION

The definitions of various entropy and other related measures which represent grayness ambiguity in an image (based on individual pixel as well as a collection of pixels) are listed below.

Linear Index of Fuzziness

$$\gamma_l(X) = (2/MN)\sum_m\sum_n |\mu_{mn} - \underset{\sim}{\mu}_{mn}| \quad (2)$$

$$= (2/MN)\sum_m \sum_n \min(\mu_{mn}, 1-\mu_{mn})$$

$m = 1, 2, \ldots M;\ n = 1, 2, \ldots N$

Quadratic Index of Fuzziness

$$\gamma_q(X) = \left(2/\sqrt{MN}\right)\left[\sum_m\sum_n \{\mu_{mn} - \underset{\sim}{\mu}_{mn}\}^2\right]^{0.5} \quad (3)$$

$m = 1, 2, \ldots M;\ n = 1, 2, \ldots N$

Entropy

$$H(X) = (1/MN \ln 2)\sum_{m}\sum_{n} S_n(\mu_{mn}) \tag{4}$$

with $S_n(\mu_{mn}) = -\mu_{mn} \ln \mu_{mn} - (1 - \mu_{mn}) \ln(1 - \mu_{mn})$

$m = 1, 2, \ldots M; n = 1, 2, \ldots N$

μ_{mn} denotes the degree of possessing some property μ by the (m, n)th pixel x_{mn}. $\underset{\sim}{\mu}_{mn}$ denotes the nearest two tone version of μ_{mn}

rth Order Entropy

$$H^r(X) = (-1/k)\sum_{i}\left\{\mu\left(s_i^\gamma\right)\log\left\{\mu\left(s_i^\gamma\right)\right\} + \left\{1-\mu\left(s_i^\gamma\right)\right\}\log\left\{1 - \mu\left(s_i^\gamma\right)\right\}\right\} \tag{5}$$

$i = 1, 2, \ldots k$

s_i^r denotes the ith combination (sequence) of r pixels in X. k is the number of such sequences. $\mu(s_i^r)$ denotes the degree to which the combination s_i^r, as a whole, possesses the property μ.

Hybrid Entropy

$$H_{hy}(X) = -P_w \log E_w - P_b \log E_b \tag{6}$$

with $E_w = (1/MN)\sum_{m}\sum_{n} \mu_{mn} \exp(1-\mu_{mn})$

$E_b = (1/MN)\sum_{m}\sum_{n}(1-\mu_{mn})\exp(\mu_{mn})$

$m = 1, 2, \ldots M; n = 1, 2, \ldots N$

μ_{mn} denotes the degree of "whiteness" of (m, n)th pixel. P_w and P_b denote probability of occurrences of white ($\mu_{mn} = 1$) and black ($\mu_{mn} = 0$) pixels respectively. E_w and E_b denote the average likeliness (possibility) of interpreting a pixel as white and black respectively.

Correlation

$$C(\mu_1, \mu_2) = 1 - 4\left[\sum_{m}\sum_{n}\{\mu_{1mn} - \mu_{2mn}\}^2\right] \Big/ (X_1 + X_2) \tag{7}$$

$C(\mu_1, \mu_2) = 1 \quad \text{if } X_1 + X_2 = 0$

with $X_1 = \sum_{m}\sum_{n}\{2\mu_{1mn} - 1\}^2$

and $X_2 = \sum_{m}\sum_{n}\{2\mu_{2mn} - 1\}^2$, $m = 1, 2, \ldots M; n = 1, 2, \ldots N$

$C(\mu_1, \mu_2)$ denotes the correlation between two properties μ_1 and μ_2 (defined over the same domain). μ_{1mn} and μ_{2mn} denote the degree of possessing the properties μ_1 and μ_2 respectively by the (m, n)th pixel.

These expressions (equations 2-7) are the versions extended to two dimensional image plane from those defined for a fuzzy set. For example, index of fuzziness was defined by Kaufmann [1], entropy by DeLuca and Termini [2], rth order entropy and hybrid entropy by Pal and Pal [3], and correlation by Murthy, Pal and Dutta Majumdar [4].

Interpretation

Let us describe the properties of these measures along with their relevance to image processing/ analysis problems. Index of fuzziness reflects the ambiguity present in an image by measuring the distance between its fuzzy property plane and the nearest ordinary plane. The term "entropy", on the other hand, uses Shannon's function in the property plane but its meaning is quite different from the one of classical entropy because no probabilistic concept is needed to define it. $H^r(X)$ gives a measure of the average amount of difficulty in taking a decision on any subset of size r with respect to an image property. If $r = 1$, $H^r(X)$ reduces to (unnormalized) H(X) of equation (4). Equation (5) is formulated based on the logarithmic behavior of gain function. Similar expression using exponential gain function [14] can also be defined. $H_{hy}(X)$ represents an amount of difficulty in deciding whether a pixel possesses certain properties or not by making a prevision on its probability of occurrence. In absence of fuzziness (i.e.,with proper defuzzification), H_{hy} reduces to two state classical entropy of Shannon, the states being black and white. Since a fuzzy set is a generalized version of an ordinary set, the entropy of a fuzzy set deserves to be a generalized version of classical entropy by taking into account not only the fuzziness of the set but also the underlying probability structure. In that respect, H_{hy} can be regarded as a generalized entropy such that classical entropy becomes its special case when fuzziness is properly removed.

All these terms, which give an idea of 'indefiniteness' or fuzziness of an image may be regarded as the measures of average intrinsic information which is received when one has to make a decision (as in pattern analysis) in order to classify the ensembles of patterns described by a fuzzy set.

$\gamma(X)$ and H(X) are normalized in the interval [0, 1] such that

$$\text{Pr 1: } \gamma_{min} = H_{min} = 0 \text{ for } \mu_{mn} = 0 \text{ for all } (m,n)(X) \tag{8a}$$

$$\text{Pr 2: } \gamma_{max} = H_{max} = 1 \text{ for } \mu_{mn} = 0.5 \text{ for all } (m,n) \tag{8b}$$

$$\text{Pr 3: } \gamma(X) \geq \gamma(X^*) (\text{or, } H(X) \geq H(X^*)) \tag{8c}$$

and

$$\text{Pr 4: } \gamma(X) = \gamma(\overline{X}) (\text{or, } H(X) \geq H(\overline{X})) \tag{8d}$$

where X* is the 'sharpened' or 'intensified' version of X such that

$$\mu_X^*(x_{mn}) \geq \mu_X(x_{mn}) \text{ if } \mu_X(x_{mn}) \geq 0.5$$
$$\text{and } \mu_{X^*}(x_{mn}) \leq \mu_X(x_{mn}) \text{ if } \mu_X(x_{mn}) \leq 0.5 \quad (9)$$

In other words, $\gamma(X)$ or H(X) increases monotonically with μ, reaches a maximum at $\mu = 0.5$ and then decreases monotonically. This is explained in Fig. 1. It is to be mentioned here that the definition of fuzzy entropy [14,16] based on exponential gain function also satisfies the aforesaid properties.

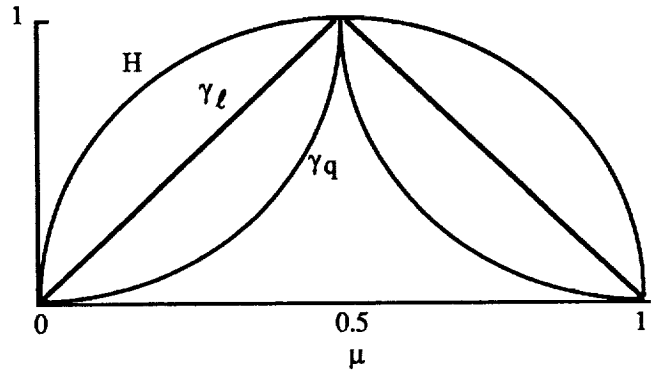


Figure 1 Variation of Fuzziness with μ.

According to property 8(c), these parameters decrease with contrast enhancement of an image. Now through processing, if we can partially remove the uncertainty on the grey levels of X, we say that we have obtained an average amount of information given by $\delta\gamma = \gamma(X) - \gamma(X^*)$ or $\delta H = H(X) - H(X^*)$ by taking a decision bright or dark on the pixels of X. The criteria $\gamma(X^*) \leq \gamma(X)$ and $H(X^*) \leq H(X)$, in order to have positive $\delta\gamma$ and δH-values, follow from Eq. (8c). If the uncertainty is completely removed, then $\gamma(X^*) = H(X^*) = 0$. In other words, $\gamma(X)$ and H(X) can be regarded as measures of the average amount of information (about the grey levels of pixels) which has been lost by transforming the classical pattern (two-tone) into a fuzzy pattern X.

It is to be noted that $\gamma(X)$ or H(X) reduces to zero as long as μ_{mn} is made 0 or 1 for all (m, n), no matter whether the resulting defuzzification (or transforming process) is correct or not. In the following discussion it will be clear how H_{hy}, takes care of this situation.

$H^r(X)$ has the following properties:

Pr 1: H^r attains a maximum if $\mu_i = 0.5$ for all i.

Pr 2: H^r attains a minimum if $\mu_i = 0$ or 1 for all i.

Pr 3: $H^r \geq H^{*r}$, where H^{*r} is the rth order entropy of a sharpened version

of the fuzzy set.

Pr 4: H^r is, in general, not equal to $\overline{H}^r$, where $\overline{H}^r$ is the rth order entropy of the complement set.

Pr 5: $H^r \leq H^{r+1}$ when all $\mu_i \in [0.5, 1]$.

$H^r \geq H^{r+1}$ when all $\mu_i \in [0, 0.5]$.

Note that the property P4 of equation 8(d) is not, in general, valid here. The additional property Pr 5 implies that H^r is a monotonically nonincreasing function of r for $\mu_i \in [0, 0.5]$ and a monotonically nondecreasing function of r for $\mu_i \in [0.5, 1]$ (when 'min' operator has been used to get the group membership value).

When all the μ_i values are same, $H^1(X) = H^2(X) = \ldots = H^r(X)$. This is because of the fact that the difficulty in taking a decision regarding possession of a property on an individual is same as that of a group selected therefrom. The value of H^r would, of course, be dependent on the μ_i values.

Again, the higher the similarity among singletons the quicker is the convergence to the limiting value of H^r. Based on this observation, let us define an index of similarity of supports of a fuzzy set as $S = H^1/H^2$ (when $H^2 = 0$, H^1 is also zero and S is taken as 1). Obviously, when $\mu_i \in [0.5, 1]$ and the min operator is used to assign the degree of possession of the property by a collection of supports, S will lie in [0, 1] as $H^r \leq H^{r+1}$. Similarly, when $\mu_i \in [0, 0.5]$ S may be defined as H^2/H^1 so that S lies in [0, 1]. Higher the value of S the more alike (similar) are the supports of the fuzzy set with respect to the property P. This index of similarity can therefore be regarded as a measure of the degree to which the members of a fuzzy set are alike.

Therefore, the value of conventional fuzzy entropy (H^1 or Eq. 4) can only indicate whether the fuzziness in a set is low or high. In addition to this, the value of H^r also enables one to infer whether the fuzzy set contains similar supports (or elements) or not. The similarity index thus defined can be successfully used for measuring interclass and intraclass ambiguity (i.e., class homogeneity and contrast) in pattern recognition and image processing problems.

The aforesaid features are explained in Table 1 when $\mu_i \in [0.5, 1]$, min operator is used to compute group membership and k in Eq. 5 is considered to be $10C_r, r = 1, 2, \ldots 6$.

$H_{hy}(X)$ has the following properties. In the absence of fuzziness when MNP_b pixels become completely black ($\mu_{mn} = 0$) and MNP_w pixels become completely white ($\mu_{mn} = 1$) then $E_w = P_w$, $E_b = P_b$ and H_{hy} boils down to two state classical

entropy

$$H_c = -P_w \log P_w - P_b \log P_b, \tag{10}$$

the states being black and white. Thus, H_{hy} reduces to H_c only when a proper

Table 1: Higher Order Entropy

Case	μ_X	H1	H2	H3	H4	H5	H6	S
1	{1,1,1,1,1,1,1,1,1,1}	0	0	0	0	0	0	1
2	{.5,.5,.5,.5,.5,.5,.5,.5,.5,.5}	1	1	1	1	1	1	1
3	{1,1,1,1,1,.5,.5,.5,.5,.5}	.5	.777	.916	.976	.996	1	.642
4	{.5,.5,.5,.5,.5,.6,.6,.6,.6,.6}	.980	.991	.996	.999	.999	1	.989
5	{.6,.6,.65,.9,.9,.9,.9,.9,.9,.915}	.538	.678	.781	.855	.905	.937	.793
6	{.8,.8,.8,.8,.8,.8,.9,.9,.9,.9}	.538	.613	.641	.649	.650	.650	.878
7	{.5,.5,.5,.5,.5,.5,.9,.9,.9,.9}	.748	.916	.979	.997	1	1	.816
8	{.7,.7,.7,.7,.7,.8,.8,.8,.8,.8}	.748	.802	.830	.841	.845	.846	.932

defuzzification process is applied to detect (restore) the pixels. $|H_{hy} - H_c|$ can therefore be acted as an objective function for enhancement and noise reduction. The lower the difference, the lesser is the fuzziness associated with the individual symbol and higher will be the accuracy in classifying them as their original value (white or black). (This property was lacking with $\gamma(X)$ and $H(X)$ measures (equations 2-4) which always reduce to zero irrespective of the defuzzification process). In other words, $|H_{hy} - H_c|$ represents an amount of information which was lost by transforming a two tone image to a gray tone.

For a given P_w and P_b $(P_w + P_b = 1,\ 0 \le P_w, P_b \le 1)$, of all possible defuzzified versions, H_{hy} is minimum for the one with properly defuzzified.

If $\mu_{mn} = 0.5$ for all (m, n) then $E_w = E_b$

and $$H_{hy} = -\log(0.5 \exp 0.5) \tag{11}$$

i.e., H_{hy} takes a constant value and becomes independent of P_w and P_b. This is logical in the sense that the machine is unable to take decision on the pixels since all μ_{mn} values are 0.5.

Let us consider an example of a digital image in which, say, 70% pixels look white, while the remaining 30% look dark. Thus the probability of a white pixel P_w is 0.7 and that of a dark pixel P_b is 0.3. Suppose, the whiteness of the pixels is not constant, i.e., there is a variation (grayness) and similar is the case with the black pixels.

Let us now consider the effect of improper defuzzification on the pattern shown in case 1 of the Table 2. Two types of defuzzifications are considered here. In cases 2-4 all the symbols with $\mu = 0.5$ are transformed to zero when some of them were actually generated from symbol '1'. In cases 5-6 of Table 2 some of the μ values

greater than 0.5 which were generated from symbol 1 (or belong to the white portion of the image) are wrongly defuzzified and brought down towards zero (instead of 1).

In both situations, it is to be noted that $|H - H_{hy}|$ does not reduce to zero. The case 7, on the other hand, has all its elements properly defuzzified. As a result, E_1 and E_0 become 0.3 and 0.7 respectively and $|H_{hy} - H_c|$ reduces to zero.

Table 2: Effect of wrong defuzzification(with $p_0 = 0.3$ and $p_1 = 0.7$)

Case	μ_X	E_1	E_0	H_{hy}	$\lvert H - H_{hy} \rvert$
1	{.9,.9,.8,.8,.7,.6,.5,.5,.4,.3}	.620	.876	.235	.375
2	{.999,.999,.9,.8,.7,.7,.3,.3,.2,.1}	.576	.776	.342	.268
3	{1,1,1,.99,.9,.9,.1,.1,0,0}	.450	.648	.542	.068
4	{1,1,1,1,1,1,0,0,0,0}	.400	.600	.632	.021
5	{.99,.99,.1,.1,.9,.8,.7,.2,.1,.1}	.630	.634	.456	.154
6	{1,1,0,0,1,1,1,0,0,0}	.500	.500	.693	.082
7	{1,1,1,1,1,1,1,0,0,0}	.300	.700	.611	0

There had been some attempts [2, 15] to combine probabilistic entropy and possibilistic entropy, they failed to have the aforesaid property of the effect of wrong defuzzification. The details of classical entropy measures (e.g., higher order, conditional and positional) of an image are available in [14, 16].

$C(\mu_1, \mu_2)$ of equation (7) has the following properties.

a) If for higher values of $\mu_1(X)$, $\mu_2(X)$ takes higher values and the converse is also true then $C(\mu_1, \mu_2)$ must be very high.

b) If with increase of x, both μ_1 and μ_2 increase then $C(\mu_1, \mu_2) > 0$.

c) If with increase of x, μ_1 increases and μ_2 decreases or vice versa then $C(\mu_1, \mu_2) < 0$.

d) $C(\mu_1, \mu_1) = 1$

e) $C(\mu_1, \mu_1) \geq C(\mu_1, \mu_2)$

f) $C(\mu_1, 1-\mu_1) = -1$

g) $C(\mu_1, \mu_2) = C(\mu_2, \mu_1)$

h) $-1 \leq C(\mu_1, \mu_2) \leq 1$

i) $C(\mu_1, \mu_2) = -C(1-\mu_1, \mu_2)$

j) $C(\mu_1, \mu_2) = C(1-\mu_1, 1-\mu_2)$

IMAGE GEOMETRY

The various geometrical properties of a fuzzy image subset (characterized by $\mu_X(x_{mn})$ or simply by μ) as defined by Rosenfeld [5,6] and Pal and Ghosh [7] are given below with illustration. These provide measures of ambiguity in geometry (spatial domain) of an image.

Area The area of a fuzzy subset μ is defined as [5]

$$a(\mu) = \int \mu \tag{12}$$

where the integration is taken over a region outside which μ=0. For μ being piecewise constant (in case of digital image) the area is

$$a(\mu) = \sum \mu \tag{13}$$

where the summation is over a region outside which μ=0. Note from equation (13) that area is the weighted sum of the regions on which μ has constant value weighted by these values.

Example 1 Let μ be of the form

0.2 0.4 0.3
0.2 0.7 0.6
0.6 0.5 0.6

Area $a(\mu)$ = (0.2+0.4+0.3+0.2+0.7+0.6+0.6+0.5+0.6) = 4.1

Perimeter If μ is piecewise constant, the perimeter of μ is defined as [5]

$$p(\mu) = \sum_{i,\,j,\,k} |\mu(i) - \mu(j)| * |A(i,\,j,\,k)| \tag{14}$$

This is just the weighted sum of the lengths of the arcs A(i, j, k) along which the regions having constant μ values μ(i) and μ(j) meet, weighted by the absolute difference of these values. In case of an image if we consider the pixels as the piecewise constant regions, and the common arc length for adjacent pixels as unity then the perimeter of an image is defined by

$$p(\mu) = \sum_{i,\,j} |\mu(i) - \mu(j)| \tag{15}$$

where μ(i) and μ(j) are the membership values of two adjacent pixels.

For the fuzzy subset μ of example 1, perimeter is

$$\begin{aligned} p(\mu) &= |0.2-0.4| + |0.2-0.2| + |0.4-0.3| + |0.4-0.7| \\ &\quad + |0.3 - 0.6| + |0.2 - 0.6| + |0.2 - 0.7| + |0.7 - 0.6| \\ &\quad + |0.7-0.5| + |0.6-0.6| + |0.6-0.5| + |0.5-0.6| \\ &= 2.3 \end{aligned}$$

Compactness The compactness of a fuzzy set μ having an area of a (μ) and a perimeter of p(μ) is defined as [5]

$$\text{comp}(\mu) = \frac{a(\mu)}{(p(\mu))^2} \tag{16}$$

Physically, compactness means the fraction of maximum area (that can be encircled by the perimeter) actually occupied by the object. In non fuzzy case the value of compactness is maximum for a circle and is equal to $\pi / 4$. In case of fuzzy disc, where the membership value is only dependent on its distance from the center, this compactness value is $\geq \pi / 4$ [6]. Of all possible fuzzy discs compactness is therefore minimum for its crisp version.

For the fuzzy subset μ of example 1, comp(μ) = 4.1/(2.3*2.3) = 0.775.

Height and Width The height of a fuzzy set μ is defined as [5]

$$h(\mu) = \int \max_{m} \mu_{mn} dn \qquad [17]$$

where the integration is taken over a region outside which $\mu_{mn} = 0$.

Similarly, the width of the fuzzy set is defined by

$$w(\mu) = \int \max_{n} \mu_{mn} dm \qquad (18)$$

with the same condition over integration as above. For digital pictures m and n can take only discrete values, and since $\mu = 0$ outside the bounded region, the max operators are taken over a finite set. In this case the definitions take the form

$$h(\mu) = \sum_{n} \max_{m} \mu_{mn} \qquad (19)$$

and

$$w(\mu) = \sum_{m} \max_{n} \mu_{mn} \qquad (20)$$

m = 1, 2, . . . M; n = 1, 2, . . . N

So physically, in case of a digital picture, height is the sum of the maximum membership values of each row. Similarly, by width we mean the sum of the maximum membership values of each column.

For the fuzzy subset μ of example 1, height is h(μ) = 0.4+0.7+0.6 = 1.7 and width is w(μ) = 0.6+0.7+0.6 = 1.9.

Length and Breadth The length of a fuzzy set μ is defined as [7]

$$l(\mu) = \max_{m} \left(\int \mu_{mn} dn \right) \qquad (21)$$

where the integration is taken over the region outside which $\mu_{mn} = 0$. In case of a digital picture where m and n can take only discrete values the expression takes the form

$$l(\mu) = \max_{m} \left(\sum_{n} \mu_{mn} \right) \qquad (22)$$

Physically speaking, the length of an image fuzzy subset gives its longest expansion in the column direction. If μ is crisp, μ_{mn} = 0 or 1; in this case length is the maximum number of pixels in a column. Comparing equation (22) with (19) we notice that the length is different from height in the sense, the former takes the summation of the entries in a column first and then maximizes over different columns whereas, the later maximizes the entries in a column and then sums over different columns.

The breadth of a fuzzy set μ is defined as

$$b(\mu) = \max_n \left(\int \mu_{mn} dm\right) \tag{23}$$

where the integration is taken over the region outside which $\mu_{mn} = 0$. In case of a digital picture the expression takes the form

$$b(\mu) = \max_n \left(\sum_m \mu_{mn}\right) \tag{24}$$

Physically speaking, the breadth of an image fuzzy subset gives its longest expansion in the row direction. If μ is crisp, $\mu_{mn} = 0$ or 1; in this case breadth is the maximum number of pixels in a row. The difference between width and breadth is same as that between height and length.

For the fuzzy subset μ in example 1, length is $l(\mu) = 0.4 + 0.7 + 0.5 = 1.6$ and breadth is $b(\mu) = 0.6 + 0.5 + 0.6 = 1.7$.

Index of Area Coverage The index of area coverage of a fuzzy set may be defined as [7]

$$\text{IOAC}(\mu) = \frac{\text{area}(\mu)}{l(\mu) * b(\mu)} \tag{25}$$

In nonfuzzy case, the IOAC has value of 1 for a rectangle (placed along the axes of measurement). For a circle this value is $\pi r^2 / (2r * 2r) = \pi / 4$. Physically by IOAC of a fuzzy image we mean the fraction (which may be improper also) of the maximum area (that can be covered by the length and breadth of the image) actually covered by the image.

For the fuzzy subset μ of example 1, the maximum area that can be covered by its length and breadth is $1.6 * 1.7 = 2.72$ whereas, the actual area is 4.1, so the IOAC $= 4.1 / 2.72 = 1.51$.

It is to be noted that $l(X)/h(X) \leq 1$ (26)

$b(X)/w(X) \leq 1$ (27)

When equality holds for (26) or (27) the object is either vertically or horizontally oriented.

Degree of Adjacency The degree to which two regions S and T of an image are adjacent is defined as

$$a(S,T) = \sum_{p \in BP(S)} \frac{1}{1 + |\mu(p) - r(q)|} * \frac{1}{1 + d(p)} \tag{28}$$

Here d(p) is the shortest distance between p and q, q is a border pixel (BP) of T and p is a border pixel of S. The other symbols are having their same meaning as in the previous discussion.

The degree of adjacency of two regions is maximum (=1) only when they are physically adjacent i.e., d(p)=0 and their membership values are also equal i.e., $\mu(p) = r(q)$. If two regions are physically adjacent then their degree of adjacency is determined only by the difference of their membership values. Similarly, if the membership values of two regions are equal their degree of adjacency is determined by their physical distance only.

In the following sections, we will be explaining how the aforesaid measures can be used for image segmentation and edge detection problems.

SEGMENTATION AND OBJECT EXTRACTION

The problem of grey level thresholding plays an important role in image processing and vision problems. For example, in enhancing contrast in a image we need to select proper threshold levels from its histogram so that some suitable non-linear transformation can highlight a desirable set of pixel intensities compared to others. Similarly, in image segmentation one needs proper histogram thresholding whose objective is to establish boundaries in order to partition the image spaces into meaningful regions. This Section illustrates an algorithm where these various information measures can be used to make this selection task automatic so that an optimum threshold (or set of thresholds) may be estimated without the need to refer directly to the histogram.

Algorithm 1

Given an L level image X of dimension MxN with minimum and maximum gray vales ℓ_{min} and ℓ_{max} respectively,

Step 1: Construct the membership plane using the standard S function (equation (29)) as

$$\mu_{mn} = \mu(\ell) = S(\ell;\ a,\ b,\ c)$$

(called bright image plane if the object regions possess higher gray values)

or $\mu_{mn} = \mu(\ell) = 1 - S(\ell;\ a,\ b,\ c)$

(called dark image plane if the object regions possess lower gray values)
with cross-over point b and a bandwidth Δb. The S function as given below in equation (29) is shown graphically in Fig. 2 for an L-level image.

$$P_{mn} = \mu_x(x_{mn}) = S(x_{mn};\ a,\ b,\ c) = 0,\ x_{mn} \le a \qquad (29a)$$

$$= 2\left[(x_{mn} - a)/(c - a)\right]^2, a \le x_{mn} \le b \qquad (29b)$$

$$= 1 - 2\left[(x_{mn} - c)/(c - a)\right]^2, b \le x_{mn} \le c \qquad (29c)$$

$$= 1,\ x_{mn} \ge c \qquad (29d)$$

with $b = (a+c)/2$, $b-a = c-b = \Delta b$.
The parameter b is the cross-over point, i.e., S(b; a, b ,c) = 0.5. Δb is the bandwidth.

Step 2: Compute $\gamma(X)$,H(X), Comp(X) and IOAC(X)

Step 3: Vary b between ℓ_{min} and ℓ_{max} and select those b for which I(X) (where I(X)) denotes one of the aforesaid measures or a combination of them) has local minima. Among the local minima let the global one have a cross over point s.

The level s, therefore, denotes the cross over point of the fuzzy image plane μ_{mn}, which has minimum grayness and/or geometrical ambiguity. The μ_{mn} plane

Step 3: Vary b between ℓ_{min} and ℓ_{max} and select those b for which I(X) (where I(X)) denotes one of the aforesaid measures or a combination of them) has local minima. Among the local minima let the global one have a cross over point s.

The level s, therefore, denotes the cross over point of the fuzzy image plane μ_{mn}, which has minimum grayness and/or geometrical ambiguity. The μ_{mn} plane

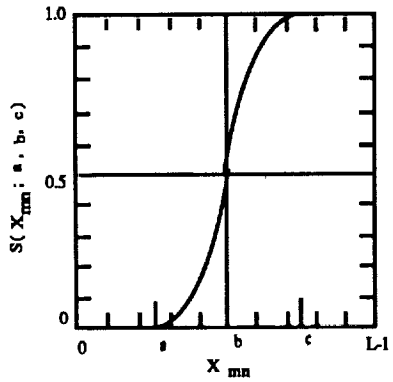


Figure 2 Standard S function for an L-level image.

then can be viewed as a fuzzy segmented version of the image X. For the purpose of nonfuzzy segmentation, we can take s as the threshold or boundary for classifying or segmenting image into object and background. (For images having multiple regions, one would have a set of such optimum $\mu(X)$ planes).

The measure I(X) in Step 3 can represent either grayness ambiguity (i.e., $\gamma(X)$ or H(X)) or geometrical ambiguity (i.e., comp(X) or IOAC(X) or a(S,T)) or both (i.e., product of grayness and geometrical ambiguities).

Faster Method of Computation

From the algorithm 1 it appears that one needs to scan an L level image L times (corresponding to L cross over points of the membership function) for computing the parameters for detecting its threshold. The time of computation can be reduced significantly by scanning it only once for computing its co-occurrence matrix, row histogram and column histogram, and by computing $\mu(l)$, $l = 1, 2, \ldots L$ every time with the membership function of a particular cross over point.

The computations of $\gamma(X)$ (or H(X)), a(X), p(X), 1(X) and b(X) can be made faster in the following way. Let h(i), i=1,2..L be the number of occurrences of the level i, c[i,j], i = 1, 2 . . L, j = 1, 2 . . L the co-occurrence matrix and $\mu(i)$, i = 1, 2.. L the membership vector for a fixed cross over point of an L level image X.

Determine $\gamma(X)$, area and perimeter as

$$\gamma(X) = \frac{2}{MN} \sum_{i=1}^{L} T(i)\, h(i) \tag{30a}$$

$$T(i) = \min\{\mu(i), 1-\mu(i)\} \tag{30b}$$

For calculating length and breadth following steps can be used. Compute the row histogram R[m, l], m = 1, . . . M, l = 1 . . L, where R[m, l] represents the number of occurrences of the gray level l in the mth row of the image. Find the column histogram C[n, l], n = 1. . N, l = 1 . . L, where C[n, l] represents the number of occurrences of the gray level l in the nth column of the image. Calculate length and breadth as

$$l(X) = \max_{n} \sum_{l=1}^{L} C[n, l].\ \mu(l) \tag{33}$$

$$b(X) = \max_{m} \sum_{l=1}^{L} R[m, l].\ \mu(l) \tag{34}$$

Some Remarks

The grayness ambiguity measure e.g., $\gamma(X)$ or H(X) basically sharpens the histogram of X using its global information only and it detects a single threshold in its valley region. Therefore, if the histogram does not have a valley, the above measures will not be able to select a threshold for partitioning the histogram. This can readily be seen from Equation (30) which shows that the minima of $\gamma(X)$ measure will only correspond to those regions of gray level which has minimum occurrences (i.e., valley region). Comp (X) or IOAC(X), on the other hand, uses local information to determine the fuzziness in spatial domain of an image. As a result, these are expected to result better segmentation by detecting thresholds even in the absence of a valley in the histogram.

Again, comp(X) measure attempts to make a circular approximation of the object region for its extraction, whereas, the IOAC(X) goes by the rectangular approximation. Their suitability to an image should therefore be guided by this criterion.

Choice of Membership Function

In the aforesaid algorithm $w = 2\Delta b$ is the length of the interval which is shifted over the entire dynamic range of gray scale. As w decreases, the $\mu(x_{mn})$ plane would have more intensified contrast around the cross-over point resulting in decrease of ambiguity in X. As a result, the possibility of detecting some undesirable thresholds (spurious minima) increases because of the smaller value of Δ b. On the other hand, increase of w results in a higher value of fuzziness and thus leads towards the possibility of losing some of the weak minima.

The criteria regarding the selection of membership function and the length of window (i.e., w) have been reported recently by Murthy and Pal [10] assuming continuous function for both histogram and membership function. For a fuzzy set "bright image plane", the membership function μ: $[0, w] \rightarrow [0,1]$ should be such that

i) μ is continuous, $\mu(0) = 0$, $\mu(w) = 1$
ii) μ is montominally non-decreasing, and
iii) $\mu(x) = 1 - \mu(w-x)$ for all $x \in [0, w]$ where $w>0$ is the length of the window.

Furthermore, μ should satisfy the bound criteria derived based on the correlation measure (equation 7). The main properties on which correlation was formulated are

P_1: If for higher values of μ_1, μ_2 takes higher values and for lower values of μ_1, μ_2 also takes lower values then $C(\mu_1, \mu_2) > 0$

P_2: If $\mu_1 \uparrow$ and $\mu_2 \uparrow$ then $C(\mu_1, \mu_2) > 0$

P_3: If $\mu_1 \uparrow$ and $\mu_2 \downarrow$ then $C(\mu_1, \mu_2) < 0$

[$\uparrow$ denotes increases and $\downarrow$ denotes decreases].

It is to be mentioned that P2 and P3 should not be considered in isolation of P_1. Had this been the case, one can cite several examples when $\mu_1 \uparrow$ and $\mu_2 \uparrow$ but $C(\mu_1, \mu_2) < 0$ and $\mu_1 \uparrow$ and $\mu_2 \downarrow$ but $C(\mu_1, \mu_2) > 0$. Subsequently, the type of membership functions which should not be considered in fuzzy set theory are categorized with the help of correlation. Bound functions h_1 and h_2 are accordingly derived [11]. They are

$$h_1(x) = 0, \quad 0 \le x \le \epsilon \tag{35}$$

$$= x - \epsilon, \quad \epsilon \le x \le 1$$

$$h_2(x) = x + \epsilon, \quad 0 \le x \le 1 - \epsilon \tag{36}$$

$$= 1, \quad 1 - \epsilon \le x \le 1$$

where $\epsilon = 0.25$. The bounds for membership function μ are such that

$$h_1(x) \le \mu(x) \le h_2(x) \text{ for } x \in [0,1].$$

For x belonging to any arbitrary interval, the bound functions will be changed proportionately. For $h_1 \le \mu \le h_2$, $C(h_1, h_2) \ge 0$, $C(h_1, \mu) \ge 0$ and $C(h_2, \mu) \ge 0$.

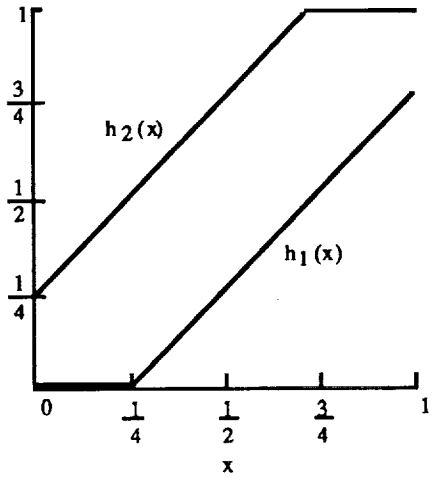


Figure 3 Bound Functions for $\mu(x)$.

The function μ lying in between h_1 and h_2 does not have most of its variation concentrated (i) in a very small interval, (ii) towards one of the end points of the interval under consideration and (iii) towards both the end points of the interval under consideration.

Figure 3 shows such bound functions. It is to be noted that Zadeh's standard S function (equation 29) satisfies these bounds.

It has been shown [10] that for detecting a minimum in the valley region of a histogram, the window length w of the μ function should be less that the distance between two peaks around that valley region.

H^r as an Objective Criterion

Let us now explain another way of extracting object by minimizing higher order fuzzy entropy (equation 5) of both object and background regions. Before explaining the algorithm, let us describe the membership function and its selection procedure.

Let s be an assumed threshold which partitions the image X into two parts namely, object and background. Suppose the gray level ranges [1 - s] and {s + 1 - L] denote, respectively, the object and background of the image X. An inverse π-type function as shown by the solid line in the Figure 4 is used here to obtain μ_{mn} values of X. The inverse π-type function is seen (from Fig. 4) to be generated by taking union of S(x ; (s - (L - s)), s, L) and 1 - S(x; 1, s, (s + s - 1)), where S denotes the standard S function defined by Zadeh (equation 29).

The resulting function as shown by the solid line, makes μ lie in [0.5,1]. Since the ambiguity (difficulty) in deciding a level as a member of the object or the background is maximum for the boundary level S, it has been assigned a membership value of 0.5 (i.e., cross-over point). Ambiguity decreases (i.e., degree of belongingness to either object or background increases) as the gray value moves away from s on either side. The μ_{mn} thus obtained denotes the degree of belongingness of a pixel x_{mn} to either object or background.

Since s is not necessarily the mid point of the entire gray scale, the membership function (solid line if Fig. 4) may not be a symmmetric one. It is further to be noted that one may use any linear or nonlinear equation (instead of Zadeh's standard S function) to represent the membership function in Fig. 4. Unlike the Algorithm-1, the membership function does not need any parameter selection to control the output.

Algorithm 2

Assume a threshold s, $1 \leq s \leq L$ and execute the following steps.

Step 1: Apply an inverse π - type function [Fig. 4] to get the fuzzy μ_{mn} plane, with $\mu_{mn} \in$ [0.5, 1]. (The membership function is in general asymmetric).

Step 2: Compute the rth order fuzzy entropy of the object H_O^r and the background

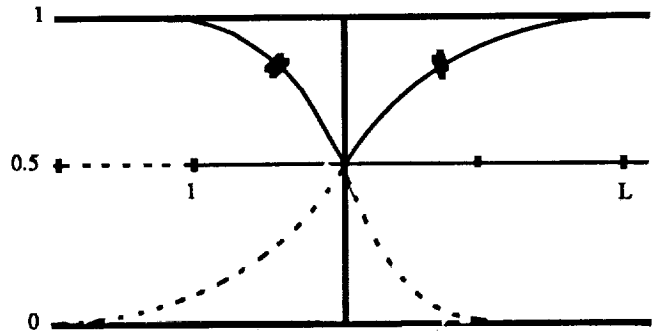


Figure 4 Inverse π function (solid line) for computing object and background entropy.

H_B^r considering only the spatially adjacent sequences of pixels present within the object and background respectively. Use the 'min' operator to get the membership value of a sequence of pixels.

Step 3: Compute the total rth order fuzzy entropy of the partitioned image as

$$H_s^r = H_O^r + H_B^r.$$

Step 4: Minimize H_s^r with respect to s to get the threshold for object background classification.

Referring back to the Table 1, we have seen that H^2 reflects the homogeneity among the supports in a set, in a better way than H^1 does. Higher the value of r, the stronger is the validity of this fact. Thus, considering the problem of object-background classification, H^r seems to be more sensitive (as r increases) to the selection of appropriate threshold; i.e., the improper selection of the threshold is more strongly reflected by H^r than H^{r-1} For example, the thresholds obtained by H^2 measure has more validity than those by H^1 (which only takes into account the histogram information). Similar arguments hold good for even higher order (r > 2) entropy.

Example 2

Figures 5 and 6 show the images of Lincoln and blurred chromosome along with the histogram. Table 3 shows the thresholds obtained by comp (X) and IOAC (X) measures for various window sizes w when Zadeh's S function is used as membership function. Lincoln image is of 64x64 with 32 gray levels whereas, chromosome image is of 64x64 with 64 gray levels.

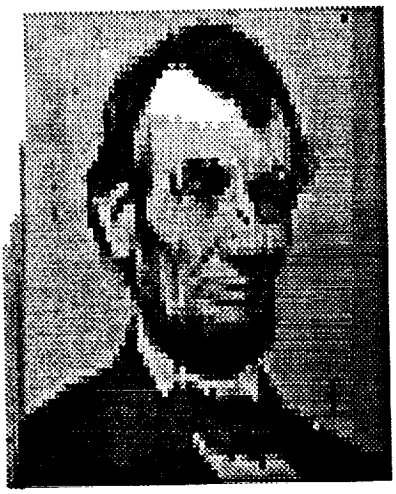


FIgure 5(a) Input.

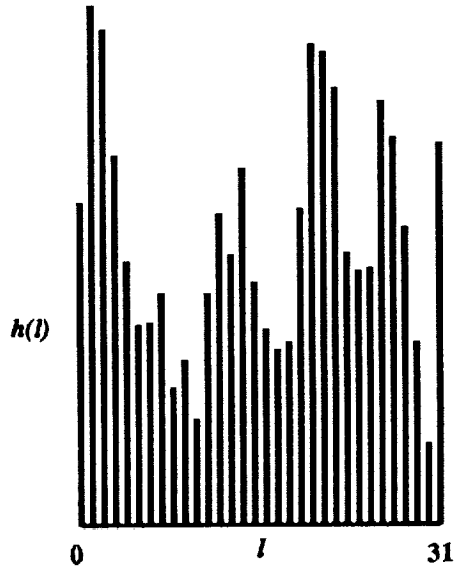


Figure 5(b) Histogram

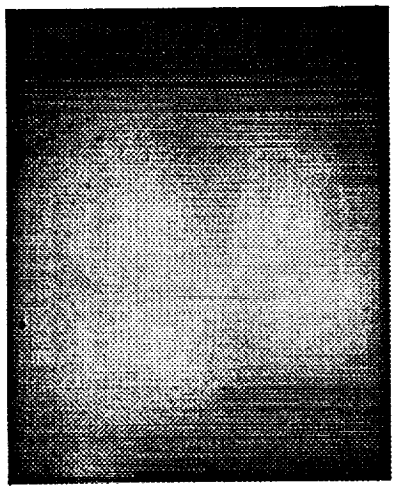


Figure 6(a) Input.

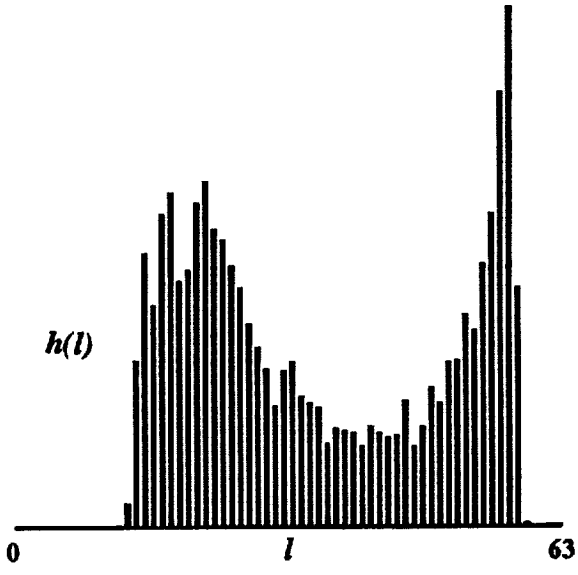


Figure 6(b) Histogram.

Figure 7(a) Threshold = 10.

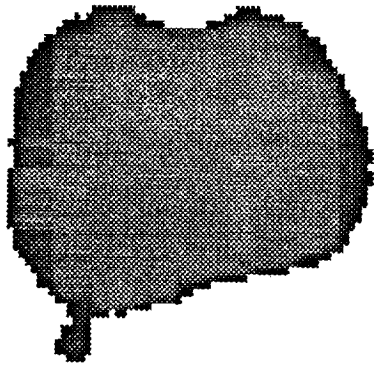


Figure 7(b) Threshold = 32.

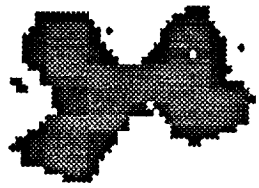


Figure 7(c) Threshold = 56.

Table 3 Various Thresholds (* denotes global minimum)

W	Lincoln	
	Comp	IOAC
8	10	11 * 23
10	10	11 * 23
12	10	11 * 23
16	9	11

W	Chromosome	
	Comp	IOAC
12	33 56	* 30 * 51
16	55	31 * 49
20	54	32 * 46
24	52	34

Threshold produced by H^2 measure (Algorithm 2) is 8 for Lincoln image. Some typical nonfuzzy thresholded outputs of these images are shown in Figure 7. Recently, transitional correlation and within class correlation have been defined [12] based on equation (7) for image segmentation which takes both local and global information into account. Automatic selection of an appropriate enhancement operator is available in [17].

EDGINESS MEASURE

Let us now describe an edginess measure [18,19] based on H^1(Equation 5) which denotes an amount of difficulty is deciding whether a pixel can be called an edge or not. Let $N^3_{x,y}$ be a 3 x 3 neighborhood of a pixel at (x, y) such that

$$N^3_{x,y} = \{(x, y), (x-1,y), (x+1,y), (x,y-1), (x,y+1), (x-1, y-1), (x-1, y+1), (x+1,y-1), (x+1,y+1)\} \quad (37)$$

The edge-entropy, $H^E_{x,y}$ of the pixel (x, y), giving a measure of edginess at (x, y) may be computed as follows. For every pixel (x, y), compute the average, maximum and minimum values of gray levels over $N^3_{x,y}$. Let us denote the average, maximum and minimum values by Avg, Max, Min respectively. Now define the following parameters.

$$D = \max \{ Max - Avg, Avg - Min\} \quad (38)$$

$$B = Avg \quad (39)$$

$$A = B - D \quad (40)$$

$$C = B + D \quad (41)$$

A π-type membership function is then used to compute μ_{xy} for all (x, y) $\in N^3_{x,y}$, such that $\mu(A) = \mu(C) = 0.5$ and $\mu(B) = 1$. It is to be noted that $\mu_{xy} \geq 0.5$. Such a μ_{xy}, therefore, gives the degree to which a gray level is close to the average value computed over $N^3_{x,y}$. In other words, it represents a fuzzy set "pixel intensity close to its average value", averaged over $N^3_{x,y}$. When all pixel values over $N^3_{x,y}$ are either equal or close to each other (i.e., they are within the same region), such a transformation will make all $\mu_{xy} = 1$ or close to 1. In other words, if there is no edge, pixel values will be close to each other and the μ values will be close to one(1); thus resulting in a low value of H^1. On the other hand, if there is an edge (dissimilarity in gray values over $N^3_{x,y}$), then the μ values will be more away from unity; thus resulting in a high value of H^1. Therefore, the entropy H^1 over $N^3_{x,y}$ can be viewed as a measure of edginess ($H^E_{x,y}$) at the point (x, y). The higher the value of $H^E_{x,y}$, the stronger is the edge intensity and the easier is its detection. As mentioned before, there are several ways in which one can define a π-type function

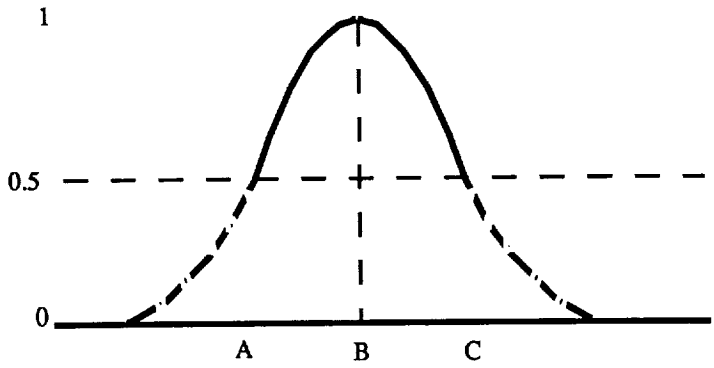


Figure 8 π function (solid line)for computing edge entropy.

(solid line) as shown in Fig. 8.

The proposed entropic measure is less sensitive to noise because of the use of a dynamic membership function based on a local neighborhood. The method is also not sensitive to the direction of edges. Other edginess measures are available in [13,20].

CONCLUSIONS

Various uncertainty and image information measures, as conveyed by entropy and fuzzy geometry, have been explained. The problems of object extraction and edge detection have been considered, as an example, to demonstrate their applications. Uncertainty in membership evaluation in these problems and its management have also been explained through bound functions. The measures comp(X) and IOAC(X) can also be used for skeleton extraction and medial axis transformation of a gray tone image [21].

Acknowledgements

This work was done while the author held an NRC-NASA research Associateship at the Johnson Space Center, Houston, Texas. The author gratefully acknowledges Dr. Robert N. Lea for his interest in this work, Ms. Dianne Rader, Ms. Kim Herhold, and Mr. Todd Carlson for typing the manuscript and Mr. Albert Leigh for his assistance in getting some results. The author is also grateful to his Institute for sanctioning him leave.

References

1. A Kaufmann, *Introduction to the Theory of Fuzzy Subsets-Fundamental Theoretical Elements*, vol 1, Academic Press, NY, 1975.
2. A De Luca and S. Termini, A definition of nonprobabilistic entropy in the setting of fuzzy set theory, *Inform and Control*, vol 20, pp 301-312, 1972.
3. N.R. Pal and S.K. Pal, Higher order fuzzy entropy and hybrid entropy of a

set, *Information Sciences* (to appear).

4. C.A. Murthy, S.K.Pal and D. Dutta Majumder, Correlation between two fuzzy membership functions, *Fuzzy Sets and Systems*, vol. 7, No-1, pp 23-38, 1985.
5. A. Rosenfeld, The fuzzy geometry of image subsets, *Patt. Recog .Lett.*, vol 2, pp 311-317, 1984.
6. A. Rosenfeld and S. Haber, The perimeter of a fuzzy set, *Technical Report*, University of Maryland, Center for Automation Research, TR-8, 1983.
7. S.K. Pal and A. Ghosh, Index of area coverage of fuzzy image subsets and object extraction, *Patt. Recog. Lett.*, (to appear).
8. L.A. Zadeh, K.S. Fu, K. Tanaka and M. Shimura, *Fuzzy Sets and Their Applications to Cognitive and Decision Processes*, Academic Press, London, 1975.
9. S.K. Pal and A. Rosenfeld, Image enhancement and thresholding by optimization of fuzzy compactness, *Patt Recog Lett*, Vol 7, pp 77-86, 1988.
10. C.A. Murthy and S.K. Pal, Histogram thresholding by minimizing graylevel fuzziness, *Information Sciences* (to appear).
11. C.A. Murthy and S.K. Pal, Bounds for membership functions: correlation based approach, *Fuzzy Sets and Systems*, (communicated).
12. S.K. Pal and A. Ghosh, Image segmentation using fuzzy correlation, *Information Sciences*, (to appear).
13. S.K. Pal and D. Dutta Majumdar, *Fuzzy Mathematical Approach to Pattern Recognition*, John Wiley and Sons, (Halsted Press), NY, 1986.
14. N.R. Pal and S.K. Pal, Object-background segmentation using new definitions of entropy, *IEE Proc.*,vol. 136, Pt. E, pp. 284- 295, July 1989.
15. W.X. Xie and S.D. Bedrosian, An information measure for fuzzy sets, IEEE Trans. Syst., man and Cyberns., vol. SMC-14, pp. 151- 156, 1984.
16. N.R. Pal and S.K. Pal, Entropy: a new definition and its applications, *IEEE Trans. Syst., Man and Cyberns.*, accepted for publication.
17. M.K. Kundu and S.K. Pal, Automatic selection of object enhancement operator with quantitative justification based on fuzzy set theoretic measure, *Patt. Recog. Lett.*, (to appear).
18. N.R. Pal, *On Image Information Measure and Object Extraction*, Ph. D. Thesis, Indian Statistical Institute, Calcutta, India, March 1990.
19. S.K. Pal and N.R. Pal, Higher order entropy, hybrid entropy and their applications, *Proc. INDO-US Workshop on Spectrum analysis in one and two dimensions*, Nov 27-29, 1990, New Delhi, NBH Oxford Publishing Co. New Delhi (to appear).
20. S.K. Pal, A measure of edge ambiguity using fuzzy sets, *Patt. Recog. Lett.*, vol 4, pp 51-56, 1986.
21. S.K. Pal, Fuzzy skeletonization of an image, *Patt. Recog. Lett.*, vol. 10, pp. 17-23, 1989.

THE SEMANTICS OF FUZZY LOGIC

Enrique H. Ruspini
Artificial Intelligence Center
SRI International
Menlo Park, California, U.S.A.

INTRODUCTION

In this brief paper,[1] we summarize the results of recent research on the conceptual foundations of fuzzy logic [9]. This research resulted in the formulation of several semantic models that interpret the major concepts and structures of fuzzy logic in terms of the more primitive notions of resemblance and similarity between "possible worlds," i.e., the possible states, situations or behaviors of a real-world system. The metric structures representing this notion of proximity are generalizations of the accessibility relation of modal logics [1].

Possibilistic reasoning methods may be characterized, by means of our interpretation, as approaches to the description of the relations of proximity that hold between possible system states that are logically consistent with existing evidence, and other situations, which are used as reference landmarks. By contrast, probabilistic methods seek to quantify, by means of measures of set extension, the proportion of the set of possible worlds where a proposition is true.

Our discussion will focus primarily on the principal characteristics of a model, discussed in detail in a recent technical note [3], that quantifies resemblance between possible worlds by means of a similarity function that assigns a number between 0 and 1 to every pair of possible worlds. Introduction of such a function permits to interpret the major constructs and methods of fuzzy logic: conditional and unconditional possibility and necessity distributions and the generalized modus ponens of Zadeh on the basis of related metric relationships between subsets of possible worlds.

[1]The present paper is a slightly revised and expanded version of a communication appearing in the Proceedings of the 1990 Iizuka Conference on Fuzzy Logic and Neural Networks.

THE APPROXIMATE REASONING PROBLEM

Our semantic model of fuzzy logic is based on two major conceptual structures: the notion of possible world, which is the basis for our unified view of the approximate reasoning problem [4], and a metric structure that quantifies similarity between pairs of possible worlds.

If a reasoning problem is thought of as being concerned with the determination of the truth-value of a set of propositions that describe different aspects of the behavior of a system, then a *possible world* is simply a function (called a valuation) that assigns a unique truth value to every proposition in that set and that, in addition, is consistent with the rules of propositional logic. The set of all such possible worlds is called the *universe of discourse*.

In any reasoning problem, knowledge about the characteristics of the class of systems being studied combined with observations about the particular system under consideration restricts the extent of possible worlds that must be considered to a subset of the universe of discourse, called the *evidential set*, which will be denoted $\mathscr{E}$.

The purpose of the inferential procedures utilized in any reasoning problem may be characterized as that of establishing if, for a given proposition $\mathscr{H}$ (the *hypothesis*), either $\mathscr{E} \Rightarrow \mathscr{H}$ or $\mathscr{E} \Rightarrow \neg\mathscr{H}$, i.e., whether existing evidence implies the hypothesis or it implies its negation. In approximate reasoning problems, as illustrated in Figure 1, such determination is, by definition, impossible: there are some possible worlds in the the evidential set where the hypothesis is true and some where it is false.

SIMILARITY FUNCTIONS AND IMPLICATION

In the view of fuzzy logic proposed by our model the purpose of possibilistic methods is the description of the evidential set by characterization of the resemblance relations that hold between its elements and elements of other sets used as reference landmarks. By contrast, probabilistic methods (i.e., probabilities usually interpreted as frequencies or as degree of personal belief) seek to measure the relative extensions of the sets $\mathscr{E} \cap \mathscr{H}$ and $\mathscr{E} \cap \neg\mathscr{H}$.

To represent similarity or resemblance between possible worlds we introduce a binary function S that assigns a value between 0 and 1 to every pair of possible worlds w and w'. A value of S equal to 1 means that w and w' are identical, while a value of S equal to 0

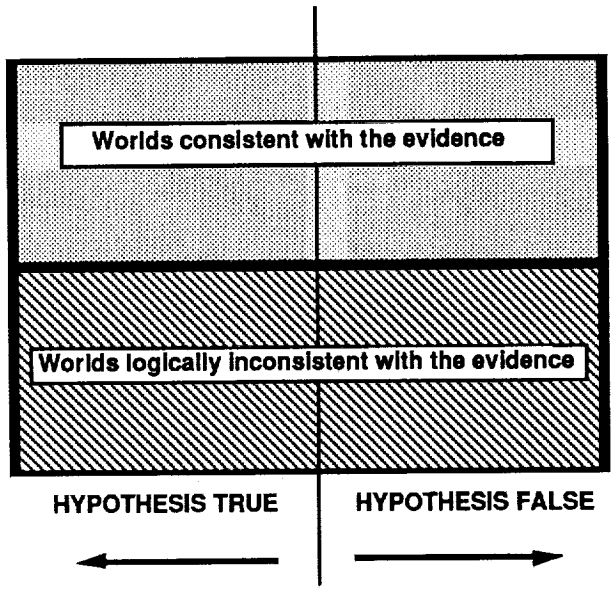


Figure 1: The approximate reasoning problem

indicates that knowledge of propositions that are true in one possible world does not provide any indication about the nature of the propositions that are true in the other.

In addition to the above requirement of reflexivity, i.e.

$$S(w, w) = 1\,,$$

we will need to impose additional axioms to assure that S captures the semantics of a similarity relation. In addition to assuming that S is symmetric, i.e.,

$$S(w, w') = S(w', w)\,,$$

we will also require that S satisfies a form of transitivity that is motivated by noting that if w, w' and w'' are possible worlds and if w is highly similar to w' and w' is highly similar to w'', then it would be surprising if w and w'' were highly dissimilar. This consideration indicates that knowledge of $S(w, w')$ and $S(w', w'')$ should provide a lower bound for values of $S(w, w'')$, as expressed by the inequality

$$S(w, w'') \geq S(w, w') \circledast S(w', w'')\,,$$

where $\circledast$ is a binary operator used to represent a real function that produces the required bound. If reasonable requirements are imposed upon the function $\circledast$, it is easy to show

that it has the properties of triangular norms: a class of functions that play a major role in multivalued logics [6].

The generalized transitivity property expressed by the above inequality may be easier to understand as a classical triangular inequality if it is noted that the function $\delta = 1 - S$ has the properties of a metric. When $\circledast$ is the Lukasïewicz norm

$$a \circledast b = \max(a + b - 1, 0)$$

, then the transitivity property of S is equivalent to the well-known triangular property

$$\delta(x, z) \leq \delta(x, y) + \delta(y, z)\,,$$

of distance functions. If $\circledast$ corresponds to the Zadeh triangular norm $a \circledast b = \min(a, b)$, then δ may be shown to satisfy the more stringent ultrametric inequality

$$\delta(x, z) \leq \max(\,\delta(x, y), \delta(y, z)\,)\,.$$

The correspondence between propositions and subsets of possible worlds simplifies the interpretation of the classical rule of modus ponens as a rule of derivation based on the transitive property of set inclusion. If three propositions p, q and r are such that the set of possible worlds where p is true is a subset of the set of possible worlds where q is true, and if such set is itself a subset of the set of worlds where r is true, then the modus ponens simply states that the set of p-worlds is a subset of the set of r-worlds.

The conventional relation of set inclusion, based on the binary truth-value structure of classical logic, allows only to state that a set of possible worlds is a subset of another or that it is not. Introduction of a metric structure in the universe of discourse, however, permits the quantification of the degree by which a set is included into another. Every set of possible worlds, as illustrated in Figure 2, is a subset of some neighborhood of any other set. The minimal amount of "stretching" that is required to include a set of possible worlds q in a neighborhood of a set of possible worlds p, given by the expression

$$\mathbf{I}(p \,|\, q) = \inf_{w' \vdash q} \sup_{w \vdash p} S(w, w')\,,$$

is called the *degree of implication*.

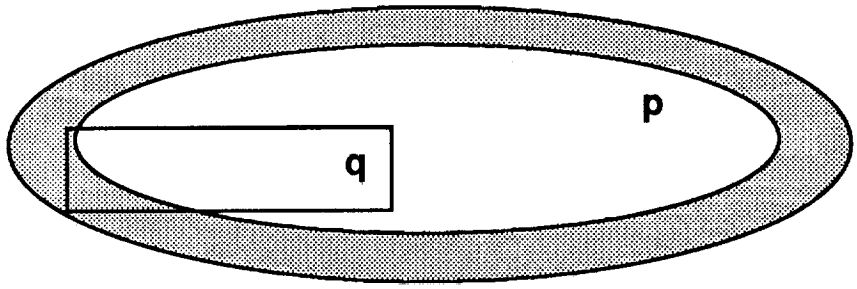


Figure 2: Degree of implication

The degree of implication function has the important transitive property expressed by

$$\mathbf{I}(p\,|\,q) \geq \mathbf{I}(p\,|\,r) \circledast \mathbf{I}(r\,|\,q)\,,$$

which is the basis of the generalized modus ponens of Zadeh. As illustrated in Figure 3, this important rule of derivation tells us how much the set of p-worlds should be stretched to encompass q on the basis of knowledge of the sizes of the neighborhoods of p that includes r and of r that includes q.

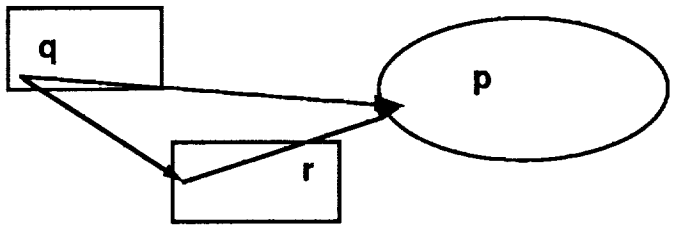


Figure 3: The generalized modus ponens

A notion dual to the degree of implication is that of *degree of consistence*, which quantifies the amount by which a set must be stretched to intersect another, and that is given by the expression

$$\mathbf{C}(p\,|\,q) = \sup_{w' \vdash q} \sup_{w \vdash p} S(w, w')\,.$$

POSSIBILISTIC DISTRIBUTIONS

Although the transitive property of the degree of implication essentially provides the bases for the conceptual validity of the generalized modus ponens, this rule of derivation is typically expressed by means of necessity and possibility distributions.

An *unconditioned necessity distribution* given the evidence $\mathscr{E}$ is any function defined over propositions that bounds by below the degree of implication function, i.e., any function satisfying the inequality

$$\mathbf{Nec}(p) \leq \mathbf{I}(p \,|\, \mathscr{E}) .$$

Correspondingly, an *unconditioned possibility distribution* is any upper bound for the degree of consistence function, i.e.,

$$\mathbf{Poss}(p) \geq \mathbf{C}(p \,|\, \mathscr{E}) .$$

The definition of conditional possibility and necessity distributions makes use of a form of inverse of the triangular norm denoted $\oslash$ and defined by the expression

$$a \oslash b = \sup\{\, c : \; b \circledast c \leq a \,\} .$$

Using this function, it is possible to define conditional possibilistic distributions as follows:
Definition: A function $\mathbf{Nec}(\cdot|\cdot)$ is called a *conditional necessity distribution* for $\mathscr{E}$ if

$$\mathbf{Nec}(q|p) \leq \inf_{w \vdash \mathscr{E}} [\mathbf{I}(q \,|\, w) \oslash \mathbf{I}(p \,|\, w)] .,$$

Definition: A function $\mathbf{Poss}(\cdot|\cdot)$ is called a *conditional possibility distribution* for $\mathscr{E}$ if

$$\mathbf{Poss}(q|p) \geq \sup_{w \vdash \mathscr{E}} [\mathbf{I}(q \,|\, w) \oslash \mathbf{I}(p \,|\, w)] .$$

GENERALIZED MODUS PONENS

The *compositional rule of inference* or generalized modus ponens of of Zadeh is a generalization of the corresponding classical rule of inference that may be used even when known facts do not match the antecedent of a conditional rule. The interpretation provided by our model explains the generalized modus ponens as an extrapolation procedure that uses

knowledge of the similarity between the evidence and a set of possible worlds p (the antecedent proposition), and of the proximity of p-worlds to q-worlds, to bound the similarity the latter to the evidential set. The actual statement of the generalized modus ponens for necessity and possibility distributions in terms of similarity structures makes use of a family $\mathscr{P}$ of satisfiable propositions that partitions the universe of discourse:

Theorem *(Generalized Modus Ponens for Possibility Distributions)*: Let $\mathscr{P}$ be a partition and let q be a proposition. If $\mathbf{Poss}(p)$ and $\mathbf{Poss}(q|p)$ are real values, defined for every proposition p in $\mathscr{P}$, such that

$$\mathbf{Poss}(p) \geq \mathbf{C}(p \,|\, \mathscr{E}), \quad \mathbf{Poss}(q|p) \geq \sup_{w \vdash \mathscr{E}} [\mathbf{I}(q \,|\, w) \oslash \mathbf{I}(p \,|\, w)],$$

then the following inequality is valid:

$$\sup_{\mathscr{P}} [\mathbf{Poss}(q|p) \circledast \mathbf{Poss}(p)] \geq \mathbf{C}(q \,|\, \mathscr{E}).$$

Theorem *(Generalized Modus Ponens for Necessity Distributions)*: Let $\mathscr{P}$ be a partition and let q be a proposition. If $\mathbf{Nec}(p)$ and $\mathbf{Nec}(q|p)$ are real values, defined for every proposition p in $\mathscr{P}$, such that

$$\mathbf{Nec}(p) \leq \mathbf{I}(p \,|\, \mathscr{E}), \quad \mathbf{Nec}(q|p) \geq \inf_{w \vdash \mathscr{E}} [\mathbf{I}(q \,|\, w) \oslash \mathbf{I}(p \,|\, w)],$$

then the following inequality is valid:

$$\sup_{\mathscr{P}} [\mathbf{Nec}(q|p) \circledast \mathbf{Nec}(p)] \leq \mathbf{I}(q \,|\, \mathscr{E}).$$

VARIABLES AND FUZZY RULES

If our attention is restricted to propositions of the form "X = x," describing the value of a variable X, and to logical combinations of these propositions, then a possibility distribution $\Pi_{Y|X}$ may be regarded, as is well known, as an elastic constraint that restricts the values of a variable Y on the basis of general background information (the evidence $\mathscr{E}$) and knowledge about possible values of another variable X.

In our similarity-based interpretation, this notion of elastic constraint is easier to understand (Figure 4) by means of the concept of compatibility relation that associates specific

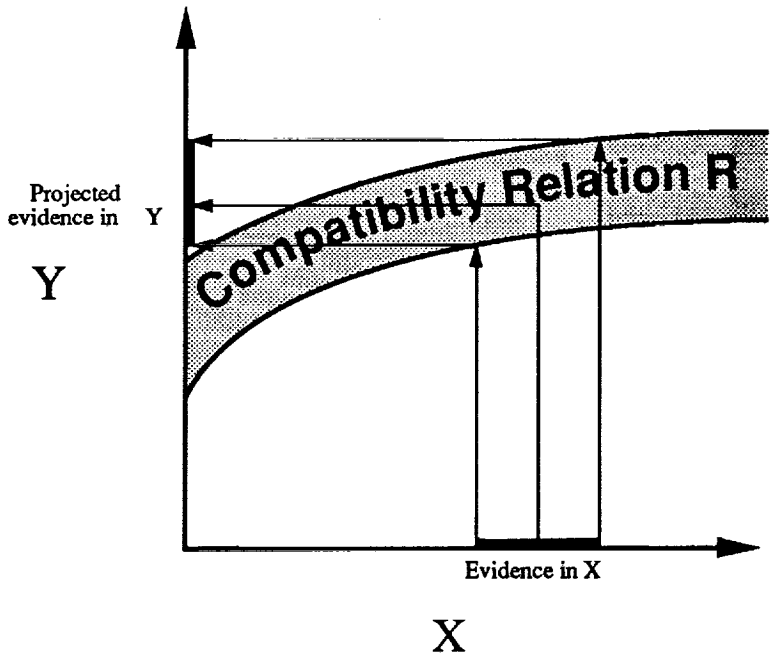


Figure 4: Compatibility Relation.

values of one variable (X) with possible values of another (Y). Using this basic notion, we may now describe two major interpretations of fuzzy rules as its similarity-based approximations by means of fuzzy-set theoretic structures.

The first interpretation, originally proposed by Zadeh [8] and further developed by Trillas and Valverde [6], is the formal translation of the statement

$$\text{If } \mu_A \text{ is a possibility for } X, \text{ then } \mu_B \text{ is a possibility distribution for } Y.$$

Using our structures we may define this particular formulation by saying that

$$\mathbf{Poss}(y|x) = \mu_B(y) \oslash \mu_A(x) \geq \mathbf{I}(y \,|\, w) \oslash \mathbf{I}(x \,|\, w)\,,$$

for every world $w \vdash \mathscr{E}$, i.e., that $\mathbf{Poss}(\cdot|\cdot)$ is a conditional possibility distribution. This distribution expresses a basic relationship between the similarity between possible evidential worlds and the core of μ_B as a "fraction" of their similarity with the core of μ_A.

Under this interpretation, the fuzzy-rule based approximation to a compatibility relation may be depicted as done in Figure 5, where it has been assumed that the underlying metric

(i.e., dissimilarity) is proportional to the euclidean distance in the plane. As illustrated in that figure, the core of the corresponding conditional possibility distribution is an (upper) approximant of a classical compatibility relation (which fans outward from the Cartesian product of the cores of A and B). Whenever several such rules are available, then each one of these rules will lead to a separate possibility distribution, which may be illustrated, as done in 7, as an approximating fuzzy relation. Combination of these estimates by intersection results in a sharper "integrated" estimate of the effect of a rule set.

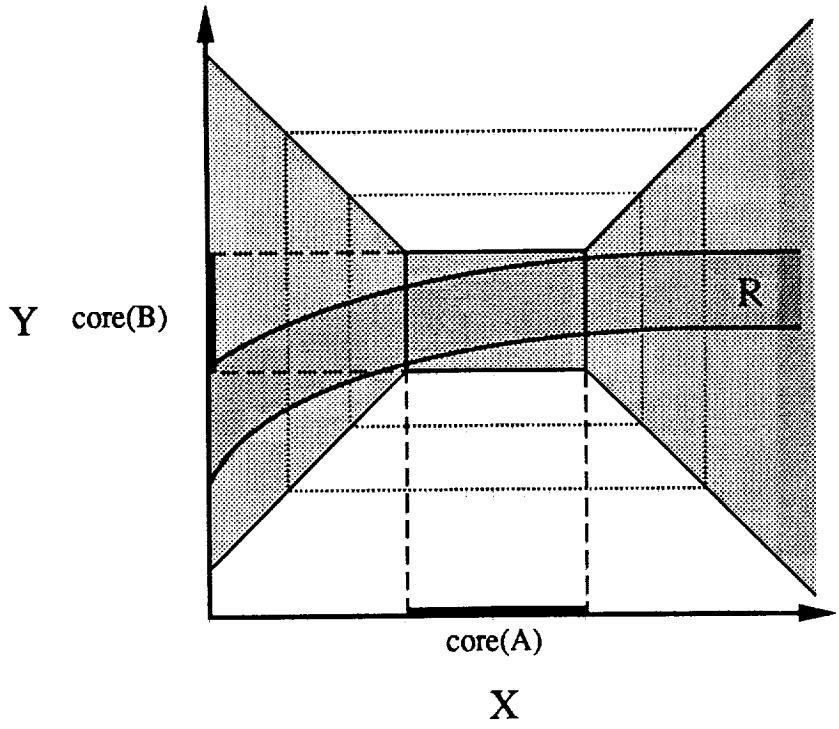


Figure 5: Rules as Possibilistic Approximants of a Compatibility Relation.

The second interpretation, originally propoded by Zadeh [7], was first applied by Mamdani and Assilian [2] to design fuzzy controllers, being also the basis for a wide variety of recent industrial products [5]. In this formulation, a number of statements of the form

$$\text{If } X \text{ is } A_k, \text{ then } Y \text{ is } B_k, \qquad k = 1, 2, \ldots, n,$$

are interpreted as a combined "disjunctive" description of the compatibility relation, rather

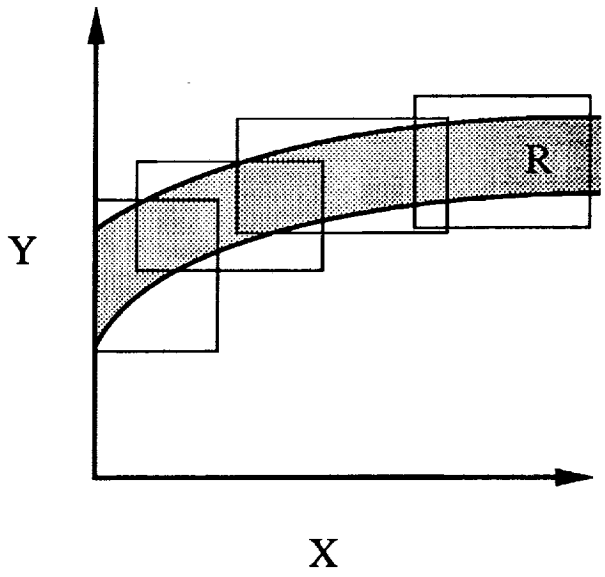


Figure 6: Rule-Sets as Possibilistic Approximants of a Compatibility Relation

than as a set of independently valid rules, as shown in Figure 6. In this case, each disjunctive approximant, corresponding to a fuzzy relation such as that illustrated in Figure 8 (with the relation "slopping" away from the cartesian products of the core of the fuzzy sets) is combined disjunctively by fuzzy set union with the other approximants.

CONCLUSION

Models based on the logical notion of possible-world provide interpretations of the major concepts and structures of fuzzy logic in terms of primitive notions of similarity and resemblance. These interpretations clearly show the basic nature of the difference between possibilistic, which are based on metrics, and probabilistic methods, which are based on set measures.

ACKNOWLEDGMENT

This work was supported in part by the US Air Force Office of Scientific Research under Contract No. F49620-89-K-0001 and in part by the US Army Research Office under Contract

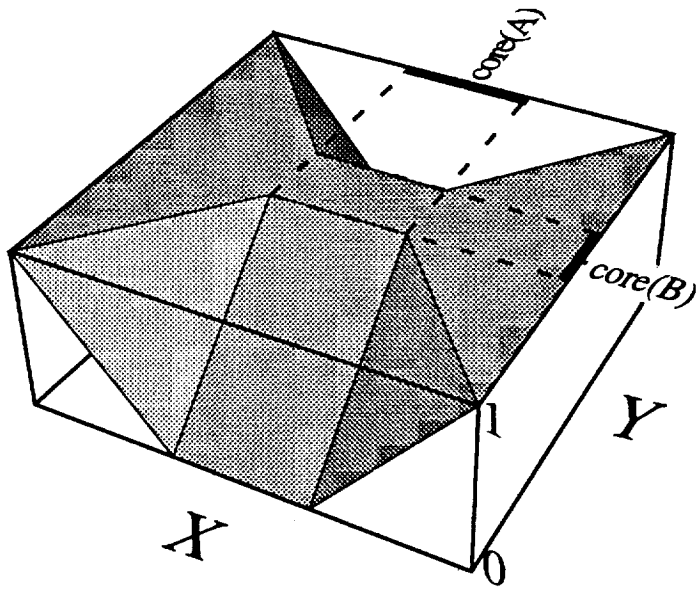


Figure 7: A Possibilistic Conjunctive Conditional Rule

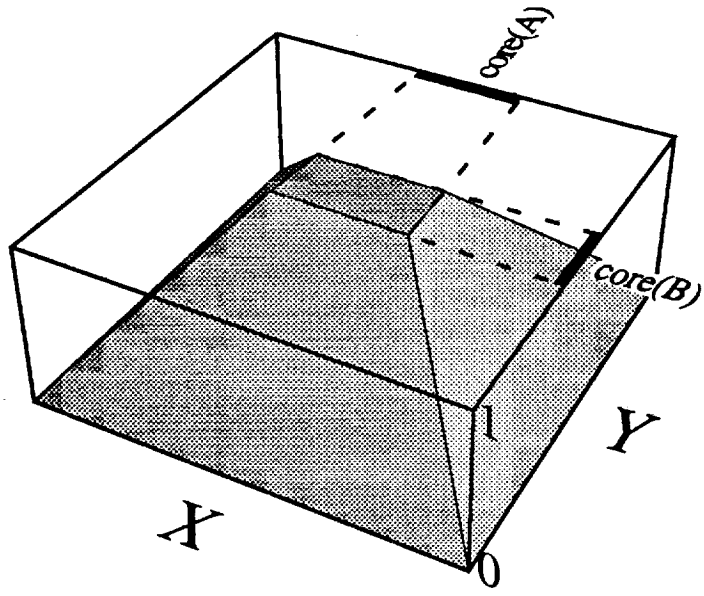


Figure 8: A Component of a Disjunctive Rule Set

No. DAAL03-89-K-0156. The views, opinions and/or conclusions contained in this note are those of the author and should not be interpreted as representative of the official positions, decisions, or policies, either express or implied, of the Air Force Office of Scientific Research, the Department of the Army, or the United States Government.

REFERENCES

[1] G.E. Hughes and M.J. Creswell. *An Introduction to Modal Logic.* New York: Methuen, 1972.

[2] E.H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *Int. J. Man-Machine Studies*, 7:1–13, 1975.

[3] E.H. Ruspini. *On the Semantics of Fuzzy Logic.* Technical Note No. 475, Artificial Intelligence Center, SRI International, Menlo Park, California, 1989.

[4] E.H. Ruspini, *Approximate Reasoning: Past, Present, Future.* Information Sciences, forthcoming, 1990.

[5] M. Sugeno. *Industrial Applications of Fuzzy Control.* Amsterdam: North Holland, 1985.

[6] E. Trillas and L. Valverde. On mode and implication in approximate reasoning. In M.M. Gupta, A. Kandel, W. Bandler, J.B. Kiszka, editors, *Approximate Reasoning and Expert Systems*, Amsterdam: North Holland, 157–166, 1985.

[7] L.A. Zadeh. A rationale for fuzzy control. *Journal of Dynamic Systems, Measurement and Control*, C94: 3–4, 1972.

[8] L.A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

[9] L.A. Zadeh. A theory of approximate reasoning. In D. Michie and L.I. Mikulich, editors, *Machine Intelligence 9*, New York: Halstead Press, 149–194, 1979.

THE SEMANTICS OF FUZZY LOGIC

Enrique H. Ruspini
Artificial Intelligence Center
SRI International

Fuzzy Logic and Neural Networks Workshop
NASA JSC, Houston, April 1990

MOTIVATION

- Provide basic characterization of Possibilistic concepts
  - ° Possibility and Necessity Distributions
  - ° Possibistic Calculus
  - ° Inferential Rules (GMP)
- Determine analogies and differences with Probabilistic Reasoning Methods
  - ° Unified Approach to Interpretation
  - ° Needs for specific formalisms/theoretical structures

FUZZY LOGIC MAY BE FORMALLY EXPLAINED BY METRIC CONCEPTS AND STRUCTURES:

Similarity, Proximity, Closeness, Resemblance

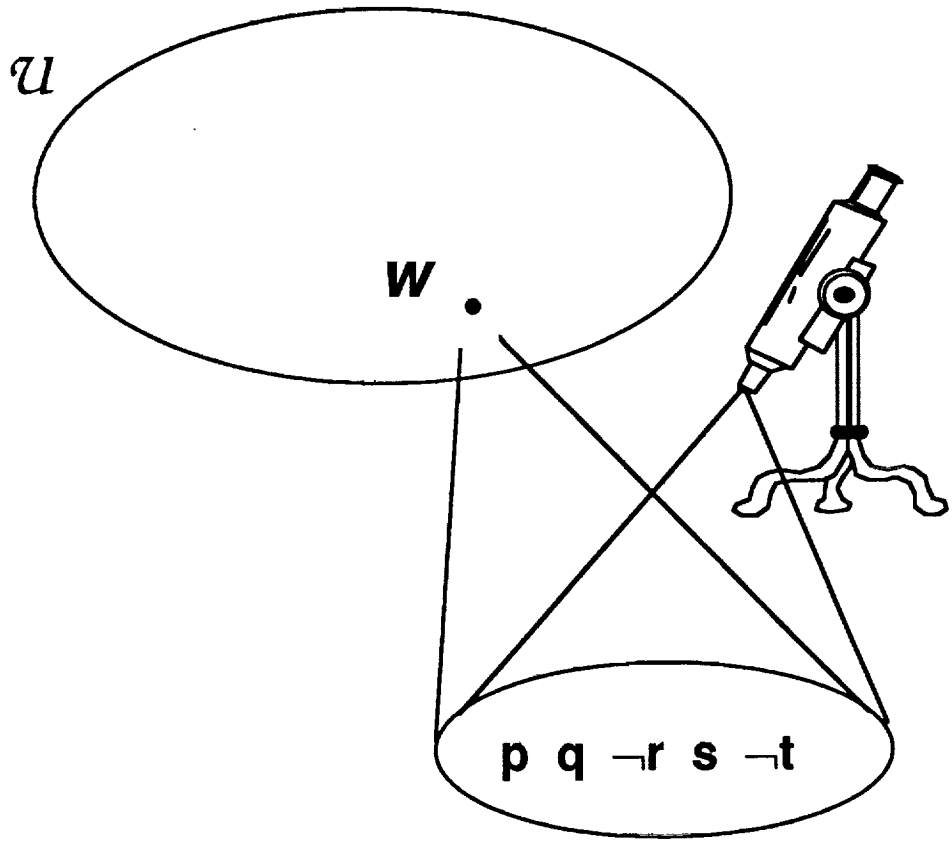
POSSIBLE WORLDS

- Possible States, Behaviors, Trajectories of a Conceptual System that is being reasoned about

  Examples: Weather System, Vehicle Control System, Portfolio Status

  - Formally equivalent to a Valuation:
  - Assignment of truth-values (i.e., T, F) to all relevant propositions about the state of system
  - Consistent with rules of logic

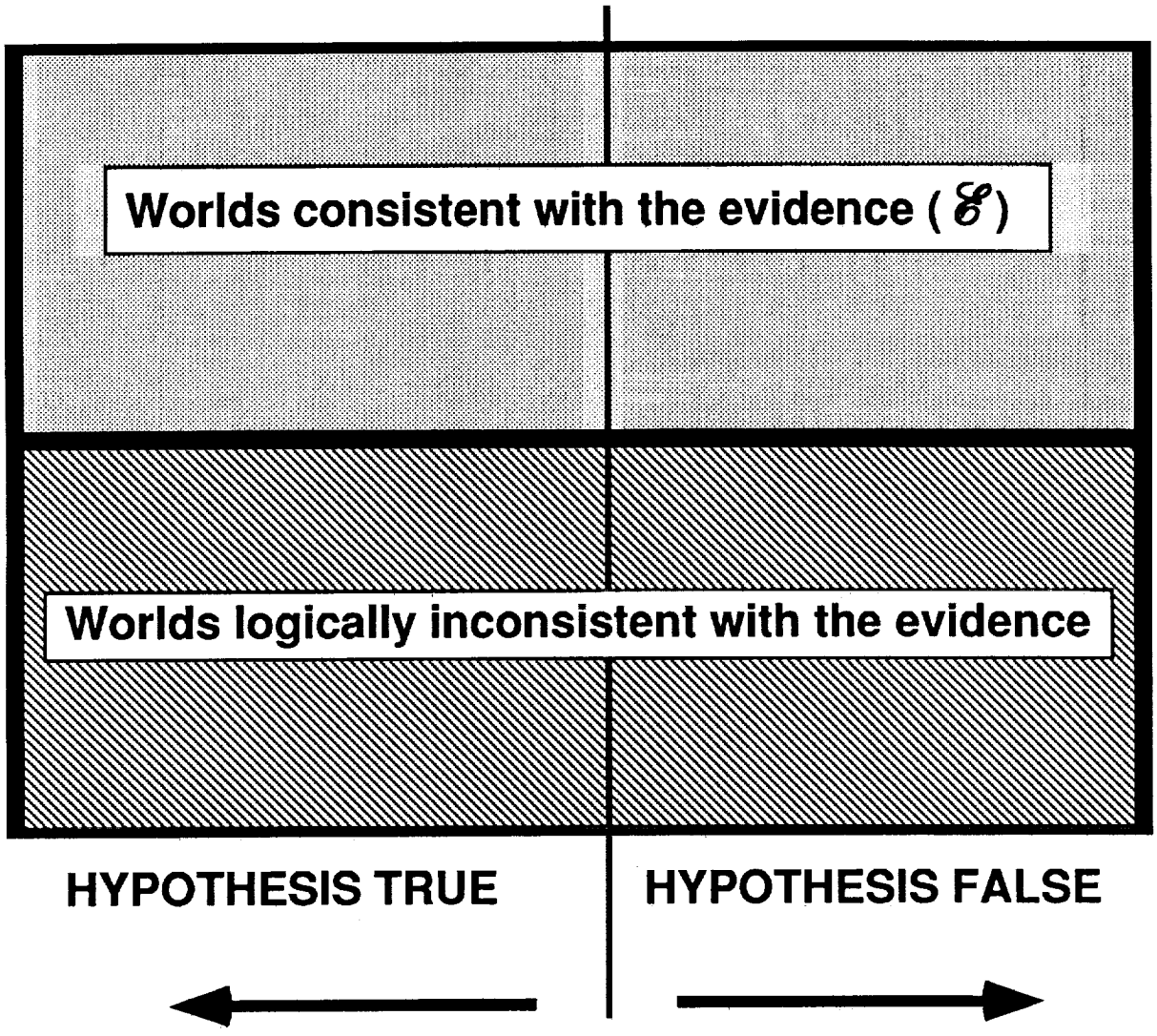
- Universe ≡ Set of all Possible Worlds



Enrique H. Ruspini
Artificial Intelligence Center

THE APPROXIMATE REASONING PROBLEM

- Conventional deductive methods fail to unambiguously determine the truth-value of a proposition of interest (hypothesis).



Enrique H. Ruspini
Artificial Intelligence Center

APPROXIMATE REASONING METHODS

- Describe properties of the Evidential Set $\mathscr{E}$

Probabilistic Reasoning :

- Based on the use of additive set measures
- Concerned with (objective or subjective) proportions of occurrence of certain events, e.g.,

$$\mu(H \mid \mathscr{E}) + \mu(\neg H \mid \mathscr{E})$$

Possibilistic Reasoning:

- Based on metric notions (distance, similarity, proximity)
- Uses measures of resemblance between subsets of possible worlds
- Oriented toward characterization of conceptual flexibility, typicality, proximity, degree of fitness

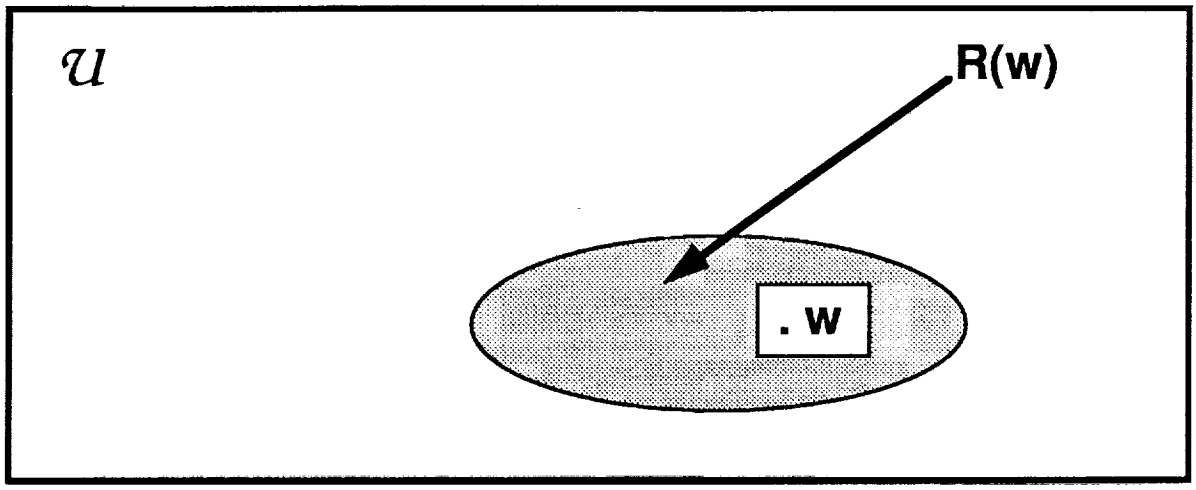
Semantic Formulation of Modal Logic

- **Basic Elements**:
  - $\mathcal{U}$: A set of possible worlds (the **universe**)
  - $\mathcal{V}$: a valuation (mapping pairs of possible worlds and propositions into truth values), e.g.,

    (w, "it rains") $\rightarrow$ T
  - R: A binary relation (between pairs of possible worlds) called the **conceivability**, **reachability**, or **accessibility** relation

- **POSSIBILITY AND NECESSITY** :
  - p is **possible** in w (w |- Πp) if and only if p is true in **some** world w' that is related to w
  - p is **necessary** in w (w |- Np) if and only if p is true in **every** world w' that is related to w
  - Different properties of R lead to different modal systems (T, S4, S5)

INTERPRETING ACCESSIBILITIES



- R= $\mathcal{U} \times \mathcal{U}$: Conventional notion of logical necessity
- O = {a propositional subset} (the "observables")

  R(w,w') if w and w' share the same "observations"
  (Necessity then models rational knowledge)
- Inevitability: Two worlds are related if they are identical up to some point in time
- Cognitive capability
- Moral Necessity
- Linguistic Modalities

WE ARE INTERESTED IN MODELING THE ABILITY OF POSSIBLE WORLDS TO EXEMPLIFY CERTAIN CONDITIONS

Enrique H. Ruspini
Artificial Intelligence Center

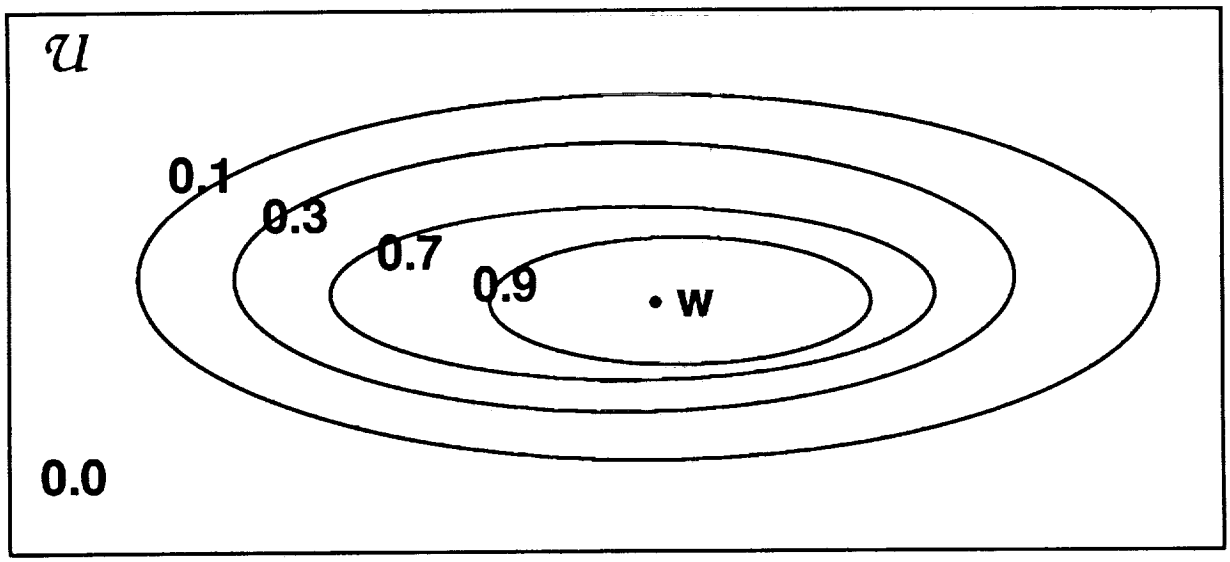
MULTIPLE ACCESSIBILITY RELATIONS

- Generalize notion of accessibility relation by consideration of a family of relations indexed by a numerical parameter between 0 and 1
- Modeling Objective:

  Define resemblance between situations so as to allow a form of analogical reasoning

  Example: *Investment advice for S (Wealth=$1,000,000) is valid (to some extent) for S' (Wealth=$999,999)*

  We want to be able to describe behavioral rules that are valid in neighborhoods of sets of possible worlds



- Defined by means of a similarity function

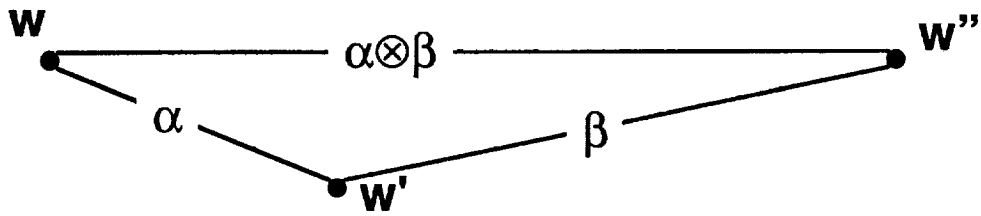
SIMILARITY FUNCTIONS

- Assigns a similarity value to pairs of possible worlds

$$S: W \times W \rightarrow [0,1]$$

  - S(w,w')=1 means that w is identical to w'
  - S(w,w')=0 means that w and w' are completely dissimilar

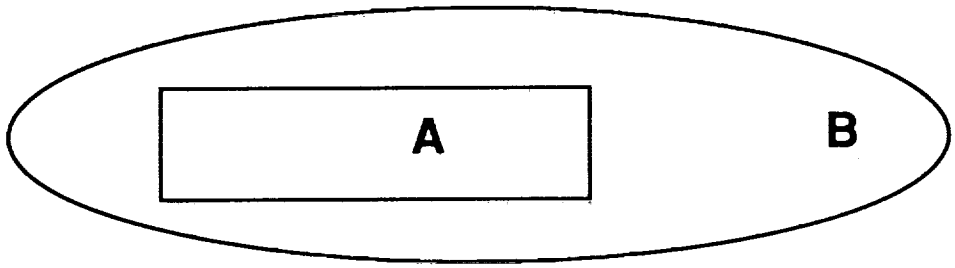
- Properties of Similarity Relations:
  - S(w,w') =1 if and only if w=w' (Reflexivity)
  - S(w,w') = S(w',w) (Symmetry)
  - $S(w,w'') \geq S(w,w') \otimes S(w',w'')$



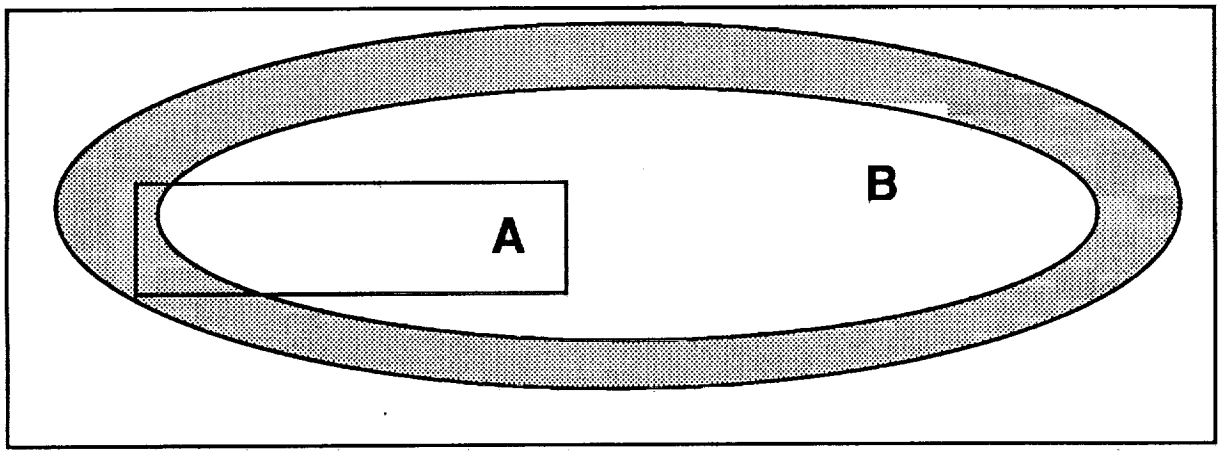
- Imposition of reasonable requirements indicates that $\otimes$ should have the properties of a continuous triangular norm (T-norm).
- $\delta = 1 - S$, is a distance function.

LOGIC and METRICS

- Metric structures allow to characterize implications between propositions (i.e., subset inclusions) in terms of similarities between subset elements:
  - If B $\supseteq$ A, then every point of A has one point of B (i.e., itself) that is similar to it to the degree 1



  - In general, every point of a subset A is similar to some degree to a point of B (i.e., falls in some neighborhood of B)



MODELS (Good, Bad, and otherwise)

- MODELS (MODAL LOGIC) :
  - q is a model of p (q |=p) if and only if every q-world is a p-world, i.e.,

$$q \Rightarrow p$$

- GENERALIZED MODELS :
  - q is a necessary model of p to the degree α if and only if every q-world is α-similar to a p-world, i.e.

$$q \Rightarrow \Pi_{\alpha} p$$

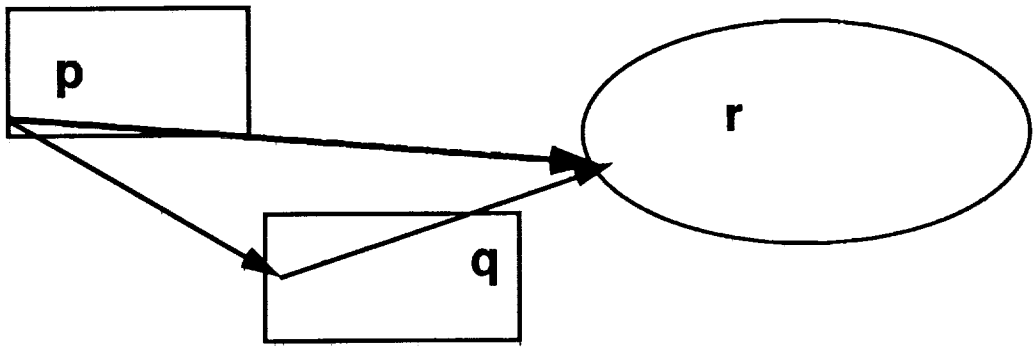
("De Re" Interpretation)

- To what degree q is a necessary model or a good example of p? ("Degree of Implication")

$$I(p \mid q) = \inf_{w \mid -q} \sup_{w' \mid -p} S(w, w')$$

DEGREE OF IMPLICATION

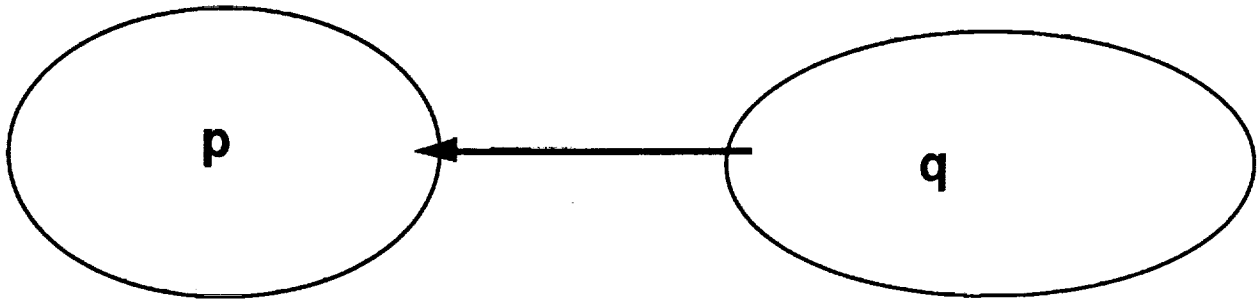
- I(p|q) measures the extent by which <u>any</u> q-world resembles <u>some</u> p-world
- Degree by which was is true in one set <u>must</u> apply on another
- Properties of I(p|q):
  - If $p \Rightarrow r$, then $I(p|q) \leq I(r|q)$,
  - If $q \Rightarrow r$, then $I(p|q) \geq I(p|r)$,
  - $I(p|q) \geq I(p|r) \otimes I(r|q)$,
  - $I(p|q) = \sup_r [I(p|r) \otimes I(r|q)]$
- Basis for <u>generalized modus ponens</u>



DEGREE OF CONSISTENCE

$$C(p \mid q) = \sup_{w \mid\!- q} \; \sup_{w' \mid\!- p} \; S(w, w')$$

- "Dual" of the degree of implication function
- Measures extent by which true propositions in one set may apply on another



UNCONDITIONED POSSIBILITY DISTRIBUTIONS

- Upper bounds of $C(p|\mathcal{E})$

$$C(p|\mathcal{E}) \leq Poss(p)$$

UNCONDITIONED NECESSITY DISTRIBUTIONS

- Lower bounds of $I(p|\mathcal{E})$

$$Nec(p) \leq I(p|\mathcal{E})$$

$$\boxed{Nec(p) \leq I(p|\mathcal{E}) \leq C(p|\mathcal{E}) \leq Poss(p)}$$

Inverse of a T-Norm

$$a \oslash b = \sup\{c : b \otimes c \leq a\}$$

$a \otimes b$	$a \oslash b$
$\max(a+b-1,0)$	$\min(1+a-b,1)$
ab	a/b, if $b > a$, 1, otherwise
$\min(a,b)$	a, if $b > a$, 1, otherwise

Conditional Distributions

- **Conditional Necessity**

$$\text{Nec}(q|p) \leq \inf_{w|-\mathcal{S}} [I(q|w) \varnothing I(p|w)]$$

- **Conditional Possibility**

$$\text{Poss}(q|p) \geq \sup_{w|-\mathcal{S}} [I(q|w) \varnothing I(p|w)]$$

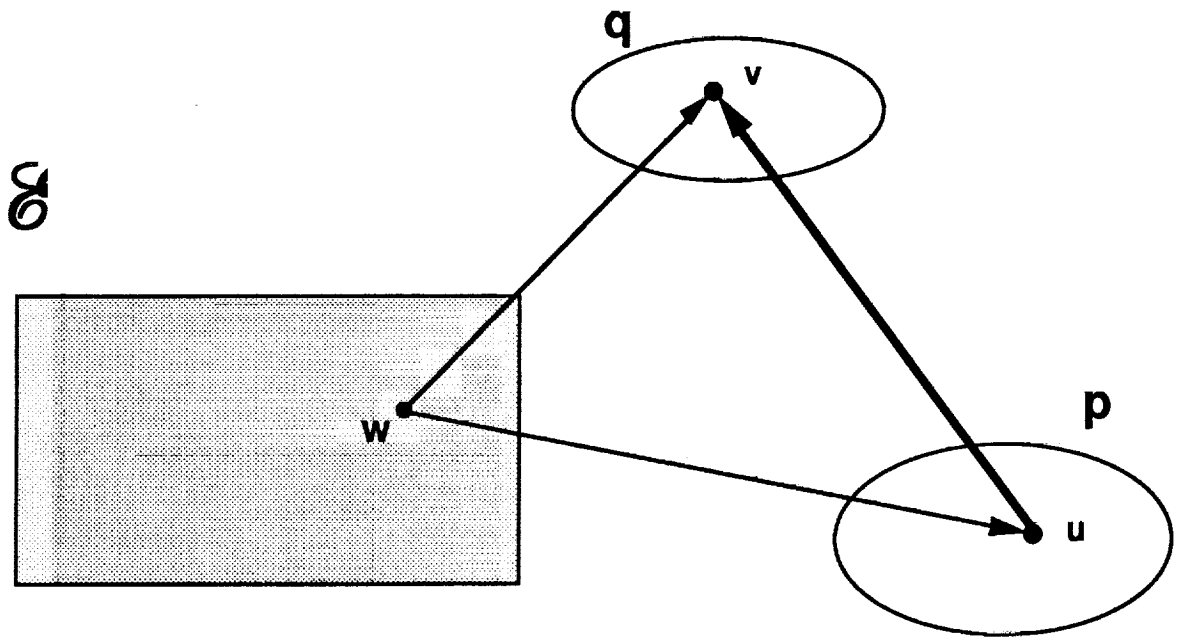
The conditional distributions measure the extent by which similarity to the consequent must or may (respectively) exceed the similarity to the antecedent

Enrique H. Ruspini
Artificial Intelligence Center

Generalized Modus Ponens

$$\sup_{\mathscr{P}} [\text{Nec}(q|p) \otimes \text{Nec}(p)] \leq I(q|\mathscr{E})$$

$$\sup_{\mathscr{P}} [\text{Poss}(q|p) \otimes \text{Poss}(p)] \geq C(q|\mathscr{E})$$



Enrique H. Ruspini
Artificial Intelligence Center

POSSIBLE WORLDS and VARIABLES

- Possible Worlds will be characterized by means of a number of variables X, Y, ...
- Each variable X takes values in a well-defined domain $\mathscr{D}(X)$, e.g.,

$$\mathscr{D}(\text{Color}) = \{ \text{Green, Red, Blue, ...} \}$$

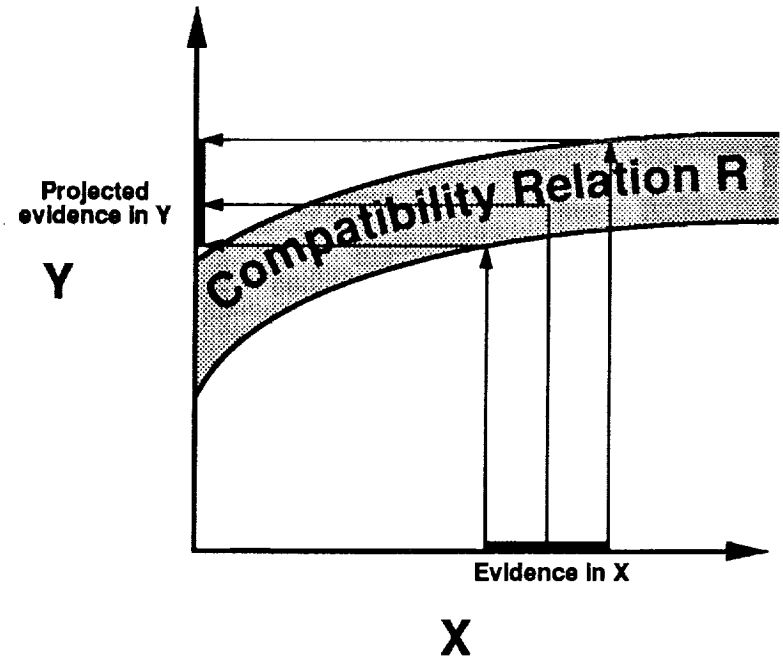
- Possible worlds correspond to a complete specification of variable values
- Partial specification of variable values defines a subset of possible worlds
- The propositions of interest are those of the form:

  "X is x," "Y is y," .....

  and their logical combinations (conjunctions, disjunctions, ...)
- [X=x] denotes the subset of all worlds where X=x

Compatibility Relations

- Define relationship between values of two system variables (in the "actual" world)
- Permits the derivation of possible values of Y from knowledge of possible values of X
- Constrain the extent of the evidential set



CONDITIONAL POSSIBILITIES from FUZZY RULES

- *If X is A, then Y is B*

- Interpretation:
  - If $\mathscr{E}$ is such that

$$\text{Poss}(x|\mathscr{E}) = \mu_A(x) \geq C(x|\mathscr{E}).$$

  then

$$\mu_B(y) \geq C(y|\mathscr{E}).$$

- The function $\Pi(y|x)$ defined by

$$\Pi(y|x) = \mu_B(y) \varnothing \mu_A(x),$$

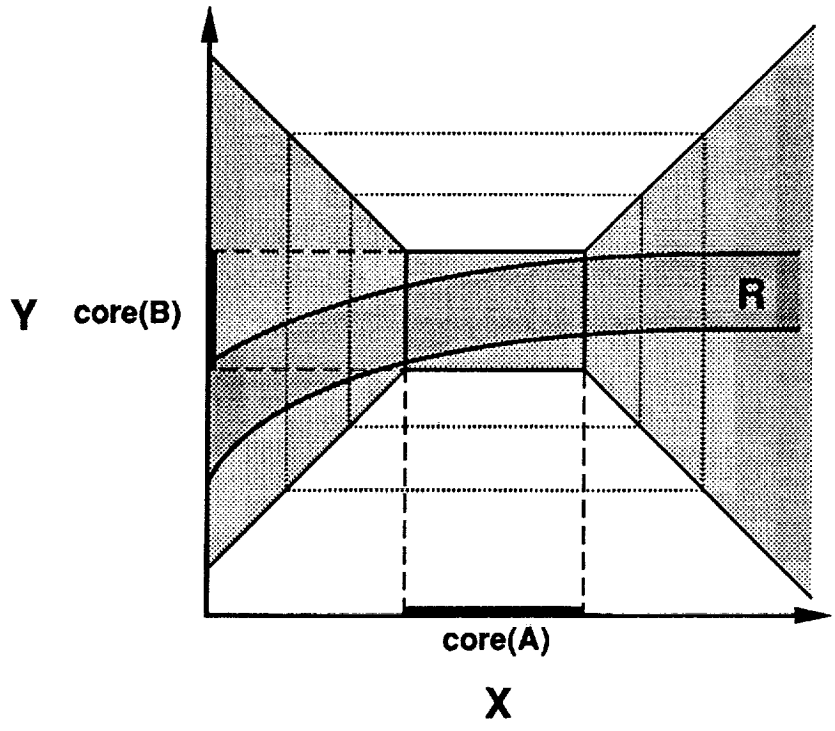
is a <u>conditional possibility</u> for y given x

Enrique H. Ruspini
Artificial Intelligence Center

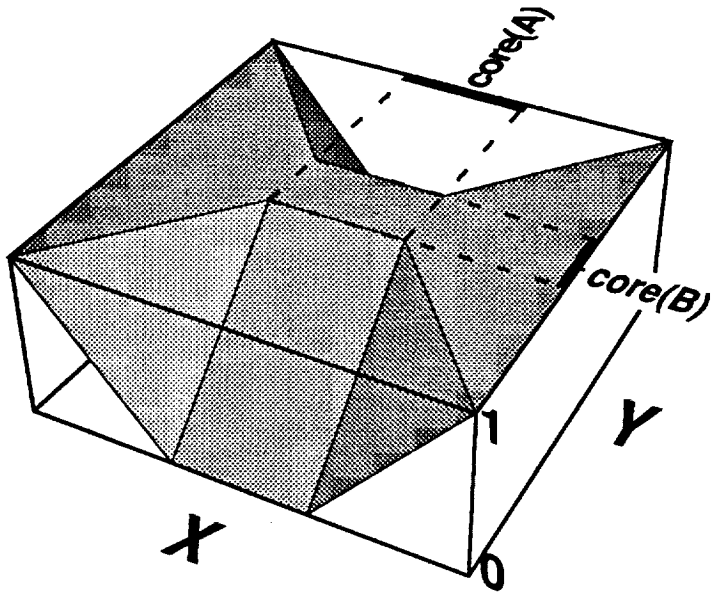
Logical Interpretation of a Conditional Rule

(Zadeh-Trillas-Valverde)

- **The conditional possibility is an "enclosing" approximation of the compatibility relation**



ZTV Interpretations as Fuzzy Relations

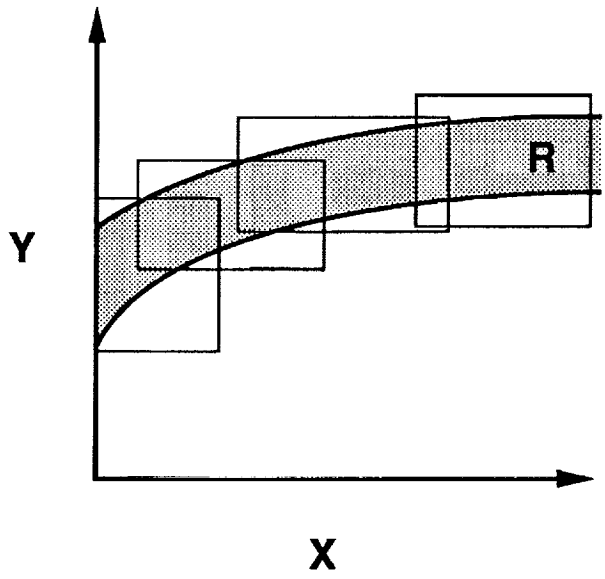


Enrique H. Ruspini
Artificial Intelligence Center

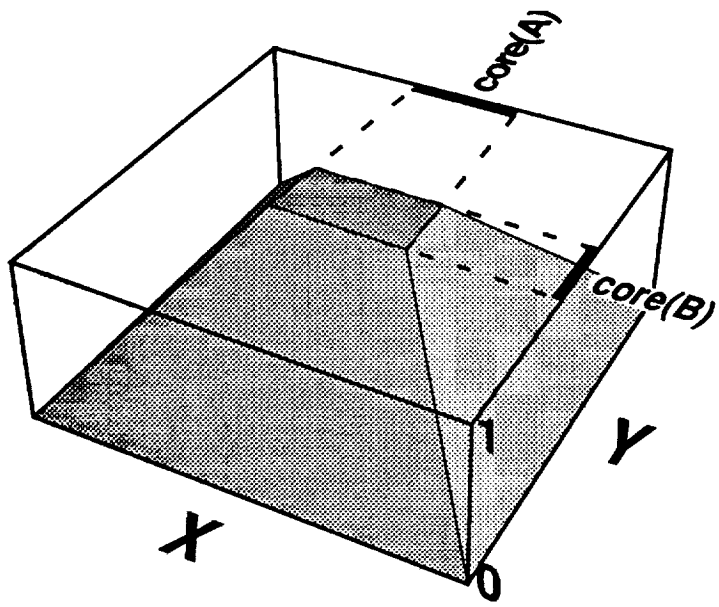
Disjunctive Interpretations of Conditional Possibility Relations

(Zadeh-Mamdani-Assilian)

- "If X is A, then Y is B" is interpreted as one of a set of regions that must be combined (by disjunction) to approximate the compatibility relation
- Relation is characterized as a "set of points" rather than as the intersection of constrainning regions



ZMA Disjunctive Approximants as Fuzzy Relations



Enrique H. Ruspini
Artificial Intelligence Center

Autonomous Robotics Research at the Artificial Intelligence Center

- FLAKEY
  - Successor of the pioneer autonomous robot SHAKEY
  - Technological Emphasis:
    - Autonomous Navigation
    - Autonomous Planning/Replanning
    - Multiple Intercommunicating Agents
    - Explicit Representation of Knowledge States
    - Integration of Sensing Activities into Plans
    - Learning
  - Fuzzy Logic/Neural Network Investigations:
    - Rule-based "blending" of Local Behaviors
    - Flexible Navigation
    - Flexible Planning/Replanning
    - NN-Based Learning

Enrique H. Ruspini
Artificial Intelligence Center

Identification, Estimation and Control of Dynamical Systems
with the Parametric Avalanche Neural Network

(Paper not provided by publication date.)

AUTHOR INDEX

NASA
National Aeronautics and Space Administration

REPORT DOCUMENTATION PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
CP 10061		

4. Title and Subtitle	5. Report Date
Proceedings of the Second Joint Technology Workshop on Neural Networks and Fuzzy Logic	April 1990
	6. Performing Organization Code PT4
7. Author(s)	8. Performing Organization Report No.
Robert N. Lea, Editor James Villarreal, Editor	S-624
9. Performing Organization Name and Address	10. Work Unit No.
Lyndon B. Johnson Space Center Information Technology Division Houston, TX 77058	11. Contract or Grant No.
12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
National Aeronautics and Space Administration Washington, D.C. 20546	Conference Publication
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

Documented here are papers presented at the Neural Networks and Fuzzy Logic Workshop sponsored by the National Aeronautics and Space Administration and cosponsored by the University of Houston, Clear Lake. The workshop was held April 11 - 13 at the Lyndon B. Johnson Space Center in Houston, Texas. During the three days, approximately 30 papers were presented. Technical topics addressed included adaptive system; learning algorithms; network architectures; vision; robotics; neurobiological connections; speech recognition and synthesis; fuzzy set theory and application, control and dynamics processing; space applications; fuzzy logic and neural network computers; approximate reasoning; and multiobject decision making.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
fuzzy logic, non-Lipschitzian dynamics, parrallel distributed models, algoriths, neural network, spatiotemporal patterns, neuron ring, buzzy controllers, signal processing, pattern recognition	Unclassified - unlimited Subject Category ????

19. Security Classification (of this report)	20. Security Classification (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified		

For sale by the National Technical Information Service, Springfield, VA 22161-2171