SELF-ORGANIZING NEURO-FUZZY CONTROL FOR UNDERWATER ROBOTIC VEHICLES

التحكم الفازى من خلال الشبكات العصبية للغواصات الآلية المنظم ذاتيا

A. Yassien* , M. Sherif* , S. Saraya* , F. Fahmy* , M.R. El-Basyouni**
*Corresponding authors are at the Computers & Systems Dept., Faculty of Engineering, Mansoura University.
** Faculty of Specific Education, Mansoura University

خلاصة :

نظرا للأهمية التي تحتلها الأذرع الآلية بالمقارنة بالآليات التي يتحكم بها الانسان فقد تم البحث في كيفية عمل متحكمات ذات مستوى كفاءة متميز يحاكى الديناميكية غير الخطية و المتغيرة في وحدة الزمن للأذرع الآلية. و من هنا وجب ايجاد نظام تحكم متطور بإمكانية التعلم و التكيف للتغيرات في ديناميكية الأذرع الآلية. نظام التحكم الفازي باستخدام الشبكات العصبية يتم من خلاله إستنباط و تهيأة القواعد أثناء التعلم من خلال عملية التعرف على الهيكل و عناصر النظام المقترح. وتتم هاتين العمليتين في ان واحد لضمان سرعة النظام. مرحلة التعرف على الهيكل يتم خلالها تحديد هيكل القواعد الفازية أما مرحلة تعلم العناصر فهي لتغيير و تعديل معاملات كل قاعدة للوصول للنظام ذو الكفاءة المطلوبة.

ABSTRACT:

Underwater robotic vehicles have become an important tool for various underwater tasks because they have greater speed, endurance, and depth capability, as well as a higher factor of safety, than human divers. However, most vehicle control system designs have been based on a simplified vehicle model, which has often resulted in poor performance because the nonlinear and time-varying vehicle dynamics have parameters uncertainty. It is desirable to have an advanced control system with the capability of learning and adapting to change in the vehicle dynamics and parameters. The proposed system is possessing neural network's learning ability. There are no rules initially in the proposed system. They are created and adapted as on-line learning proceeds via simultaneous structure and parameter identification. The identification process of the controller includes both structure and parameter learning.

1. INTRODUCTION

In the past decades, there is growing interest in neuro-fuzzy systems (NFS) as they continue to find success in a wide range of applications. Unfortunately, scientists have discovered that most existing neuro-fuzzy systems [1,2,3]exhibit several major drawbacks that may eventually lead to performance degradation. One of the drawbacks is the curse of dimensionality or fuzzy rule explosion. This is an inherent problem in fuzzy logic control systems; that is, too many fuzzy rules are used to approximate the input-output function of the system because the number of rules grow exponentially with the number of input and output variables. The second one is their lack of ability to extract input-output knowledge from a given set of training data. Since neuro-fuzzy systems are trained by numerical input-output data, the cause-effect knowledge is hidden in the training data and is difficult to be extracted. The third one is their inability to re-structure their internal structure; that is, the fuzzy term sets and the fuzzy rules in their hidden layers. The key advantage of neural fuzzy approach over traditional ones lies on that the former doesn't require a mathematical description of the system while modeling. Moreover, in contrast to pure neural or fuzzy methods, the neural fuzzy method possesses both of their advantages: it brings the low-level learning and computational power of neural networks into fuzzy systems and provides the high-level human-like thinking and reasoning of fuzzy systems into neural networks [3]–[5].

A fuzzy system consists of a bunch of fuzzy IF-THEN rules. Conventionally, the selection of fuzzy IF-THEN rules often relies on a substantial amount of heuristic observation to express proper strategy's knowledge. Obviously, it is difficult for human experts to examine all the input–output data

Accepted January 20, 2002

from a complex system to find a number of proper rules for the fuzzy system. To cope with this difficulty, several approaches to generating fuzzy IF-THEN rules from numerical data, an active research topic in the neural fuzzy area, have been proposed [2],[3].[5]–[12]. Generally, these approaches consist of two learning phases, the structure learning phase and the parameter learning phase. The structure as well as the parameter learning phases are done simultaneously in the proposed self-constructing neural fuzzy inference network by performing them both for each incoming data. This ability makes the proposed controller suitable for fast on-line learning.

One important task in the structure identification of a neural fuzzy network is the partition of the input space, which influences the number of fuzzy rules generated. We propose in this paper an on-line input space partitioning method, which is an aligned clustering-based approach. This method can produce a partition result like the one shown in Fig.1.Basically, it aligns the clusters formed in the input space, so it reduces not only the number of rules but also the number of membership functions under a pre-specified accuracy requirement. The proposed method creates only the significant membership functions on the universe of discourse of each input variable by using a fuzzy measure algorithm.

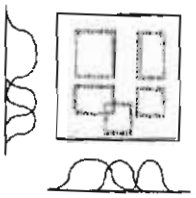


Figure 1. A partition result using Aligned clustering based approach

2.STRUCTURE OF THE VEHICLE :

The vehicle is a closed-framed sphere with eight thrusters and one manipulator (see Figure 2). It is capable of maneuvering with six degrees-of-freedom (DOF) motion: sway, surge, heave, roll, pitch, and yaw, and it can continuously operate for 3 hours with 25 rechargeable (5 v) lead-acid batteries. Vehicle's onboard assembly is a compact and efficient unit located on the upper mounting dish within the spherical hull. The overall architecture was designed for easy expansion to include additional components. Four vertical thrusters and four horizontal thrusters are used for the omni-directional motion. Each vertical thruster is paired with a horizontal thruster, and the four pairs are positioned 90 degrees apart along the equator of the spherical vehicle as shown in Fig. 2b(b).[13]

The vehicle possesses inherent which has been developed with a vehicles hydrodynamic thruster redundancy. Vertical motion is possible using all vertical thruster or just two thrusters. Also, all six motion is possible with all eight thrusters or just six thrusters (three horizontal thrusters & three vertical thrusters)

In Fig. 2b(b), numbers 1- 4 annotated to the thrusters correspond to the vertical thrusters, and numbers 5-8 correspond to the horizontal thrusters. With one vertical thruster failure and one horizontal thruster failure, vehicle can still generate motion in all directions Even when two horizontal thrusters fail, vehicle navigation to a target location in x-y plane can still be achieved with careful path planning. Each thruster is equipped with a hall-effect sensor that measures the output voltage of each thruster Thruster failure can be easily detected by monitoring these thruster motor outputs. The blade's spin direction for each thruster was designed so that the resulting moment from the blades angular motion would counteract each other. This way, the net angular moment would be zero The vehicle navigation sensors include a pressure transducer, 8 acoustic sonars, and an inertial navigation system (INS). Fig. 3 shows vehicle's navigation and control block diagram. The vehicle position (x, y, z) and its linear velocity are estimated by Kalman filter using measurements from the eight sonars The vehicles' depth (z) and its velocity are also obtained using measurements from the pressure sensor, vehicle's pitch, roll and heading which are measured by the INS [13]

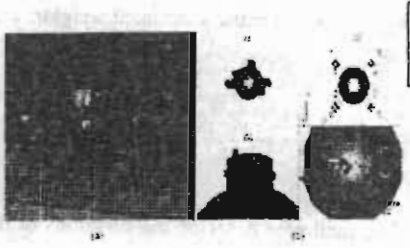


Figure 2
Closed frame sphere with eight thrusters and one manipulator

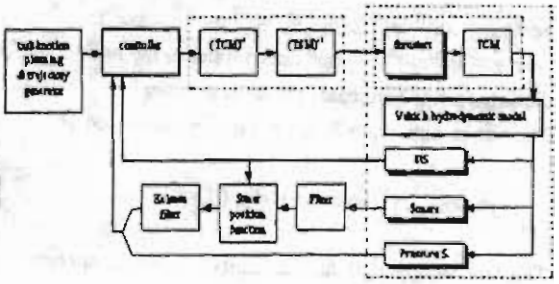


Figure 3.Control block diagram

3.STRUCTURE OF THE NEURO-FUZZY CONTROLLER :

A fuzzy inference system consists of a set of fuzzy IF-THEN rules that map values from the input space to values of the output space. The basic structure of a fuzzy control/decision system consists of four major components:

- A knowledge base, which consists of a set of fuzzy rules and a database that defines the membership functions used in the fuzzy rules,
- An inferenee mechanism that performs reasoning process to derive an output,
- Fuzzification modules, which transform the crisp inputs into appropriate fuzzy sets.
- Defuzzification modules, which convert the inferred fuzzy sets into a crisp output

In the knowledge base, the antecedents of fuzzy rules partition the input space into a number of linguistic term sets while the consequent eonstituent could be chosen as a fuzzy membership function to describe the control/decision
action on a given region.
Here the proposal technique is to investigate a neuro-fuzzy system with a fuzzy rule base consisting of *J* fuzzy rules in the following IF-THEN rule structures:

Rule j: **IF** x_1 is A_{j1} and ···and x_n is A_{nj} .
THEN y_1 is f_{j1} and ···and y_m is f_{mj} (1)

Where j = 1, 2, ···, J, *xi*(i = 1,2, ···, n) and y_k(k = 1,2, ···, m) are the input and output variables, respectively, and *Aij* are the input fuzzy term sets. f_{mj} are output fuzzy term sets .

In general, fuzzy systems are linguistically understandable because they use fuzzy term sets in both antecedents and consequents. Much researches [3],[8] have been conducted with remarkable success in this type of fuzzy systems due to its diversified selection in the shape of membership functions, fuzzy reasoning methods, and defuzzification procedures.

The proposed on-line self-organizing neuro-fuzzy Inference network is a six-layer feed-forward multilayer network. Each layer consists of nodes, each of which has some finite "fan-in" of connections represented by weight values from other nodes, and "fan-out" of connections to other nodes [13]
Each node in the system structure consists of one input integration function and one output activation function. The input integration function (f(·))combines information, activation, or evidence provided by links from other nodes and is expressed as.

$node_{(in)} = f(u_1^{(l)}, u_2^{(l)}, \ldots, u_p^{(l)}; w_1^{(l)}, w_2^{(l)}, \ldots, w_p^{(l)})$,

where $u_1^{(l)}, u_2^{(l)}, \ldots, u_p^{(l)}$ are inputs to the node, $w_1^{(l)}, w_2^{(l)}, \ldots, w_p^{(l)}$ are the associated weights, and the superscript l indicates the layer number.

The output activation function $(a(\cdot))$ is expressed as:

$$node_{(out)} = a^{(l)}(node_{(in)}) = a^{(l)}(f) = u^{(l+1)}$$

The link weight is treated as unity unless we specify it. We shall next describe the functions of the nodes in each of the six layers of the system.

Layer 1:

No computation is done in this layer. Each node in this layer, which corresponds to one input variable, only transmits input values to the next layer directly. That is:

$$f = u_i^{(1)}$$

and,

$$a^{(1)} = f \qquad (2)$$

The link weight in layer one $[w_i^{(1)}]$, is unity.

Layer 2:

Each node in this layer corresponds to one linguistic label (small, large, etc.) of one of the input variables in Layer 1.

In other words, the membership value which specifies the degree to which an input value belongs to a fuzzy set is calculated in Layer 2.

There are many choices for the types of membership functions for use, such as triangular, trapezoidal, or Gaussian ones. With the choice of Gaussian membership function, the operation performed in this layer is:

$$f[u_{ij}^{(2)}] = -[u_i^{(2)} - m_{ij}]^2 / \sigma_{ij}^2$$

and

$$a^{(2)}(f) = e^f \qquad (3)$$

where m_{ij} and σ_{ij} are, respectively, the center (or mean) and the width(or variance) of the Gaussian membership funetion of the jth term and the ith input variable x_i. Hence, the link weight in this layer can be interpreted as m_{ij}.

Similarly, we can deduce the rest of layers' equations.

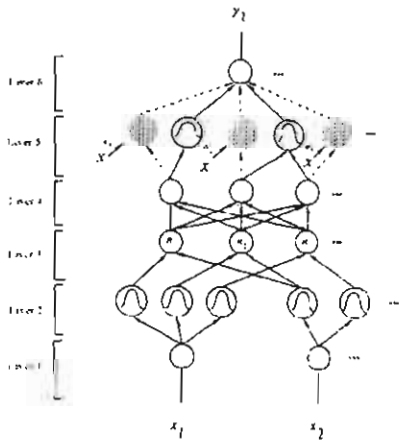


Figure 4. Structure of the proposed system

4. LEARNING ALGORITHMS FOR THE NEURO-FUZZY CONTROLLER :

Two types of learning : structure and parameter learning are used concurrently for constructing the proposed controller.

The structure learning :

Includes both the precondition and consequent structure identification of a fuzzy IF-THEN rule.

The precondition structure identification:

Corresponds to the input-space partitioning and can be formulated as a combinational optimization problem with the following two objectives:

- to minimize the number of rules generated
- and to minimize the number of fuzzy sets on the universe of discourse of each input variable.

The consequent structure identification:

The main task is to decide when to generate a new membership function for the output variable and which significant terms (input variables) should be added to the consequent part (a linear equation) when necessary.

The parameter learning:

Based upon supervised learning algorithms, the parameters of the linear equations in the consequent parts are adjusted by either Least Mean Square(LMS) or Recursive Least Squares(RLS) algorithms [14] and the parameters in the precondition part are adjusted by the back propagation algorithm to minimize a given cost function.

Algorithm Structure

Self-Organizing Phase:

The controller-learning algorithm consists of a self-constructing learning phase and a self-organizing learning phase. The learning algorithm proceeds upon receiving an on-line incoming training data by performing self-constructing learning phase. In this phase of learning, structure learning and parameter learning are performed. An aligned clustering-based method [14] is used to construct the number of rules and also the number of membership functions for each input variable. A rule in the controller corresponds to a cluster in the input space.

For each incoming pattern $x = [x_1, x_2, \ldots, x_n]^T$, the firing strength of each rule can be interpreted as the degree of the training pattern belonging to a corresponding rule. Similarly, for each individual input variable, the firing strength of each term set can be interpreted as the degree of the input belonging to the corresponding term set. Thus, the information of firing strength can be used to determine whether to add a new rule or a term set for the incoming pattern. Once the criterion for generating a new rule is established, the next procedure is to determine whether the generation of a new term set is required for each input variable. If the largest firing strength of the existing term sets is less than the threshold of adding a new term set, then a new term set is formed. This same idea is used to establish the output term sets.

There are no rules (i.e., no nodes in the network except the input–output nodes) in the controller initially. They are created dynamically as learning proceeds upon receiving on-line incoming training data by performing the following learning processes simultaneously:

1) input/output space partitioning,

2) construction of fuzzy rules,

3) parameter identification.

In the above, learning process 1), and 2) belong to the structure learning phase and 3) belongs to the parameter learning phase.

1)Input–Output Space Partitioning:

The way the input space is partitioned determines the number of rules extracted from training data as well as the number of fuzzy sets on the universal of discourse of each input variable. Geometrically, a rule corresponds to a cluster in the input space, with m_i and D_i representing the center and variance of that cluster. For each incoming pattern x the firing strength of a rule can be interpreted as the degree the incoming pattern belongs to the corresponding cluster. For computational efficiency, we can use the firing strength directly as this degree measure:

$$F^i(x) = \Pi_i \, u_i^{(3)} = e^{-[D_i(x-m_i)]^T[(D_i(x-m_i)]} \quad (4)$$

where $F^i \in [0,1]$. In the above equation, the term $[Di(x-mi)]^T [(Di(x-mi)]$ is, in fact, the distance between x and the center of cluster I . Using this measure, we can obtain the following criterion for the generation of a new fuzzy rule.

Let $x(t)$ be the newly incoming pattern. Find :

$$J = \arg \max_{1 \le I \le c(t)} F(x) \quad (5)$$

where $c(t)$ is the number of existing rules at time t .If $F^J \le \bar{F}(t)$, then a new rule is generated where $\bar{F}(t) \in (0,1)$ is a pre-specified threshold that decays during the learning process.

Once a new rule is generated, the next step is to assign initial centers and widths of the corresponding membership functions. Since the goal is to minimize an objective function and the centers and widths are all adjustable later in the parameter learning phase, it is of little sense to spend much time on the assignment of centers and widths for finding a perfect cluster.

Hence, we can simply set :

$$m_{[c(t)+1]} = x \quad (6)$$

$$D_{[c(t)+1]} = -\,l\ \beta . diag\left[\frac{1}{\ln(F^J)} \ldots\ldots \frac{1}{\ln(F^J)}\right] \quad (7)$$

according to the first-nearest-neighbor heuristic [9] where β decides the overlap degree between two clusters. Similar methods are used in [15], [16].

In the proposed controller, the width is taken into account in the degree measure, so for a cluster with larger width (meaning a larger region is covered), fewer rules will be generated in its vicinity than a cluster with smaller width. this is a more reasonable result.

After a rule is generated, the next step is to decompose the multidimensional membership function formed in [14] and [15] to the corresponding 1-D membership function for each input variable. For the Gaussian membership function, the task can be easily done as:

$$e^{[D_i(x-m_i)]^T[(D_i(x-m_i)]} = \Pi_j e^{-[(x_j-m_{ij})^2/\sigma^2_{ij}]} \quad (8)$$

Where m_{ij} and σ_{ij} are, respectively, the projected center and width of the membership function in each dimension. To reduce the number of fuzzy sets of each input variable and to avoid the existence of highly similar ones, we should check the similarities between the newly projected membership function and the existing ones in each input dimension [14]

The whole ***algorithm*** for the generation of new fuzzy rules as well as fuzzy sets in each input variable is as follows. Suppose no rules are existent initially:

Let $\mu(m_i,\sigma_i)$ represent the Gaussian membership function with center mi and σi width .

IF [x.d] is the first incoming pattern **THEN** do

PART 1. {Generate a new rule with center $m_1 = x$, width $D_1 = diag\ (1/\sigma_{init}\ .\ .\ .\ 1/\sigma_{init})$

where σ_{init} is a prespecified constant After decomposition, we have n one dimensional membership functions, with center $m_{1i} = x_{1i}$ & width $\sigma_{1i} = \sigma_{init}$.

$i = 1, 2, \ldots, n$

}

ELSE for each newly incoming x, **do**

PART 2. { $F(x) = \Pi_{i=1}^{n} e^{-(x_i - m_{ij})^2 / \sigma^2_{ij}}$ where $F \in [0,1]$ is the firing strength of cluster (rule) j for the incoming pattern x.

find $J = arg \max_{1 \le j \le c(t)} F(x)$.

where c(t) the number of existing rules at time t.

IF $F \ge \bar{F}_{in}(t)$ **THEN**

do nothing

ELSE

$c(t+1) = c(t) + 1$. generate a new fuzzy rule, with

$m_{c(t+1)} = x$.

$$D_{c(t+1)} = -1\ \beta . diag\left[\frac{1}{\ln(F^J)} \ldots \ldots \frac{1}{\ln(F^J)}\right]$$

After decomposition, we have

$m_{new,i} = x_i$. $\sigma_{new,i} = -\beta . Ln(F^J)$, $i = 1 \ldots n$

Adding Term Set Nodes :

Find largest firing strength of the existing fuzzy term set nodes for each input i, $L\_F(x_i)$.

IF $L\_F(x_i) \ge \bar{F}_{term}(t)$, **THEN** assign

{ $m_{ki(t+1)} = m_{L\_Fi}$ and $\sigma_{ki(t+1)} = \sigma_{L\_Fi}$ }

ELSE do

Adopt a new membership function with

$m_{ki(t+1)} = x_i$ and $\sigma_{ki(t+1)} = \sigma_{init}$ and set

$k_i(t+1) = k_i(t) + 1$, where $k_i(t)$ is the existing number of a fuzzy term-set nodes for input i at time t.

}

}

Where, the threshold $\bar{F}_{in}(t)$ determines the number of rules generated, $\bar{F}_{term}(t)$ threshold determines number of term-set nodes, $\bar{F}_{init}(t)$ determines the number of output clusters generated

2)Construction of Fuzzy Rules :

We have to decide the consequent part of the generated rule. Suppose a new input cluster is formed after the presentation of the current input-output training pair (x,d); then the consequent part is constructed by the following algorithm:

IF there are no output clusters **THEN**

do {*PART 1* in Process 1), with x replaced by d }

ELSE

do {

find $J = arg \max_{1 \le j \le c(t)} P(x)$,

IF $P \ge \bar{F}_{out}(t)$ **THEN**

connect input cluster $c(t+1)$ to the existing output cluster J

ELSE

generate a new output cluster connect input cluster $c(t+1)$ to the newly generated output cluster.

}

The algorithm is based on the fact that, different preconditions of different rules may be mapped to the same consequent fuzzy set .

3) Parameter Identification :

After the network structure is adjusted according to the current training pattern, the network then enters the parameter identification phase to adjust the parameters of the network optimally based on the same training pattern. Note that the following parameter learning is performed on the whole network after structure learning, no matter whether the nodes (links) are newly added or are existent originally The idea of backpropagation is used for this supervised learning. Considering the single-output case for clarity, the goal is to minimize the error function

$$E = \tfrac{1}{2} [\, y(t) - y^d(t)]^2 \qquad (9)$$

where $y^d(t)$ is the desired output and $y(t)$ is the current output.

Self-Adaptive Phase :

In the self-constructing learning phase, the controller network is only able to grow the numbers of rules and term sets. Some of the fuzzy rules or term sets could become inappropriate after several learning iterations. This phenomenon can be easily observed when the learning curve becomes flattened. That is, performing the parameter learning can no longer further reduce the error caused by the existing rules and term sets. Hence, a performanee examination procedure is proposed that will enable the controller network to prune inappropriate rules and term sets based on a cumulative error performance index [17] and a similarity measure [3], respectively.

In our observation, inappropriate rules always cause errors. In order to examine the performance of each rule, a cumulative error performance index is established for eaeh rule. For each incoming pattern, the error signal is not only back-propagated to adjust the parameters but is also accumulated for the dominant rule (the rule with the largest firing strength of the existing rules). The cumulative error will be served as a performance index for identifying an inappropriate rule. The larger the cumulative error, the worse the performance of the rule. Thus, based on the cumulative error performance index, the rules with bad performance are the candidates to be pruned Note that the number of rules to be pruned should be exponentially decreasing as the error becomes smaller [18]. The following significance index is established to examine each rule

$$S^j = \frac{\sum_{i=1}^{I} \sum_{k=1}^{N} \rho_j}{I \times N} \qquad (10)$$

where $S^j \in [0,1]$ represents the significance of the j^{th} FBF, $\rho_j \in [0,1]$ is the normalized firing strength of the j^{th} rule in Layer 4, I is the number of iterations, and N is the total number of training data in each iteration. ρ_j is defined as

$$\rho_j(x) = \frac{\prod_{i=1}^{n} \mu_{A_i^j}(x_i)}{\sum_{j=1}^{J} \prod_{i=1}^{n} \mu_{A_i^j}(x_i)} \qquad (11)$$

where. $\mu_{A_i^j}(x_i)$ is Guassian membership function defined by :

$$\mu_{A_i^j}(x_i) = Exp[-(\frac{x_i - m_i^j}{\sigma_i^j})^2] \qquad (12)$$

Where m_i^j and σ_i^j are the centers and widths of the Gaussian functions. respectively .

To determine the rule for pruning, we find

$$k = arg \min_{1 \le j \le c(t)} S^j \qquad (13)$$

where c(t) is the number of existing rules at time t. If $S^k \le \varphi$. then the K^{th} rule (FBF) is a candidate to be pruned because it is the least significance (or it was least fired among all the rules). where $\varphi \in [0.1]$ is a pre-speeified threshold. Next. using the similarity measure. the highly similar term sets are combined by simply taking the average of centers and selecting the larger width to form a new membership function. This self-adapting learning phase performs well and is able to prune some insignificant rules and fuzzy term sets from the controller system to speed up the learning convergence. This learning phase results in a more concise network structure without sacrificing the performance. Figure 5 shows the flow chart of the proposed system's learning algorithm .the learning algorithm proceeds upon receiving every incoming training data by performing the self-organizing learning phase and the self-adapting learning phase simultaneously.

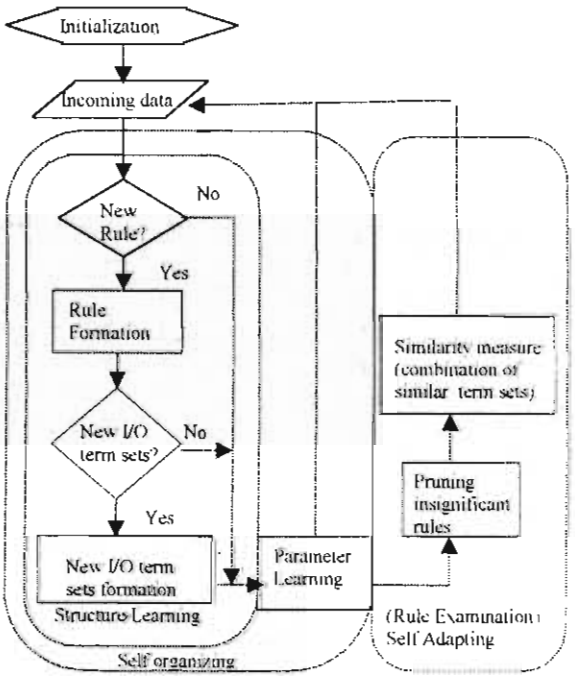


Figure 5. Flow Chart of the controller learning algorithm

5.COMPUTER SIMULATIONS :

In this application, controller is applied to model a control system for an autonomous underwater vehicle . It is a closed-framed sphere with eight thrusters and one manipulator.

Computer simulations were conducted to demonstrate:

(1) The self-organizing learning and self-adapting abilities of the proposed system. In order to verify the performance of the controller system .

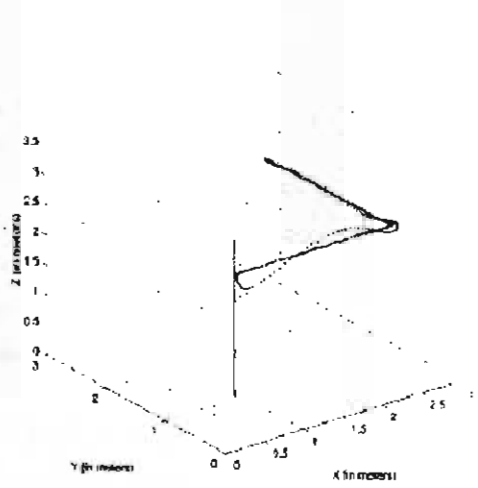
The better performance of the proposed controller as compared to an adaptive controller [19].

Simulations are conducted on the dynamic model [19]. Here we select the initial parameters. K_i = 0.01I, D = I, fi =1, δ_i=1, and the initial value of $\theta_i = 0.0$ for the adaptive controller (taken from [19]). By treating the control signals generated by the adaptive controller as desired control signals, our objective is to train the proposed system using the generated training patterns. Each training pattern consists of six inputs of position errors and velocity errors, $\mathbf{x} = [\, e_x , e_y , e_z , \dot{e}_x , \dot{e}_y , \dot{e}_z \,]^T$, and three outputs of forces for x, y, and z axes, $y = [F_x , F_y , F_z]^T$. We neglect the moments about the x, y,and z axes because they are always very close to zero. In order to reduce the learning burden, we decompose the training patterns into three subsets, each of which consists of the position error, velocity error, and force of each axis; that is, the training patterns generated by the adaptive controller are injected separately to three different controller systems, namely cont_x, cont_y, and cont_z. To generate the training data, all the possible error trajectories are fed into the adaptive controller and the corresponding force outputs are collected. 80 position errors and 80 velocity errors are randomly sampled from ±0.2 meters (m) and ±0.1 meters/sec for x and y axes, and ±0.4m and ±0.1 m/sec for z axis. Selecting the learning rate $\eta = 0.001, \bar{F}_{in}(t) = 0.25, \bar{F}_{term}(t) = 0.6, \sigma_{init} = 0.1$, the similarity threshold $\rho = 0.75$, and the significance threshold $\varphi = 10^{-2}$, each subsystem is trained with 6,400 training patterns in each epoch. Again, to compare the effects of the self-adapting phase (rule examination procedure), two simulations are conducted: (a) without performing the self-adapting learning phase, and (b) with performing the rule examination for every 10 epochs. In this simulation, initially there is no rule in the controller . Table 1 shows the comparison of simulations (a) and (b). By performing the self-adapting learning phase, a significant reduction in the number of fuzzy rules as well as fuzzy term sets for the proposed controller is observed.

Simulation	Controller	No.of Fuzzy term sets		Initial Rule No.	Final rule No.
		Input 1	Input 2		
(a)	Cont_x	6	6	31	31
	Cont_y	6	6	31	31
	Cont_z	7	8	33	33
(b)	Cont_x	5	6	31	26
	Cont_y	5	6	31	26
	Cont_z	7	7	33	27

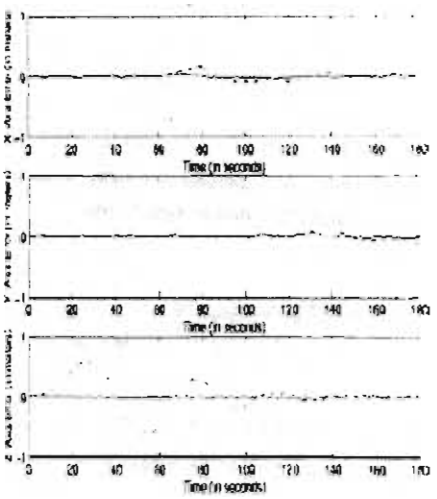
Table 1. Comparison between the two simulations

In our second simulation, the vehicle is asked to perform an edge-following for x, y, and z axes and a diagonal movement by using the trained controller (from the first simulation result) and the adaptive controller. In the first planned movement, the vehicle starts at 0.5m from the bottom of the pool, moves up vertically to 2.5 m, moves along the x axis for 2.0 m, and then moves along the y axis for 2.0 m. In the second planned movement, the vehicle is asked to move downward and upward diagonally for 2.0 m. Figure 6 shows the desired trajectory, the actual trajectories from the adaptive controller and the proposed controller for the edge-following simulation. Figure 7 shows the error curves of the adaptive controller and the proposed controller for the edge-following simulation along the x, y, and z axes Similarly, for the diagonal motion simulation, Figure 8 shows the desired trajectory, the actual trajectories from the adaptive controller and the proposed controller. Figure 9 shows the error curves of the adaptive controller and the proposed controller for the diagonal motion simulation along the x, y, and z axes. From these simulations and figures, the results show that the performance of the proposed controller is much better than the existing adaptive controller for controlling the vehicle.



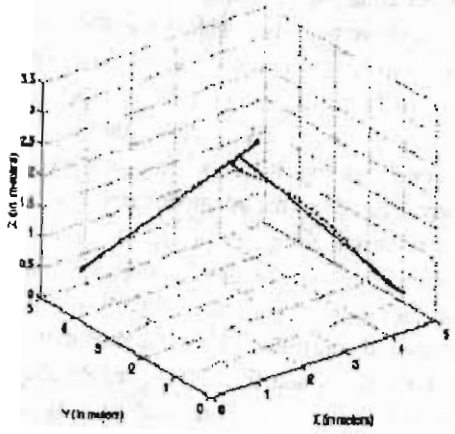
desired (dotted) Adaptive controller (x) proposed controller(solid)

Figure 6 comparison of desired trajectory, adaptive trajectory, proposed system trajectory for the edge following motion .



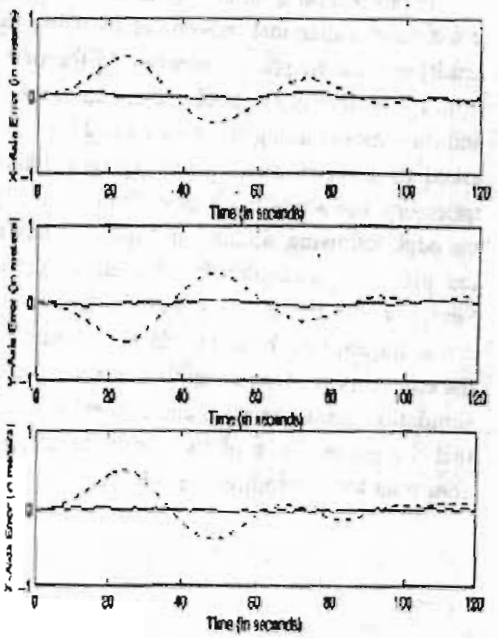
Adoptive controller (doshed) proposed controller(solid)

Figure 7 Error curves of adaptive controller and proposed controller For edge following simulation .



desired (dotted) Adaptive controller (x) proposed controller(solid)

Figure 8 comparison of desired trajectory, adaptive trajectory, proposed system trajectory for the diagonal motion simulation



Adaptive controller (dashed) proposed controller(solid)

Figure 9 Error curves of adaptive controller and proposed controller For the diagonal motion simulation

CONCLUSIONS

This paper described an on-line self-adaptive neural fuzzy inference system with three different types of IF-THEN rule structures. The proposed system is a multi-layered feedforward neural network that realizes a traditional fuzzy logic system and is capable of self-organizing and self-adapting its internal node connectivity and learning the parameters of each node based on incoming training data.

The multi-layered feedforward connectionist structure of the proposed system incorporates fuzzy basis functions as a universal approximator for better system performance. Computer simulations on a real-world application ,modeling a control system for autonomous underwater vehicles, have been conducted to validate the effectiveness of the self-organizing and self-adapting abilities of the proposed system.

REFERENCES

[1] H. R. Berenji and P. Khedkar, "Learning and Tuning Fuzzy Logic Controllers through Reinforcements,"IEEE Trans. Neural Networks, vol.3, no.5, pp.724-740, September, 1992.

[2] J. R. Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference System," IEEE Trans. Syst. Man Cybern., vol.23, no.3, pp.665-685, 1993.

[3] C. T. Lin and C. S. G. Lee, Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems,Prentice Hall PTR, 797 pages, 1996.

[4] B. Kosko. Neural Networks and Fuzzy Systems. Englewood Cliffs. NJ:Prentice-Hall. 1992

[5] C. T. Lin. Neural Fuzzy Control Systems with Structure and Parameter Learning. New York. World Scientific. 1994.

[6] C. T. Sun. "Rule-based structure identification in an adaptive-network-based fuzzy inference." IEEE Trans. Fuzzy Syst., vol. 2, pp. 64--73. Feb.1994.

[7] C. T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system." IEEE Trans. Comput., vol. 40, pp. 1320–1336,Dec. 1991.

[8] C.T.Lin and C.S.George Lee , "Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems." IEEE Trans. Fuzzy Syst., vol. 2. pp.46–63. Feb. 1994.

[9] C. J. Lin and C. T. Lin. "Reinforcement learning for ART-based fuzzy adaptive learning control networks." accepted for publication in IEEE Trans. Neural Networks. vol. 7. pp. 709–731. May 1996.

[10] C. T Lin. "A neural fuzzy control system with structure and parameter learning." Fuzzy Sets Syst., vol. 70. pp. 183–212. 1995

[11] L. X. Wang and J. M. Mendel. "Generating fuzzy rules by learning from examples." IEEE Trans. Syst., Man. Cybern., vol. 22. no. 6. pp.1414–1427. Nov./Dec. 1992.

[12] H. Ishibuchi. K. Nozaki. N. Yamamoto. and H. Tanaka. "Selecting fuzzy IF-THENrules for classification problems using genetic algorithms." IEEE Trans. Fuzzy Syst., vol. 3. pp 260–270. Aug 1995

[13] Jeen-Shing Wang , C.S. George Lee and J.Yuh, "An On-Line Self-Organizing Neuro-Fuzzy Control For Autonomous Underwater Vehicle",proc. of 1999 IEEE,Int'l conf. on Robotics & Automation, Detroit,MI, pp.2416-2421, May 10-15,1999 .

[14] C. F. Juang and C. T. Lin, "An On-Line Self-Constructing Neural Fuzzy Inference Network and
Its Applications," IEEE Trans on Fuzzy Systems,vol.6, no.1, pp.12-32, 1998.

[15] J. Platt. "A resource allocating network for function interpolation."Neural Computat., vol 3. pp 213–225. 1991.

[16] J. Nie and D. A. Linkens. "Learning control using fuzzified self-organizing radial basis function network." IEEE Trans. Fuzzy Syst., vol.40. pp. 280–287. Nov. 1993

[17] J. S. Wang. C. S. G. Lee, and C. H. Juang,"Structure and Learning in Self-Adaptive Neural Fuzzy Inference Systems." Proc. of the Eighth Int'l Fuzzy Syst. Association World Conf., Taipei. Taiwan,pp.975-980. August 17-20, 1999

[18] Jeen-Shing Wang and C.S. George Lee. "Structure and Learning in Self-Adaptive Neural fuzzy Inference Systems", Int'l Journal of Fuzzy System, vol. 2, No 1. March 2000

[19] S K. Choi. "An Adaptive-Learning Control System for Underwater Robotic Vehicles." Ph.D Thesis. Department of Mechanical Engineering. Univ of Hawaii, 1995