Control of a Robot Based on Intelligent Systems

Pedro Ponce,Ricardo Fernandez , Eduardo Azcue, Jose Silva, Juan Silva
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY-CAMPUS CIUDAD DE MÉXICO –DEPARTAMENTO DE MECATRÓNICA
Calle del Puente 222 Ejidos de Huipulco Tlalpan
México City

ABSTRACT. *In this work a novel neuro-fuzzy controller was designed and implemented using an artificial six-legged arthropod (hexapod). The core of the controller is based on both artificial neural networks based on trigonometric series and fuzzy clusters. A new scheme was optimised and the performance of the hexapod was adequate. The implementation was done by a Motorola microcontroller. Programming the controller law into a Motorola microcontroller gave flexibility to the platform proposed. Simulation and experimental results are included to validate the theory presented in this paper.*

Keywords: *Neuro-Fuzzy controller , Fuzzy Logic, Hexapod and Neural networks.*

1 Introduction

The artificial intelligence controller has being used in many industrial and research applications, as mobile robots, specially in speed and position control [1]. For these applications, conventional closed loop control systems have been tested and have shown to be reliable in different conditions in the hexapod. The hexapod is a classical example of a non linear system that has been solved in many ways but remain a prototypical case study for optimization and the testing of new control techniques.

The hexapod has a great importance for its application in practical systems. In the automation process, it provides a framework for understanding the control of many systems. There has been considerable work in other industrial applications as well. The hexapod is also a relevant model for understanding the way in which structures with two or more feet (such as human being and some robots) may walk while keeping balance [2]. Several solutions to the hexapod problem are known, so that research has increasingly emphasized the more complex cases of hexapod. These systems have more inputs and outputs that need to be controlled, which makes them rather nonlinear. A lot of control techniques have been tested. As the control law is based on a conventional PID control is quite complex for multiple-input, multiple-output (MIMO) systems. Because of that, modern control theories are generally used in the control design of these systems. These techniques include state feedback, adaptive-control strategies, neural-network modeling to simulate possible combinations of input/output control, adaptive or intelligent neural-network controllers and, more recently, the integrated application of neural networks and fuzzy-logic. This is so because fuzzy control requires an expert control law for the hexapod formulated in terms of if-then rules. A recently designed controller, described in an IEEE publication, uses genetic algorithms, neural networks and fuzzy logic to tune a PID controller. Many neural-network architectures have been proposed to control an hexapod [2]. The control learning of an hexapod by means of a neuro-fuzzy controller was proposed by P. Ponce [3], who provided the system with a trigonometric series as nodes which represents harmonics and fuzzy clusters.

2 Prototype description

The prototype used in this work has the following basic elements: an hexapod [4], a Motorola microcontroller [5], a Basic stamp [6] and sensors[7]. As can be seen, it is divided in four main components. The sensors are Devantech SRF04 [7] ultrasonic sensors. This ultrasonic ranger has an approximate range from 3cm to 3m. The sensor has a logic signal used to trigger a pulse and the echo is returned on another input. It is only needed to supply a short 10 μS pulse to trigger the start of the ranging. The SRF04 will send out an 8 cycle burst ultrasound at 40 KHz and raise its echo signal high. It then listens for an echo, and as soon as it detects the signal, it lowers the echo input again. The echo input is therefore a pulse whose width is proportional to the distance to the object.

By measuring the pulse, it is possible to calculate the range in inches or centimeters. If nothing is detected then the SRF04 will lower its echo input anyway after about 36mS. The timing of the echo input is made by the microcontroller located on the Basic Stamp Board. Three sensors were used to cover the front area of the robot. Each sensor can measure within a range of 45 degrees. Three of them were positioned at the front of the robot, two of them separated by 45 degrees from the center line. This sensibility provides enough range to cover (135 degrees), this is a necessity in order to have a good navigation controller.

A BASIC Stamp II microcontroller is used to measure the echo pulse sent by the sensors. The logic lines used to acquire the measurement of each sensor are connected to the I/O pins of the Education Development Board - Serial Version [6], that simultaneously will be transmitted through the serial input and received by the MC0S12E64 serial input.

The implementation of the controller was done by a Motorola microcontroller (MC9S12E64) [5].

As shown in figure 5, some of the main elements of the MC9S12E64 include a 25 MHz bus operating at 5 V for 40 nS minimum instruction cycle time and a 16 MHz bus operation at 3.3 V for 62.5 nS minimum instruction cycle time. It is code compatible with the 68HC11 and 68HC12. It has a C optimized architecture producing extremely compact code. It is self-timed, fast programming, Fast Flash Page Erase 20 mS (1024 bytes), it is very fast while compiling and loading. It can program 16 bits in 20 μS while in burst mode. It has a Virtual EEPROM implementation, and Flash array usable for EE extension (64k Flash memory). It includes a fast, easy conversion from analog inputs like temperature, pressure and fluid levels to digital values for CPU processing through a 16-channel A/D converter. It also provides digital control capabilities of external analog devices in two 8 bit digital to analog converters (DAC). A flexible, programmable timer system exists in three or four channels. 16 bit timers with each channel programmable for input capture or output compare. There are two pulse width modulators included and three serial communications interfaces for exact baud rate matching; having asynchronous communication between the MCU an a terminal, computer or a network of microcontrollers. It includes up to 90 input/output lines with programmable pull/ups or pull/downs and dual drive capability. All the navigation control is programmed using the CodeWarrior Software Interface [8], used to program the entire Motorola microcontrollers and lets the programmer do it in “C” language. The MC9S12E64 board sends through serial line to the PSC (Parallax Servo Controller) the data that each servomotor needs to perform the robot routine cycles.

The Parallax Servo Controller [9], PSC, controls up to 16 servos, and may be networked together so that two PSCs can control 32 servos using a single I/O line. The PSC manages all of the servo pulses so the host controller does not have to. Additionally. If the ramping function allows to, individually, set the speed of each servo. With ramping, the only signal needed to move the servo is to indicate where to go, and just how fast to get there.

Some of the main features of this board include the possibility of a runtime selectable baud rate, where a serial message switches the baud rate from 2400 to 38k. 16 servos can be driven simultaneously, with 0-180o rotation, and 2 μS resolution. The speed can be selectable with servo ramping, choosing from one of the 63 ramp rates for each servo. The user may also request position of any servo at any time. It requires a 5 VDC supply, so in order not to damage the servos a 9 VDC power supply was used, because they require 4-7.5 VDC. The PSC is connected with the MC9S12E64 board using a three-conductor cable that has a 5 VDC on one of its terminals, Ground and the serial line on its other two terminals. Through this cable, we send the necessary data (baud rate, channel, servo positioning, and speed ramp) to control each of the 12 servomotors. The PSC both transmits and control every one of the servo pulses.

3 Neuro-fuzzy controller [3]

In this paper the neuro-fuzzy controller proposed by Pedro Ponce [2 and 3] is developed and implemented. The fuzzy system is based on the Sugeno inference [10] but the outputs of the systems are trigonometric functions . The 3 basic steps of the algorithm were the followings:

- First the fuzzy cluster algorithm is used for tuning the member function,
- Second the if-then rules were established using the partition space.
- Third the trigonometric series functions were applied as outputs

A block diagram of the neuro-fuzzy system developed is showed in figure 1 and the figure 2 shows the training block diagram.

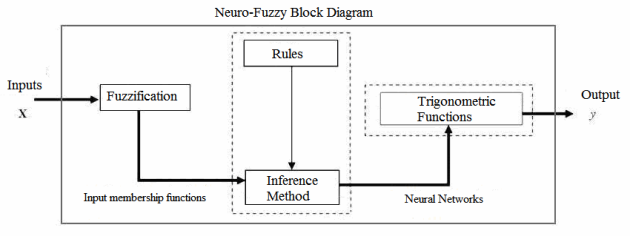


Fig.1 Neuro-fuzzy controller.

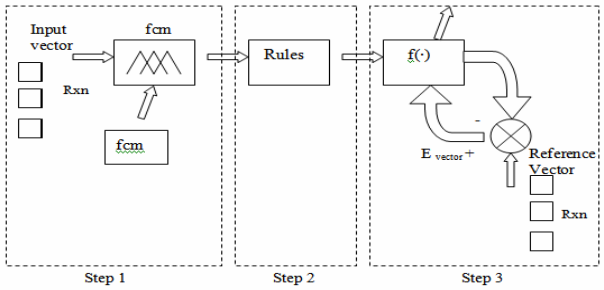


Fig. 2 Training algorithm.

The first component of the algorithm is expressed in the following conditions, all the fuzzy sets are associated with linear membership functions, the truth value of preposition is written as [10]:

$$[X \text{ is } B \text{ and } y \text{ is } C] = B(X) \wedge y(C) \quad (1)$$

The format of implication (R) used is:

R: if (X1 is B1, ……, Xk is Bk) then y =g (*) (2)

Where: g(*) are trigonometric functions.

R1 if X1 is B1, …….., and Xk is Bk then Y=g(*)

.....

Rn if Xn is Bn, …….., and Xn is Bn then Y=g(*) (3)

In order to group the different points in a –n-number of fuzzy clusters, the FCM (Fuzzy C-Means) method was used

The objective of this method is to assign the degree of membership of a n-element set to each one of the defined clusters.

Assuming that X ={x1,x2,…..xn} is a s-dimension set with a n-number of points and that we wish to divide in a c number of clusters v={v1,v2,….vc}, each cluster can be represented by the center of its cluster v1. The objective is that the quadratic sum of error of the distance from the points to the center reaches its minimum. The objective function is described as (4) [11, 12, 13 and 14]:

$$J_{FCM} = \frac{1}{2}\sum_{x=1}^{N}\sum_{i=1}^{c}\mu_{x,i}^{m} d^2(z_x, v_i) \quad (4)$$

Where $\mu_{x,i}$ (x=1,2,...,N, i=1,2, ...,c) is a membership value that represents the fuzzy membership of the point x, that belongs to a set i, vi(i=1,2,…,c) is the centroid of each cluster and zx(x=1,2,…N) is the set of data, m is the fuzzification value, d2 (zx,vi) is the Euclidean distance between zx and vi, N is the number of points and c is the number of clusters.

The fuzzy partition was obtained by the iterative optimization of the objective function according to [14]:

1) Choose the vi main centroids (prototypes).
2) Calculate the degree of membership for all the data sets in all the clusters using equation number (5):

$$\mu_{x,i} = \frac{\left(\frac{1}{d^2(z_x, v_i)}\right)^{1/(m-1)}}{\sum_{i=1}^{c}\left(\frac{1}{d^2(z_x, v_i)}\right)^{1/(m-1)}} \quad (5)$$

3) Calculate the new centroids Vi1 (3):

$$v_i^1 = \frac{\sum_{x=1}^{N}\mu_{x,i}^{m} z_x}{\sum_{x=1}^{N}\mu_{x,i}^{m}} \quad (6)$$

And assign the new degrees of membership $\mu_{x,i}$ to $\mu_{x,i}^1$ according to (4).

4) If max x,i[$|\mu_{x,i} - \mu_{x,i}^1|$] < e, then stop, else go to 3, where e is a finishing criteria between 0 and 1 and $d^2(z_x,v_i)$ is given by

$$d2(zx, vi) = ||zx - vi||2 \quad (7)$$

Applying this method it was possible to find the degrees of membership for each one of the clusters along with their centroids.

In order to obtain the membership that the points have to each one of the clusters, it is necessary to assign the number of clusters in which they are going to be grouped with the FCM method. To determine this number the Partition Coefficients algorithm was used. This algorithm is based on iterative calculations in which the PC number is obtained, considering the different possible cluster number. At the end, the highest PC number

corresponds to the optimal cluster number in which the elements can be grouped. The formula of this method is shown in (8) [14]:

$$PC(U,c)=\frac{\sum_{j=1}^{n}\sum_{i=1}^{c}(u_{ij})^2}{n} \tag{8}$$

Where the optimum number of clusters is defined in (9):

$$\max_{c}\left[\max_{\Omega_c \in U}\{PC(U;c)\}\right] \tag{9}$$

At the time that this algorithm was implemented, an optimum number of 2 clusters was obtained, however, 3 clusters were defined in order to avoid system oscillations. Once the number of clusters was determined to gather the information, the method FCM (Fuzzy C-Means) was used to assign the membership degree to each one of the elements containing the clusters. In order to obtain the membership degrees of multiple intermediate data to each one of the clusters, the following methodology was used:

Every case was arranged considering the membership degree to the cluster in ascending order. Six lists were obtained with the arranged data for each one of the clusters. Later, the hyper plane equations were obtained according the following pseudo code:

Start
N=0
For n=1:32
If(n=<32){
Obtain the hyper plane curve with the formula
$f(x,y,z)=a_0+b_0x+c_0y+d_0z$
With the list cases of every cluster from n until n+4}
Else STOP

At the end of this pseudo code, we have 32 equations for each one of the clusters, which describe the membership function of each cluster. Each hyper plane equation has a range, which corresponds to the maximum and minimum values of the hexapod sensors (S1, S2 and S3). If one wants the cluster membership degree of one point out of the list, it has to be analyzed in which hyper plane equations could be evaluated; this is possible when the evaluating case was between the range. We observe this in the following equations:

$$\{\min(x_1,x_2,x_3,x_4)\}<x<\{\max(x_1,x_2,x_3,x_4)\} \tag{10}$$

$$\{\min(y_1,y_2,y_3,y_4)\}<y<\{\max(y_1,y_2,y_3,y_4)\} \tag{11}$$

$$\{\min(z_1,z_2,z_3,z_4)\}<z<\{\max(z_1,z_2,z_3,z_4)\} \tag{12}$$

If the evaluating case is between the range of a hyper plane, the case is evaluated with the equation of its hyper plane. Later, the error degree is obtained which is represented by the output. In order to calculate the error, the distance of the evaluating point and each of the cases which generated the hyper plane is measured. Those distances are obtained with the equations from (12), to (16). If the evaluating element was between one or more hyper plane ranges, the minimum value was the one which defined the hyper plane that will evaluate that element.

Vector Equations are these:

$$V_1=\sqrt{(x_1-x)^2+(y_1-y)^2+(z_1-z)^2} \tag{12}$$

$$V_2=\sqrt{(x_2-x)^2+(y_2-y)^2+(z_2-z)^2} \tag{13}$$

$$V_3=\sqrt{(x_3-x)^2+(y_3-y)^2+(z_3-z)^2} \tag{14}$$

$$V_4=\sqrt{(x_4-x)^2+(y_4-y)^2+(z_4-z)^2} \tag{15}$$

$$V_T=\sqrt{(V_1)^2+(V_2)^2+(V_3)^2+(V_4)^2} \tag{16}$$

If the evaluated element was not in one of these cases, a subtraction between the values of the input and the values of each one of the cases was done, and an evaluation with the hyper plane equation of the closest element was made.

Once that the membership degree for each one of the clusters is obtained, the real output of the system can be obtained by multiplying the degree of membership for each cluster times the value obtained with the evaluation of the neural network based on trigonometric functions.

According to Kolmogorov´s theorem a hidden layer in a conventional neural network is enough to make an approximation of a non linear system, Kolmogorov showed that a continuous function of many variables can be represented exactly by the superposition of continuous one-dimensional functions of the original input variables. Any mapping of a function of n-dimension inputs (n>=2)

to a m-dimension output can be implemented exactly by a network with a hidden layer for:

$\phi : I^n \rightarrow R^m, \phi(x)=y$, where I is the closed interval (0,1]. Figure 3 shows the basic element of a conventional neural network (neuron).

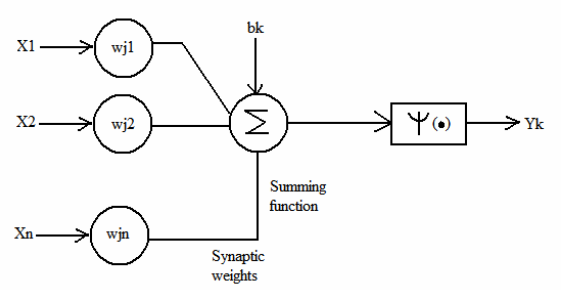


Fig. 3. Conventional Neuron.

Where:

$$Y_k = \Psi(u_k + b_k)$$

$$Z_k = \sum_{j=1}^{n} \lambda^n \Psi(x_j + \varepsilon k) + k$$

$$Y_i = \sum_{k=1}^{2n+1} g_i(Z_k)$$

$$\lambda = constant$$

$$\Psi = Continuous function$$

$$g_i = real\_continuous\_output\_node$$

This theorem could be applied to trigonometric functions instead of conventional activation functions $\Psi(\bullet)$ if the theorem is expanded. The topology based on activation functions constructed by trigonometric series are easier to understand, and every node is equal to a single harmonic component of the series. It can be described by the following expressions:

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}\left(a_n \cos(n\omega x) + b_n \sin(n\omega x)\right) \quad (17)$$

In our system, we employed as activation functions the sin functions, but a similar topology could be generated for cos functions. According to this, equation 17 could be defined by equation number 18.The selection of sin or cos functions depends on the system that will be approximated [3].

$$f(x,y,z) = \lambda_1 + \sum_{n=1}^{\infty}\lambda_2 \sin(nx + \lambda_5) + \sum_{n=1}^{\infty}\lambda_3 \sin(ny + \lambda_6) + \sum_{n=1}^{\infty}\lambda_4 \sin(nz + \lambda_7) \quad (18)$$

It is important to observe that all the coefficients of equation 18 could be defined using analytical expression [12], but one of the goals of this topology is to be able to map without using primary functions which describe the relation between multiple- inputs and multiple-outputs. The equation 18 could be represented by a neural network topology, where x,y,z are the inputs and λ_i =(a_0, *C, D, E, θ, α, β) are* the weights of the network [3]. This topology only has two layers, where the first layer represents the harmonics (nodes) and the second one represents a weight and the summing function. Figure 8 shows the new topology. In order to find the coefficients (weights) could be applied a supervised training method could be applied the back-propagation method, this was possible because the nodes are based on continuous functions. The back-propagation method is defined using as the error energy function.

$$E = \frac{1}{2}\sum_p \sum_i (d_{pi} - Y_{pi})^2, \quad E = \sum_p E_p \quad (19)$$

Where:

$$E_p = \frac{1}{2}\sum_i (d_{pi} - Y_{pi})^2$$

$$\frac{\partial E}{\lambda_i} = \sum_p \frac{\partial E_p}{\partial \lambda_i}$$

$$\frac{\partial E_p}{\partial \lambda_i} = \sum_k \frac{\partial E_p}{\partial a_k}\frac{\partial a_k}{\partial \lambda_i}$$

$$\delta i = \frac{\partial Ep}{\partial ai} = \frac{\partial E_p}{\partial Y_i}\frac{\partial Y_i}{\partial a_i}$$

$$\delta k = -(d_{pk} - Y_{pk})\delta \quad (20)$$

The basic neural network topology based on trigonometric series is showed in figure 4, and it is observed that every node is a component of the series (harmonic).

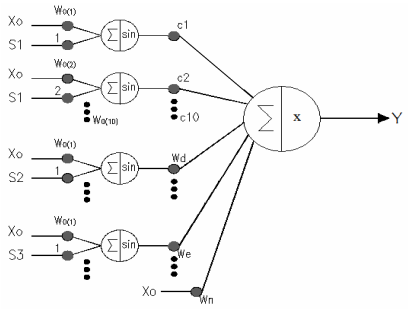


Fig. 4. Neural Network Topology [2]

Once that the membership degree for each one of the clusters is obtained, the real final of the system can be obtained by multiplying the degree of membership for each cluster times the value obtained with the evaluation of the neural network topology of the corresponding cluster and finally dividing by the sum of the degrees of membership

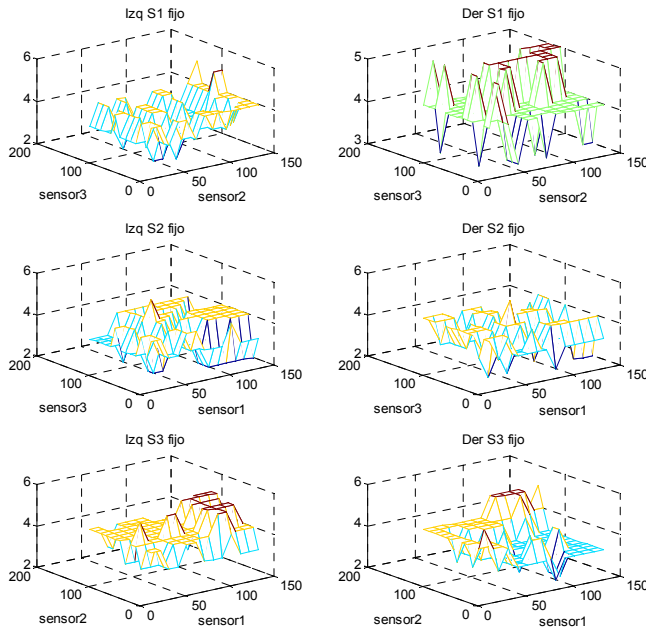
for each one of the clusters. This expression is shown in (21):

$$Output = \frac{\mu_{x1} * x_{x1} + \mu_{x2} * x_{x2} + \mu_{x3} * x_{x3}}{\mu_{x1} + \mu_{x2} + \mu_{x3}} \qquad (21)$$

Where: $\mu_{x,i}$ is the degree of membership of the element to the cluster i and x_{xi} is the value of the output of the neural network related with the cluster i evaluated with the inputs of point x.

4 Results

The figure 5 shows the surface generated for the controller which were evaluated during the decision process, it could be observed the correct performance of the controller. The figure 6 shows the experimental result when a unknown path was presented to the hexapod. Not only could the hexapod travel around the walls, but also the hexapod never collided against the walls.



Fix value =40

Fig. 5. Described path by the hexapod.

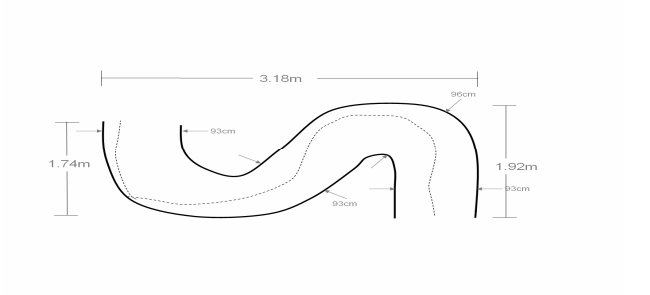


Fig. 6. Described path by the hexapod.

5 Conclusions

A novel topology based on fuzzy and neural networks was implemented, and it was able to control the hexapod in a proper manner. The system was tested under unknown situations for the system and the neural network approximation based on trigonometric series allowed to present a correct generalization. The methodology gives a new possibility to solve problems using neuro-fuzzy controllers based on both trigonometric series and fuzzy logic. The proposed systems presents a good approximation for non linear systems as the case of the robot. Not only does the robot figure out the training path but also any unknown path.

References:

[1] FIRA Association, www.fira.org
[2] Neural-networks based on trigonometric functions internal report proposed by P. Ponce, Instituto Tecnologíco de Monterrey Campus Ciudad de México.
[3] A Neuro fuzzy-controller based on trigonometric series , to appear, P. Ponce
[4] http://www.crustcrawler.com
[5] Motorola/Metrowerks CodeWarrior IDE. Version 5.5.1272 ©Metrowerks Corporation
[6]Basic stamp manual
[7] Devantech SRF04 Manual
[8] Motorola/Metrowerks CodeWarrior IDE. Version 5.5.1272 ©Metrowerks Corporation
[9]www.parallax.com
[10] M. Sugeno and T. Takagi, “A New Approach to Design of Fuzzy Controller”, Advances in Fuzzy Sets, Possibility Theory and Applications 1983.
[11] Yu**,** Huang, “A New Weighting Fuzzy c-means Algorithm”
[12] Kaya, Metin “An Algorithm for Image Clustering and Compression”
[13] James C. Bezdek, \A Convergence Theorem for The Fuzzy ISODATA Clustering Algorithms", IEEE TransactionOn Pattern Analysis And Machine Intelligence, Vol.pami-2(1), pp.1-8, 1980.
[14] Rhee,Oh “A Validity Measure for Fuzzy Clustering and Its Use in Selecting Optimal Number of Clusters”