$_{EN}INV^{IE}_{2006}$
Encuentro de Investigación en Ingeniería Eléctrica
Zacatecas, Zac, Abril 5—7, 2006

Implementation of a Neuron-based Autonomous Mobil Robot on a DSP

G. Miramontes de L., C. Sifuentes, G., J. I. de la Rosa V., J. Villa H.,
M. Araiza E., E. García D.
Laboratorio de Procesamiento Digital de Señales,
Facultad de Ingeniería Eléctrica,
Universidad Autónoma de Zacatecas
Zacatecas ZAC-98000.
TEL: +(492)9239407, ext. 1518
email: gmiram@uaz.edu.mx, cgsifuen@uaz.edu.mx

Abstract — An autonomous mobil robot has been implemented on a Digital Signal Processor (for real time operation) using neural networks as the main part of the program that runs on the processor. The neural network was based on a single layer perceptron (SLP) with two neurons, four inputs coming from four sonar sensors, and two outputs to control the direction of two CD motors. The goal of the mobil robot is to avoid obstacles while it runs randomly in a given room. Two training sets were tested to provide two different reactive behaviors, the first one with forward direction, left turn and right turn, and the second which includes reverse direction. Both behaviors were compared.

Resumen— Se implementó un robot móvil autónomo en un Procesador Digital de Señales (para operación en tiempo real) usando redes neuronales como la parte principal del programa que se ejecuta en el procesador. La red neuronal está basada en un perceptrón de una sola capa (SLP) con dos neuronas, cuatro entradas alimentadas por sensores de sonar, y dos salidas para controlar la dirección de dos motores de corriente directa. El objetivo del robot móvil es evitar obstáculos mientras recorre aleatoriamente un cuarto. Se probaron dos patrones de entrenamiento para tener dos comportamientos reactivos diferentes, el primero con dirección hacia adelante, giro a la derecha y giro a la izquierda, y el segundo que incluye marcha hacia atrás. Se compararon ambos comportamientos.

Index Terms* — *Mobil robot, neural networks, perceptron, sonar.

I. INTRODUCTION

AUTONOMOUS mobil robots are today a very interesting problem that can be solved with the aid of digital signal processors. The recent technological progress on embedded processors, sensors technology and mechatronic systems allows the construction of these systems with such capacities as perception, decision and action according to the environment.

Specialized applications can be found, for example, in space explorations, but today they can be found in many other places, for example at the industry or at home with smart vacuum cleaners. In fact, there is a new applications domain aimed to introduce advanced and secured robotized systems in our living space.

The main objective of this work is to start developing a robust advanced system with signal processing technology and neural networks theory. The system will increase gradually the functions and tasks it can perform. For example, in the near future, a direction of arrival algorithm will me included using the microphone inputs of the DSP board.

This paper is organized as follows: In section II, the basic steps of the design are presented. In this section some details about the perceptron used in the construction of the robot are introduced. The description of the training patterns and the development of the MATLAB code is also presented [1]. In section III the hardware construction of the robot is presented. Section IV contains a comparison in the behavior of the robot under the two training rules used in the project. Finally section V presents some conclusions and a brief discussion of planed projects.

II. THEORICAL FOUNDATION

In 1943 McCulloch and Pitts introduced the idea of neural networks as computational machines. Rosenblatt in 1958 proposed the perceptron as the first model for supervised leaning machines. The perceptron is the simplest form of a neural network and it is used for classification of linearly separable patterns.

The perceptron is basically a linear combiner which feeds a decision device. This decision device is based on a non-linear decision rule, usually a hard limiter or threshold function with an output normally set at ±1. A perceptron consists of a single neuron with adjustable weights [2],[4]. The weights of the perceptron are adapted on an iteration-by-iteration basis in order to minimize some cost function. Usually, the error-correction rule for the implementation of the adaptation algorithm is used.

The structure or architecture of the network can be a pair of nodes, two neurons with, for example, 2-dimensional inputs is shown in Figure 1. The corresponding desired output for each class will be denoted as:

$$\begin{aligned} ClassA &\rightarrow d = [+1\ \ -1] \\ ClassB &\rightarrow d = [-1\ \ +1]. \end{aligned}$$

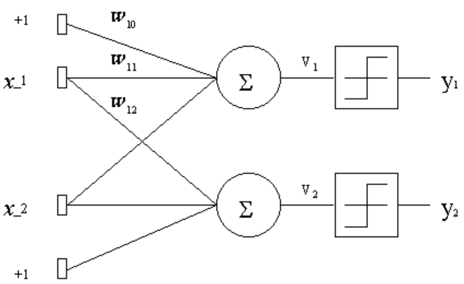

Figure. 1. A 2-neuron perceptron.

The output y_k is given by the decision rule $\varphi(\nu)$. In fact the function $\varphi(\nu)$ may be considered as an stochastic law, which determines the probability of the values $y = \pm 1$, for the hard limiter decision rule or as a continuous function, if the neuron are assigned analog values, as the case where $\varphi(\nu) = \tanh(\nu)$ [3].

A. *The* MATLAB *implementation of the SLP.*

MATLAB code was written for the implementation of the neural network. The Neural Network Toolbox was not used. It is strongly recommended to avoid unnecessary do-loops when writing MATLAB code. This is mainly to exploit the benefits of the MATLAB "language". In this way we can obtain a very efficient program in terms of computation time and the structure of the code itself. To explain this, let's show (schematically) the steps involved in the development of a complete matrix of equations. Let $\mathbf{W}$ be a column matrix of the synaptic weights with elements $[w_{k1} \ \ w_{k2} \ \ \ldots \ \ w_{kL}]^T$ for a k-th single neuron. Then, the induced local field will be,

$$\nu_k = [-\mathbf{W}-] \begin{bmatrix} | \\ \mathbf{X} \\ | \end{bmatrix} = \sum_j w_j x_j, \qquad (1)$$

resulting in a scalar ν_k.

For multiple neurons (fully connected),

$$\mathbf{V_k} = \begin{bmatrix} -\mathbf{W_1}- \\ -\mathbf{W_2}- \\ \vdots \\ -\mathbf{W_k}- \end{bmatrix} \begin{bmatrix} | \\ \mathbf{X} \\ | \end{bmatrix} = \mathbf{W^T X}, \qquad (2)$$

where $\mathbf{W}$ is a $L \times k$ matrix of weights with a column vector for each neuron k, and L is the length of the input vector. The output will be:

$$\sum_j W_{kj} x_j = \begin{bmatrix} | \\ \mathbf{V_k} \\ | \end{bmatrix} \qquad (3)$$

where $\mathbf{V_k}$ is a column vector with elements $[\nu_1 \, \nu_2 \, \ldots \, \nu_k]^T$.

In order to obtain the correct dimensions on the matrices involved in the calculations of the error signals, a matrix $\mathbf{D}$ containing the desired output vector for each input is constructed. Of course, there is a corresponding class for every input point. $\mathbf{D}$ will be constructed as:

$$\mathbf{D} = \begin{bmatrix} 1 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ \vdots \end{bmatrix} \cdots \begin{bmatrix} 0 \\ \vdots \\ 1 \end{bmatrix}. \qquad (4)$$

The length of each column vector depends on the dimensionality of the problem. So, the computation of the error signals is given by:

$$\mathbf{E} = \mathbf{D} - \mathbf{Y}. \qquad (5)$$

Finally, the adaptation algorithm can be applied, based on the error-correction rule as:

$$\mathbf{W} = \mathbf{W} + \eta \mathbf{E} \mathbf{X} \qquad (6)$$

This last step has to be done in a for-loop mode because the dimensions of each element in (6) do not allow a single instruction in the code.

B. Neuron design

For the construction of the mobil robot the network is based on two neurons and four inputs. The outputs follow a McCulloch-Pitts model. There are sixteen possible combinations at the inputs. For training, a fixed number of input patterns are randomly chosen.

Four classes can be defined according to the four possible combinations at the output, as shown in Table I, and assuming M1 is the left motor and M2 is the right motor.

TABLE I
DEFINITION OF CLASSES.

M1	M2	action
1	1	forward
1	-1	turn right
-1	1	turn left
-1	-1	backwards

TABLE II
TRAINING RULES.

S1	S2	S3	S4	M1	M2
1	1	1	1	-1	-1
-1	1	1	1	-1	1
1	1	-1	-1	1	-1
-1	-1	-1	-1	1	1
1	-1	1	1	1	-1
1	1	-1	1	-1	1
⋮	⋮	⋮	⋮	⋮	⋮

Part of the training rule is shown in Table II. All sixteen possible combinations have to be defined in the training set since the patterns are randomly chosen for every epoch.

C. *The cross-validation*

The network design problem is to chose within a given set of structures the "best" model according to some criteria. The first step is to split the data set into a training data set and a testing data set. The testing data set is divided again in two disjoint sets: estimation and validation data sets. The goal is to validate the model using a different data set than used for the estimation. In this way it is possible to measure the performance of several models and then to chose the best one. Let r, (0,1), be the partition parameter. Then $(1-r)N$ will be the size of the estimation set and rN will be the size of the validation set, where N is the number of training patterns.

In our case, the number of patterns is not only fixed but completely deterministic, even so the generalization property of the network was tested using a partition parameter r equal to 0.2. This means 80% of the patterns (thirteen) will be used for the estimation and 20% (three) will used for validation.

III. CONSTRUCTION OF THE MOBIL ROBOT

The robot was constructed using a sonar system for collision avoidance, two CD motors and a 56000 family DSP. Once defined the training rules, the neural network was trained and the synaptic weights were used in the actual implementation of the neural network on the DSP board. The sonar system was constructed with one transmitter unit based on a pest repeller (Black and Decker EP1100 Ultrasonic Pest Repeller) and four receiving ultrasonic sensors. On the DSP board, a general purpose input output port was programmed for four binary inputs and two binary outputs. The outputs control the direction of the CD motors. Figure 2 shows the final construction of the mobil robot where the DSP on top can be observed.

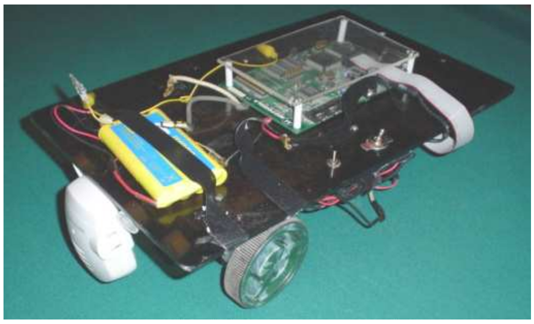

Figure. 2. Complete mobil robot.

A. *Perception system*

The perception system is based on a sonar unit constructed with a modified commercial bugs repeller and four ultrasonic sensors. The transmitter unit produces a pulsating 40 kHz signal. In front of a solid object the signal is reflected and picked up by ultrasonic sensors. The detected signals are amplified and fed to envelope detector circuits. A level comparator allows to set the range of the detection, delivering a logic signal (0 or 1) as inputs for the DSP. Figure 3 shows the sonar system and amplification circuitry.

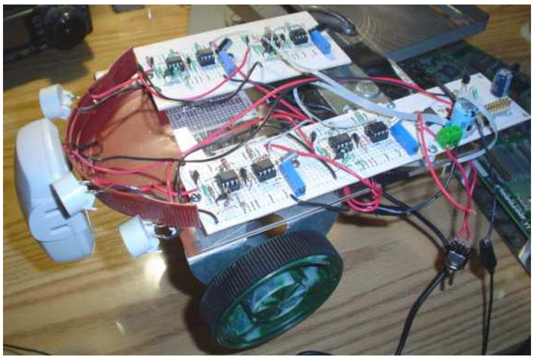

Figure. 3. Sonar system and amplifiers.

B. *Traction unit*

The traction of the robot was implemented with two 6 volts CD motors, as shown in Figure 4. One motor is located in the front left side of the robot and the other one in the front right side. An H bridge and one control line allow the change of direction of each motor. When on motor goes forward and the other motor goes backward a turn, left or right, is produced.

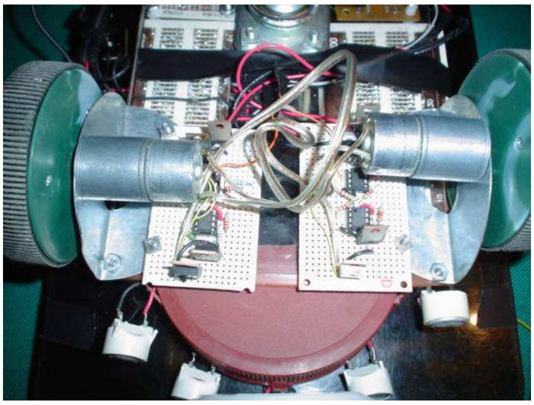

Figure. 4. Traction system of the mobil robot.

IV. RESULTS

The generalization property of the neural network was tested and the results were correct, i.e., three input patterns were chosen at random and left out of the training phase. When these three patterns were presented to the network the classification was correct.

After implementation of the neural network on the DSP and the complete construction of the robot it was tested in a unknown environment. The response of the robot to walls and obstacles was good enough to avoid collisions.

Two different training rules were implemented. The first one did not include backward direction. For any possible input pattern only three possible responses were considered: forward, right turn, and left turn. It was supposed the possibility to enter in an infinite loop, that is, once the robot had detected a wall in front of it, it would get back until the point where no obstacle was detected, then in the absence of a reflected signal the robot would go forward and get back again. With this strategy the robot had a good behavior, but sometimes the turns were very close to the wall leaving not enough space to run freely. The problem with this training rule could be if the robot enters in a very narrow space.

The second training rule did include backward direction. Surprisingly, the robot never entered in an infinite loop during the tests. It was found anyway that when the robot passed a space where it passed before, approximately in the same direction and the same angle, it showed the same behavior.

V. CONCLUSIONS

After testing the behavior of the mobil robot it was found that the neural network was enough to allows "autonomous exploration" in an unknown environment. The implementation of the neural network on a DSP was straightforward and as easy as the implementation of any digital filter.

One drawback of the robot is the lack of detection for non sound reflexive objects, but a project is in progress to include more capabilities like light and voice commands detection. Other functions have to be implemented like the following of a given trajectory or tracking a moving target.

With this project, a DSP-based mobil robot platform was constructed. It is expected that with this platform future developments will be easier to tackle down. Further improvements will include direction of arrival estimation and tracking capabilities as a way to gain experience in advanced DSP projects.

References

[1] Matlab, *The Language of Technical Computing.* The Mathworks, Inc. 24 Prime Park Way, Natick, MA 01760-1500.

[2] Anil K. Jain, Jianchang Mao, K. M. Mohiuddin, *Artificial Neural Networks : A Tutorial* COMPUTER Vol. 29, No. 3 March 1996, pp. 31-44.

[3] Müller B., Reinhardt J., Strickland M. T., *Neural Networks an introduction* Second Edition, Springer, 1995.

[4] Haykin, S., *Neural Networks a compresive foundation* Second Edition, Prentice Hall, 1999.