SOLVING VEHICLE PARKING & BARRIER CROSSING PROBLEMS WITH NEURAL NETWORKS

Arzu Babaev babaev@uludag.edu.tr
Onur GÜNGÖR onurgungor@sekizdesekiz.com
Murat ERKAN muraterkan@sekizdesekiz.com

Uludağ University,
Faculty of Engineering and Architecture, Department of Electronics Engineering, 16059
Bursa, TURKEY

Key words : Neural network control applications, Vehicle parking problem, Barrier crossing problem

ABSTRACT

Neural network (NN) control applications are designed to have a basic desicion mechanism with minimum error rate and less number of iterations. In this study NN methods are used to solve two different control applications, Vehicle Parking and Barrier Crossing.

I – INTRODUCTION

In designing NN control applications, one of the important challenges is to determine the structure and the input – output values of the decision mechanism. The second step of the method is to train the input – output data, which will form the mathematical model of the application. In this study NN methods are used to solve two different control applications, Vehicle Parking and Barrier Crossing. This paper is organized as follows. Section 2 gives the architecture of NN. Section 3 defines vehicle parking problem. Section 4 and Section 5 respectively gives solution stages of the problem and simulation results. Sections 6, 7, 8 follows the same steps for the barrier crossing problem.

II – NN ARCHITECTURE

NN is an effective method to demonstrate, save, and process the data. NN can be considered as a basic input –output system[1,2].

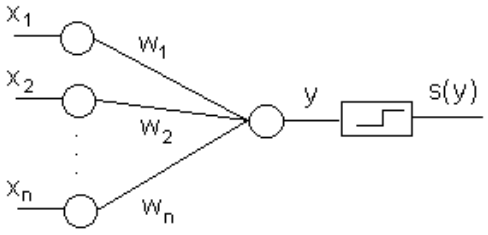

Figure 1 : Basic NN architecture

$$y = \sum_{j=1}^{n} w_j x_j \text{ and } s(y) = \begin{cases} 1, if\ y > \theta \\ 0, if\ y \le \theta \end{cases}$$

The circles in Figure 1 are called neurons, s(y) is the threshold function, θ is the value of the threshold. We use continuous sigmoid function shown below as the threshold function.

$$s(y) = \frac{1}{1 + e^{-y+\theta}}$$

Finding the elements of w_j means training the NN.

When the number of training data is too many then the training of NN will be difficult. For this kind of situation we need a hidden layer in the NN structure as shown in figure 2.

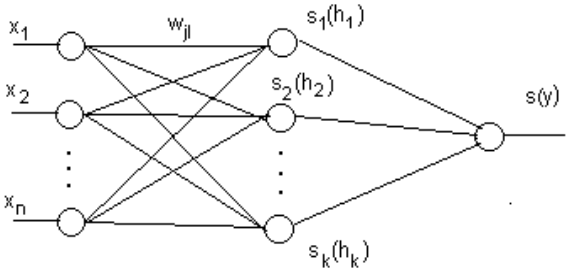

Figure 2 : NN with a hidden layer

$$h_l^t = \sum_{j=1}^{n} w_{jl} x_j^t \quad , \quad \bar{y}^t = \sum_{l=1}^{k} v_{li} s_l(h_l^t)$$

h_l^t are the output values for the hidden layer and $\bar{y}^t$ is the output value of the NN.

III – DEFINITION OF THE VEHICLE PARKING PROBLEM

Let's explain the problem. Position of the vehicle is defined by (x, y, θ) parameters[3]. Our aim is to move the vehicle for parking to the specified position (0, y_t, 90°). x and y are the coordinates of the position, θ is the angle of the position (figure 3)

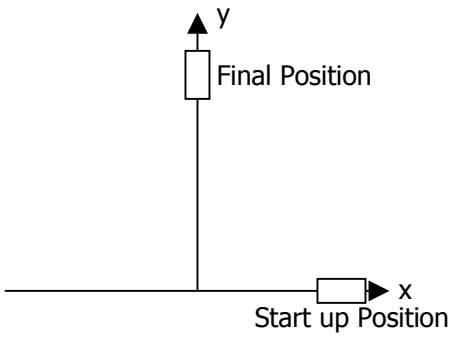

Figure 3 : Graphical comment of the problem

Start up position is taken as any (x, 0, θ) point on x coordinate. y coordinate is defined between [0, ∞] and not considered in this problem. Consequently 3 input (x : position, θ : position angle, -1 : bias) and 1 output (φ : rotation angle of steering vehicle) contoller is obtained. The intervals of parameters are chosen as

x Є [-10, 10], θ Є [0°, 180°], φ Є [-90°, 90°]

IV – SOLUTION OF THE VEHICLE PARKING PROBLEM

To solve the problem, first the input data values have to be determined and this values have to be processed by training algorithm to obtain w_{ij} and V_{ij} matrixes. They are found as

$$W_{34} = \begin{bmatrix} -2.257 & 6.611 & 1.775 & -7.874 \\ -2.145 & 6.202 & 1.391 & -7.265 \\ -0.767 & 10.436 & 2.575 & -2.901 \end{bmatrix}$$

and

$$V_{41} = \begin{bmatrix} 4.1099 \\ -9.456 \\ -2.657 \\ 8.736 \end{bmatrix}$$

In the simulation program these matrixes are used to calculate the new position and the new position angle of the vehicle during each iteration. After running the specified numbers of iteration, the vehicle gets the final destination. In each iteration the calculated angle is added to the final angle of the vehicle and this provides the vehicle to choose the right route.

V – SIMULATION RESULTS

Simulation program has been developed in Visual Basic 6.0 and Training program has been developed in Turbo C++. The program outputs for four different cases are given in the figures below. In each simulation different start up positions and angles are chosen.

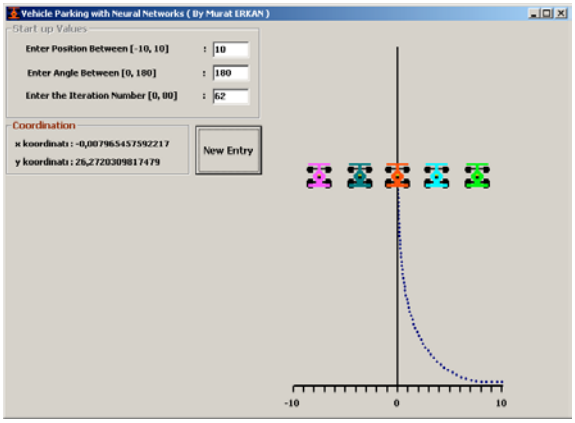

Figure 4 : Solution for x=10, θ=180

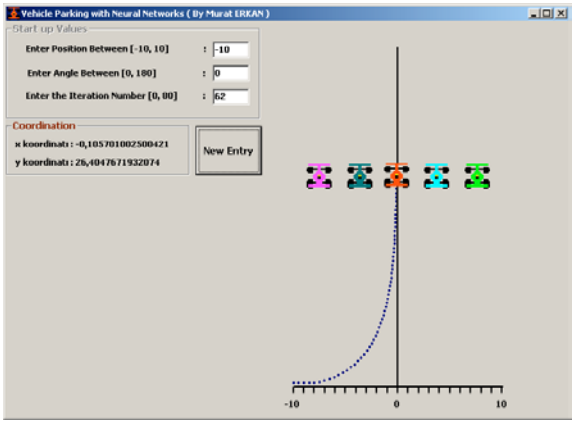

Figure 5 : Solution for x=-10, θ=0

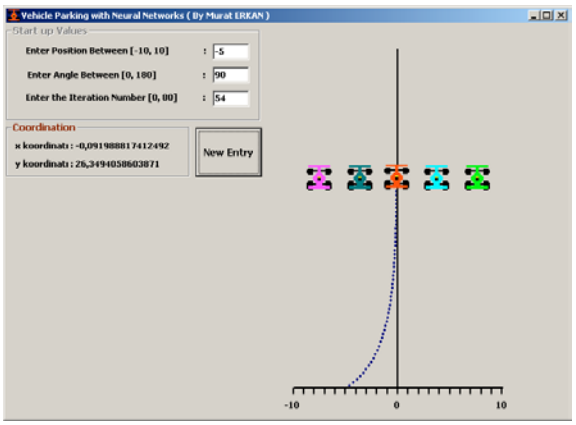

Figure 6 : Solution for x=-5, θ=90

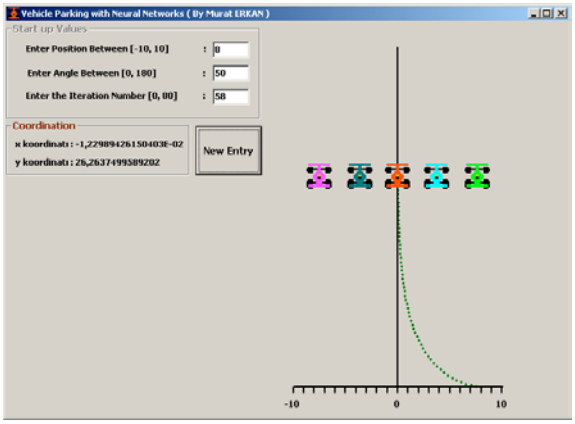

Figure 7 : Solution for x=8, θ=52

VI – DEFINITION OF THE BARRIER CROSSING PROBLEM

The second problem in this article is Barrier Crossing[4]. In this problem there are barriers in both sides of the vehicle's moving path. Our aim is to provide a suitable route for our vehicle by preventing it to crash into a barrier. Position of the vehicle is defined by (x, y, θ) parameters.

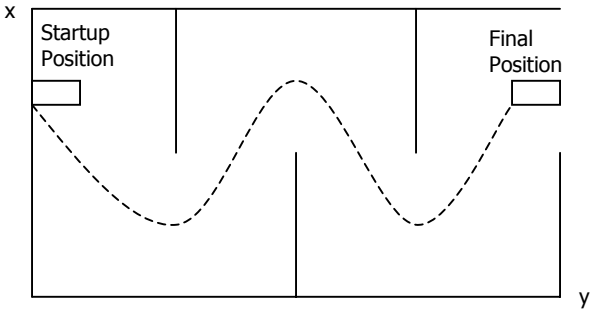

Figure 8 : Graphical comment of problem

Start up position is taken as any (x, 0, θ) point on x coordinate. y coordinate is defined between [0, 4]. Consequently 3 input (x : position, θ : position angle, m : side of the barrier) and 1 output (φ : rotation angle of steering vehicle) contoller is obtained. The intervals of parameters are chosen as

$x \in [-2, 2]$, $\theta \in [0^o, 180^o]$, $\varphi \in [-150^o, 150^o]$

VII – SOLUTION OF THE BARRIER CROSSING PROBLEM

To solve the problem, first the input data values have to be determined and this values have to be processed by training algorithm to obtain w_{ij} and V_{ij} matrixes. They are found as

$$W_{34}=\begin{bmatrix} -0.491 & 2.313 & 0.773 & 1.505 \\ -0.669 & -1.989 & 0.662 & -0.436 \\ -0.268 & 0.341 & 0.234 & -1.315 \end{bmatrix}$$

and

$$V_{41}=\begin{bmatrix} 6.698 \\ -2.956 \\ -5.643 \\ 6.991 \end{bmatrix}$$

In the simulation program these matrixes are used to calculate the new position and the new position angle of the vehicle during each iteration. After running the specified numbers of iteration, the vehicle gets the final destination. In each iteration the calculated angle is added to the final angle of the vehicle and this provides the vehicle to choose the right route.

VIII – SIMULATION RESULTS

Simulation program has been developed in Visual Basic 6.0 and Training program has been developed in Turbo C++. The program outputs for four different cases are given in the figures below. In each simulation different start up positions and angles are chosen.

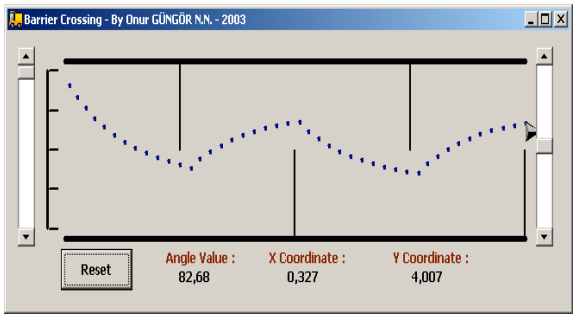

Figure 9 : Solution for x=-2, θ=90

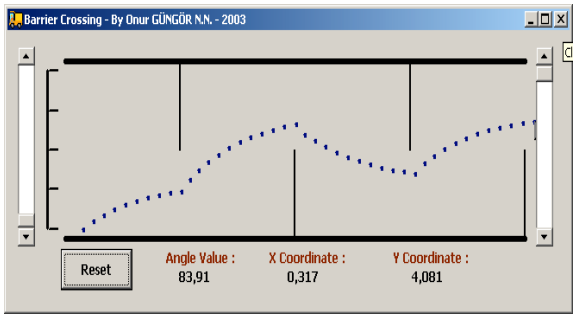

Figure 10 : Solution for x=2, θ=0

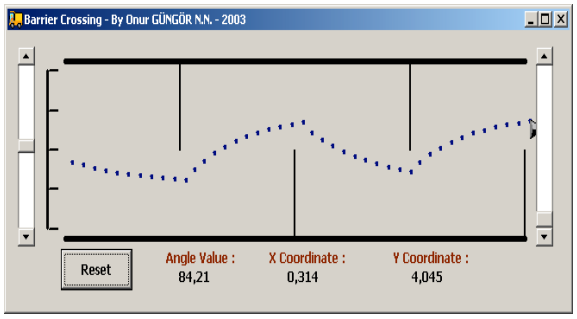

Figure 11 : Solution for x=0, θ=180

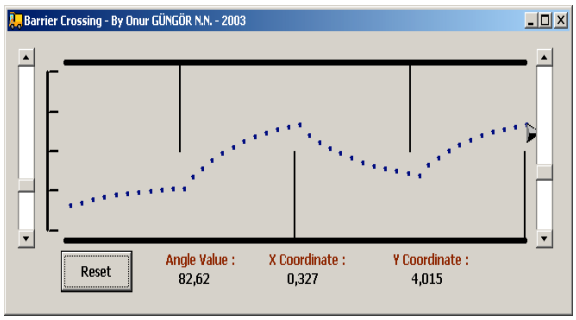

Figure 12 : Solution for x=1, θ=120

IX – CONCLUSION

In this study, N.N. solutions for 2 different control problems are examined. Program outputs which are taken with various start up values both for start up position and position angles can be found in the article.
After these two practical solutions, it can be said that the N.N. method is a basic but an efficient approach for solving these kind of control automations. Even with complex problems, if appropriate training data values are chosen, the satisfactory results can be obtained with N.N. controllers.

REFERENCES

1. B. Kosko . Neural Networks and Fuzzy Systems Prentice Hall Inc, 1992.
2. L.H. Tsoukalas, R.E. Uhrig. Fuzzy and Neural Approaches in Engıneering. John Wiley & Sons, 1997.
3. A. Babaev, I. Gucuyener. Takagi - Sugeno biçiminde kural tabanlı kontrol sistemi. Second Int. Symp. on Mathematical & Computational Applications. Baku, September, pp. 236-241. 1999.
4. A. Babaev, S. Arı. Simplification of Structure of a Neuro-Fuzzy Controller.-IMS-98, Sakarya, August, 1998.