ISSN 2229-5518

Optimized Fuzzy Logic Training of Neural Networks for Autonomous Robotics Applications

Ammar A. Alzaydi, Kartik Vamaraju, Prasenjit Mukherjee, Jeffrey Gorchynski

Abstract— Many different neural network and fuzzy logic related solutions have been proposed for the problem of autonomous vehicle navigation in an unknown environment. One central problem impacting the success of neural network based solutions is the problem of properly training neural networks. In this paper, an autonomous vehicle controlled by a feed-forward neural network is trained in real time using a fuzzy logic based trainer and the standard back-propagation learning algorithm. The experimental results presented demonstrate the feasibility of real time training using a constrained hardware platform. They also show the impact of racetrack complexity on the training process as well as the impact of the neural network size on the learning speed and error convergence during the training process. The results are then used to develop an optimization procedure that is used to determine the optimal neural network size for the given problem domain and experimental platform.

Index Terms—Autonomous, Autonomous navigation, Autonomous robotics, Fuzzy logic, Navigation, Neural network, Real-time training

1 INTRODUCTION

Designing an autonomous vehicle for navigation in an unknown environment is a challenging problem that encompasses many complex tasks such as obstacle avoidance and path planning. The complex dynamics of vehicle control and the unknown nature of the environment being navigated limit the feasibility of many theoretical methods for autonomous vehicle navigation. Many intelligent methods have been developed involving fuzzy logic and neural networks to solve problems related to autonomous navigation. Fuzzy logic based systems, such as [1], [2], utilize a semantic rule base consisting of IF {antecedent} THEN {consequent} rules that govern the actions of the vehicle based on fuzzified sensory data. These systems do not necessarily require advanced modelling of vehicle dynamics yet can provide control that is more tolerant to sensory error. Fuzzy logic based systems suffer from the inability to adapt to unforeseen scenarios due to the static nature of their fuzzy rule base. Alternatively, neural network based methods overcome this barrier through the use of learning algorithms. Fuzzy-neural networks merge the advantages of both fuzzy logic and neural networks together to create a potentially superior implementation [3], [4]. But while fuzzy-neural networks appear to represent an ideal solution to the problem of autonomous navigation in an unknown environment, there remain to be significant practical barriers preventing widespread adoption.

All methods that involve neural networks suffer from the problem of generating relevant, useful training data and implementing viable learning algorithms for real-time usage. Many different learning algorithms have been used to train neural networks [5], [6], [7]. These methods however have limited viability in training scenarios in which the vehicle must learn fast to respond to unknown obstacles using a limited supply of online training data. Regardless of the exact training algorithm used, there is still the problem of constructing a viable set of training data. It is highly difficult and impractical for an expert to generate the potentially thousands of input-output data vectors necessary to train the neural network, and while methods exist for ensuring useful training data [8], [9], these methods are difficult to apply to an arbitrary scenario.

One solution to the problem of providing viable training data is to create a fuzzy logic trainer that supplies the necessary data to train the neural network that is driving the vehicle [10]. In this scenario, a vehicle is controlled by a neural network with randomized weights and is placed within a racetrack. As the vehicle navigates the racetrack the fuzzy logic trainer generates data that is used to train the neural network using the standard back-propagation learning algorithm. The neural network is trained in real time as it navigates through a racetrack. Since the ideal output used to train the neural network is provided entirely by the fuzzy logic trainer, the pattern recognition behaviour of the neural network results in the neural network approximating the fuzzy logic trainer performance. The mathematical error between fuzzy logic trainer and the neural network results in differences in the navigational performance that lead to competitive, if not superior navigational performance by the neural network. The level of noise in the training data also tends to reduce the probability of the back-propagation algorithm converging to local minima. The problem of generating an optimal training data set is therefore equivalent to creating a viable racetrack. The success of training a neural network in this scenario is directly comparable to training a neural network in any type of environment in which the neural network must learn fast with a limited amount of training data.

In this paper, this training process is investigated experimentally by implementing a fuzzy logic trainer on a robot vehicle controlled by an untrained neural network. A feed-forward neural network with a single hidden layer is used to control the steering angle and speed of the vehicle. The goal of this paper is to experimentally determine the feasibility of real-time training of a neural network for autonomous naviga-

————————————————

• *Ammar A. Alzaydi: B.Sc., M.A.Sc., Ph.D. Student, Mechanical and Mechatronics Engineering. University of Waterloo, Waterloo, On., Canada. E-mail: aalzaydi@engmail.uwaterloo.ca*

tion, as well as to determine the optimal number of neurons for the hidden layer for the given experimental design. The process used to select the optimal number of neurons is motivated primarily by experimental results as opposed to other theoretically based methods. This process is not specific to autonomous navigation and can be applied to other applications as well. In addition, results are presented demonstrating the effect of additional neurons in the hidden layer with respect to both training efficiency and overall driving speed. The successful implementation of real time training using this experimental platform should scale easily as the complexity of implemented systems increase, and therefore the experimental results should be directly applicable to real time training scenarios involving different neural network topologies and hardware implementations.

2 Experimental Platform

A robot vehicle was built to navigate indoor racetracks. A single Hokuyo URG-04LX laser range-finder (Lidar) was used to supply sensory data to two Atmega1280 based microcontrollers. These microcontrollers communicated to a single steering servo motor and to a Sabretooth 2x10 motor controller to move the vehicle. The fuzzy logic and neural network implementations were designed for real time autonomous navigation even when constrained by the significant computational limitations imposed by the 8-bit microcontrollers.

The fuzzy logic trainer and neural networks were implemented on the two microcontrollers. One microcontroller was used as an interface to the sensor, and used the fuzzy logic trainer and neural networks to generate control packets for steering angle and speed. The control packets from the first microcontroller were sent to the second microcontroller, which acted as an interface to the steering servo motor and the speed motor. The purpose of using two different microcontrollers was to allow for easy expansion with additional sensors, since there would be a greater availability of processing power and input/output ports. The fuzzy logic trainer and neural networks were implemented on the same microcontroller mainly for simplicity in data transmission; there was a minimal performance difference if the neural network was shifted to the second microcontroller.

The main performance issue with the experimental platform was reaction time. The Lidar used provided data at a maximum rate of 10 Hz, but the overall reaction time of the vehicle taking into account computation time was about 5-7 Hz. In order for real time navigation to be realistic, the vehicle must have been able to detect and respond to obstacles at a fast rate relative to its speed. The navigation speed used during the subsequent experiments was therefore minimized so as to ensure that the vehicle could appropriately detect and respond to obstacles. A quicker reaction time would have allowed for a faster average speed as well as a faster speed during the neural network training process.

3 Fuzzy Logic Trainer

The navigational logic for the fuzzy logic trainer was loosely based on the potential fields based method of navigation. The potential fields' method assumed that both the vehicle and all nearby obstacles possess an electrostatic charge. The negative gradient of the resulting electrostatic field represents the direction that the vehicle should travel. Similarly, the fuzzy logic trainer derived forces that were used to characterize the environment. A set of four forces were used as the input to the fuzzy logic speed calculation and a single net force was then calculated to be used as the input to the fuzzy logic steering angle calculation. The only input to the fuzzy logic trainer was the sensory data provided from the Lidar, which resulted in a control strategy that was suitable for training a neural network. The kinematics of the vehicle being used was taken into account partially through the design of the fuzzy logic membership functions and fuzzy rule base. The fuzzy logic navigational performance could have been improved through the use of additional sensors and inputs to the fuzzy logic trainer or with the use of additional filtering. However, this would have required significant amounts of processing power and increased the complexity of the implementation without necessarily improving the training process of the neural network.

First, the sensory data provided by the Lidar was processed. The Lidar provided a set of 256 data points representing the distance at which an obstacle was detected over its 180° field of view at about 0.70° angular intervals. This set of data points was then filtered to remove data that was considered out of range. A minimum range of about 10 cm was selected to ensure that excessively low measurements were ignored and a maximum range of about 70 cm was selected to ensure that the vehicle was not reacting to obstacles that were too far away.

The filtered set of data points was then converted into a set of four forces that represent the proximity of an obstacle within each of the 45° quadrants (Left-Extreme, Left, Right and Right-Extreme). This was done by taking the sum of the reciprocal of each filtered data point that falls within the respective quadrant, using the equation:

$$Force = \sum_i \frac{1}{d_i} \text{ where } d_i \text{ is a data point} \qquad (1)$$

The resulting four force values were then multiplied by an experimentally chosen constant that represented the required sensitivity of the Lidar to its surroundings. The force values were scaled to be within 0 to 255. A value of 0 indicated that there were no obstacles within range in the respective quadrant, whereas a value of 255 indicated the presence of an obstacle that had an average position that was close to the vehicle. This scaling simplified the design of later components such as the fuzzy logic membership functions. These four forces values were used as the input to the fuzzy speed calculations, and were further processed to produce a single force that was used as the input to the fuzzy steering angle calculation.

The four forces (LE, L, R, and RE) were used as the inputs to both the neural network and the fuzzy logic speed calculation. In order to perform the fuzzy logic steering angle calculation however, the four input forces were used to create a single force that represented the direction of the most significant obstacle. This was done by taking the difference of Left-Side (LS) and Right-Side (RS) forces. The LS force was generated using the logic:

$$LS\,Force = \begin{cases} \dfrac{LE+L}{2} & if\ LE \geq \alpha\ and\ L \geq \alpha \\ \max(LE, L) & if\ LE < \alpha\ or\ L < \alpha \end{cases} \quad (2)$$

Where α was a minimum threshold value. The RS force was generated using the same logic, but using RE and R forces instead of LE and L respectively. The result of the difference of LS and RS forces was a net force that represented the net presence of obstacles. A strong positive force indicated that the obstacle was more prevalently on the left, and a strong negative force indicated that the obstacle was more prevalently on the right. The magnitude of the net force was a function of both the proximity of the obstacle as well as whether the obstacle was clearly on one side of the vehicle rather than directly in front. This was due to the averaging of data points to produce the underlying forces and due to the usage of a single differential force rather than multiple individual forces.
The various membership functions that were used for the subsequent fuzzy logic calculations were all exponential curves. Exponential curves tended to better match the nonlinear dynamics of the input forces than the more commonly used triangular membership curves. For defuzzification, the centroid method was used, since this method encompassed the entire decision space more completely than alternative methods such as the mean of maximum method. The centroid method however was a significant computational burden for the experimental platform, and so the precision of the method was minimized so as to maximize speed while minimizing numerical error. This was done by using fewer points to represent the functions that were being used in the sums.

3.1 Fuzzy Steering Angle Controller

The fuzzy rule base for the steering angle is described as:

1) If (force) is NEGATIVE, go LEFT
2) If (force) is ZERO, go STRAIGHT
3) If (force) is POSITIVE, go RIGHT

Where NEGATIVE, ZERO and POSTIVE were fuzzy variables in the input space, as shown in Fig. 1, and LEFT, STRAIGHT and RIGHT were fuzzy variables in the output space, as shown in Fig. 2. The number of required rules was minimized by using a single net force as the input rather than the four forces generated prior. Using the four forces to design the fuzzy rule base did not result in any noticeable performance improvements.

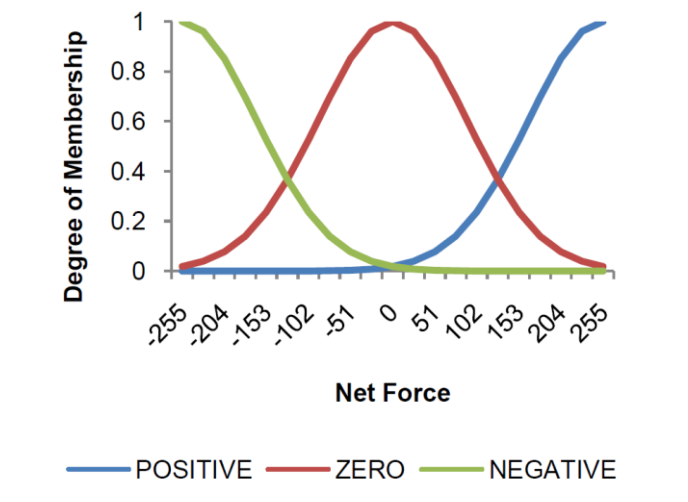

Fig. 1. Steering Angle Input Membership Functions

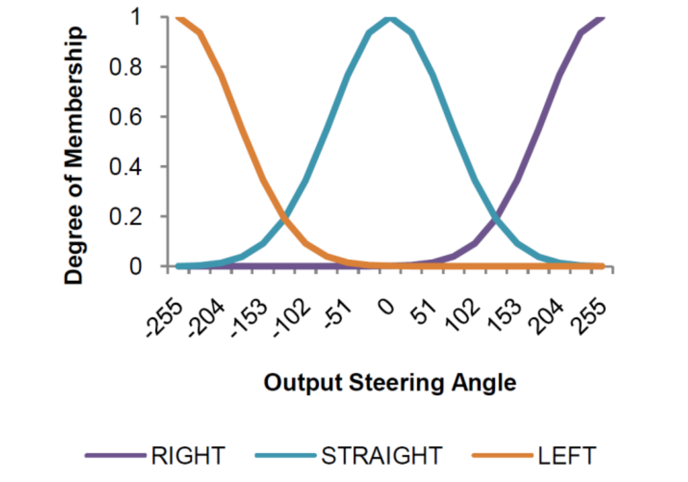

Fig. 2. Steering Angle Output Membership Functions

3.2 Fuzzy Speed Controller

The fuzzy rule base for the speed controller consisted of:

1) If (LE) AND (L) AND (RE) AND (R) are LOW, go FAST
2) If (LE) AND (RE) are HIGH AND If (L) AND (R) are LOW, go MEDIUM
3) If (LE) AND (L) AND (RE) AND (R) are HIGH, go SLOW

Where LOW and HIGH were fuzzy variables in the input space, as shown in Fig. 3, and FAST, MEDIUM and SLOW were fuzzy variables in the output space, as shown in Fig. 4. The minimum operator was used as the fuzzy AND operator. The general logic was to move slower when more obstacles are present in order to allow the vehicle the opportunity to react accordingly; that is to move more slowly during a sharper turn or in the presence of more obstacles. The vehicle operated within a minimum and maximum speed and the fuzzy speed calculation was used to specify a speed within this range. The design of the fuzzy logic trainer was such that the vehicle could navigate racetracks with smooth curves and simple obstacles. The fuzzy rule base was therefore designed accordingly.

The kinematics of the vehicle and the current vehicle velocity were being ignored when the speed was calculated. Ignoring these variables resulted in a significant simplification that had minimal negative impact on the vehicle performance. The maximum allowable speed was also reduced to compensate for the slow reaction time of the vehicle. The minimum allowable speed was increased to ensure that the vehicle could reliably overcome the friction of the driving surface as well as to ensure that speed could be maintained while turning. These speed constraints caused the overall variation in speed to be low, allowing simplified or suboptimal speed calculations to have a relatively minimal impact on the vehicle performance. Together, these restrictions allowed for effective performance in the experimental racetracks even with a minimal set of fuzzy rules.

4 Neural Netowrk

Artificial neural networks are a type of non-linear statistical model that are used to model patterns between inputs and outputs. Through the use of learning algorithms, neural networks possess the ability to learn patterns from a unique set of input and output data points. Neural networks are typically represented as a set of neurons connected together using with a set of weights. Each neuron generates an output between 0 and 1 based on its activation function. Each input to a neuron is multiplied by an associated weight, and this vector of input-weight pairs is used by the activation function to generate the neuron output. Learning is achieved through the systematic modification of the weights of the neural network. A simple two-layer feed-forward neural network is used to control the autonomous vehicle in this case, as shown in Fig. 5. This is among the simplest neural network topology that is suitable for autonomous navigation.

A feed-forward neural network was implemented to control the vehicle. The network received the four force values (LE, L, R and RE) as its input and produced the steering angle and speed values as its output. The neural network consisted of an output layer of two neurons and a hidden layer of 1-10 neurons. Only a single hidden layer was used since the presence of additional hidden layers was not necessary for successfully modeling the fuzzy logic trainer with an arbitrary level of precision [11-12]. The sigmoid function was used for the activation of each neuron and the sum of the product of the weights and inputs to each neuron was used to determine the activation state of the neuron. The back-propagation algorithm was used to train the neural network with no additional modifications such as a momentum term. The optimal global learning rate for the neural networks was determined in simulation to be within the 0.5-1.5 range, and so a global learning rate of 1

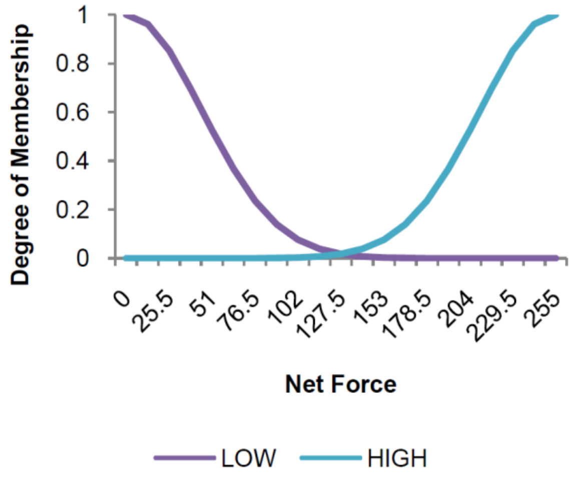

Fig. 3. Speed Input Membership Functions

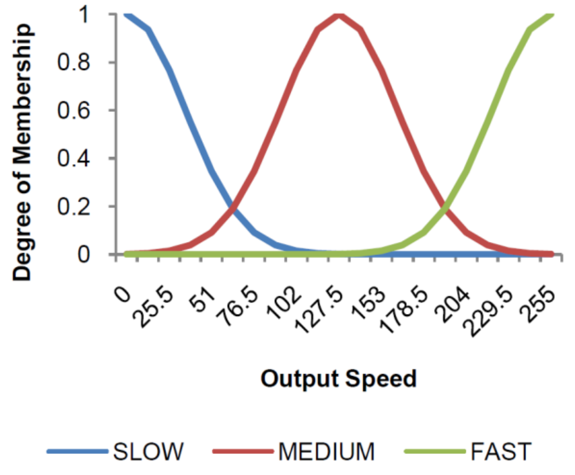

Fig. 4. Speed Output Membership Functions

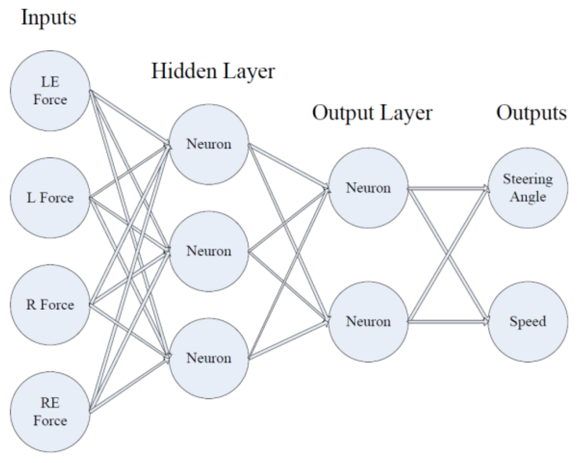

Fig. 5. Sample Feedforward Neural Network for Vehicle

was used for all tested neural networks. The process flow for the fuzzy logic trainer and neural network was:

1. The set of 4 force values were sent to both the fuzzy logic trainer and to the neural network as input. The force values were normalized to improve the efficiency of the neural network learning process.

2. The fuzzy logic trainer calculated the ideal steering angle and speed based on the input. This output was used to train the neural network.

3. The forward propagation algorithm was used to determine the steering angle and speed calculated from the neural network.

4. The outputs from both the neural network and the fuzzy logic trainer were used as inputs to the error back-propagation algorithm. The weights of the neural network were modified to minimize the calculated error:

$$\begin{aligned} e_{steering} &= e_{fuzzy\ steering} - e_{neural\ network} \\ e_{speed} &= e_{fuzzy\ speed} - e_{neural\ network} \end{aligned} \quad (3)$$

5. Steps 3-4 were repeated twice, resulting in a total of 3 iterations of the back-propagation algorithm for a given set of input force values. The number of iterations was selected to be the smallest number that would allow for feasible experimental performance. Additional iterations would require more processing power without necessarily improving the overall training process during experiments or simulation.

6. The final corrected output from the neural network was then used to control the vehicle. This was done to ensure that the training of the neural network would be relative to the performance of the neural network itself rather than be dependent on the fuzzy logic navigation. The control action of the neural network effectively generated a feedback loop in which the fuzzy logic trainer would train the neural network based on the surroundings of vehicle as well as the difference between the neural network and fuzzy trainer outputs. The training behaviour could easily be verified during experiment, since the navigational behaviour changed in real time.

Sensory data was not provided fast enough to properly train the neural network in real time. By repeating Steps 3-4 multiple times, the output produced from the neural network was significantly more accurate than if Steps 3-4 had only been run once. The training speed improvements that resulted from using multiple iterations of the back-propagation algorithm for a single input data vector outweighed the resulting computational penalties. While this was not the ideal method of training the neural network, this method improved the feasibility of real time training when insufficient training data was available.

The output from the fuzzy logic trainer was scaled to be within a range of 0.2 to 0.8. This was done to compensate for the lack of bias terms within the neural network and was verified in simulation to improve the overall training efficiency of the neural network. An output of 0.2 corresponded to a maximum turn left at minimum speed and an output of 0.8 corresponded to a maximum turn right at maximum speed. The training process for the neural network was designed to train the neural network in a manner that would ensure that the initial neural network output would be as accurate as possible.

5 Experimental Results and Analysis

Before experiments were performed using the racetracks, a simulation was performed to determine the number of epochs that the neural network would require to converge to a minimum level of error and to determine the rough error rates experienced during the learning process. An epoch was equivalent to one lap around the racetrack. The simulation consisted of a set of 10 neural networks, each with a different number of neurons in its hidden layer, being trained using a set of 256 unique training vectors. The input training vectors that were used were equally distributed over the entire range of possible inputs (since each of the input forces was bounded from 0 to 255). Mean squared error was measured using the fuzzy logic output and the neural network output over each epoch. Both the training process and the error metric were the same in simulation and in experiments, ensuring that simulation results could effectively be used to generate viable hypotheses for subsequent experiments.

The simulation demonstrated that the neural networks would converge to a minimum error within 10 epochs, as shown in Fig. 6. In addition, the larger neural networks appeared to have lower initial error and to converge to a lower error after training. However, while the larger neural networks appeared to more accurately model the fuzzy logic trainer, the rate of convergence appeared to be faster with less neurons in the hidden layer of the neural networks. The later experiments were designed using the assumption that a maximum of 10 epochs would be necessary to train each neural network.

Two separate racetracks were then designed to test the learning capability of various neural networks. A simple racetrack, as shown in Fig. 7, was developed to prove that the neural network can successfully be trained in real time. The second, more challenging racetrack, as shown in Fig. 8, was developed to demonstrate the repeatability of the experimental results in a different scenario. Each racetrack generated about 200-300 unique training vectors during each epoch, which was directly comparable to the number of training vectors used during simulation.

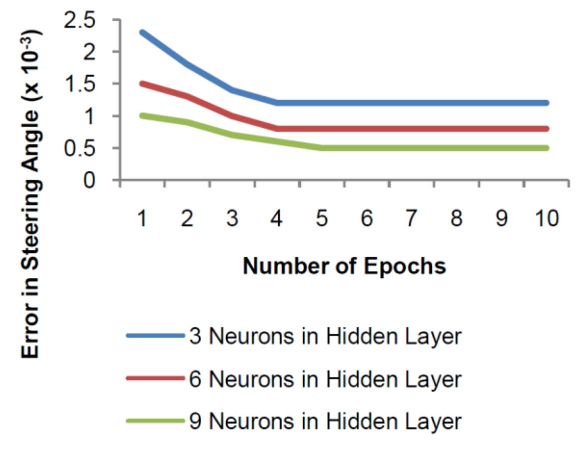

Fig. 6. Error in Steering Angle During Simulation

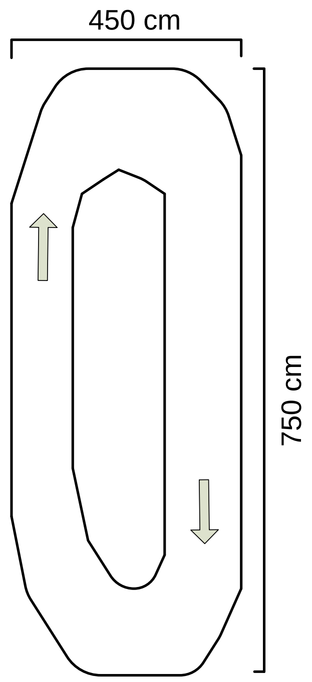

Fig. 7. Race Track #1

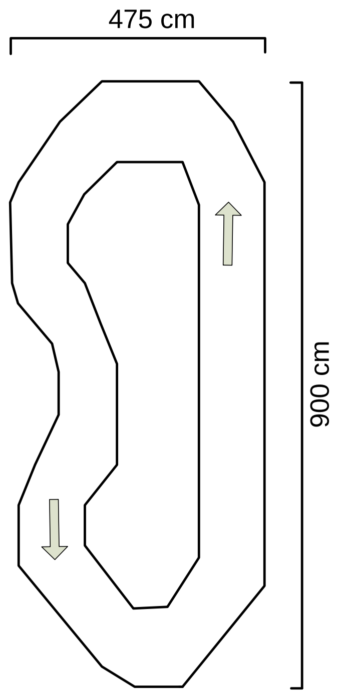

Fig. 8. Race Track #2

The experimental results demonstrated that the neural networks could successfully learn to navigate the racetrack when trained using a fuzzy logic trainer, as shown in Fig. 9. All the neural networks appeared to converge to roughly equivalent levels of minimum error and the rates of convergence were directly comparable to what was demonstrated during simulation.

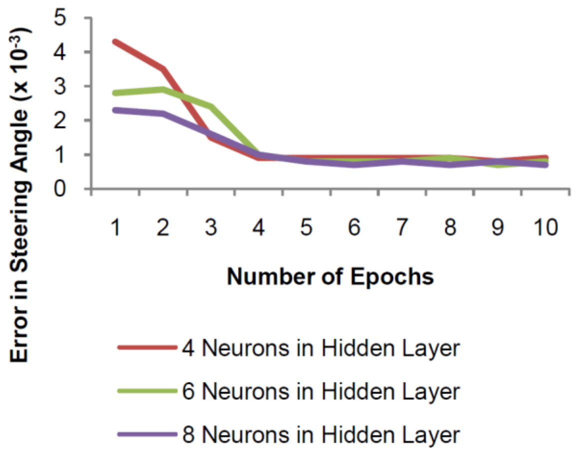

Fig. 9. Error in Steering Angle Over 10 Epochs

5.1 Experiment #1

The vehicle was placed in Racetrack #1 and navigated 10 epochs while being controlled by an untrained neural network. Neural networks containing 2, 4, 6, 8 and 10 neurons respectively in their hidden layers were tested. The goal of this experiment was to determine if a neural network could successfully be trained while attempting to navigate a racetrack. It was also intended to determine if the observations made during simulation were valid experimentally.

All the tested neural networks appeared to be fully trained in about 5 epochs. Using this result, the experiment was then repeated, but after 5 epochs the vehicle was allowed to navigate the racetrack while being controlled exclusively by the now trained neural network. This was done in order to determine the performance of the trained neural network without interaction with the fuzzy logic trainer.

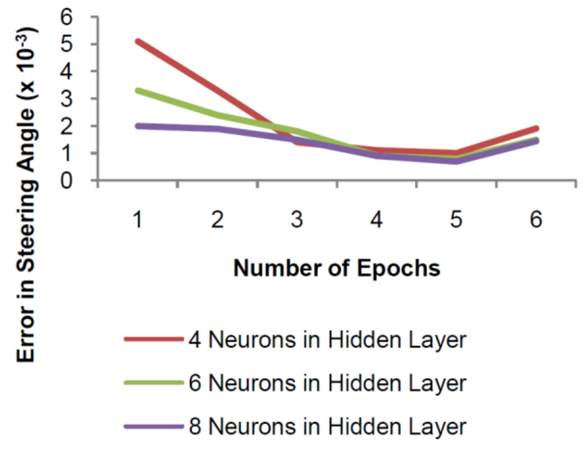

Fig. 10. Error in Steering Angle Over 5 Epochs

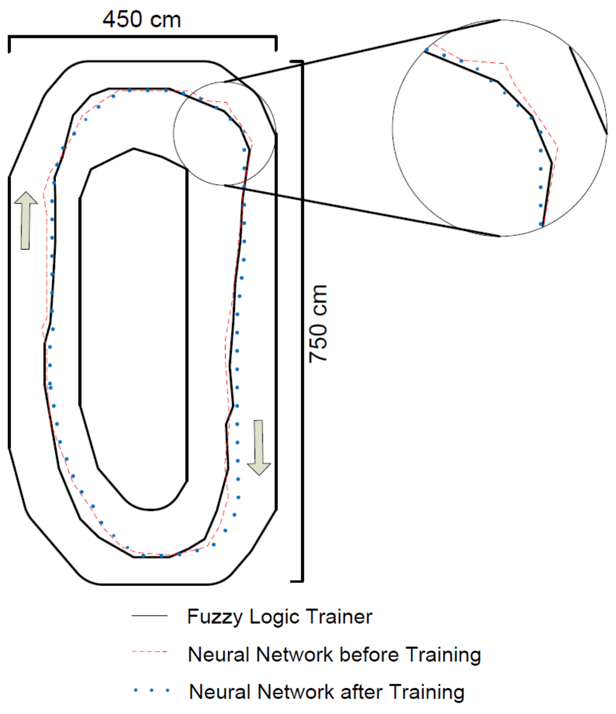

Fig. 11. Approximate Trajectory of Vehicle During Experiment #2

The experimental results verified that the neural network was successfully able to learn to navigate the racetrack after 5 epochs of training. The error rates of the tested neural networks after training were all relatively similar regardless of the size after the neural networks were trained, as shown in Fig. 10. When the trajectory of the vehicle while being controlled by a neural network with 6 neurons in its hidden layer was recorded, there was not a significant difference between the fuzzy logic trainer and neural network trajectories, as shown in Fig. 11. The neural network after training appeared to have a smoother trajectory that was more consistently in the centre of the racetrack than the less consistent trajectory before training. The performance of the neural networks was directly comparable to the performance using only the fuzzy logic trainer.

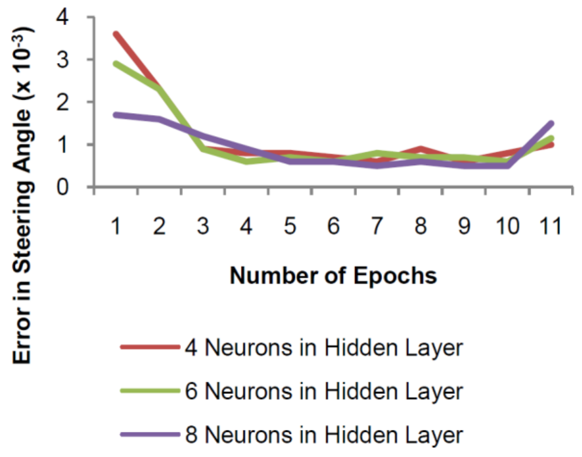

Fig. 12. Error in Steering Angle During Experiment #2

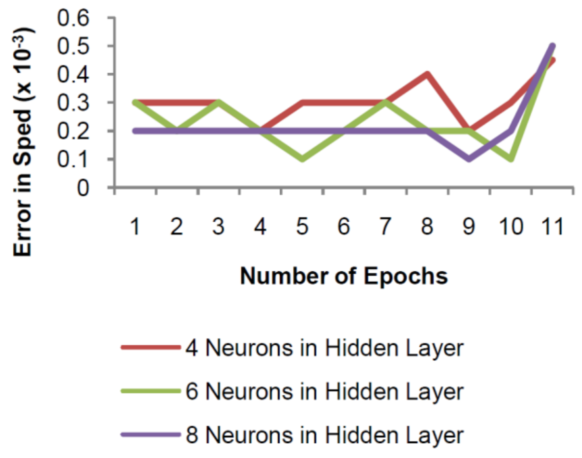

Fig. 13. Error in Speed During Experiment #2

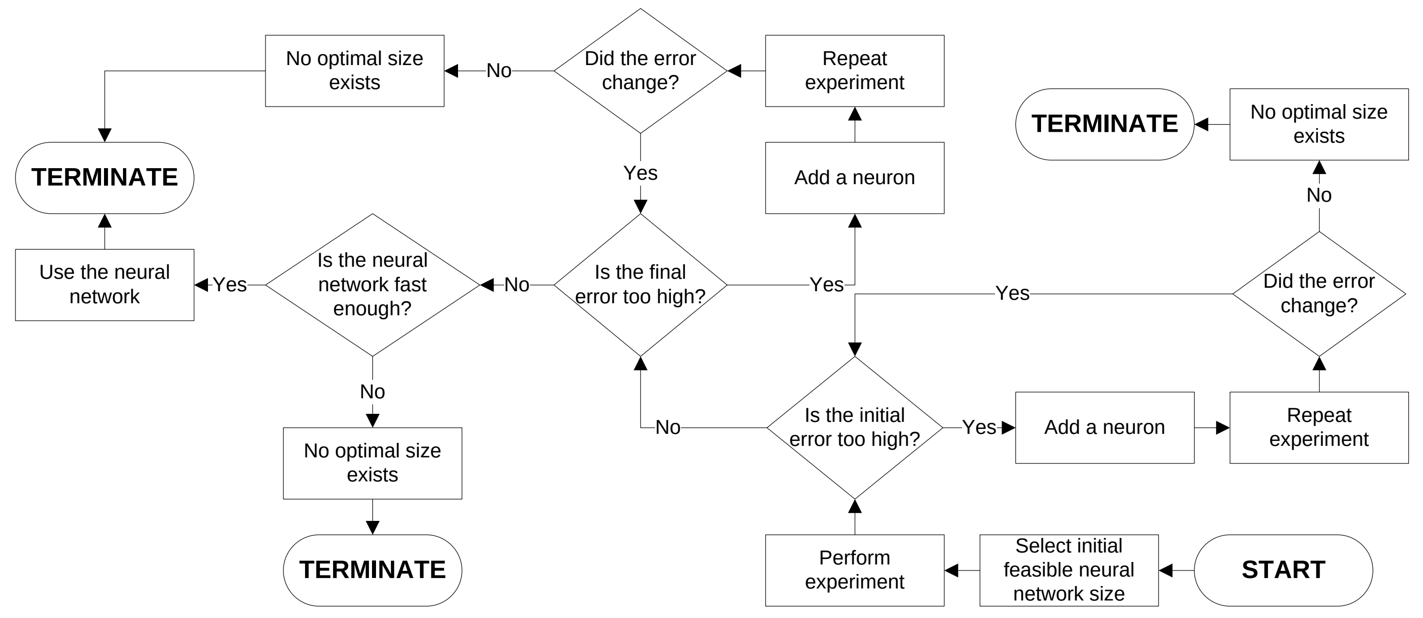

Fig. 14. Neural Network Size Selection Process Flow

5.2 Experiment #2

In order to select an optimal neural network size, it was necessary to determine the impact of racetrack complexity on the overall learning capability of the neural networks. The vehicle was placed in Racetrack #2 and navigated 10 epochs while being controlled by an untrained neural network. After the neural network had completed the 10 training epochs, an additional epoch was performed in which the trained neural network navigated the racetrack without interaction with the fuzzy logic controller. By comparing the observed results against the previous experimental results, more generalized conclusions could then be made neural network sizing.

The experimental results demonstrated that the general training behaviour was not significantly impacted by racetrack complexity, as shown in Fig. 12 and Fig. 13. As more neurons were added to the hidden layer, the error during the initial epoch reduced. The additional neurons also increased the number of epochs required for the neural network to converge to its minimum error. However, all the tested neural networks had similar performance after training, and so increasing the size of the neural network did not appear to have a significant benefit as far as minimizing the error.

In this case, the steering angle error converged within about 5 epoch cycles whereas the speed error converged within the initial epoch cycle to a more accurate value. This was due to the design of the fuzzy speed rules and the experimental racetracks. As a result, the back-propagation algorithm minimized the global error of both the speed and steering angle outputs and so caused fluctuations in the speed error while minimizing the steering angle error. Using separate neural networks for each output would have removed a significant amount of the fluctuations in the error of the outputs.

The vehicle speed during the training process was about half of the vehicle speed using only the fuzzy logic trainer. After training, the vehicle speed of the vehicle using the neural network was roughly equivalent to the vehicle speed using the fuzzy logic trainer. The neural network implementation was computationally more efficient than the fuzzy logic trainer, resulting in a slightly greater reaction time with the neural network that would have allowed for increases in the average vehicle speed.

6 Selection of Optimal Neural Network Size

There are several criteria that impact the selection of the optimal number of neurons in the hidden layer of the neural network:

1. The hidden layer must be large enough to approximate the fuzzy logic trainer with sufficient accuracy. This criterion was demonstrated through experimental results to not be a significant factor as a wide range of tested hidden layer sizes converged to roughly the same level of error.

2. The computational burden of the neural network must not minimize its practicality during training or autonomous driving. All the tested neural network sizes met this criterion.

3. The initial error during the training process must be low enough to allow the neural network to navigate the unknown racetrack. While all tested neural networks met this criterion, the probability of a

successful test run increased as the hidden layer size increased. The neural networks with less hidden layer neurons were generally unreliable during the initial test run.

4. The neural network must require a minimal number of epochs to successfully 'learn' the given racetrack. This requirement maximized the practicality of the neural network, since the training process was such that it was impractical to complete the hundreds of epochs that are possible during simulation. The experimental results effectively require the neural network size to be minimized for this criterion to be met.

Based on these criteria, a selection process was designed to determine a neural network size that is within an optimal range, as shown in Fig. 14. It was assumed that a neural network would be trained in real time over a limited number of epochs in an experimental environment. An initial neural network size was assumed to exist and be chosen such that it was as small as possible while still being large enough to be able to solve the given problem. Adding an additional neuron was assumed to reduce the initial and final error of the neural network, similarly to the tested neural networks. If a neural network existed that simultaneously met all of the criteria, then the process would terminate with an optimal size. If there was no solution that satisfied all the criteria, then the process would terminate because no optimal solution was found. This indicated that either the learning process must be modified, that the error constraints were too stringent, or that the computational limitations were too severe. Using the selection process, a neural network consisting of about 4-6 neurons in its hidden layer was determined to be optimal. The initial error rate was too high with less than 4 neurons and the computational penalty did not justify greater than 6 neurons. The exact sizes were relative to the difficulty of the racetracks that were being navigated and to the number of obstacles.

7 SUMMARY

It was clearly demonstrated that a fuzzy logic trainer could be used to train a neural network in real time as the neural network controlled a vehicle navigating an unknown racetrack. It was also demonstrated that even with a minimal fuzzy logic implementation and significant hardware constraints the neural networks were able to learn suitably to navigate the tested. The experimental results demonstrated that usable real time autonomous navigation could be implemented without elaborate control strategies or significant computational power.

In addition, it was observed that the number of neurons in the hidden layer primarily impacted the initial accuracy of the neural network and the rate of convergence to a minimum level of error. While increasing the size of the neural network appeared to improve performance in simulation, it did not significantly improve performance during experiments. The neural networks were also observed to perform competitively when compared against the fuzzy logic trainer even with a minimal number of training epochs. While the observations were not necessary valid for all possible neural network or fuzzy logic trainer implementations, the emphasis on initial feasibility rather than final error reduction was a noteworthy shift from traditional neural network size selection processes'.

Based on these observations a selection process was developed to determine the optimal number of neurons for the given neural network. The process of optimizing the size of a neural network was designed to be applied in experiment driven applications in which other, more theoretical optimization methods would be unsuitable. The optimal number of neurons in the hidden layer for the tested application was determined to be about 4-6 based on this selection process. While the selection process may not produce feasible or optimal results in all possible neural network topologies, in similar feed-forward neural network applications the selection process should produce usable, near-optimal neural network sizes.

ACKNOWLEDGMENTS

The authors would like to acknowledge the financial support and encouragement of Yousef Awad Al-Ahmadi, Director, Ideas Development Co. Ltd. Furthermore, the authors would like to acknowledge the contributions of Arun Das, Byung Yoon Kim and Sanjay Singh in hardware design, mechanical design, and design of the experimental setup for the autonomous vehicle.

REFERENCES

[1] Shigeki Ishikawa. "A Method of Indoor Mobile Robot Navigation by Using Fuzzy Control." IEEE International Workshop on Robots and Systems, pp 1013-1018, November 1991.

[2] Patrick Reignier. "Fuzzy logic techniques for mobile robot obstacle avoidance." Elsevier - Robotics and Autonomous Systems, vol. 12, pp 143-153, 1994.

[3] L.H. Tsoukalas, E. N. Houstis, G. V. Jones. "Neurofuzzy Motion Planners for Intelligent Robots." Journal of Intelligent and Robotic Systems vol. 19, pp 339-356, 1997.

[4] Jean Bosco Mbede, Pierre Ele, Chantal-Marguerite Mveh-Abia, Youssoufi Toure, Volker Graefe, Shugen Ma. "Intelligent mobile manipulator navigation using adaptive neuro-fuzzy systems." Elesevier - Information Sciences, vol. 171, pp 447-474, 2005.

[5] Shunming Li, Jianghui Xin, Weiyan Shang, Shen Huan, Wenqing Xiu. "The algorithm of obstacle avoidance based on improved fuzzy neural networks fusion for exploration vehicle." WSEAS Transactions on Systems and Control, vol. 3, iss. 4, pp 140-150, March 2009.

[6] Terence D. Sanger. "Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network." Neural Networks, vol. 2, pp 459-473, 1989.

[7] Hieu Trung Huynh, Yonggwan Won. "Online training for single hidden-layer feedforward neural networks using RLS-ELM." IEEE International Symposium on Computational Intelligence in Robotics and Automation, pp 469-

473, December 15-18 2009.

[8] Shumeet Baluja. "Evolution of an Artificial Neural Network Based Autonomous Land Vehicle Controller." IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, vol. 26, iss. 3, pp 450-463, June 1996.

[9] Dean A. Pomerleau. "Efficient Training of Artificial Neural Networks for Autonomous Navigation." Neural Computation, vol.3, iss. 1, pp 88-97, Spring 1991.

[10] Kazuyuki Hara, Kenji Nakayama. "Selection of Minimum Training Data for Generalization and On-line Training by Multilayer Neural Networks." IEEE International Conference on Neural Networks, vol. 1, pp 436-441, June 3-6 1996.

[11] Bernd Freisleben, Thomas Kunkelmann. "Combining Fuzzy Logic and Neural Networks to Control an Autonomous Vehicle." Second IEEE International Conference on Fuzzy Systems, vol. 1, pp 321-326, 1993.

[12] G. Cybenko. “Approximation by Superpositions of a Sigmoidal Function.” Mathematics of Control, Signals and Systems, vol. 2, pp 303-314, 1989.