An Intelligent Mobile Robot Controller Using Neuro-Fuzzy Techniques

X

By

Ayman A. Abu-Baker

Supervisor

Dr. Khaldoun Tahboub

Co-Supervisor

Dr. Munaf S. N. Al-Din

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Industrial Engineering

Faculty of Graduate Studies

University of Jordan

August 2002

This thesis was successfully defended and approved on 19th of August 2002

Examination Committee	**Signature**
Dr. Khaldon Tahboub, Chairman Industrial Engineering Department University of Jordan	[illegible]
Dr. Munaf S. N. Al-Din, Co-Supervisor Electrical Engineering Department Al-balqa' Applied University	[illegible]
Dr. Zain El-Abideen Tahboub, Member Industrial Engineering Department University of Jordan	[illegible]
Prof.. Mohamad Z. Khader, Member Electrical Engineering Department University of Jordan	[illegible]
Dr. Moh'd Al-Ebenee, Member Biomedical Engineering Department Jordan University for Science and Technology	[illegible]

Dedication

To My Father And Mother

To My Brother Omar

To My Sisters

Acknowledgement

I am very grateful for the advice, encouragement, and education provided by my adviser and co-adviser Dr. khaldoun Tahboub and Dr. Munaf S. N. Alden. I would like to express my deep appreciation and thanks to the examination committee Dr. Zain El-Abideen Tahboub, Prof. Moh'd Z. Khader and Dr. Moh'd Al-Ebenee for their professional criticisms and valuable suggestions and aid, and, most of all, long term support.

It is also a pleasure to acknowledge to Eng. Abdel latef abu-Khadejah, Eng. Doried Mismar and Eng. Imad Shehadeh who have contributed to my research effort on robot and thus the production of this thesis.

Finally, none of this would have been possible without my family's constant encouragement and love.

Table of Contents

List of Tables

List of Figures

Abbreviations and Symbols

$\mu(x)$	: Membership function of a crisp input x.
dl	: Left distance measured from the left front sensor of the robot.
dc	: Front distance measured from the front sensor of the robot.
dr	: Right distance measured from the right front sensor of the robot.
tr	: Target orientation (the angle between the center of the robot and the target).
LB	: The target is Left Big.
LS	: The target is Left Small.
Z	: The target is in the same direction.
RS	: The target is Right Small.
RB	: The target is Right Big.
Sa	: Turning Angle of the robot.
TLB	: Turn the robot Left Big
TLS	: Turn the robot Left Small
TZ	: Turn the robot Zero
TRS	: Turn the robot Right Small
TRB	: Turn the robot Right Big
FLC40	: Forty-Rules Fuzzy Navigator System
FLC625	: Improvement Fuzzy Navigator System
VS	: Very Small Distance Detected
S	: Small Distance Detected
M	: Medium Distance Detected
L	: Large Distance Detected
VL	: Very Large Distance Detected
NNT	: Neural Networks
NFC	: Hybrid (Neuro-Fuzzy) Controller

An Intelligent Mobile Robot Controller Using Neuro-Fuzzy Techniques

By

Ayman A. Abu-Baker

Supervisor

Dr. Khaldoun Tahboub

Co-Supervisor

Dr. Munaf S. N. Al-Din

Abstract

An autonomous mobile robot is a robot that is capable of moving from a starting point towards a target point while avoiding collision with possible obstacles along its path without human intervention. The field of mobile robots is finding great application possibilities on the factory floor, as service robots, land exploration, etc. Research issues concerning autonomous mobile robots include real-time path planning, obstacle detection and avoidance, and avoiding trap situations.

This thesis addresses the problem of moving a mobile robot autonomously from a starting point to a target point. The objective is to make the robot move along a collision free trajectory until it reaches its target. The approach taken here utilizes a hybrid neuro-fuzzy method where the inference engine is replaced by a collection of five parallel neural networks in order to reduce computational time for real-time applications. Initially, a classical fuzzy logic controller has been constructed for the path planning problem. The inference engine required 625 if-then rules for its implementation. Simulation experiments on the developed path planning control algorithm have shown that it supercedes existing implementations in the literature. Then a neuro-fuzzy controller was constructed. The five neural networks were trained using data sets randomly selected from the original fuzzy matrix. Simulation experiments were conducted to test the performance of the developed controller and the results proved that the approach to be practical for real time applications. For example, the inference CPU time of fourty rules fuzzy navigator is measured to be 132 μS with slightly or severely colliding with obstacles. To avoid collision with obstacles, the inference engine rules were increased to 625. This improvement increased the inference CPU time by 13 times. A model of neuro-fuzzy controller with single node achieved better performance – no collision with obstacles- with inference CPU time reduction by 38% of 625 inference engine time.

Finally, the developed neuro-fuzzy controller was tested on a prototype mobile robot which was constructed as part of a research project funded by the Faculty of Graduate Studies. Experimentation with the robot proved the practicality of the developed approach of this work.

INTRODUCTION

1.1 Introduction:

A robot device is an instrumented mechanism used in science or industry to take the place of a human being. It may or may not physically resemble a human or perform its tasks in a human way, and the line separating robot devices from merely automated machinery is not always easy to define. In general, the more sophisticated and individualized the machine is, the more likely it is to be classed as a robot device.

The blend of automation development with the progress of automation and cybernetics began to take shape in the 1950s and 1960s. One line of development was represented by the synthetic robot, a computer simulation model of a human being. An example was SAMMIE, created at the University of Nottingham in England and designed to show how a human would perform in various environments, such as an aircraft cockpit. Another type of research involved the construction of such experimental automata as a mechanical horse built by a laboratory team at the General Electric plant, in Schenectady, N.Y., which would be ridden over rough ground at 48 Km/hr; and an android, constructed at Massachusetts Institute of Technology, which could be programmed to walk down a corridor, enter a room, position furniture so as to facilitate retrieval of a book, and return to its starting point.

Modern industrial robot devices aim to substitute to a machine for a man in hostile environments, cut costs by replacing expensive hand labor with cheap dependable machines; and provide versatile all-purpose robots at predictable cost. The alternative is special device for each operation, with higher cost. Robots of high level are capable of adapting to changes in environment, but are also capable of making decisions by selecting the proper solution from among several alternatives stored in memory units. Robotic control involves:(1) giving instructions (programming) to device and stored information (memory) within it; (2) gaining the knowledge of the state or position of the device and processing this information to decide a course of action (feedback); (3) transformating this feedback into action (response). The trend in the field of industrial robots since the mid 1990's is mainly focused on the development of intelligent robots, that can perform the required tasks without the human intervention by using what is called intelligent controllers, it can accept and remember instructions. Such a robot is intended to improve man's sense, or add new ones, and then to do his bidding.

The mobile robot is constructed as manipulator for auto sensing, which is capable to navigate in an unknown topology with moving and stationary obstacles. In the design of this kind of autonomous system, two important aspects need to be addressed carefully. The first of which is the design of a nonlinear and non-analytical controller, whereas the second deals with the reliability of such a system. It can be stated that human experience represented by a set of linguistic rules can be a sensible solution to this kind of control problems. Such controllers are called fuzzy logic controllers, which are designed to mimic human experience. Although, fuzzy logic controllers are robust, but deriving and fine-tuning the entire rule set of a fuzzy logic controller is a tedious and difficult problem. Furthermore, fuzzy controllers can be very slow in response when used for large-scale problems.

Another type of intelligent controllers based on neural networks can also be used for such problems. Neural controllers are basically a set of simple processing units connected together through weighted connectors, which are trained with predefined patterns to perform a specified task. The major problem when using neural controllers is that discrete input representations may cause such systems to be unstable. In addition, a large number of training patterns are usually required to enhance the controller and this in turn will increase the training time.

The main objectives of this research are to develop a fuzzy and hybrid (Neuro-Fuzzy) controller that can easily move the robot to the target with improved resolution. The fuzzy controller is designed to avoid the robot from collision with obstacles, to accomplish that, the fuzzy logic controller was modified to hybrid (Neuro-Fuzzy) controller.

1.2 Statement of the Problem

The mobile robot is required to explore several paths in a maze, in a pattern of successive combinations of left and right turns. Its task is to reach a favored position at the end of one channel. The mobile robot uses a kind process, sequentially adopting cyclic pattern of the left and right turns. Eventually it ends up in the desired position, at which time a signal is injected, causing the robot to record the correct pattern. The mobile robot is assumed to be equipped with three physical ultrasonic sensors and one virtual sensor. The physical sensors are used to detect obstacles in front of the robot, on the right side, and on the left side, respectively. The maximum distance that can be sensed by these sensors is assumed to be 6 meters. The virtual sensor is used to guide the robot towards the target. This sensor is especially needed when the target direction of movement is totally blocked by an obstacle. The virtual sensor will guide the robot back towards the target once the obstacle is avoided.

Henceforth, the robot travels quickly and accurately along the track to accomplish any job that has been assigned. It is assumed here that the robot will not face any traps (or get into a situation where it is required to backtrack or turn around). Such a problem is not dealt with in this research and is left as a recommendation for future work.

The four sensors provide the path planning system (in our case a fuzzy logic system) of the robot with three distances front (dc), right (dr), left (dl), and target orientation (theta), respectively. From these inputs, the fuzzy logic controller will workout a decision in which direction should the robot move in order to reach the target. The fuzzy logic controller will go through three stages, i.e., fuzzification, inference, and defuzzification.

The fuzzy logic controller (FLC) was analyzed and tested for different cases based on the same parameters and rules used by Xu and Tso(Xu and Tso,1996). The robot motion results have been considered with relation to different cases. Problems were recorded and investigated and the reasons behind the failure of this robot, in these cases, were related to the limited number of the sets used (FAR, NEAR), and the limited angle of orientation (turning angle), which are five sets. Because of this limitation, the robot touches the obstacles slightly in all cases considered. To avoid these problems, a relaxation of the rules was done by increasing the number of sets for the input distances from two to five sets, accordingly the number of rules was increased to 625 activation rules. The results obtained from this improved fuzzy logic controller have been improved. The robot avoids collision with the obstacles, but the main problem in using that improved controller is the processing time. It is very long, since the number of rules is high and requires more time to create a decision and this will affect the response time of the robot.

To solve the above-mentioned problem of processing time, the inference engine was replaced with a neural network. The system is investigated by considering the results of the marriage between both systems (Fuzzy logic and neural networks). The outcome has been efficient and accurate but it requires training, which was introduced for that system. To do the training, the sample turns out to be very large (83521 × 20) and the system faces problems. To overcome that huge sample, the NN was structured as five parallel networks, each has 20 input nodes and one output node, and this improves the performance and the response time, which is the heart of this research. The results obtained with the input to the NN are fuzzy and the output is Fuzzy too as before.

1.3 Scope of the Problem

In this research, it is proposed to investigate a hybrid neuro-fuzzy controller in order to benefit from both advantages of fuzzy logic and neural networks in grasping human experience with continuous representation.

This thesis describe Baker current work in the area of Hybrid neuro-fuzzy controller, including the tools used to marry the Neural Networks with the improvement fuzzy using better simulation technology. The Hybrid controller produced, has good capability with better performance and response time. The simulation results were tremendous since no collision at all with the obstacle. The restructured of the fuzzy logic controller to Hybrid with the same fuzzy input and fuzzy output have been accomplished. This thesis discusses the shortcomings of the fuzzy logic and the techniques used to overcome its limitation. Work has been concentrated on development of modeling capabilities that go beyond simulation.

1.4 Organization of the Thesis

This thesis falls into six chapters; in this chapter FLC in the past and the present works in the area of Robot is presented. This work involves a number of system applications that use knowledge- engineering tools such as inference engine, neural network and hybrid neuro-fuzzy controllers. *Chapter 2,* discusses exemplary the theoretical background of the fuzzy logic, neural networks and hybrid neuro-fuzzy approach showing how knowledge can be acquired from these controllers through the use of examples. The author describes a model that allows a computer programs to be created by example, and discusses research issues involved in developing such systems. *Chapter 3,* a literature review, describes the fuzzy logic and neuro-fuzzy approaches of mobile robot navigation by different researchers. *Chapter 4* presents the development of the author proposed neuro- fuzzy navigator system, it relates the results of an ongoing investigation of FLC 40, it discusses the development of Hybrid navigator system. This research particularly noteworthy for it is one of a few efforts involved in the fairly new area of parallel distribution system using knowledge- based simulation to test various cases. The author then presents a set of problem solvers developed for autonomous mobile robot. Finally, some experimental findings that affect the performance and response time of such robot are described. Moving toward *Chapter 5,* describes the design of the simulation software and the components of the simulation screen, and discussing the hardware design for the robot to suit the different controllers. Also it details with the experimental results of the three controllers. Finally it highlights on the work done by the author, which details the attempts to marry the improvement fuzzy logic with the neural networks, which benefits from both systems. A comparison between different systems is outlined.

In this study, the objectives of the present work are outlined, and the way of information can be represented in a hybrid system was described. Some preliminary attempts to create programs that help the hybrid input information into the computer was described and finally the usefulness of providing robot with this type of tool was discussed.

Finally Chapter 6 has presented and illustrated the conclusions of this work in addition to providing the interested reader with the recommendations for the future work in the field of autonomous mobile robot field.

THEORETICAL BACKGROUND

2.1 Introduction

One of the primary issues in artificial intelligence (AI) has been the choice between two fundamentally different (and often viewed as competing) approaches to building intelligent systems — traditional symbolic AI and numeric (sub-symbolic) artificial neural networks (ANNs). This has been an issue engaging the AI community for three decades, and there have been attempts to bridge the gap between these two paradigms in order to take advantage of the relative merits of each (Zadeh, 1996).

In an attempt to model the human mind/brain it has been necessary to oversimplify the structure (resulting in ANNs) and the function (resulting in precisely defined symbolic-AI programmes) of the brain. Symbolic AI attempts to preprogram intelligence into a deterministic algorithm. On the other hand, most ANNs are equipped with relatively weak forms of learning (i.e. tuning a fixed set of parameters or weights). It has been argued (Uhr and Honavar, 1994) that despite the seemingly different approaches that symbolic AI and ANNs take to building intelligent systems, they both share common origins, and are both based on the hypothesis that cognition can be modeled by computation. Tasks performed by ANNs can be performed by symbolic AI and vice versa as both paradigms rely on different but essentially equivalent models of computation (Uhr and Honavar, 1994). Decades of collective experience by theoreticians and practitioners in several areas of computer science have shown that it is efficiency, robustness and elegance that determine the best approach. Hybrid systems resulting from the integration of concepts and technologies drawn from both traditional AI systems and ANNs clearly demonstrate the potential benefits for the design of truly robust, flexible and adaptive intelligent systems in a wide application domain. This chapter concentrates on one of many promising approaches for developing hybrid intelligent systems known as soft computing (SC). SC is not a single methodology, rather it is a partnership. The principal partners at this juncture are fuzzy logic (FL), neuro-computing (NC), and probabilistic reasoning (PR), which subsumes genetic algorithms (GA), chaotic systems, belief networks and parts of learning theory.

2.2. Fuzzy Logic

Fuzzy logic is one of the newest paradigms in the control system community, and it is originally invented in 1965 by Lotfi A. Zadeh (Tanaka, 1996). The basic idea in using fuzzy logic is to represent human thought by allowing a user to describe a system using natural words, and create a control mechanism based on "expert rules" or in other words, rules developed through experience. A fuzzy controller is developed without creating a mathematical model of the system. For example, a fuzzy logic code may contain the following linguistic rule:

If the angle of the input line is high, then turn the vehicle quickly to the right.

The blend of these simple rules allows the system to be easily analyzed, controlled, modified, and understood.

The foundations of Fuzzy Logic are as old as the discipline of the traditional logic theory. Classical logic theory is based on the principle of "crisp" sets, or the law of the excluded middle which says X must be in set A or in set not-A. Everything falls in one group or another; there is nothing that is both a day of the week and not a day of the week. Often, classical set theory assigns a membership value to the universe 1 for true statements, 0 for false. However, human thoughts and behaviors allow consideration of vagueness and imprecision. The concepts of fuzzy logic attempt to describe the universe in a more natural manner where each element of the set being considered, or the **universe of discourse**, to belong to a set *to a certain degree*. This degree of belonging is represented in fuzzy sets by a membership value that varies between 0 and 1. Of course, classical set theory is a special case of fuzzy set theory that contains a discontinuous break in the membership of a set in the jump between true and not true. Fuzzy logic allows a more continuous, smooth variation from membership to non-membership in a set. For example, in mobile robot navigation, one of the important aspects is the distance relative to some reference object, which is maximum distance can be measured by an ultrasonic sensor. Therefore, the universe of discourse of the distance measurement is the continuous set of distances between 0 and 6 meters. Using the concepts of classical set theory, a statement could be made that asserts distances closer than two meters are considered near, while distances that are above 4 meters are considered not-near, and distances less than 1 meter are considered near, however the logic begins to break down near the transition point. It seems unreasonable to call a distance "near" that is just below 2 meters and a distance not near just above 2 meters, especially if these distances differ by a fraction of a centimeter. Fuzzy Logic allows the *membership function* of the set of near distances to vary continuously from 0 to 10 meter, from near to not near. Graphical descriptions of the "crisp" and "fuzzy" membership functions are shown below in Figure 2.1 and 2.2.

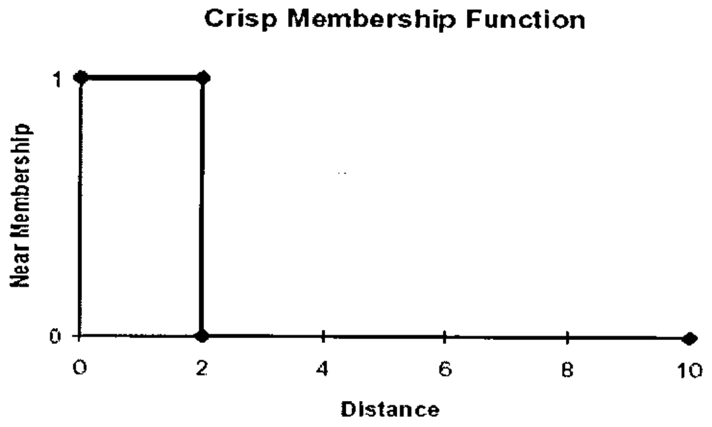


Figure 2.1: Example of a Crisp Membership Function

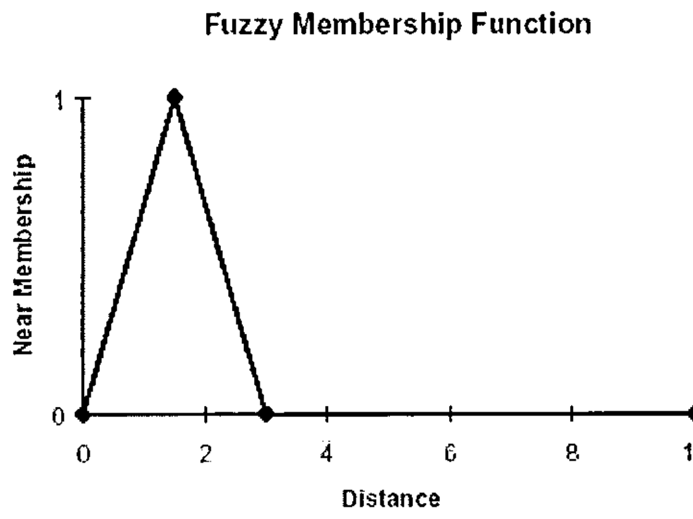


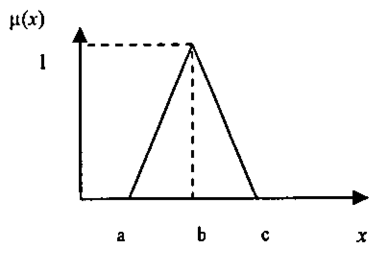
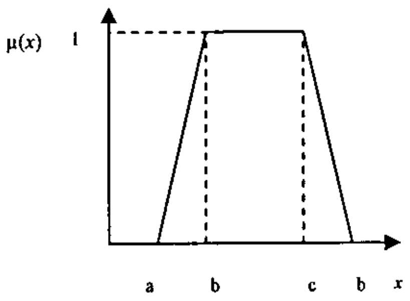
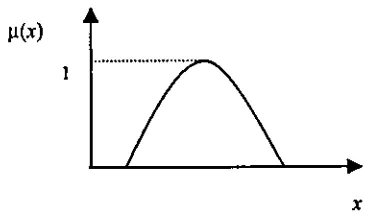
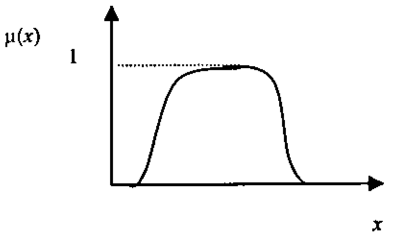
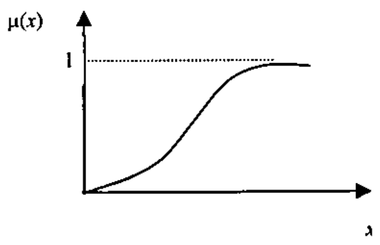
Figure 2.2: Example of a Fuzzy Membership Function

A **fuzzy set** is denoted by an ordered set of pairs, the first element of which denotes the element (x) and the second ($\mu_A(x)$) the *degree of membership:*

$$A = \{(x, \mu_A(x)) | x \in X\} \quad (2.1)$$

Where **a membership function** $\mu_A(x)$ represents how each point in the input space is mapped with a degree of membership to the set in consideration and takes values in the interval [0,1]. Membership functions can come in many shapes from bell curves, to s-functions, to even crisp "classical" membership functions. Although shapes of membership functions vary, ranges of the function usually vary between 0 and 1. Table (2.1) presents the mathematical and graphical representations of the most widely used membership functions.

Table 2.1: The Fuzzy Membership Functions

Membership Function	Mathematical Representation	Graphical Representation
Triangular	$T(X,a,b,c)=\begin{cases}0 & X\le a\\ \frac{(X-a)}{b-a} & a\le X\le b\\ \frac{(c-X)}{(c-b)} & b\le X\le c\\ 0 & X\ge c\end{cases}$	
Trapezoid	$T(X,a,b,c,d)=\begin{cases}0 & X\le a\\ \frac{(X-a)}{b-a} & a\le X\le b\\ 1 & b\le X\le c\\ \frac{(d-X)}{(c-b)} & c\le X\le d\\ 0 & X\ge d\end{cases}$	
Gaussian	$gaussmf(X,a,b,c)=e^{-\frac{1}{2}\left(\frac{X-c}{\sigma}\right)^2}$	
Generalized bell	$gbellmf(X,a,b,c)=\frac{1}{1+\left\|\frac{X-c}{b}\right\|^{2b}}$	
Sigmoid	$sigmf(X,a,c)=\frac{1}{1+e^{-a(X-c)}}$	

the operator is min) gives the fuzzy set A and B which is represented by its membership function $\mu_{A \ and \ B}$ (x) (solid line on Figure). The application of T-conorm (here max operator) on these fuzzy sets gives the fuzzy set represented with solid line on the Figure2.3

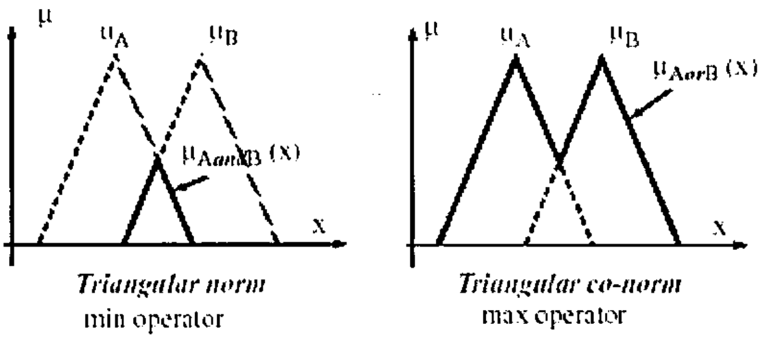


Figure 2.3 Fuzzy Operators

2.2.1 Fuzzy Propositions and Rule-Based System

We can use fuzzy sets to represent *linguistic variables.* Linguistic variables are in general used to represent the process states and control variables. Their values are defined in linguistic terms or fuzzy values. For example, for the linguistic variable Speed can define a set of term: S(Speed) = {very slow, slow, medium, high, very high}. The set of terms: S(Speed) can be characterized as fuzzy sets whose membership functions are shown in Figure 2.4. Every fuzzy set in a universe of discourse represents one *linguistic value* or *label.*

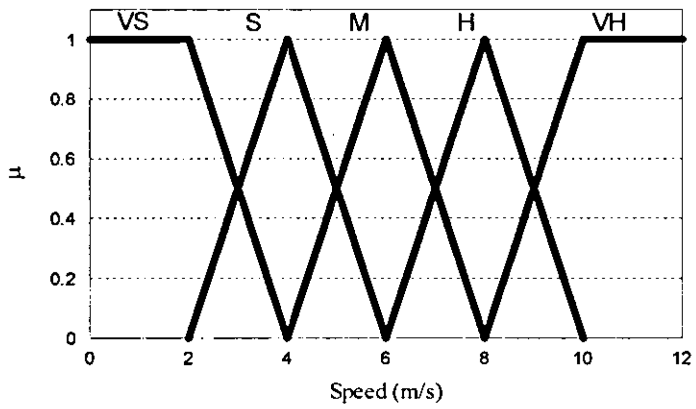


Figure 2.4 Fuzzy Propositions

As mentioned above, the principal idea of fuzzy logic systems is to express the human knowledge in the form of linguistic *if-then* rules. Every rule has two parts:

• Antecedent part (premise), expressed by **if...** and

• Consequent part, expressed by: **then...**

The antecedent part is the description of the state of the system, and the consequent is the action that the operator who controls the system must take. There are several forms of *if-then* rules. The general is:

If (a set of conditions is satisfied) ***then*** (a set of consequences can be inferred).

Zadeh (Zadeh, 1973) was the first who introduced a notion of fuzzy rule in the form:

Example: ***If*** the *distance to obstacle* is big, ***then*** the *speed* is high.

The general form of this rule is:

Rule: ***If*** *x* is A, ***then*** *y* is B.

distance to obstacle (*x*) and *speed* (*y*) are *linguistic variables. x* represents the state of the system, and *y* is control variable and represents the action of the operator. Big (A) and high (B) are *linguistic values* or *labels* characterized by appropriate membership functions of fuzzy sets. They are defined in the universe of discourse of the linguistic variables x and y.

The generalization of a control rule *R* which has two conditions in the antecedent part has a following form:

Rule: ***If*** *x* is A *and y* is B, ***then*** *z* is C

Also:

$R = (A \text{ } and \text{ } B) \rightarrow C$

The control rule is implemented by a *fuzzy implication* (fuzzy relation) and is defined as follows:

$\mu_R = \mu_{A \text{ } and \text{ } B \rightarrow C}(x\,y\,z) = \mu_A(x) \text{ } \boldsymbol{and} \text{ } \mu_B(y) \rightarrow \mu_C(z)$

A, B and C are fuzzy sets defined in the universes of discourse for x, y and z. μ_R is the *strength or weight* (level of the truth) for the statement R and μ_A is the *level of membership* (membership value) of the variable x in fuzzy set A.

2.2.2. General Structure of Fuzzy System

The structure of fuzzy system is very similar to the structure of classical AI expert system represented on Figure 2.5.

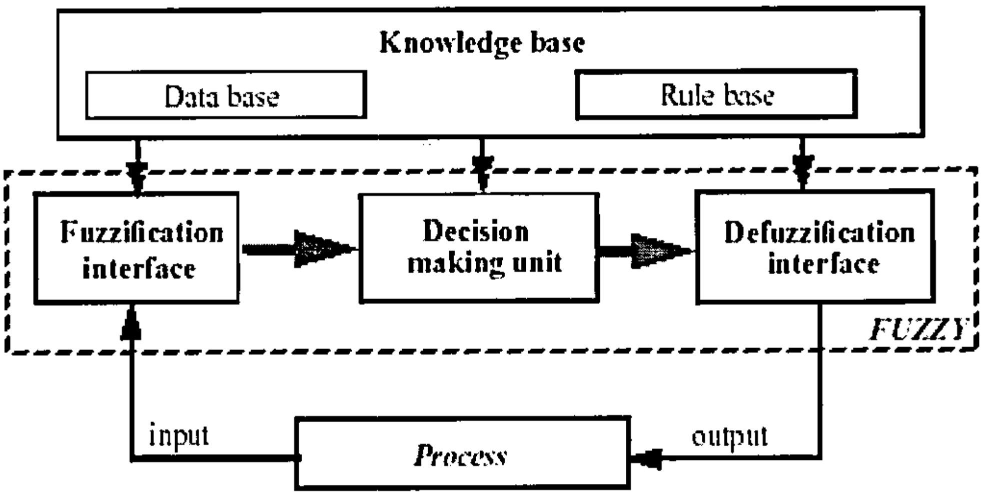


Figure 2.5 Structure of Classical AI Expert System

Every fuzzy system is composed of four principal blocks Figure 2.5:

1. **Fuzzification interface:** Under Fuzzification, the input variables (crisp inputs) are transformed into degrees of match with linguistic variables according to the membership functions defined over their universe of discourse.

2. **Decision unit:** Decision unit, or sometimes called inference engine, the fuzzy values for the premise of each rule is computed, and applied to the conclusion part of each rule. This results in one fuzzy subset to be assigned to each output variable for each

rule. Usually only MIN or PRODUCT are used as inference rules. In MIN inferencing, the output membership function is clipped off at a height corresponding to the rule premise's computed degree of truth (fuzzy logic AND). In PRODUCT inferencing, the output membership function is scaled by the rule premise's computed degree of truth.

3. **Knowledge base:** Knowledge base module in a fuzzy expert system are usually used to store the rules, parameters for membership functions and the rule base engine.

4. **Defuzzification interface:** The Defuzzification process converts a fuzzy output set to a crisp value. There are a number of defuzzification methods, three of the more common techniques are the *center-of-gravity, center-of-area* and *mean-of-maxima* methods. In the *center-of-gravity* method, the crisp value of the output variable is computed by finding the variable value of the center of gravity of the membership function for the fuzzy value. In the *mean-of-maxima* method, one of the variable values at which the fuzzy subset has its maximum truth-value is chosen as the crisp value for the output variable.

The ***FUZZY*** part of the system from Figure 2.5 is represented on the Figure 2.6.

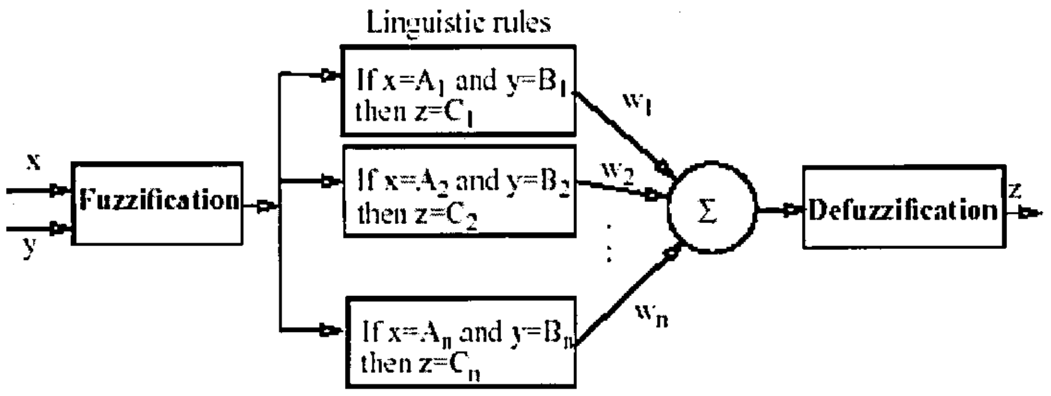

Linguistic rules
x
y
Fuzzification
If x=A1 and y=B1 then z=C1
If x=A2 and y=B2 then z=C2
If x=An and y=Bn then z=Cn
w1
w2
wn
Σ
Defuzzification
z

Figure 2.6 Fuzzy System

There are several types of fuzzy reasoning (Jang, 1992). The most important, in the literature, are:

• **Type 1**: The output membership function has to be monotonically non-decreasing (Tsukamoto, 1979). Then, the overall output is the weighted average of each rule's crisp output induced by the rule strength and output membership functions.

• **Type 2**: The operation applied to the qualified fuzzy outputs is max, and the crisp value of output is, most usually, the center of gravity of resulting fuzzy set. There are, also, the other methods to find the crisp output like: bisector of area, maximum criterion, mean of maxima, ..etc (Lee, 1990).

• **Type 3**: Takagi and Sugeno method. Each rule's output is a linear combination of input variables. The crisp output is the weighted average of each rules output.

Suppose that the fuzzy inference system has two inputs and one output. If we have to combine two rules:

R1: *If* x is A1 and y is B1 *then* z is C1

R2: *If* x is A2 and y is B2 *then* z is C2

the decision procedure for the three types of fuzzy inference systems is shown on the Figure 2.7. Fuzzy operator ***and*** is min. One can notice that operator ***or*** is different for every type of system.

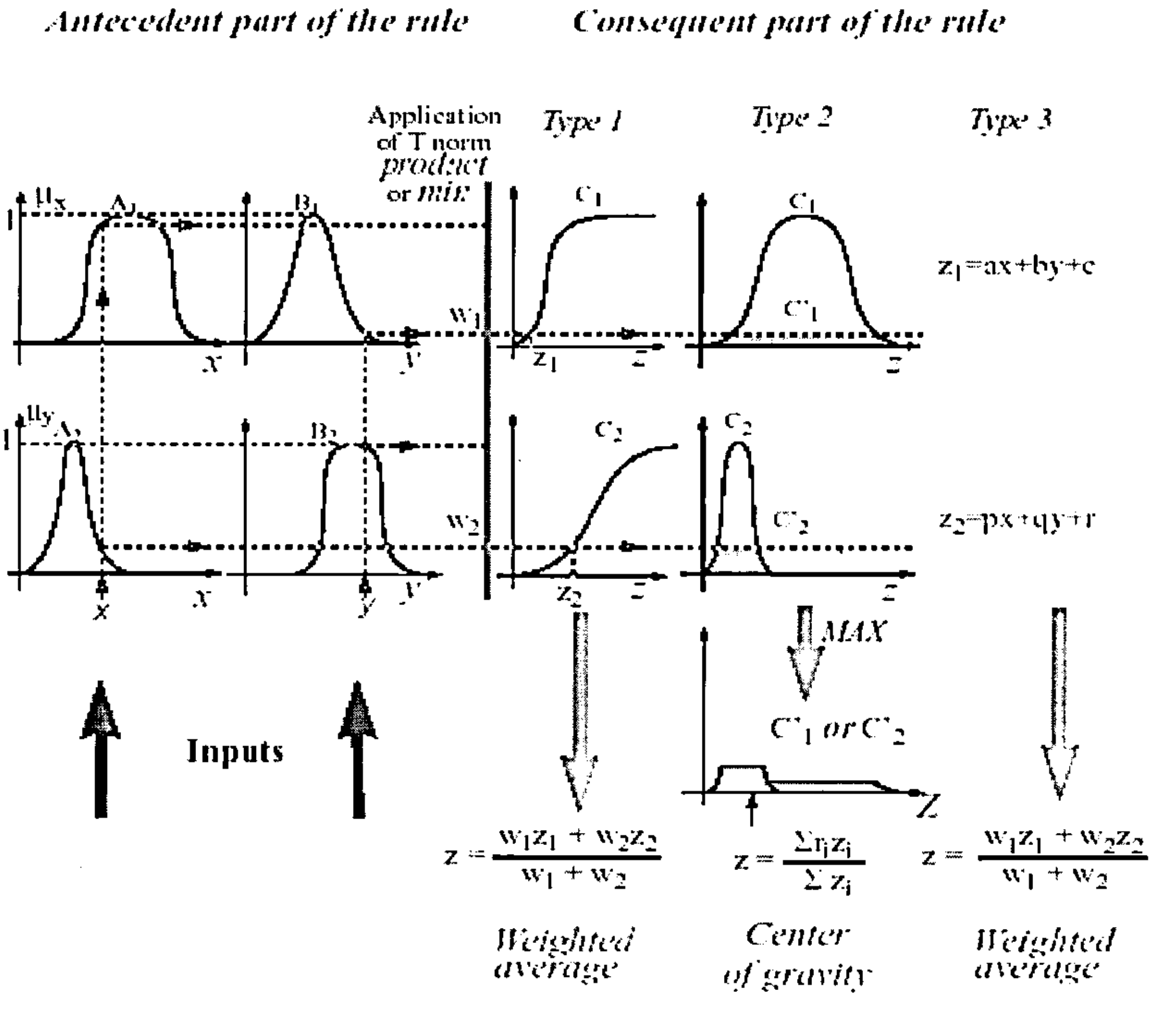

Antecedent part of the rule
Consequent part of the rule
Application of T norm product or min
Type 1
Type 2
Type 3
$z_1=ax+by+c$
$z_2=px+qy+r$
MAX
C'_1 or C'_2
Inputs
$z=\frac{w_1z_1+w_2z_2}{w_1+w_2}$
$z=\frac{\sum r_iz_i}{\sum z_i}$
$z=\frac{w_1z_1+w_2z_2}{w_1+w_2}$
Weighted average
Center of gravity
Weighted average

Figure 2.7 Three Types of Fuzzy Inference System

2.3 Neural Networks

Artificial neural networks ANN (simply, Neural Network NN) can be loosely defined as large sets of interconnected simple processing units, which have a parallel execution method to perform a common global task. These units usually undergo a learning process, which automatically updates network parameters in response to a possibly evolving input environment. The units are often highly simplified models of the biological neurons found in the brain. Thus, a neural network is a network of many very simple processors (units), each possibly having a small amount of local memory. The units are connected by unidirectional communication channels ("connections"), which carry numeric (as opposed to symbolic) data. The units operate only on their local data and on the inputs they receive via the connections.

Neural Networks can -in principle- compute any nonlinear function, i.e. they can do everything a normal digital computer can do. Especially anything that can be represented as a mapping between vector spaces that can be approximated to arbitrary precision by feed-forward NNs (which is the most often used type). In practice, NNs are especially useful for mapping problems, which are tolerant of some errors, have lots of example data available, but to which hard and fast rules cannot easily be applied.

2.3.1 Biological Neural Network

A Neural Network tries to mimic the animal brain. Biological neural networks are much more complicated in their elementary structures than the mathematical models used for ANNs.

The cerebral cortex is the largest part of the brain and is composed of numerous interconnected neurons arranged in very thin sheets that are highly convoluted. There are approximately 10^{11} neurons in the human brain, and each is connected to a thousand to up to 10,000 other neurons. The artificial neural networks have, but a very small fraction of the number of neurons in the brain. Likewise, the interconnectivity of current artificial systems is extremely far from the massive interconnections in the brain.

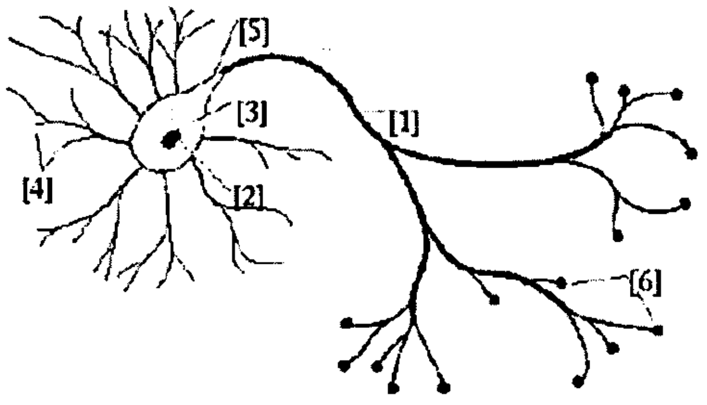

[5]
[3]
[1]
[4]
[2]
[6]
1.Axon 2. Nucleus 3.Soma (Body) 4. Dendrite 5. Axon Hillock 6. Terminals (Synapses)

Figure 2.8 the Biological Neural Network

The brain is mainly composed of massively interconnected cells called *neurons* as shown in Figure 2.8. A typical neuron has a cell body called the *soma*, and root-like structures through which input signals are received, called *dendrites*. These dendrites connected to other neurons through what is called a *synapse*. Signals collected through the numerous dendrites are accumulated in the soma. If the accumulated signal exceeds the neuron's threshold, then the cell is activated. This results in an electrical spike that is sent down the output channel called the *axon*. Otherwise, the cell remains inactive. The efficiency by which signals from one neuron is sent to another neuron via a particular synapse is called the *synaptic efficiency*. Synapses are truly channels through which electrical impulses cross. Chemicals known as neuro-transmitters regulate these electrical impulses.

The similarities between biological and artificial neural networks are obvious, as far as the huge differences are taken in consideration. An artificial neural network is an information processing system that has certain performance characteristics in common with biological neural networks. Neural networks, sometimes referred to as connectionist models, are parallel-distributed models that have several distinguishing features.

2.3.2 Model of Artificial Neurons

The main structure of a neural network is set of processing units, an activation state for each unit, which is equivalent to the output of the unit, connections between the units, a propagation rule, which determines the effective input of the unit from its external inputs, an activation function, which determines the new level of activation based on the effective input and the current activation and an external input (bias, offset) for each unit.

A processing unit Figure 2.9, also called a neuron or node, performs a relatively simple job; it receives inputs from neighbors or external sources and uses them to compute an output signal that is propagated to other units.

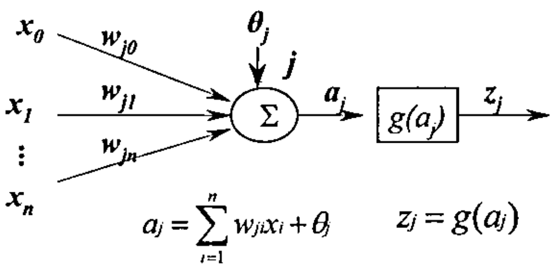


Figure 2.9 Processing Unit

Within the neural systems there are three types of units: input units, which receive data from outside of the network, output units, which send data out of the network and hidden units, whose input and output signals remain within the network. Each unit j can have one or more inputs $x_0, x_1, x_2, \ldots x_n$, but only one output z_j. An input to a unit is either the data from outside of the network, or the output of another unit, or its own output.

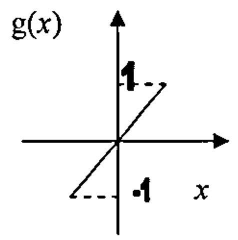
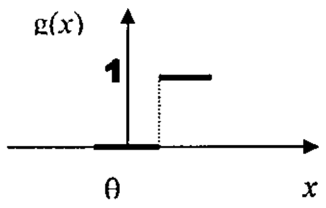
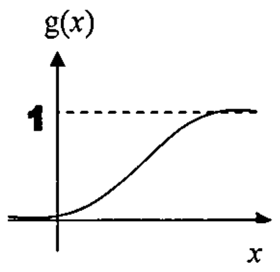
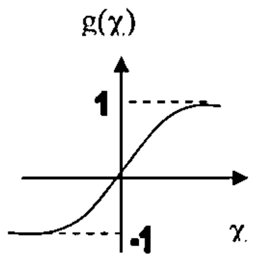
Each non-input unit in a neural network combines values that are fed into it via synaptic connections from other units, producing a single value called net input. The function that combines values is called the *combination function*, which is defined by a certain propagation rule. In most neural networks we assume that each unit provides an additive contribution to the input of the unit with which it is connected.

Lots of combination functions usually use a "bias" or "threshold" term in computing the net input to the unit. For a linear output unit, a bias term is equivalent to an intercept in a regression model. It is needed in much the same way, as the constant polynomial ‘1’ is required for approximation by polynomials.

Most units in neural network transform their net inputs by using a scalar-to-scalar function called an *activation function*, yielding a value called the unit's activation. Except possibly for output units, the activation value is fed to one or more other units. Activation functions with a bounded range are often called squashing functions. Some of the most commonly used activation functions are given in Table 2.2

In general, neural network models can be classified in a number of ways. Using the network architecture as basis, there are three major types of neural networks *feed forward networks*, *recurrent network* and *competitive networks*. According to the network architecture they can be classified into two broad categories; *static architecture* and *dynamic architecture*. Another basis for classifying neural network models is according to the mode of learning adapted. In this case, there are two major categories; *supervised learning and unsupervised learning*.

Table 2.2 The Activation Functions

Activation Function	Mathematical Representation	Graphical Representation
Identity	$g(x) = x$	
Binary step	$g(x) = \begin{cases} 1 & \text{if}(x \geq \theta\ ) \\ 0 & \text{if}(x < \theta\ ) \end{cases}$	
Sigmoid	$g(x) = \dfrac{1}{1+e^{-x}}$	
Bipolar sigmoid	$g(x) = \dfrac{1-e^{-x}}{1+e^{-x}}$	

2.3.3 Supervised and Unsupervised Network Learning

The functionality of a neural network is determined by the combination of the topology (number of layers, number of units per layer, and the interconnection pattern between the layers) and the weights of the connections within the network. The topology is usually held fixed, and a certain training algorithm determines the weights. The process of adjusting the weights to make the network learn the relationship between the inputs and targets is called *learning*, or *training*. Many learning algorithms have been invented to help find an optimum set of weights those results in the solution of the problems. They can roughly be divided into two main groups:

1. Supervised Learning - The network is trained by providing it with inputs and desired outputs (target values). These input-output pairs are provided by an external teacher, or by the system containing the network. The difference between the real outputs and the desired outputs is used by the algorithm to adapt the weights in the network Figure 2.10. It is often posed as a function approximation problem - given training data consisting of pairs of input patterns x, and corresponding target t, the goal is to find a function $f(x)$ that matches the desired response for each training input.

2. Unsupervised Learning - With unsupervised learning, there is no feedback from the environment to indicate if the outputs of the network are correct. The network must discover features, regulations, correlations, or categories in the input data automatically. In fact, for most varieties of unsupervised learning, the targets are the same as inputs. In other words, unsupervised learning usually performs the same task as an auto-associative network, compressing the information from the inputs.

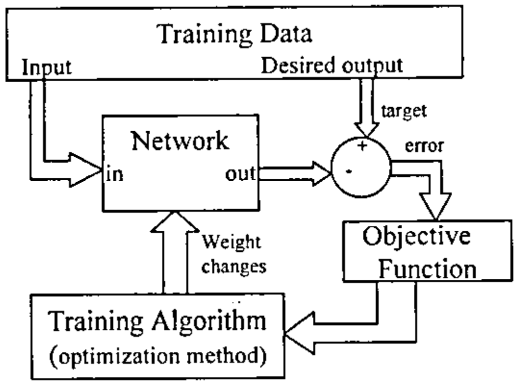


Figure 2.10 Supervised learning model

2.3.4 Feed-Forward Neural Networks

A layered feed-forward network consists of a certain number of layers, and each layer contains a certain number of units. There is an input layer, an output layer, and one or more hidden layers between the input and the output layer. Each unit receives its inputs directly from the previous layer (except for input units) and sends its output directly to units in the next layer (except for output units). Unlike the *Recurrent network,* which contains feedback information, there are no connections from any of the units to the inputs of the previous layers nor to other units in the same layer, nor to units more than one layer ahead. Every unit only acts as an input to the immediate next layer. Obviously, this class of networks is easier to analyze theoretically than other general topologies because their outputs can be represented with explicit functions of the inputs and the weights.

The network of Figure 2.11 is a network with one hidden layer. We can extend it to have two or more hidden layers easily as long as we make the above transformation further.

One thing we need to note is that the input units are very special units. They are hypothetical units that produce outputs equal to their supposed inputs. No processing is done by these input units.

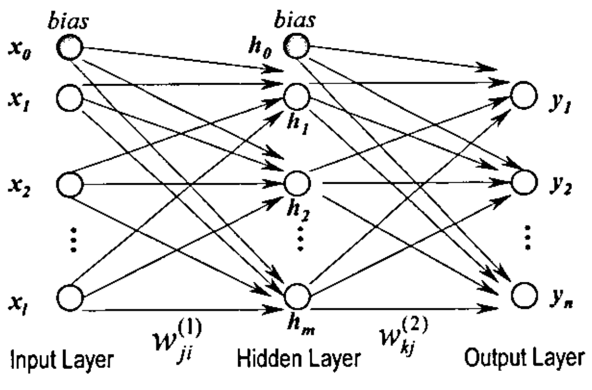


Figure 2.11 Feed-forward neural network

For most networks, the learning process is based on a suitable error function, which is then minimized with respect to the weights and bias. If a network has differential activation functions, then the activations of the output units become differentiable functions of input variables, the weights and bias. If we also define a differentiable error function of the network outputs such as the sum-of-square error function, then the error function itself is a differentiable function of the weights. Therefore, we can evaluate the derivative of the error with respect to weights, and either using the popular gradient descent or other optimization methods can then use these derivatives to find the weights that minimize the error function. The algorithm for evaluating the derivative of the error function is known as *back-propagation*, since it propagates the errors backward through the network.

To clarify the back-propagation training, consider a general feed-forward network with arbitrary differentiable non-linear activation functions and a differential error function. we know that each unit j is obtained by first forming a weighted sum of its inputs of the form,

$$a_j = \sum_i w_{ji} z_i \qquad (2.2)$$

where z_i is the activation of a unit, or an input. We then apply the activation function

$$z_j = g(a_j) \qquad (2.3)$$

Thus the task becomes to find the δ_j for each hidden and output unit in the network.

For the output unit, δ_k is very straightforward,

$$\delta_k = \frac{\partial E_p}{\partial a_k} = \frac{\partial E_p}{\partial y_k} g'(a_k) \qquad (2.4)$$

For a hidden unit, δ_k is obtained indirectly. Hidden units can influence the error only through their effects on the unit k to which they send output connections so

$$\delta_j = \frac{\partial E_p}{\partial a_j} = \sum_k \frac{\partial E_p}{\partial a_k} \frac{\partial a_k}{\partial a_j} \qquad (2.5)$$

The first factor is just the δ_k of unit k so

$$\delta_j = \frac{\partial E_p}{\partial a_j} = \sum_k \delta_k \frac{\partial a_k}{\partial a_j} \qquad (2.6)$$

For the second factor we know that if unit j connects directly to unit k then $\partial a_k / \partial a_j = g'(a_j) w_{kj}$, otherwise it is zero. So we can get the following *back-propagation* formula,

$$\delta_j = g'(a_j) \sum_k w_{kj} \delta_k \qquad (2.7)$$

which means that the values of δ for a particular hidden unit can be obtained by propagating the δ's backwards from units later in the network, as shown in Figure 2.12 Recursively applying the equation gets the δ's for all of the hidden units in a feed-forward network, no matter how many layers it has.

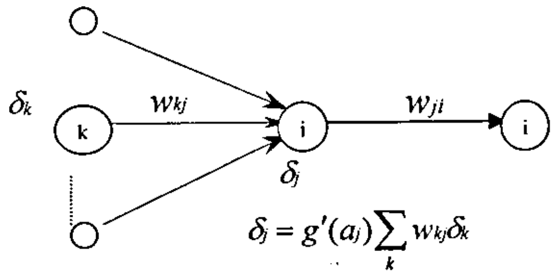


Figure 2.12 Backward propagation

One of the most obvious similarities between fuzzy systems and NN is that they both can handle extreme nonlinearities in the system. The shape of the membership function in fuzzy systems and the threshold function in NN are the same. Multiply-add operation of artificial neurons is very close to MAX-MIN operation of approximate reasoning. The MIN operation of input fuzzy variables conducted at each proposition of IF parts of fuzzy inference rule correspond to a product of input to the neuron and synaptic weights. The MAX operation to obtain the final inference value from THEN part of these plural inference rules corresponds to the input sum within a neuron.

These reasons lead to the idea of merging these two approaches. There are two possible methods: a method by which individual merits are combined, and another by which analogies between these are overlapped. One possibility is to endow learning functions to fuzzy logic systems, or to conduct pattern processing before fuzzy logic is applied. The other is to incorporate fuzzy logic and linguistic rules into neural nets structure.

The most widely researched hybrid neuro-fuzzy systems are based on the learning capabilities of ANNs in which they are exploited within the framework of fuzzy logic. In some systems, ANNs can be used to generate and tune the membership functions in a fuzzy system Figure 2.13. A number of models have been suggested for such hybrid systems, e.g. fuzzy ART (Carpenter, et al., 1992), Fuzzy LVQ (Tsao et al., 1992) and radial basis functions (Wang, 1992). The process of obtaining and tuning the fuzzy rules is one that is particularly suitable for ANNs, resulting in substantial reductions in cost and development time. Here, gradient descend methods have been used to define the shape and position of membership functions. This hybrid method has been used to design triangular, Gaussian, sigmoidal and bell-shaped membership functions (Asakawa and Takagi, 1994).

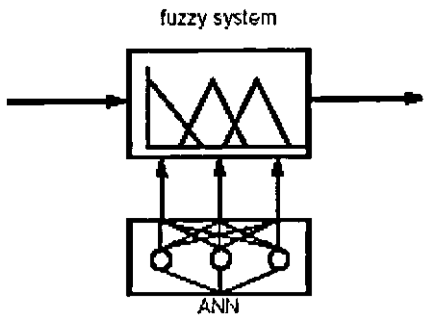


Figure 2.13 ANN for Tuning Membership Functions.

A number of applications have been reported in which fuzzy systems and ANNs are employed in series (Medsker, 1995). In such situations, either the sensor output is not suitable for direct input to the fuzzy system in which case an ANN preprocesses the input to the fuzzy system Figure 2.14(a), or the output of the fuzzy system is not suitable for direct interface with the external devices and an ANN is used as a postprocessor to perform a mapping or conversion not easily achievable by other analytical techniques see Figure 2.14(b). For example, Toshiba's microwave-oven-cum-toaster estimates the temperature and the number of pieces of bread using an ANN and decides the optimum toasting time by using fuzzy reasoning, i.e. its model resembles closely Figure 2.14(a).

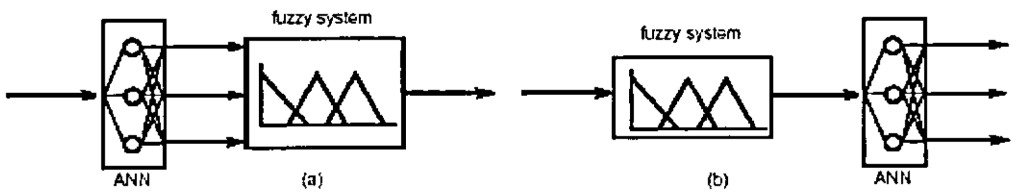


Figure 2.14 Fuzzy and NNT Application

In some applications ANN and fuzzy systems are used in parallel. One possible configuration uses a fuzzy system as the main system and an ANN to fine-tune the output to suit users' personal preferences. The ANN learns from the fine adjustments made by the user and corrects the output of the fuzzy system Figure 2.15.

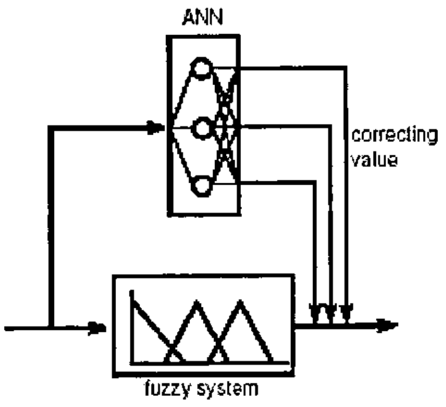


Figure 2.15 Parallel ANN and Fuzzy Systems

Another class of systems, known as neural fuzzy systems, have been used for knowledge acquisition and learning. In such systems, experts' knowledge in symbolic form is used to initialize a structured ANN. The ANN is then trained using the input/output from an actual system. Symbolic knowledge is then acquired from the trained ANN in fuzzy logic representation Fig 2.16.

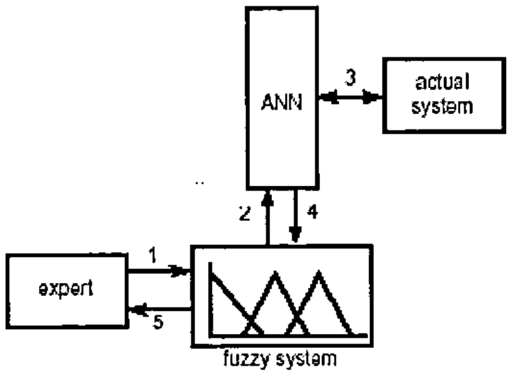


Figure 2.16 Neural/Fuzzy System

A significant number of researchers have concentrated their efforts on implementing fuzzy systems on a neural network representation. These include fuzzy weights, fuzzy neurons and fuzzy neural networks in which the neural network layers perform the fuzzification and defuzzification on crisp input/output data. The structure of fuzzy neural networks is shown in Figure 2.17. There are three groups of layers each performing one of the three functions of a fuzzy system.

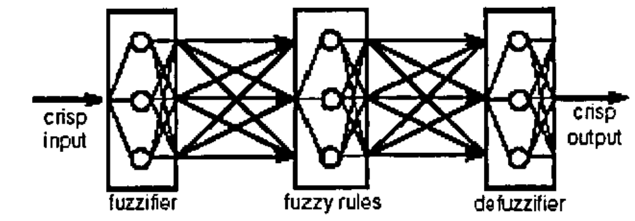


Figure 2.17 Fuzzy/ Neural Networks

The initial layers process crisp input data by assigning groups of nodes to the labels of linguistic variables and implementing membership functions in these nodes. The output of these layers goes to layers that function as fuzzy rules operating on fuzzy input. The final layer aggregates the results of applying the rules and defuzzifies the results (Medsker, 1995). GARIC is an example of such a scheme, which uses a five-layer neural network. It has been used in the space shuttle orbital operations control by NASA.

2.5 Concluding Remarks:

Fuzzy systems, including fuzzy logic and fuzzy set theory, provide a rich and meaningful addition to standard logic. The mathematics generated by these theories is consistent, and fuzzy logic may be a generalization of classic logic. The applications which may be generated from or adapted to fuzzy logic are wide-ranging, and provide the opportunity for modeling of conditions which are inherently imprecisely defined, despite the concerns of classical logicians. Many systems may be modeled, simulated, and even replicated with the help of fuzzy systems, not the least of which is human reasoning itself. On the other hand the neural network and Hybrid (Neuro-Fuzzy) controller were also discussed in this chapter.

In the next chapter the work of the research in the field of the autonomous mobile robot will be discussed. The next chapter consist two major parts, the first one is about the research that done in the mobile robot navigation system using fuzzy logic controller and the next section discussed the mobile robot navigation system using the Hybrid (Neuro-Fuzzy) controller.

LITERATURE REVIEW

3.1 Introduction:

Autonomous operation of a mobile robot in real environment posses a series of problems, so the researchers from the past works in this interesting field try to solve the robot problems and to improve the work of the autonomous mobile robot. In the past, several researches had been proposed to solve the robot navigation such as artificial potential field (APF), the edge detection (ED), obstacle boundary following (OBF), the goal oriented recursive path planning (GORP), vector field histogram (VFH), the fuzzy logic (FL), neural network (NN) and genetic algorithm (GA). In the last years, the mobile robot navigation using new techniques of fuzzy logic, neural network and genetic algorithm were investigated.

Mustafa (1999) began directing his attention toward providing aids such as neural networks to solve the problems encountered with the mobile robot navigation system. One of the initial goals of this researcher was to develop an easy to use interactive environment in which the robot to work. However development lacked this real environment because it had to be off-line application i.e. the starting point, target point and the environment have to be known. A second noteworthy application of genetic algorithm of mobile robot was developed, it was designed to solve the problems in real environment. The resulting technique provided ample opportunities for researcher to exchange ideas. The application was based gain on off-line techniques. (Samhuri M., 2000)

The papers presented in this literature review cope with a number of approaches for tackling the problem of robot navigation. As one of the principal fields of research in robotics is the development of techniques for the guidance of autonomous robots, this review highlights fuzzy and neuro-fuzzy approaches in designing and implementing controllers responsible for the mobile robot navigation.

Most of the current researches were in the mobile robot navigation field and especially in the obstacle avoidance and localization of the robot. So the pervious work can be divided to two categories

- Fuzzy Logic Approaches to Mobile Robot Navigation.
- Neuro-Fuzzy Approaches to Mobile Robot Navigation.

3.2 Fuzzy Logic Approaches to Mobile Robot Navigation

Saffiotti (1997) proposed to decompose the controller of the mobile robot to small units of control or behavior since this decomposing had to coordinate the execution of different units in order to obtain a globally coherent behavior. A major challenge of autonomous robotics was the large amount of uncertainty in the real world environment. On the other hand the exact and complete prior knowledge of these environments couldn't be known since many details were usually unknown, the position of the people and objects also couldn't be predicted a prior, passage ways maybe blocked and so on. On the other hand, knowledge acquired through sensing affected by uncertainty and imprecision. Number of robotics architectures had been proposed that try to cope with the above problems by making a clever use of the prior and sensor information. So the fuzzy logic controller was investigated in this paper to apply the behavior arbitration, the arbitration policy determines which behavior (s) should influence the operation of the robot at each moment, and thus ultimately determines the task actually performed by the robot. This policy had two advantages when it is applied to the mechanisms of fuzzy logic, which is the ability to express partial and concurrent activation of behaviors and the smooth transition between behaviors. As a result the fuzzy context rules provide a flexible means to encode behavior arbitration strategies. Like fuzzy control rules allow them to write the complex function in a modular way, using a logical format. On the other hand the fuzzy context rules also provide a convenient representation for plans, an important requirement for a plan representation that it be shared between planner and the controller.

Thongchai et al., (1999) describe,"How fuzzy control can be applied to a sonar-based mobile robot". Behavior-based fuzzy control for HelpMate behaviors was designed using sonar sensors. The fuzzy controller provides a mechanism for combining sensor data from all sonar sensors which present different information. The behavior-based approach is implemented as an individual high priority behavior. The highest-level behavior is called the task-oriented behavior, which consists of two subtasks, wall following and goal seeking. The middle level behavior is obstacle avoidance. The lowest level is an emergency behavior. Each behavior was built as an atomic agent based on the intelligent machine architecture (IMA). The HelpMate robot can follow the wall, go to the goal, and avoid obstacles detected by the sonar sensors. *HelpMate* is a mobile robot which has sonar sensors, lidar sensor, stereo color vision, etc... The data from sonar sensors is the distance between the sonar and object. The sonar sensor arrangement for HelpMate is shown. The sonar sensors' locations are important for designing the fuzzy controller.

The application of *fuzzy control* to sonar-based obstacle avoidance for Help Mate has been implemented. All sonar sensors send data to the inputs of fuzzy controllers. Each fuzzy controller for obstacle avoidance has nine inputs and two outputs. All fuzzy controllers are implemented under the behavior-based approach. Behavior-based approaches have been established as a main alternative to conventional robot control. A *Fuzzy Inference Engine* is used to combine the fuzzy IF-THEN in the *fuzzy rule-base,* and to convert input information into output member-ship functions. An inference mechanism emulates the expert's decision-making in interpreting and applying knowledge about how to perform good control. Emergency Behavior, Obstacle Avoidance Behavior and task Oriented Behaviors are discussed.

The behavior-based fuzzy control for the mobile robot was implemented. The results show that this complicated system can be solved. An emergency behavior has the highest priority to stop the robot if the obstacles are located closer than the emergency distance. The larger the obstacles, the better the sonar data are. The obstacle avoidance behavior also works better with larger obstacles. "The HelpMate robot can avoid all obstacles that are detected by sonar sensors. The task behaviors show that HelpMate can move to the goal as the user assigns and moves to follow the wall by using the wall following behavior. The distance between the robot and wall can be controlled by using the fuzzy controller. However, in some situations the robot may stop if the objects are moved very quickly close to the robot".

Thongchai (1999) describes a method to build a learning system for mobile robots based on fuzzy control technique, where range sensors are analyzed and used. In this work, a learning system for a mobile robot is designed based on fuzzy control technique. Laser and compass readings are used as the fuzzy inputs. The learning system generates a set of fuzzy rules for building behaviors of mobile robots. The learning system generates a set of fuzzy rules based on the robot's behaviors. Two examples illustrate the learning capabilities: 1) learn-to-avoid-obstacle and 2) learn-to-detect-landmark behaviors. Reinforcement learning is applied to correct the fuzzy rules. Fuzzy controllers are used to control the robot using a set of fuzzy rules. The results of learn-to-avoid behavior indicate a good learning performance in dynamic environments. The learning technique is one way that most autonomous mobile robots are used. The Fuzzy control consists of fuzzification, fuzzy rules, and defuzzification processes. The paper introduces the rules and the learning system, which serves to learn to avoid obstacles. The Fuzzy controller has a set of fuzzy rules given to the robot after learning. Furthermore, the "Learn-To-Detect-Landmark" was adopted to avoid obstacles in a dynamic environment. Additionally, two sets of fuzzy rules can be combined and used to control the robot; therefore, the number of times that the robot gets stuck should occur less often when using these two results.

The results show that the robot has a better learning rate in the dynamic environment, by using a set of fuzzy rules from learn-to-avoid-obstacle behavior in dynamic environment. It gives better results when comparing learn-to-avoid-obstacle behavior in a static environment.

Aguirre and González (2000) in their paper, a hybrid deliberative-reactive architecture for mobile robot navigation for integrating planning and reactive control and attention is focused on the design, coordination and fusion of the elementary behaviors. The complex behavior is therefore generated by combining simpler behaviors. Elementary behaviors are built up as fuzzy knowledge bases. Where they present a methodology for designing fuzzy behaviors based on regulatory control and propose several measures in order to evaluate the robot motion. Two different fuzzy methods of combining these simpler behaviors were explained. The idea of the first combination method is to combine the preferences from each behavior before defuzzification by means of the intersection of the outputs. Regarding the second method, the final output is obtained by a linear combination of each defuzzified fuzzy output. The study of the robot motion is based on two main measures: the achievement of the control objective and the total bending energy (TBE) of the trajectory. The objective of the first measure is to test the quality of control when the robot has reached the control objective and also during the whole run. The total bending energy, on the other hand, evaluates both the smoothness and the length of the trajectory.

The architecture is demonstrated by performing navigation tasks, first in simulated environments and later in the real world in an office-like environment with a Nomad 200 mobile robot. The results of the experiments show that the robot achieves every objective of control from every behavior and the trajectory is a smooth trajectory in spite of the interaction between several behaviors. As regards the combination methods, their experimentation shows that the influence of the combination method may be an important factor because in some cases there can be significant differences in the trajectory of the robot according to the combination method used. These measures show that the method which combines the outputs before defuzzification has better results than the method which combines the defuzzified outputs.

Proychev and Petrov (1996) proposed a fuzzy logic controller to solve the navigation tasks of the mobile robot in unknown environment. This controller helps the robot to avoid collision with obstacles appeared in its desired 2D trajectory, where the fuzzy navigation was connected directly with a set of infrared sensors and it was designed to drive adaptively the robot in the environment. On the other side the fuzzy logic controller that was used, worked as an obstacle-avoiding controller using preliminary designed fuzzy rules that ensure avoiding the robot from collision with the unknown obstacles appeared during the motion and with the known stable obstacles. This fuzzy controller receives the input signals from the infrared sensor, which was switched from passive state (0) to active state (1), when the distance to the obstacle in its direction is less than a given one (two or three of the robot diameter). The output of the fuzzy controller is steering angle θf, which had a distributed uniform in the range (-π/2, π/2) through a seven fuzzy sets. As a result the navigation algorithm proposed ensures the shortest way from the given initial or current robot posture to the desired posture in unknown environment. The fuzzy controller was applied to the simulation software and the algorithm was interesting the well work in a real environment.

In their paper Xu and Tso (1996) proposed automatic reaction of mobile robot. Two types of reaction strategies were workout. The paper tried to achieve autonomy of the mobile robot by introducing a system that improves the sensing of its environment and interpreting the sense information to perform obstacle avoidance and reach the target. The paper also investigated the action of the mobile robot in real time and in an unstructured environment. The system through fuzzy reasoning and using the heuristic rules synthesis, the obstacle position, the target orientation and the robot's direction of movement are synthesized. The navigation control of the robot is then realized through coordinating rules based on fuzzy reasoning. Some simulation results of mobile robot able to avoid arbitrary placed obstacles through self- reaction are also given for illustration.

The paper also introduces reaction strategies for motion in unstructured environments, fuzzification of the problem and knowledge rules for reaction plus fuzzy reasoning and simulation results. The paper concluded that fuzzy reasoning rules for avoiding obstacles are derived from the experience of human reaction to the environment. Proper reactions are realized by taking account of both the target information and the obstacle information in any situation. The obstacles are detected by three groups of ultrasonic sensors, and the target is detected by an optical rangefinder. The target location plays an important role in deciding the turning of the robot when the obstacles are close to the robot. In this paper number of rules peculiar to the mobile robot are developed and coordinated by fuzzy inference, and aggregation is elaborated. The effectiveness of the approach proposed is demonstrated by means of simulation experiments, where a robot moves in different types of environment with obstacles of various shapes presents.

Other researchers Xu et al. (1998) discussed fuzzy reactive control of automatic navigation of an intelligent mobile robot in an unknown environment. The main idea of the implemented system depended on a switching strategy, which acts as preprocessors of the controller. The study employs a Nomand 200 mobile robot, which had a sonar system that used to detect the long ranging. The paper also discussed the fuzzy reactive control interns of robot reactive strategy, fuzzy reactive control, rule base for robot fuzzy reaction and fuzzy reasoning. An important aspect of the paper was real/ virtual target switching which involves phenomenon and problem of local trapping and wandering. As a fundamental of component reasoning for building a real/ virtual target switching strategy in which, the real target is locally shifted to a virtual location temporarily according to the current sensed environment and the pervious target orientation. The local trapping can be easily avoided using this real / virtual target switching means. They conclude that reactive control scheme, involving the sensing and reaction strategies, is proposed for the real time navigation of mobile robot. Application of fuzzy logic to this control problem enables the robot, while moving toward the target, to react effectively to unknown arbitrary environments, handle the coarse range information of sonar sensors and coordinate the different reactive behavior types. The fuzzy reactive control scheme is technically more attractive in view of short reaction time and fast decision-making process.

3.3 Neuro-Fuzzy Approaches to Mobile Robot Navigation

Song and Sheen (1997) developed and successfully implemented a pattern recognition approach to reactive navigation based on real time sensory information. A heuristic fuzzy neuro network is developed for pattern – mapping between quantized ultrasonic sensory data and the velocity commands to the robot. The design goal was to enable an autonomous mobile robot to navigate safely and efficiently to target position in a previously unknown environment. To build the desired mapping between perception and motion, usefully heuristic rules were combined with the Fuzzy Kohonen Clustering Network (FKCN).
Through the process of prototype- pattern assignment, the proposed fuzzy neural network can be adopted for reactive motion control. By employing a small number of rules, satisfactory performance had been achieved. The amount of computation is therefore reduced a great deal and enhances the real time performance of reactive control. Furthermore, this method offers a straightforward mapping between control rules and human like heuristic. The mobile robot can therefore demonstrate human like tendencies for continuous motion without stopping for obstacles.
The advantage of this method that it provided much faster response to unexpected events and is less sensitive to sensor misreading than conventional approach. It allows continuous, fast motion of the mobile robot without any need to stop for obstacles. The effectiveness of the proposed method is demonstrated in a simulation test of the mobile robot.

In Yung and Ye (1999) paper, an alternative training approach to the EEM based training method is presented and fuzzy reactive navigation architecture is described. Using the rule base learned from the new method, the proposed fuzzy reactive navigator fuses the obstacle avoidance behavior and goal seeking behavior to determine its control actions The new training method is 270 times faster in learning speed; and is only 4% of the learning cost of the EEM method, where adaptability is achieved with the aid of an environment evaluator.

The system tries to determine a *collision free path* that enables the vehicle to travel through an obstacle course from an initial configuration to a goal configuration. It describes the process of finding such path, which is also known as the *path-planning problem* and classifies it into: global path planning and local path planning. *Global path planning methods*, which are usually conducted off-line in a completely known environment, was discussed. Some *heuristic approaches,* which rely on either calculating the *potential fields* or performing a search through a *state space mode*. On the other hand, the *local path planning* techniques, also known as the obstacle avoidance methods, are suggested as potentially more efficient in vehicle navigation when the environment is unknown or only partially known. This approach utilizes the on-line information provided by sensors such as the ultrasonic sensor, laser range finder, radar, and vision sensor, to tackle the uncertainty. An efficient local path planning method is the potential field method. The practicality of the potential field method therefore hinges on how well these problems can be resolved. In this paper, an alternative training approach to the EEM-based training method is presented and fuzzy reactive navigation architecture is described. The new training method employs a simple environment for constructing the fuzzy rule base and has five distinct advantages over the existing EEM. Using the rule base learned from the new method, the fuzzy reactive navigator fuses the sensor information from the sensor groups, the obstacle avoidance behavior and goal seeking behavior to determine its control actions, where adaptability is achieved with the aid of an environment evaluator. The paper presents the suggested navigator, the vehicle model and sensor arrangement, and the coordinate systems and navigation task. It also describes the architecture of the navigator and the obstacle avoider, which uses a fuzzy control for obstacle avoidance. The paper presented a fuzzy navigator that performs well in complex and unknown environments, using a rule base that is learned from a simple corridor-like environment. The principle of the navigator is built on the fusion of the obstacle avoidance and goal seeking behaviors aided by an environment evaluator to tune the universe of discourse of the input sensor readings and enhance its adaptability. For this reason, the navigator has been able to learn extremely quickly in a simple environment, and then operate in an unknown environment, where exploration is not required at all. A comparison of the navigator using the rule base obtained from the new training method and the EEM, shows that the new navigator guarantees a solution when the EEM-based navigation fails and its solution is more acceptable.

Hagras and Sobh (1998) discussed the control of autonomous intelligent robotic agent operating in unstructured changing environments. Online learning as a useful method producing intelligent machines for inaccessible environments was introduced. In these environments it is required to perform online learning through interaction with the real environment and performing any adaptation within short time intervals. Under such conditions, robotic agents have to be adaptive. The paper introduces this issue and explains the difficulty robots are facing in unstructured environments and "how learning and computational intelligence can help the robots to adapt and give them the necessary intelligence they need to face the challenges they encounter in their environments". The paper underlined the first fundamental requirement for robotic agents, which is that the agents must be grounded in that they must be able to carry on their activities in the real world, in real time according to the above definition of robotic agents. It throws light on the adaptive behavior, which cannot be considered as a product of an agent considered in isolation from the world, but can only emerge from strong coupling of the agent and its environment. The methods used to develop robotic agents in the system involve also learning and adaptation in unstructured environments. In this paper, the hard unstructured environments were presented. The paper also introduced novel learning and adaptation techniques in robotics operating in unstructured environments.

Cao et al. (2000) describe a Neuro-fuzzy control method for the navigation of an AGV robot. An overall system design and development was presented. The Neuro-Fuzzy computation and its application for mobile robot navigation were discussed. The system that was to be controlled is an electrically propelled mobile vehicle named Bearcat II, which is a computer controlled intelligent system. As autonomous navigation requires a number of heterogeneous capabilities, including the ability to execute elementary goal-achieving actions, like reaching a given location; to reach in real time to unexpected events, like the sudden appearance of an obstacle; to determine the robot's position; and to adapt to changes in the environment, the paper introduces a Neuro-fuzzy control method for navigation of an Autonomous Guided Vehicle (AGV) robot. The paper introduces neural network and fuzzy logic control techniques as a mean to improve real-time control performance for mobile robot due to its high robustness and error-tolerance ability. The paper tries to handle a new approach of the perceptual environment feature identification and classification, which are based on the analysis of the classifying neural network and the Neuro-fuzzy algorithm. The paper distinguishes between Reactive Navigation and Planed Navigation in that, while a mission is assigned or a goal location is known, the robot does not plan its path but rather navigate itself by reacting to its immediate environment in real time. The paper stresses on how robust autonomous performance can be achieved by using minimal computational capabilities, as opposed to the enormous computational requirements of path planning techniques. The autonomous mobile robot, the Bearcat II., which has been used in the research paper, was constructed "of an aluminum frame designed to hold the controller, obstacle avoidance, vision sensing, vehicle power system, and drive components. Two independently driven DC motors were used for vehicle propulsion as well as for vehicle steering. Also, all the subsystem level components were chosen to be modular in design and independent in terms of configuration so as to increase adaptability and flexibility". The navigation system takes range data as input, processes it in order to find regions that can safely drive over, and generates commands for steering the vehicle based on the distributions of these untraversable regions. The paper gives a description of the design and development of the navigation system of the mobile robot. The system uses a visual sensor system for the control of the mobile robot, under neuro-fuzzy technology. The neuro-fuzzy Hybrid System, was discussed with the architecture and the implementation of fuzzy rules, as well as the training of the neuro-fuzzy system. The paper notices that the neuro-fuzzy systems offer the precision and learning capability of neural networks, and that they are easy to understand like fuzzy system. Furthermore, the modified and new rules can be extracted from a properly trained neuro-fuzzy system, to explain how the results are derived. The paper concludes that some new technologies can be developed on the Bearcat II test bed to improve the learning speed, adjust learning and momentum rates, etc.

Ng and Trivedi (1998) introduce a Neural integrated Fuzzy controller (NiF-T), which integrates the fuzzy logic representation of human knowledge with the learning capability of neural networks, is developed for nonlinear dynamic control problems. It covers integrated sensing, control, actuator modules, real-time performance, ability to successfully handle noisy sensor signals, reactive controller design which captures high-level, linguistically based human expertise in a set of fuzzy rules, training of neural networks directly with fuzzy rules instead of numerical sample data, as well as learning capabilities and general applicability. NiF-T architecture comprises "of three distinct parts: 1) Fuzzy logic Membership Functions (FMF), 2) a Rule Neural Network (RNN), and 3) and Output-Refinement Neural Network (ORNN). FMF are utilized to fuzzify sensory inputs. RNN interpolates the fuzzy rule set; after defuzzification, the output is used to train ORNN. The weights of the ORNN can be adjusted on-line to fine-tune the controller". Only five rules were used to train the wall following behavior, while nine were used for the hall centering. Also, a robot convoying behavior was realized with only nine rules. For all of the described behaviors—wall following, hall centering, and convoying, their RNN's were trained only for a few hundred iterations and so are their ORNN's trained for only less than one hundred iterations to learn their parent rule sets.

The authors use the navigation and convoying as representatives of a class of nonlinear control problems involving noisy sensory signals and real-time feedback requirements.

The NiF-T model has three main parts: Fuzzy logic Membership Functions (FMF), Rule Neural Network (RNN), and Output-Refinement Neural Network (ORNN). The training and testing procedures for both the RNN and ORNN were multilayer feedforward neural networks using *Back Propagation* (BP). RNN maps input vectors to output vectors. RNN's objective function was defined. The Mobile Robot Navigation the authors chose was indoor rather than outdoor, the controller developed in that system can be applied to the Intelligent Vehicle Highway System (IVHS) by changing the input sensory information, such as using cameras to detect the road boundary.

The paper outlines two behaviors, wall following and hall centering. The robot was supposed to hug wall at any specified distance and to center itself in a hallway for the wall-following and hall centering behaviors, respectively. The Control Scheme was defined so that the robot can be designated to follow at any specified distance from the right or left wall. The paper concludes that a Neural integrated Fuzzy controller (NiF-T), which integrates the fuzzy logic representation of human knowledge with the learning capability of neural networks, was developed for nonlinear dynamic control problems. With the help of fuzzy logic, numerical sample data were no longer needed for the network training. Instead, the reliable expert rules were used to train the networks. In addition, membership functions provide continuous input representations. They make the controller less sensitive to slight variations in the physical parameters and control goal. Likewise, with the help of neural networks.

3.4 Concluding Remark:

The main objective of the mobile robot navigation is to reach the target without collision with any stationary or dynamic obstacles. Achieving this goal, a multi-methods but artificial intelligent (AI) is used since AI is one of the most significant tool.

The Navigation problems of the mobile robot in unknown environment have been solved by using many techniques. Saffiotti(1997) decompose the controller of a mobile robot into a small units of control or behavior in order to a globally coherent behavior. The sonar system was used in robot to avoid collision with obstacles (Thongchai, 1999; Thongchai et al., 1999). Another sensing tool has been used by Proychev and Petrov (1996), which is infrared sensor to detect the unknown environment. So the sensor can detect the distance and the robot can avoid collision with the obstacle. The difficult problem of the mobile robot faces is the traps. These traps have been solved by using the switching theory of the real/virtual target which help the robot to go out from the traps. (Xu et al., 1998).

Neuro- Fuzzy have been investigated to solve the navigation problem of the mobile robot. Song and Sheen (1997) combined useful heuristic rules with the Fuzzy Kohonen Clustering

Network (FKCN) to enable the mobile robot to navigate safely and efficiently to target position in a previous unknown environment. On the other hand, an online learning for the unknown environment has been adopted to help the robot by using the necessary intelligence data to avoid any obstacle. (Hagras and Sobh, 1998; Cao et al., 2000)

The focus of this research, as will be addressed in chapter 4 and 5, has been on the utilization of a fuzzy inference system for mobile robot navigation. It based on simple FIS that consists of 40 rules, an enhanced system has been developed which consists of 625 rules. The performance of the two systems is going to be compared according to many criteria. In addition to that, neural networks have been used to replace the inference engine of the 625 FIS in order to reduce the large CPU time required by this system.

DEVELOPMENT OF PROPOSED NEURO-FUZZY NAVIGATOR SYSTEM

4.1 Introduction

The objective of this chapter is to develop the proposed neuro-fuzzy system for a mobile robot navigation. The approach taken here is based on improving a classical fuzzy controller proposed by Xu and Tso (1996). Accordingly, the approach of Xu and Tso is first investigated and its drawbacks are clarified. Then, the proposed improvements on the fuzzy controller are presented and the performance of the developed controller is examined. Finally, the development of the neuro-fuzzy system is presented.

In order to pave the way for the discussion in this chapter, the scope of work stated in chapter one is reiterated here. The problem addressed in this research is as follows: The mobile robot is required to move from a starting point towards a target point that are both known to the robot prior to its execution of the task. The robot should avoid any fixed (not moving or static) obstacles that may be found a long its path. The path itself and the locations of the obstacles are not known a priori and should be determined by the navigation system as the robot is moving. It is assumed here that the robot will not face any traps (or get into a situation where it is required to backtrack or turn around). Such a problem is not dealt with in this research and is left as a recommendation for future work.

The mobile robot is assumed to be equipped with three physical ultrasonic sensors and one virtual sensor. The physical sensors are used to detect obstacles in front of the robot, on the right side, and on the left side, respectively. The maximum distance that can be sensed by these sensors is assumed to be 6 meters. The virtual sensor is used to guide the robot towards the target. This sensor is especially needed when the target direction of movement is totally blocked by an obstacle. The virtual sensor will guide the robot back towards the target once the obstacle is avoided.

The four sensors provide the path planning system (in our case a fuzzy logic system) of the robot with three distances front (dc), right (dr), left (dl), and target orientation (theta), respectively. From these inputs, the fuzzy logic controller will workout a decision in which direction should the robot move in order to reach the target. The fuzzy logic controller will go through three stages, i.e., fuzzification, inference, and defuzzification as shown in Figure 4.1.

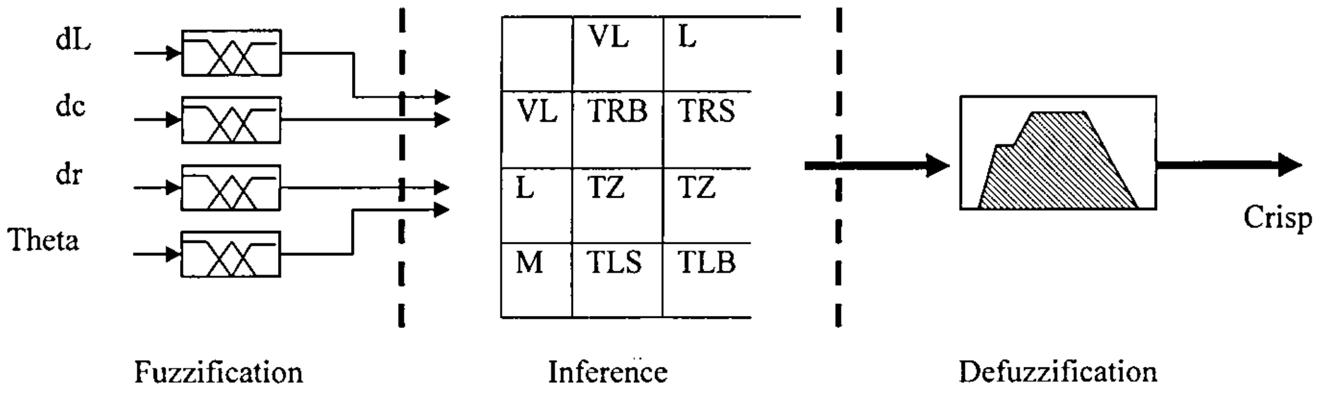


Figure 4.1: Fuzzy Logic Controller Stages

In the fuzzification stage, each input is related to a fuzzy set that has a number of sub-sets with different shapes. The crisp (sensor) input is translated into fuzzy linguistic labels (VL, L,…) with membership values ($\mu_{VL}(x)$, $\mu_L(x)$,..) of the input crisp value of that set.

In the inference stage, the first step is to construct a fuzzy rule base engine. The construction of this fuzzy rule requires the knowledge of the motions and actions that the mobile robot needs to execute subject to the sensor input data at some instant. The number of the rules to construct the rule-base depends on the number of inputs (n) and the number of fuzzy sets allocated to each input (m). Consequently, the number of rules required is m^n. As shown in Figure 4.1, the inference engine block receives the linguistic labels with their corresponding membership values. Each rule in the inference engine which has the same linguistic labels is activated. Finally, the Defuzzification stage receives the output of the inference engine and converts them into crisp values using one of the defuzzification methods described in chapter two.

In the following section, the Forty-rule Fuzzy system of Xu and Tso is investigated. Then, in section 4.3, our proposed improvement over the Xu and Tso system is dealt with. Finally, in section 4.4, the neuro-fuzzy system developed in this research is dealt with.

4.2 Description of Forty -Rule Fuzzy Navigator System:

Xu and Tsu (1996) discussed the reaction of a mobile robot computed in real time during its movement towards a target in an open field with obstacles present. Two types of reaction strategies were worked out based on human experience, and reaction rules (Fuzzy Rules) governing the system behavior. These strategies were based on synthesized reactions to the different situations such as obstacle position, target orientation and robot's direction of movement. Hereafter, the forty-rule fuzzy system proposed by Xu and Tso will be referred to as FLC40.

In their work, the robot had three ultrasonic sensors, one was in the center of the robot and the other two sensors are 60 degrees across the sides as shown in Figure 4.2. The distances measured by the three sensors are inputs to the fuzzy controller. These distances were:

(i) Distance from an obstacle to the left side of the robot (left distance (dl)).

(ii) Distance from an obstacle to the front of the robot (front distance (dc)).

(iii) Distance from an obstacle to the right side of the robot (right distance (dr)).

Another input to the fuzzy controller is the target orientation (tr), which is the angle between the robot direction of the movement and the line connected to the center of the robot with the target.

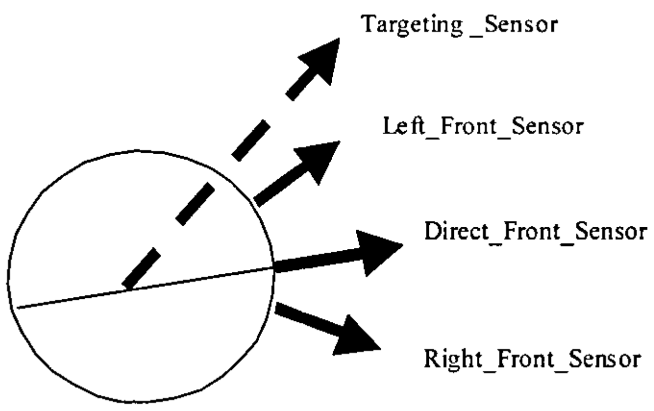


Figure 4.2: Mobile Robot with Frontal Sensors

4.2.1 Structure of the FLC40

In the FLC40 system, the input distances were represented as fuzzy sets with two linguistic labels (FAR, NEAR) within a universe of discourse of [0 to 6] meters for each input as shown in Figure 4.3.

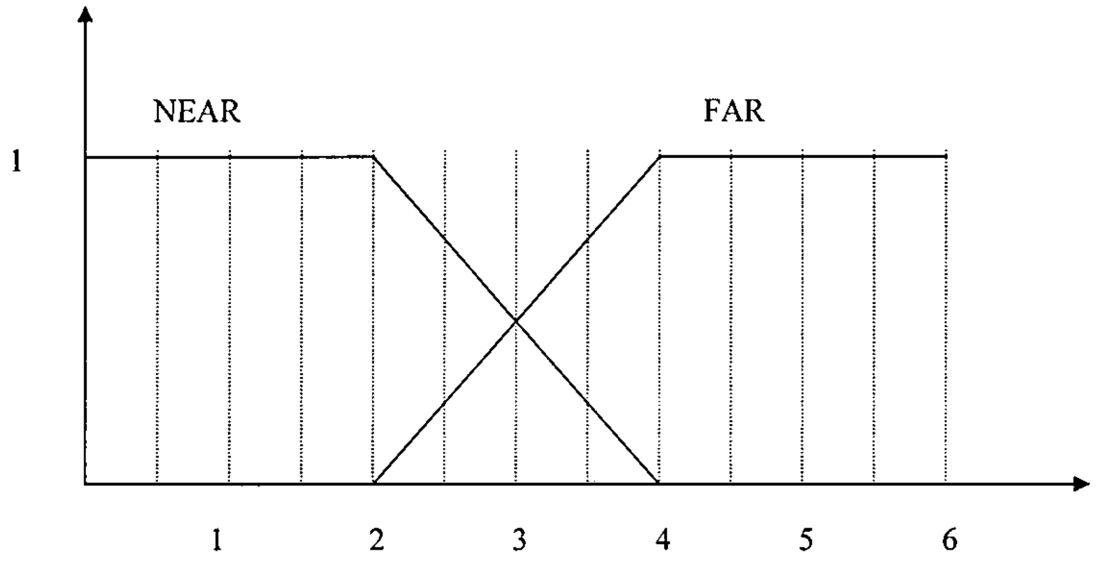
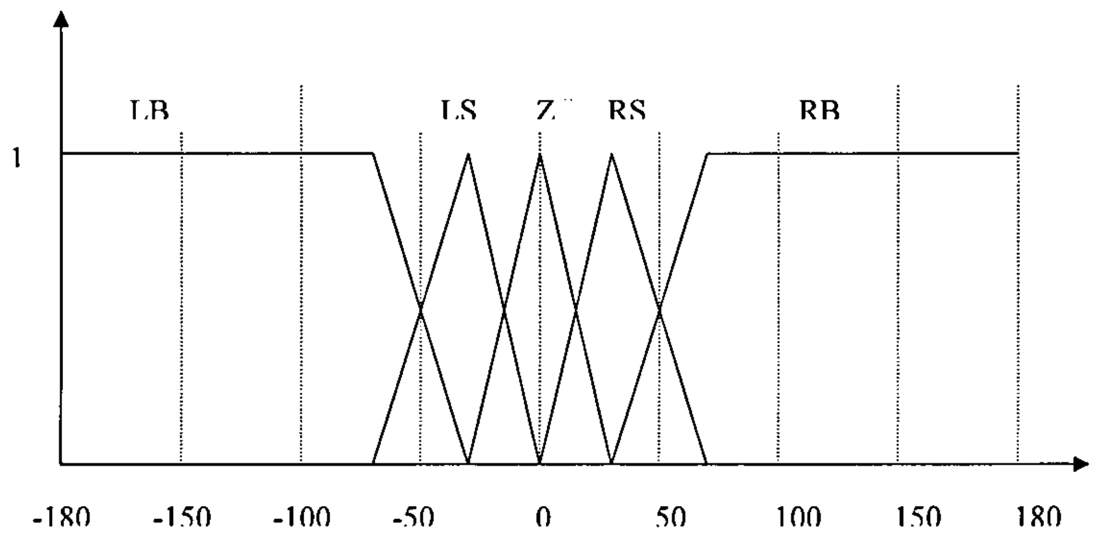


Figure 4.3: Membership functions of obstacle distances dr, dc and dl.

The target orientation (tr) was expressed by five linguistic labels (LB, LS, Z, RS, and RB) within a universe of discourse of (-180^{O} to 180^{O}) as shown in Figure 4.4.



LB: target_ Left Big   LS: target_ Left Small   Z: target_Zero

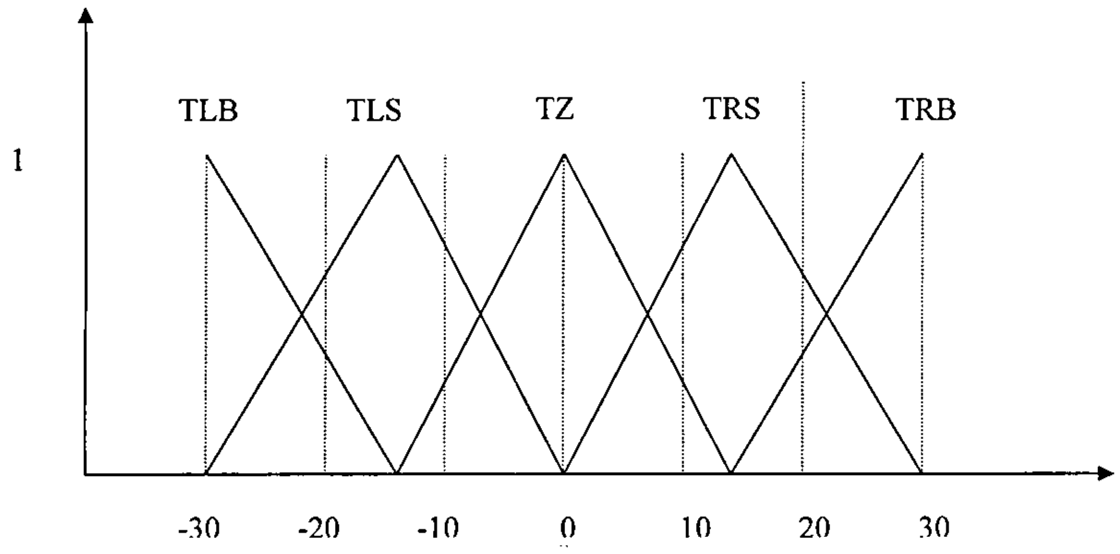
RS: target_ Right Small   RB: target_ Right Big

Figure 4.4: Membership function of the target orientation

The rule base of the inference engine was constructed with forty activations, since there were two membership functions for each distance and five membership functions for the orientation angle, thus 2×2×2×5 = 40 activation rules. The rules were of the form:

IF dr is FAR And dc is NEAR And dl is FAR And tr is LB THEN Sa is TLB

Where, **Sa** is the output action from the inference block. The output from this block was called the turning angle (Sa) of the robot. The output variable **Sa** was represented by five linguistic labels (TLB, TLS, TZ, TRS, TRB) within a universe of discourse of (-30 to 30) as shown in Figure 4.5.



TLB: Turn_ Left _ BigTLS: Turn_ Left _ Small TZ: Turn_ Zero

TRS: Turn_ Right _ Small TRB: Turn_ Right _ Big

Figure 4.5: Membership Function Turning Angle (tr).

The full details of the rules are presented in Appendix 1. A sample of the reaction rules is given below:

If dr is FAR and dc is FAR and dl is FAR and tr is LB THEN Sa is TLB

If dr is FAR and dc is FAR and dl is NEAR and tr is LB THEN Sa is TZ

If dr is FAR and dc is NEAR and dl is NEAR and tr is LB THEN Sa is TRS

Case 1

Two parallel vertical obstacles and the target close to the left side are considered as shown in Figure 4-7. Applying FLC40 to that case, the following observations appear.

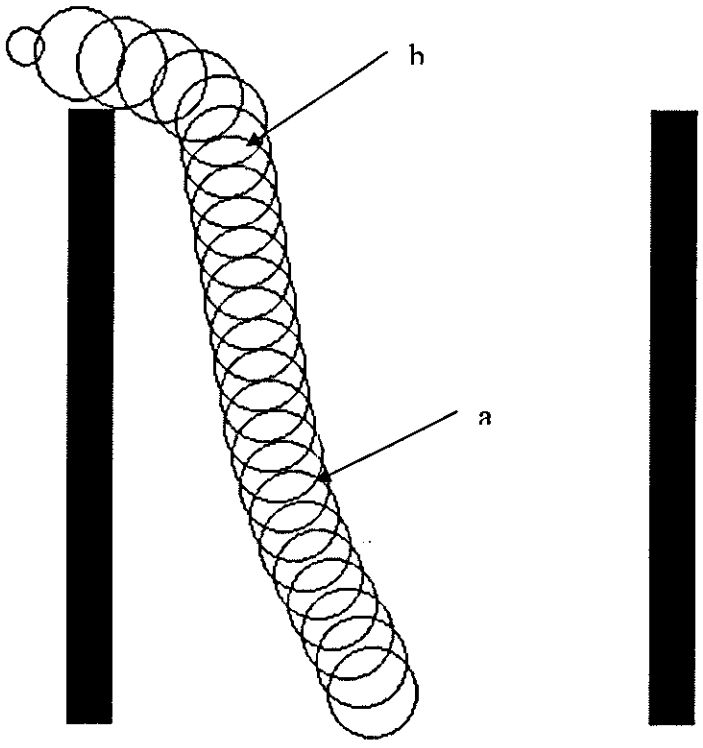


Figure 4- 7 FLC40, Case 1

- When the robot is in the home position, it takes into consideration two; things target position and avoiding obstacles. The robot in this case detects the position of the target and move towards it until it reaches point (a).

- In point (a) all the sensors of the robot detect the FAR distance (the obstacle is pass) so the activation rule is:

  IF dr is FAR and dc is FAR and dl is FAR and tr is LB THEN   Sa is TLB

Then, after that, the robot turns sharply to the left and moves in the same direction, but the left sensor detects that there is an obstacle with less than 0.5 meter far from the robot, and then activation rule becomes

IF dr is FAR and dc is FAR and dl is NEAR and tr is Z THEN Sa is TZ

Finally the robot goes in the same direction and touches the obstacle.

Case 2

The target, in this case, is above a horizontal obstacle as shown in Figure 4.8

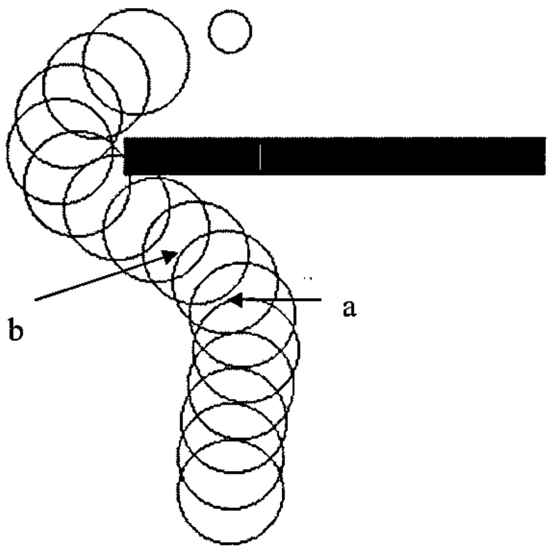


Figure 4-8 FLC40, Case 2

In this case the robot detects the target point is straightforward and the obstacle is FAR. So the robot moves towards the target as shown in Figure 4-8.

Appling the FLC40 to this case, the following results appear:

- When the robot sensors detect the obstacle in point (a) it turns to the left because the activation rules in this point are:

  IF dr is NEAR and dc is NEAR and dl is NEAR and tr is Z THEN Sa is TLB.

IF dr is FAR and dc is NEAR and dl is NEAR and tr is LS THEN Sa is TRS

As a result, the robot turns to left but this time the obstacle is too close so the robot touches the edge of the obstacle.

- Also in point (b) the left and front sensors detect the obstacle but the obstacle is close and the activation rule is:

  IF dr is FAR and dc is NEAR and dl is NEAR and tr is LS THEN Sa is TRS

  So the robot again touches the edge of the obstacle.

Case 4

In this case there are two vertical obstacles as shown in Figure 4-10 and the target is on the top left hand side.

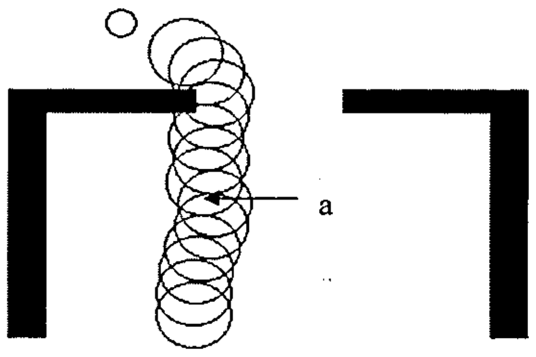


Figure 4-10 FLC40, Case 4

In this special case, the robot must move through the narrow gate between the two obstacles to meet the target. The target is on the left hand side, but the robot moves approximately in a straight line since there are two obstacles and the robot must avoid collision with them. When applying the FLC40, the following notes appear.

- In point (a) the left sensor can detect the obstacle. So the robot still works in the same direction since the activation rule is:

IF dr is FAR and dc is FAR and dl is NEAR and tr is LB THEN Sa is TZ

As a result the robot touches the obstacle.

The cases above show that the FLC40 fails in critical points. The reason behind that, due to the limited number of sets that are used in this controller, two fuzzy sets (FAR, NEAR) are not enough to control the robot for different situations with smoothing motion. The two fuzzy sets (FAR and NEAR) characterize the controller to a semi on/off system. So the detected distance is FAR or NEAR only and this can make the controller unstable in certain conditions.

4.3 Development of the Improved Fuzzy Navigation System

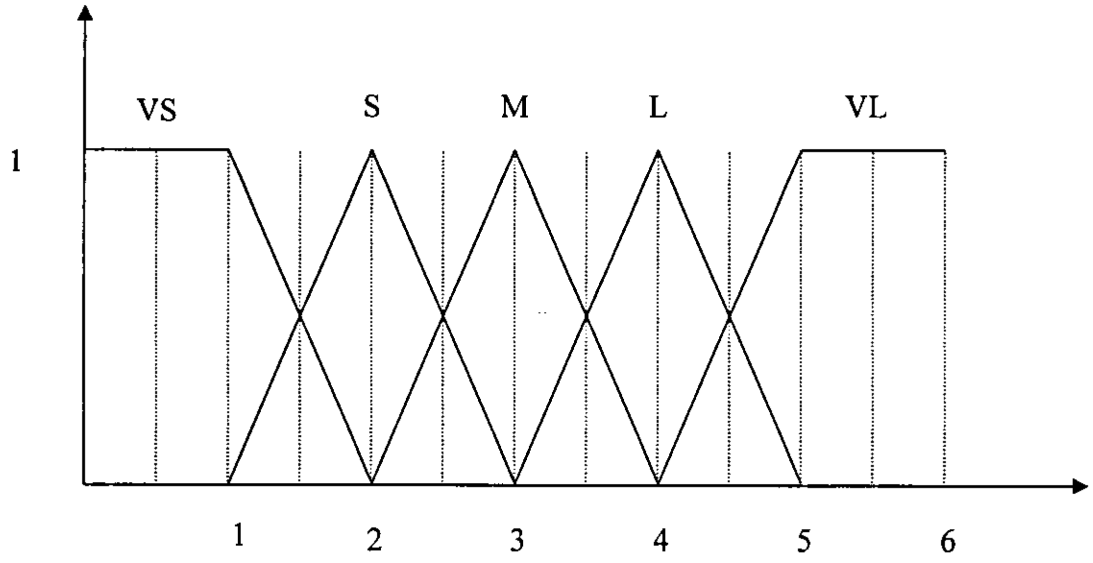
As has bee already noted, the FLC40 is not capable to avoid collision with the edges of the obstacles in all cases. The main reason behind that failure is the low resolution due to two fuzzy sets, i.e., FAR and NEAR. An improvement to the system can be easily made by increasing the number of fuzzy sets in order to achieve better resolution. In this research, it is proposed to increase the fuzzy sets to five linguistic labels (VL, L, M, S, VS) as shown in Figure 4.11. The fuzzy sets in this case become shorter than before, so the accuracy and the performance of the controller are improved. As the number of sets is increased the fuzzy rules are increased as well to 625 activation rules ($5 \times 5 \times 5 \times 5 = 625$ activation rules) and these are presented in Appendix 2.

As an example a sample is presented where the activation rules are:

IF dr is VL and dc is VL and dl is VL and tr is LB THEN Sa is TLB

IF dr is M and dc is L and dl is VL and tr is RB THEN Sa is TRS

IF dr is S and dc is VL and dl is S and tr is RS THEN Sa is TZ



VL: Very Large *L: Large* *M: Medium* *S: Small* *VS: Very Small*

Figure 4-11 Distance Linguistic Labels

The fuzzy sets of the distance are increased whereas the number of the sets for target orientation (tr) and the turning angle (Sa) are still the same as in the FLC40 since the affected factor is increasing the number of sets for the distance signals.

The previous cases are considered in this enhancement fuzzy controller will be called FLC625.

Case 1

As shown in Figure 4- 12, in this case the robot can avoid collision with the obstacles when applying this fuzzy controller. The robot movements are presented as below

- In the home position, the robot detects the target position that is to the left hand side and the parallel obstacles. The robot turns to the left towards the target and moves in parallel with the left obstacle until it reaches point (a)

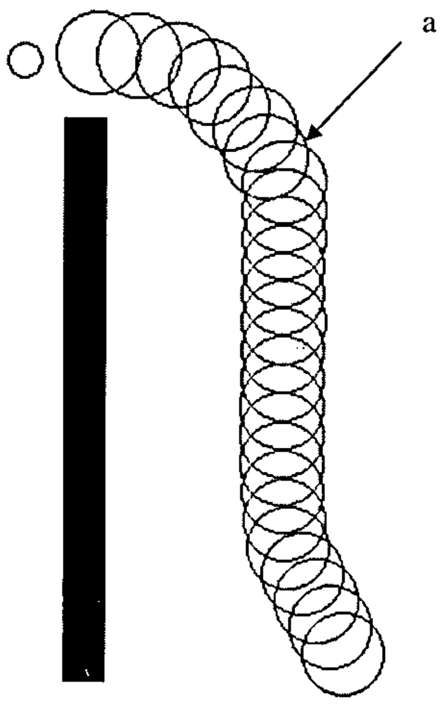


Figure 4- 12 FLC625, Case 1

- Obstacles are passed in point (a); the robot sensors are free (detect Very Large distance), so the robot turns sharply to the left since the activation rule is:

  IF dr is VL and dc is VL and dl is VL and tr is LB THEN Sa is TLB

  The robot moves until it reaches the target

__Case 2__

Figure 4- 13 below shows the result of the robot movement when the fuzzy logic controller is applied. The movements are discussed as below

- In this case, the first step, the robot moves towards the target as the obstacle is far away from the robot and the target is straightforward.

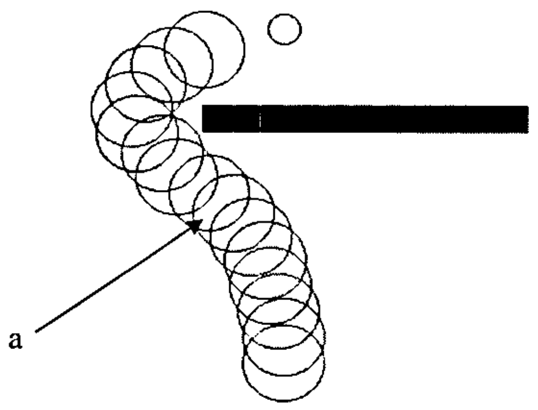


Figure 4- 13 FLC625, Case 2

- The robot keeps moving until it reaches a critical point, which is point (a). In this point, the right sensor is very close to the obstacle and the other sensors indicated that no obstacles are found and the target becomes in the right hand side of the robot. When entering these input signals to the fuzzy controller the following rule activated:

  IF dr is VS and dc is VL and dl is VL and tr is RS THEN Sa is TLS

  This time, the robot turns to the left sharply since there is one set activated and kept going until it passes the obstacle, then the robot detects the target position and turns towards the target.

Case 3

As indicated in Figure 4-14, the fuzzy logic controller is applied and the following results appear

- From the home position, since the input signals are very large distances from the sensor, and the target is approximately in the right hand side, the robot turns to the right and moves towards the target until it reaches a specific point which is point (a).

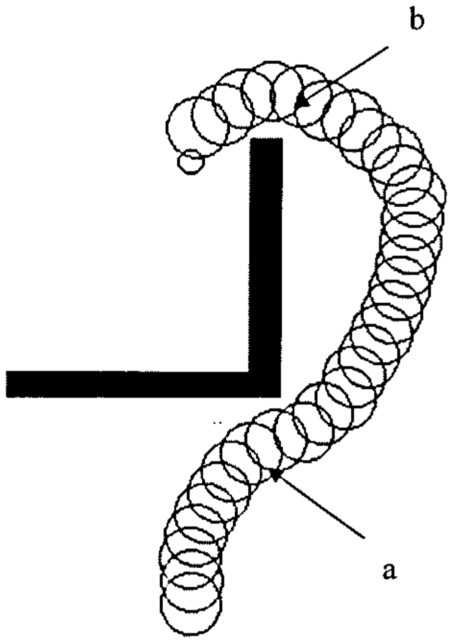


Figure 4-14 FLC625, Case 3

- In point (a), the left sensor of the robot indicated that the obstacle is close to the robot and the front sensor indicated that the obstacle is not too much close to the robot so the activation rules according to these data are:

  IF dr is VL and dc is L and dl is VS and tr is LS THEN Sa is TRS

  IF dr is VL and dc is M and dl is VS and tr is LS THEN Sa is TRS

  Accordingly the robot turns to the right to avoid collision with the obstacle depending on the defuzzification of these rules. After that, the robot moves towards the target until it reaches point (b).

- In point (b) all the robot sensors detect very large distance, and then turns towards the target until it reaches it.

Case 4

This case is one of the difficult cases that the robot tries to avoid collision with the obstacles and goes through a narrow passage to reach the target as shown in Figure 4-15.

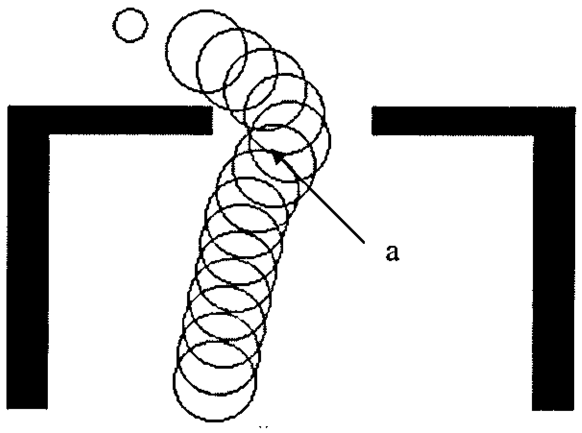


Figure 4-15 FLC625, Case 4

When implementing the fuzzy controller to this case the following results appear:

- When the robot begins the movement, it detects that the left obstacle is close so the robot turns to the right a little and keeps moving until it reaches point (a)
- In point (a) the left sensor is very close to the obstacle and the right sensor indicates that the obstacle is not too much far away. So the activated rules are:

  IF dr is S and dc is VL and dl is VS and tr is LB THEN Sa is TRS

  IF dr is M and dc is VL and dl is VS and tr is LB THEN Sa is TRS

  The robot turns to the right to avoid collision with the obstacle and keeps moving until it passes the obstacle. Then the robot turns to the left towards the target.

4.4 Development of the Neuro-Fuzzy Navigation System

The main problem in the fuzzy logic controller is the inference block, which consists of a large number of rules. To solve this problem, a neural network is used instead of the inference engine, hence a hybrid neuro-fuzzy controller is developed. The advantage of using neural networks to replace the inference engine is the expected reduction in the computational time. This advantage becomes very critical when hardware neural networks are implemented.

4.4.1 Structure of the Proposed Hybrid Neuro-Fuzzy System

As shown in Figure 4- 16, the inference engine is replaced by neural network, which also has the same facility that the input variable is fuzzy and the output variable from the neural network is also fuzzy. The neural network receives the fuzzy inputs from the fuzzification block and produces fuzzy linguistic labels as an output.

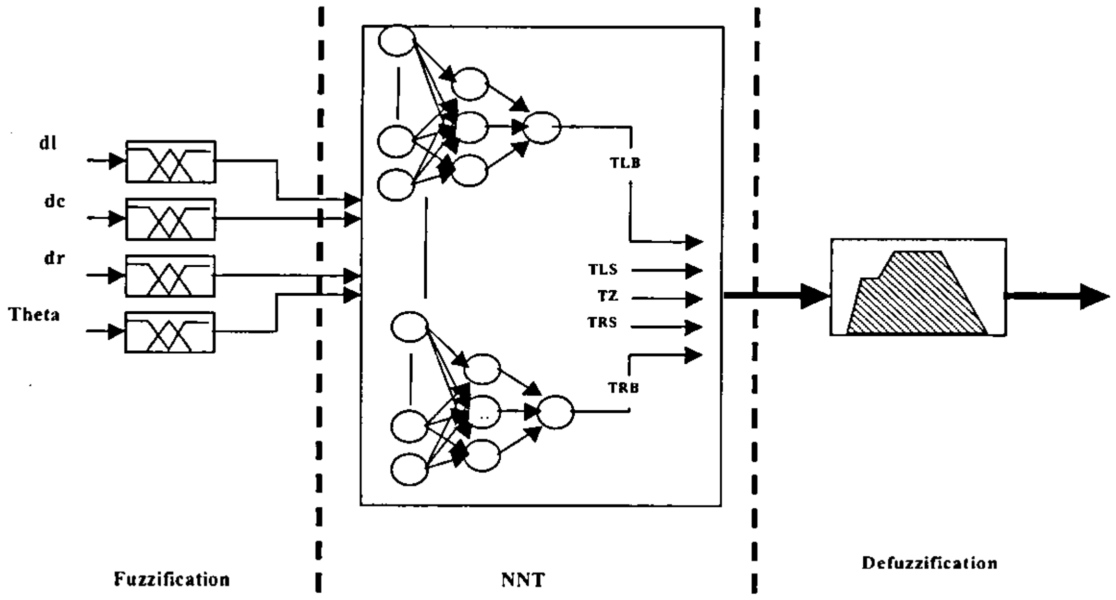


Figure 4-16 the Hybrid Neuro- Fuzzy Architecture

There are many types of neural networks, one of which is a feedforward neural network type used in this research. The architecture of the feedforward neural network is based on the input, hidden and output layer, where each layer has a special specification.

The input layer is designed to receive the input signals from the fuzzification block and translate it to the other. The input layer consists of 20 nodes, in which five nodes are given for each input (The same number of the fuzzy set of the input variables). This layer has the same membership function, which is binary sigmoid function as shown in Figure 4- 17.
In this thesis, the number of the hidden layers and the hidden nodes are chosen for the experimental work. The main objectives were to test the neural network for training purposes. In these trials (experiments) the one with maximum output error will be rejected.
From this layer the output of the neural network is translated to the defuzzification block in order to create the decision.

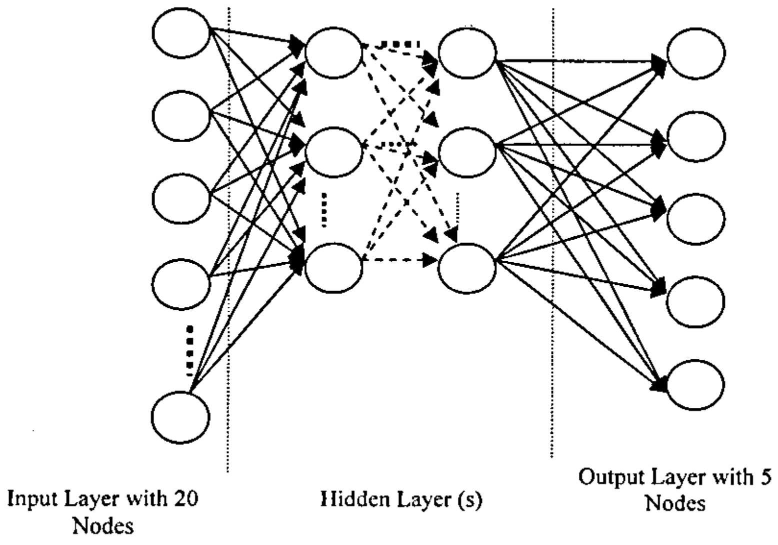


Figure 4-17 the Neural Networks Layers

4.4.2 Creating the Input and the Output Data

The neural network training in the feedforward method, the inputs, and the outputs data must be known. The inputs data are the fuzzy linguistic labels from the fuzzification block and the fuzzification block takes the crisp inputs from the environment (sensors). As shown in Figure 4- 11, the distance range is [0 –6] meters, but the interval from 1 to 5 is considered since the distance interval [0,1] has the linguistic label as 1 meter (VS) and the distance interval [5,6] has also the linguistic label 5 meters (VL). The same thing is for the angle orientation in which the interval that considered is [-65, 65] degrees since the sided sets have the linguistic labels as –65 and 65 respectively. The distance is divided into seventeen points with step of 0.25 meter between each point and step of 8.125 degree for the orientation angle input. So the resulted matrix for each input will be seventeen rows with five columns corresponding to the fuzzy sets of the input. As a result of the combinations of all these four matrixes, the input matrix of the neural network will be 83521 rows with 20 columns where the number 83521 comes from the combination of the all matrix rows $17 \times 17 \times 17 \times 17 = 83521$ and the number 20 comes from the sum of all the inputs sets $5 + 5 + 5 + 5 = 20$. To find the output of this huge numbers 83521×20 that is entered to the inference block to determine the desired output, which is 83521×5 where the number 5 is the fuzzy sets of the target turns.

After the inputs and the outputs matrices are found, then we have to use a back-propagation method with MATLAB software to train the neural network.

4.4.3 Training Neural Network

In this thesis, the training supervised method is used as framework of learning system based on the back-propagation method is used to train the neural network. The back-propagation method is used since the input and the output data are known, uses a low size from the memory compared with the other methods and it has a high speed in learning. Before the training begins a certain parameters must be adjusted

- The neural network (input, hidden and output layers) is built.
- The inputs and the outputs data are known.
- The membership function for each layer is labeled.
- The type of learning is defined.
- Weights Initialization are set.

The first three steps were discussed in the previous titles. The type of learning that is used will be the Levenberg-Marquardt (TRAINLM) as this method is faster, more accurate and with minimum error (MATLAB Package, version 6.1).

On the other hand, initializations of the weights are done in the MATLAB by choosing random values of the weights from the interval [-1, 1]. When these parameters are adjusted then the training of the neural networks begins.

Applying the huge number of input data to train the neural network, the following problems appear:

1. When applying the input matrix (83521 × 20) to a proposed neural network consists of only three layers (input layer with 20 nodes, hidden layer with three nodes and output layer with 5 nodes), the neural network can't be trained since the computer is out of memory [*].
2. A random uniform sample is taken from the input matrix (83521× 20) to decrease the input data and to enable the software to be trained. The random sample is (20000 × 20). This input matrix is entered to the neural network and the following results appear:

[*] The computer that was used in the training is PIV 1.8 GHZ Full Cash (512) with 768 MRAM.

i. A simple neural network is used which consist of two layers (input layer with 20 nodes and output layer with 5 nodes). After training of the neural network the maximum output error becomes very large 0.9091.

ii. Improving the neural network to three layers (input layer with 20 nodes, hidden layer with 5 nodes and output layer with 5 nodes) the maximum output error becomes high 0.881.

iii. New hidden nodes are added to the neural network to become input layer with 20 nodes, hidden layer with 10 nodes and output layer with 5 nodes. Also the maximum output error is high 0.78.

iv. After that, a new hidden layer is added to make the network as an input layer with 20 nodes, the first hidden layer with 5 nodes, the second hidden layer with 5 nodes and the output layer with 5 nodes. The maximum output error of this network still high. It is about 0.777.

v. Finally new nodes are added to the previous neural network to become 10 nodes in the first hidden layer where the maximum error is large 0.7011.

As has been shown in the pervious results, it is clear that the maximum output error is always high. To overcome this, another way has to be established to decrease this error. This research is investigating these ways. The multi neural network is used in this thesis where the input layer consists of 20 nodes and the output layer consists of only one node with four copies of this neural networks. As a result, the system consists of five neural networks with parallel inputs and five separated outputs as shown in Figure 4- 18.

The results of these methods are illustrated in the following points:

1. The input matrix (83521 × 20) can be trained in this method and the results are presented as follows:

i. When applying the input matrix to a simple neural network (input layer with 20 nodes, hidden layer with 5 nodes and output layer with 1 node) the maximum error is 0.6933, which is high and the percentage of the errors is 12.8 %, which is above 0.1 of the whole data.

ii. After increasing the number of nodes in the hidden layer to be 10 the maximum output error is 0.545 and percentage of the errors above 0.1 becomes 9.7 % of the whole data.

iii. Another hidden layer is added to the network to become two hidden layers. Both of the layers consist of 5 nodes. The maximum error in this case is 0.432 and the percentage of the errors above 0.1 is 1.84 % of the whole data.

iv. The number of nodes are increased in first layer to become 10 nodes. And the result is good since the maximum error becomes 0.2913 and the percentage of errors above 0.1 is 0.47 %.

v. Finally, the number of nodes for the second hidden layer is increased to 10 nodes. But in this case the output is out of memory.

2. Another training data has been done by choosing uniform random data from a large data population. This sample is about (20000 × 20) input matrix. When this input matrix is applied to the modification neural network, the following results appear:

i. When applying the input matrix (20000 × 20) to a simple neural network, which consists of input layer with 20 nodes, one hidden layer with 5 nodes and one node output layer, the maximum output error is about 0.8541 and the percentage of errors above 0.1 is 22.3%.

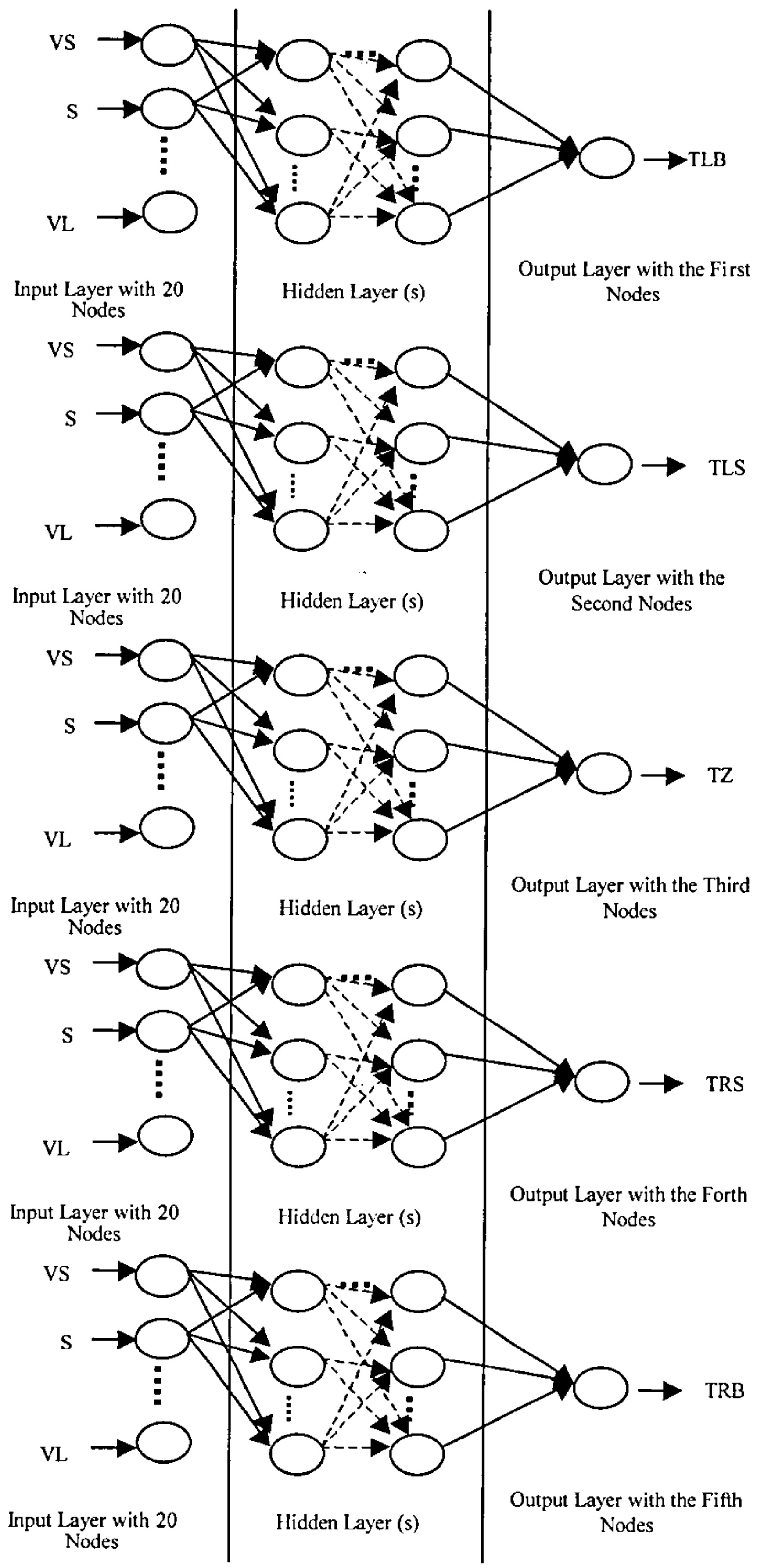

VS
S
VL
Input Layer with 20 Nodes
Hidden Layer (s)
Output Layer with the First Nodes
TLB
VS
S
VL
Input Layer with 20 Nodes
Hidden Layer (s)
Output Layer with the Second Nodes
TLS
VS
S
VL
Input Layer with 20 Nodes
Hidden Layer (s)
Output Layer with the Third Nodes
TZ
VS
S
VL
Input Layer with 20 Nodes
Hidden Layer (s)
Output Layer with the Forth Nodes
TRS
VS
S
VL
Input Layer with 20 Nodes
Hidden Layer (s)
Output Layer with the Fifth Nodes
TRB

Figure 4- 18 Five Neural Network with Parallel Inputs and Five Separated Outputs

ii. The number of the hidden nodes are increased to 10 nodes but still the maximum error is large 0.758 and the percentage error above 0.1 is 22.8 %.

iii. A new hidden layer is added to the network, where both of the hidden layers consist of 5 nodes. And the result is still high error value with high percentage in the error above 0.1.

iv. Finally the nodes in the first hidden layer are increased to 10 nodes. This time the maximum error is increased to 0.82 and percentage of errors above 0.1 is 22.45%.

It is clear as indicated by the pervious results, that the most significant network architecture is done by using two hidden layers the first of which is with 10 nodes and the second is with 5 nodes and applying 83521 × 20 input matrix.

4.4.4 Experimental Procedure

Having finished the neural networks with its matrix weights, they are translated to the simulation software instead of the inference engine. When this neuro fuzzy controller is applied to the pervious cases, the following results appear in Figure 4. 19 a, b, c and d:

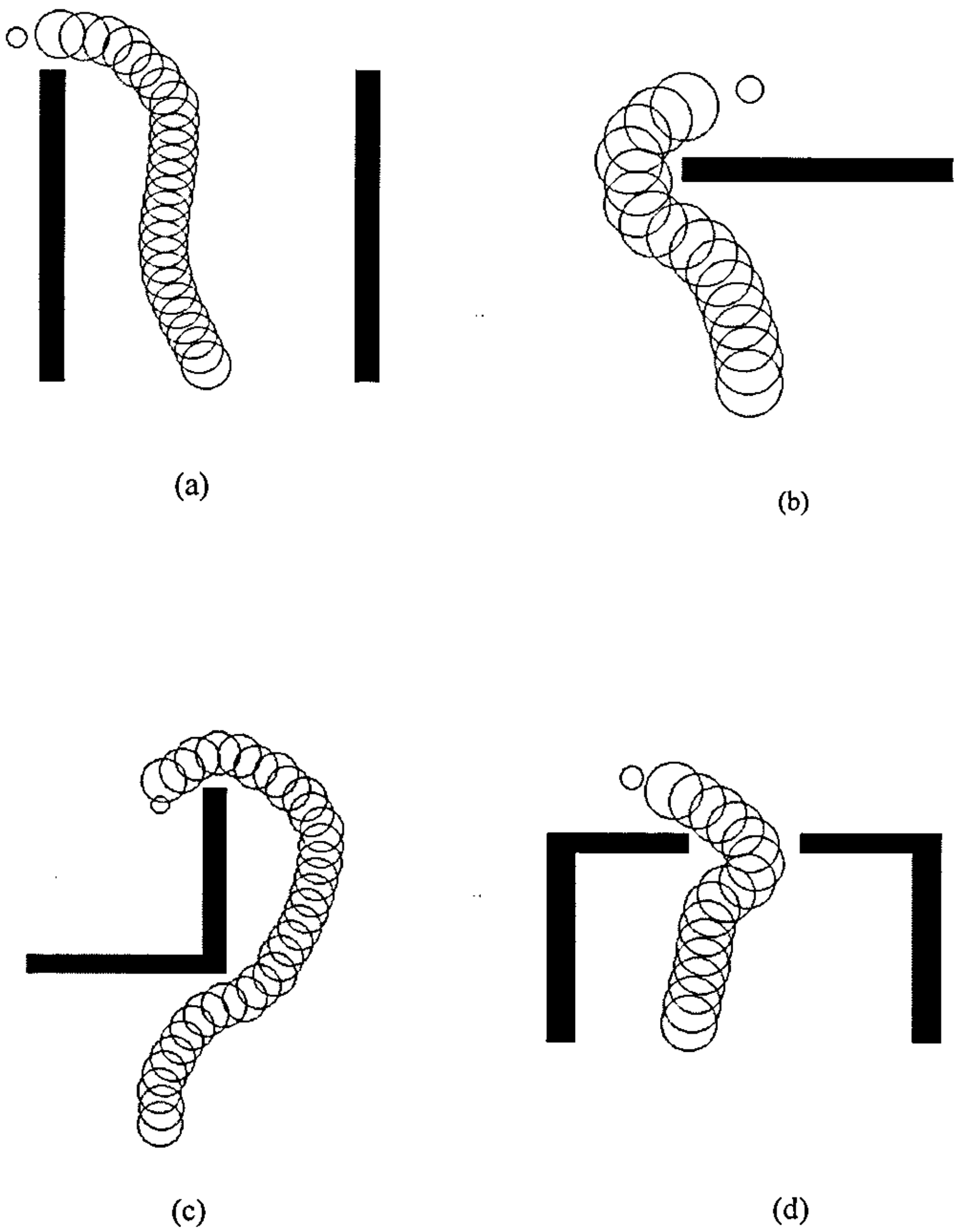


Figure 4.19 a) NFC, Case 1, b) NFC, Case 2, c) NFC, Case 3, d) NFC, Case 4

The previous cases have shown that the Neuro- Fuzzy controller has better performance as the improvement fuzzy controller. The error in this controller comes from the percentage error from training the neural network since the data are very huge, and there isn't enough memory to train the data in a large hidden layers or a large number of nodes in order to decrease the error.

4.5 Concluding Remarks:

In the beginning, the forty-rule fuzzy navigator is built to implement some critical point of the test cases. As a result, this controller can't avoid the obstacle since the number of the sets in the fuzzy distance is very small (two sets). Improvement of this controller is applied to improve the performance of the robot, in these critical points, thus, relaxation of the set is introduced. Comparing the forty-rules fuzzy navigator with the improvement fuzzy Navigator system, it becomes clear that the improvement fuzzy navigator system has very good performance compared to the forty-rules fuzzy navigator system. But there is a problem in the improvement of the fuzzy navigator system concerning the number of rules, which is huge (625 activation rules), so compiling and running these rules need a long time to create the decision. In general, the rule base is the main disadvantage of the fuzzy logic controller, so when the number of inputs is large and the number of sets for each input is large as well, the number of rules will be very large where these rules need long time in order to create decision i.e. the inference CPU time is 13 times the FLC40 (1736μS). The neural network has been investigated in order to decrease the processing time of the inference engine as will be discussed in the next chapter. Next chapter will illustrate the simulation software, which has three main categories, the first of which is to simulate the robot motion, the second one is the hardware control of the robot and the third is building the neuro-fuzzy controller.

SIMULATION AND EXPERIMENTAL RESULTS

5.1 Introduction

This chapter consists of two major parts. The first part presents the simulation software and its components. Simulation screens, map development and calibration modules, path planning modules are all presented in the first part of the chapter. The second part is devoted to the evaluation of the developed FLC625 and NFC of chapter 4. In the second part, a comparison of performance between FLC40 and FLC625 is conducted. Then, the performance of the NFC system is compared to FLC625. The testing of the developed systems on a prototype mobile robot is also presented here. The performance measures used for the comparison are the smoothness of the generated path, degree of obstacle avoidance, and computation time requirements.

5.2 Simulation Software

The software developed in this research has two purposes. First, it has been designed for the purpose of simulating the performance of the mobile robot according to three approaches, i.e., FLC40, FLC625, and NFC. Secondly, the software has also been designed for the actual motion control of a mobile robot by generating the required motion commands to be transmitted to the robot controller via the parallel port.

In this thesis, Visual Basic is used for the development of the software. Visual Basic works in an object-oriented environment with easy and suitable programming and graphics tools. The procedural steps of the software are shown in Figure 5.1.

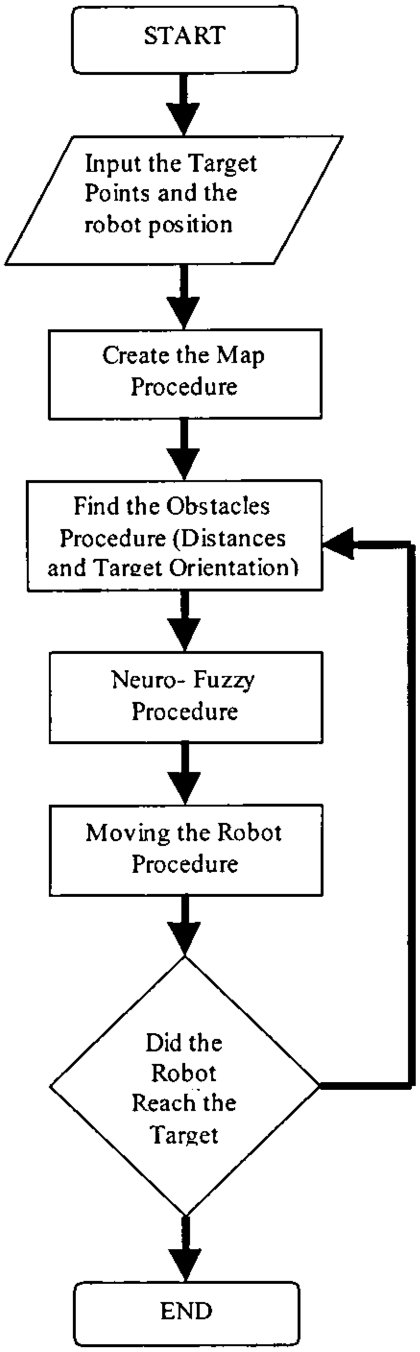

START
Input the Target Points and the robot position
Create the Map Procedure
Find the Obstacles Procedure (Distances and Target Orientation)
Neuro- Fuzzy Procedure
Moving the Robot Procedure
Did the Robot Reach the Target
END

Figure 5.1: Flowchart of the Simulation Software.

The initial screen for preparing the simulation is shown in Figure 5.2 which consists of:

1. Target position.
2. Detected distances.
3. Turning angle.
4. Robot position.
5. Simulation screen.

6. Control the robot movement.
7. New robot position.

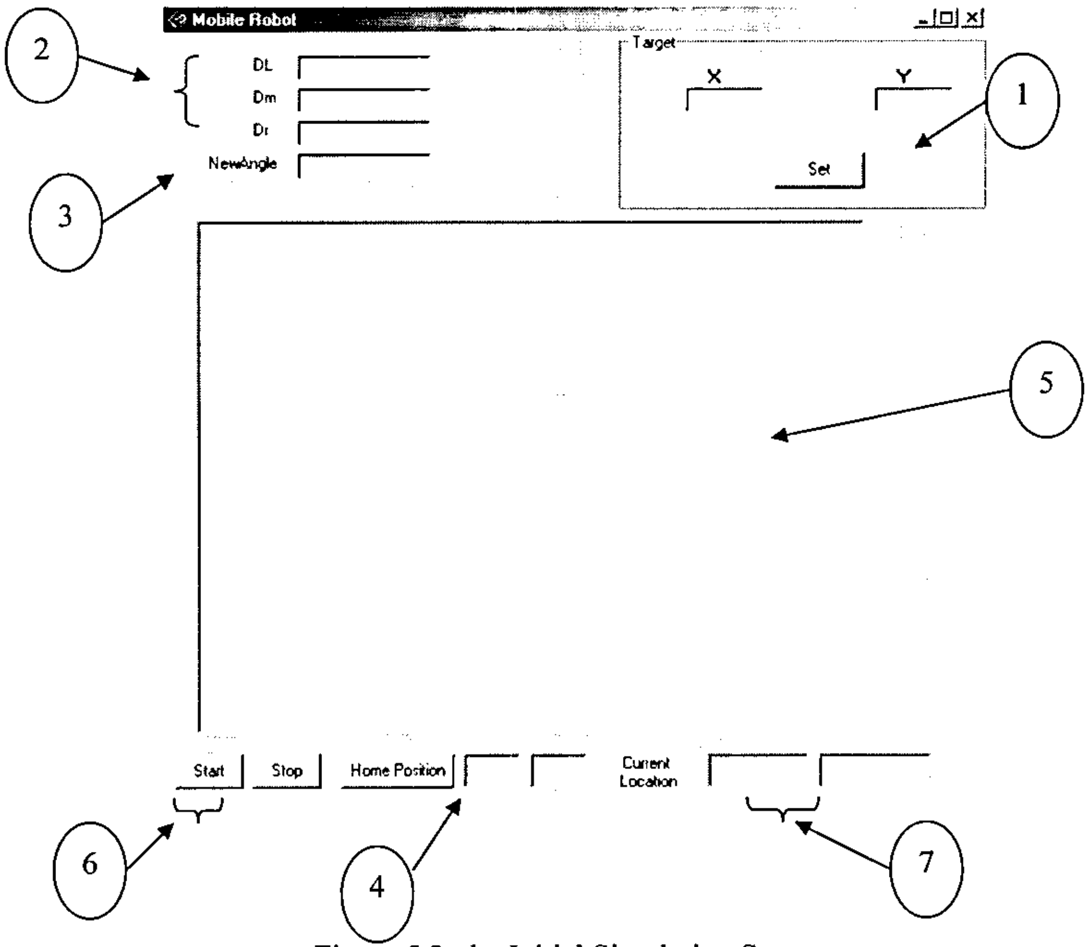
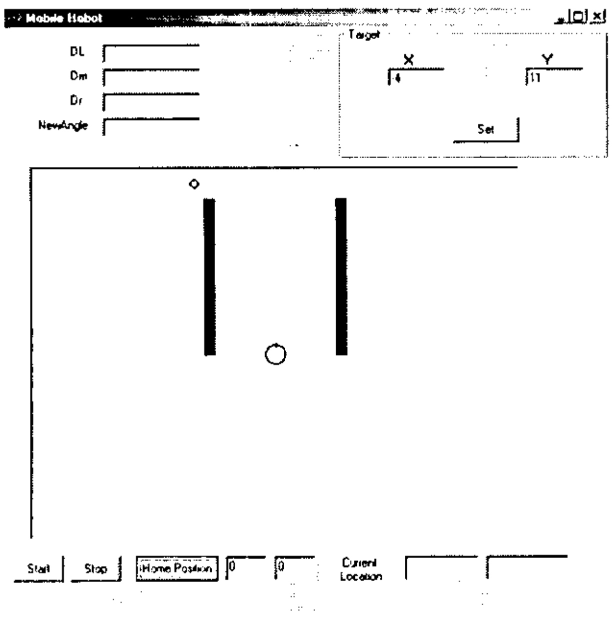


Figure 5.2: the Initial Simulation Screen.

The steps in the execution of the program are outlined as follows:

1. Initially, the user enters the position of the target and click *set,* so as to set the target parameters on the map.
2. The robot position is entered and then clicks the *home position* button to show the map, the robot and target position as shown in Figure 5.3.



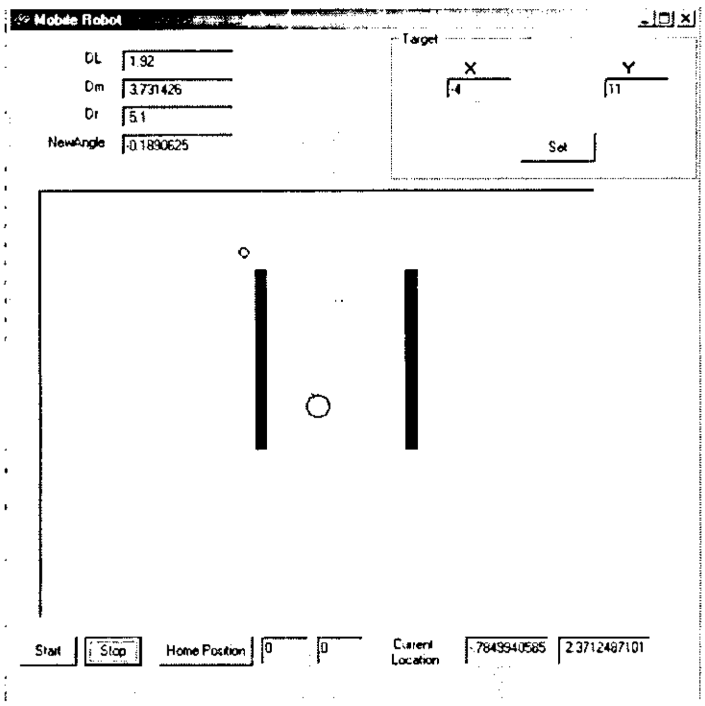
X: -4

Y: 11

Home Position: 0 0

Figure 5.3. the Simulation Screen Initialization

3. The simulation program begins when the *Start* button is clicked, as a result of this, the distance readings appear in the distance blocks and the turning angle also appears in its block. The new position of the robot will be recorded on the right hand side and two blocks are allocated, as shown in Figure 5.4.



DL 1.92

Dm 3.731426

Dr 5.1

NewAngle -0.1890625

X: -4

Y: 11

Home Position: 0 0

Current Location: -.7843940585 2.3712487101

Figure 5.4: Running Simulation Screen

5.2.1 The Simulation Part

The simulation part is a tool to represent the algorithm performance. It can perform the following tasks:

1. Creating the maps

In this region the software can be programmed to create maps with different sizes and shapes. All the cases in the pervious chapter are programmed with the assumption that the obstacles either in front of one or on both sides of the robot.

2. Finding the obstacles

In practical cases, the robot can find the obstacle by using ultrasound sensor. The ultrasound sensor is considered in the simulation software by applying the actual theory of the ultrasound sensor. In this simulation the ultrasound is designed with a detection area from –15 to 15 degrees for each sensor. In this area, 30 lines are transmitted from –15 degree to 15 degrees with one line for each degree as shown in Figure 5.5. The actual reaction of the ultrasound sensor can be satisfied in this simulation by the creation of this area.

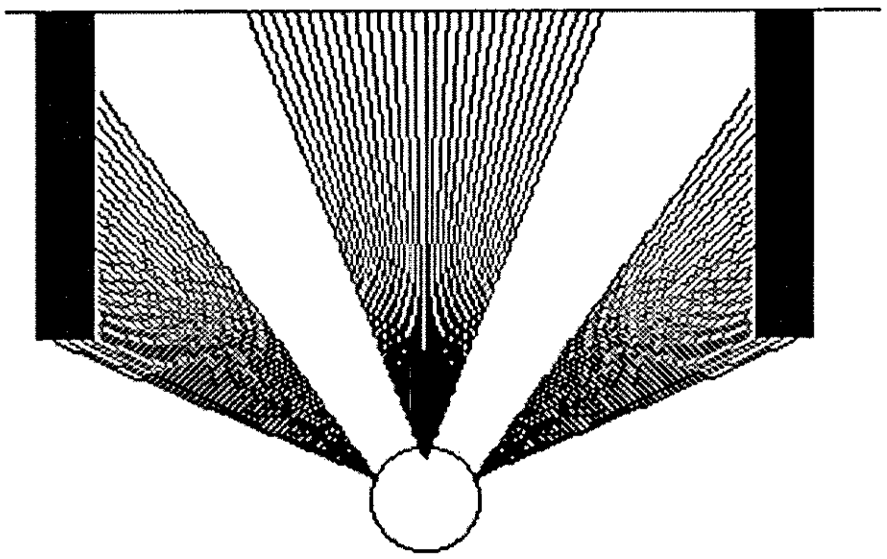


Figure 5.5: the Ultrasound in the Simulation

When 30 lines are transmitted to scan an area, any reflective line- in the simulation, this indicates an intercept of a blue pixel- will indicate the obstacle distance between the robot and the obstacle.

3. Moving the robot

The detection of the obstacle through simulation translates the distance to the Neuro- Fuzzy controller. The defuzzification of the Neuro – Fuzzy controller finds the crisp decision, which translates that to the moving robot procedure allocated in the simulation. In this procedure the robot moves from the old position to the new position according to the decision that comes from the defuzzification block.

5.2.2 The Experimental Mobile Robot Prototype

A prototype mobile robot has been constructed as part of a research project funded by the Faculty of Scientific Research. In this research, this mobile robot is used to test the applicability of the Neuro-Fuzzy controller in real life situations. In the following paragraphs, a brief description of the construction of the robot is provided.

1. Mechanical construction of the mobile robot

The mechanical part of the mobile robot is built by using two DC motors, fiberglass, gears and wheels. As shown in Figure 5.6, the robot is driven by two DC motors into a different direction by decreasing or increasing the speed of one of the motor or both. The motors are coupled to the wheels through gears, which are used to decrease the speed of the motors and increase the torque. A fiberglass base is used to cover the motors and wheels in order to fit the battery and the controlling board on it.

Software described in 4.4.2. In this procedure, the output variables of the fuzzification entered to the neural network in parallel, so the output of these neural networks translated to other procedure (which is Defuzzification). In the Defuzzification, the center of gravity method is used to translate the fuzzy value into crisp value (which is the decision)

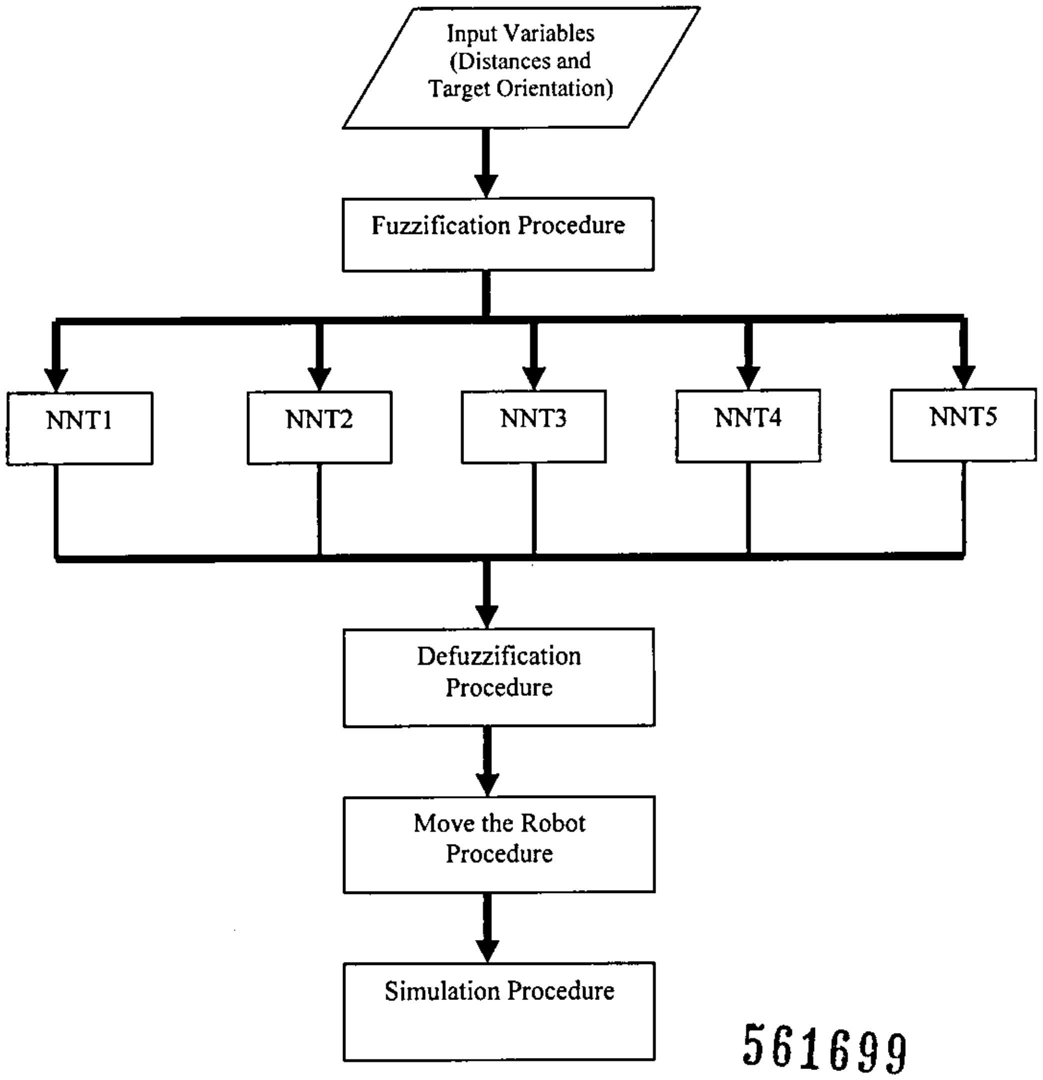


Figure 5.7: Neuro-Fuzzy Controller Interface

5.3 Experimental Results

Number of cases are implemented in the FLC40, FLC625 and NFC. The performance of these controllers and its results are discussed below.

5.3.1 Forty-rule Fuzzy navigator system

The FLC40 is implemented in a number of cases and the detailed results of each case are shown below:

Case 1

Case 1 had two parallel vertical obstacles and the target is on the left side above the left obstacle as shown in Figure5-8. Applying the fuzzy controller, the following notes appear:

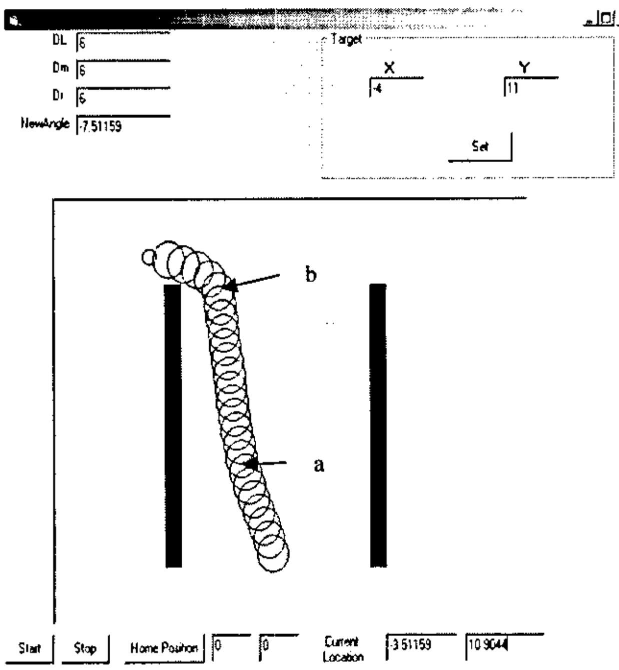

DL 6
Dm 6
Dr 6
NewAngle -7.51159
Target X -4 Y 11
Home Position 0 0
Current Location -3.51159 10.9044

Figure 5- 8 FLC40, Case 1

- When the robot is in the home position, two things are taken into consideration, target position and avoiding obstacles. In this case, the robot in the home position, detects the target, which is on the left-hand side, and the sensors don't detect any obstacles. The inputs signals to the fuzzy controller from the sensor readings are the left distance which is FAR, and front distance which is FAR and the right distance is FAR and the target orientation is left big. According to these data the activation rule is:

IF dr is FAR and dc is FAR and dl is FAR and tr is LB THEN Sa is TLB

After defuzzification of the rule, the robot is turned to the left sharply.

- The robot continues to move towards the target until it reaches point (a). In this point, the left sensor and the front sensor of the robot detect the obstacle which is not FAR. So the following rules are activated

  IF dr is FAR and dc is FAR and dl is NEAR and tr is LB THEN Sa is TZ

  IF dr is FAR and dc is NEAR and dl is NEAR and tr is LB THEN Sa is TRS

  According to these rules, the robot action turns slightly to the right and continues its movement in the same direction since the activation rule is

  IF dr is FAR and dc is FAR and dl is NEAR and tr is LS THEN Sa is TZ

- In point (b) all the sensors of the robot detect the FAR distance (since it pass the obstacle) so the activation rule is:

  IF dr is FAR and dc is FAR and dl is FAR and tr is LB THEN Sa is TLB

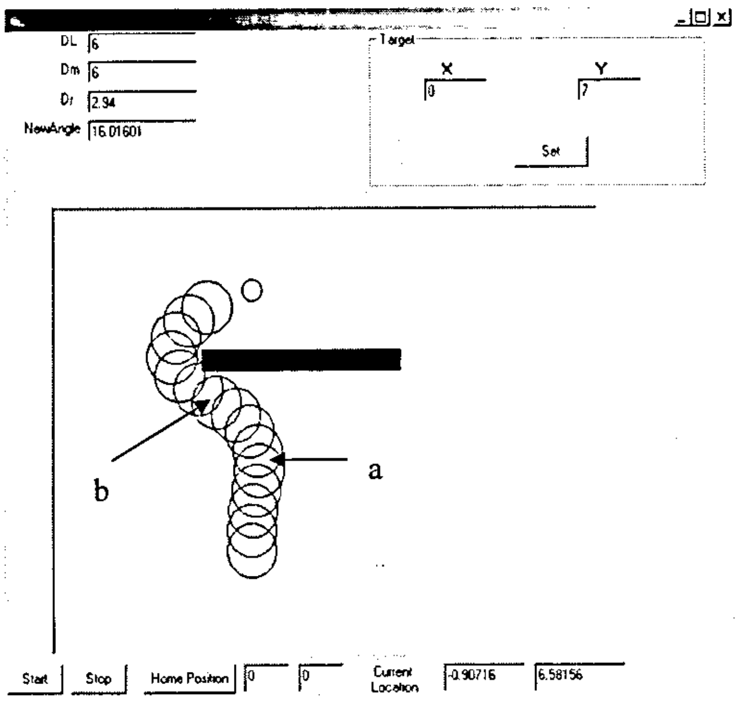
  Again the robot turns sharply to the left and moves in the same direction but the left sensor detects that there is an obstacle with less than 0.5 meter far from the robot. So the activation rule is:

  IF dr is FAR and dc is FAR and dl is NEAR and tr is Z THEN Sa is TZ

  The robot goes in the same direction and touches the obstacle.

Case 2

In this case, the target position is on the top of the horizontal obstacle as shown in Figure 5.9



DL 6
Dm 6
Dr 2.94
NewAngle 16.01601

Target
X 0
Y 7

Home Position 0 0
Current Location -0.90716 6.58156

Figure 5-9 FLC40, Case 2

The Fuzzy logic controller is applied to this case and the following results appear:

- In home position, the robot detects that the target in a straightforward sense and the obstacle is FAR since the distance is 5 meters. According to these data the robot turns into a zero angle since the activation rule is

  IF dr is FAR and dc is FAR and dl is FAR and tr is Z THEN Sa is TZ.

- When the robot sensors detect the obstacle in point (a) it turns to the left because the activation rules in this point are:

  IF dr is NEAR and dc is NEAR and dl is NEAR and tr is Z THEN Sa is TLB.

  IF dr is FAR and dc is NEAR and dl is FAR and tr is Z THEN Sa is TLS.

Where as in point (b) the right sensor detects the obstacle and the activation rule is

IF dr is NEAR and dc is FAR and dl is FAR and tr is RS THEN Sa is TZ.

Corresponding to this rule, the robot moves in the same direction and touches the obstacle.

Case 3

In this case the target above a horizontal obstacle and there is another vertical obstacle as shown in Figure 5- 10.

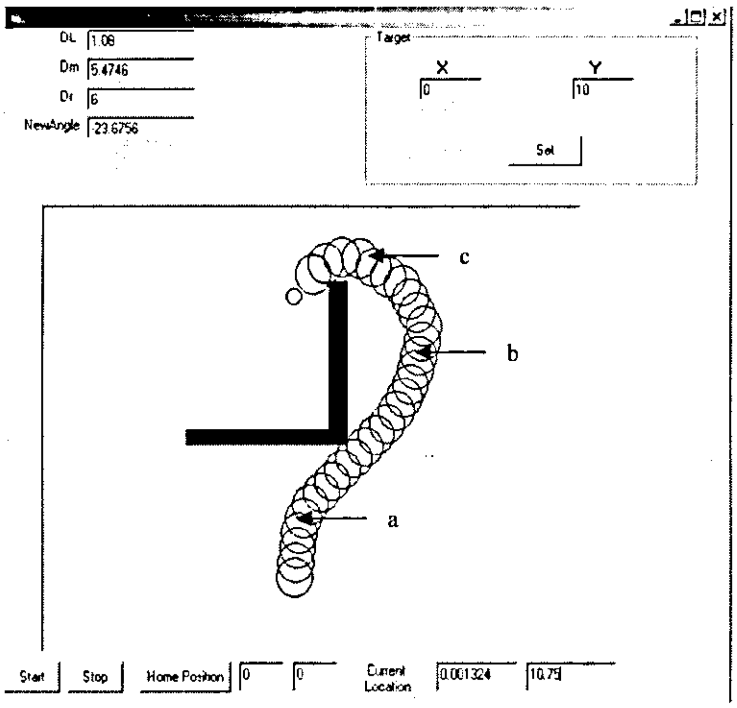

DL 1.08
Dm 5.4746
Dr 6
NewAngle -23.6756
Target X 0 Y 10
Home Position 0 0
Current Location 0.001324 10.75

Figure 5-10 FLC40, Case 3

The fuzzy logic controller is applied to this case and the following notes appear:

- The distance between the robot and the obstacle is the same as for case 2, so all the robot sensors detect the FAR distances. But here the target is approximately in the right hand side of the robot, so the robot initially turns slightly to the right towards the target since the activation rule is:

  IF dr is FAR and dc is FAR and dl is FAR and tr is RS THEN Sa is RS.

- When the robot reaches point (a) it becomes is close to the obstacle and the left and front sensor are detecting the obstacle, so the distances become NEAR for both sensors. According to these signals, the activation rules are

  IF dr is FAR and dc is NEAR and dl is NEAR and tr is LS THEN Sa is TRS

As a result, the robot turns to the right slightly.

- After passing the horizontal obstacle, the robot tries to avoid collision with the vertical obstacle to reach the target. So the left sensor detects the obstacle and accordingly the activation rule is:

  IF dr is FAR and dc is FAR and dl is NEAR and tr is LS THEN Sa is TZ

  As a result, the robot moves in the same direction until it reaches point (b).

- In point (b) all the robot sensors detect FAR distance, so the robot turns towards the target since the activation rule is:

  IF dr is FAR and dc is FAR and dl is FAR and tr is LS THEN Sa is TLS

- Finally in point (c) the robot reaches the edge of the vertical obstacle and the left sensor detects the obstacle and the robot moves in the same direction since the activation rule is:

  IF dr is FAR and dc is FAR and dl is NEAR and tr is LS THEN Sa is TZ

  After that, all the sensors of the robot detect FAR distance so the robot turns to the left towards the target and touches the edge of the vertical obstacle according to the following rule:

  IF dr is FAR and dc is FAR and dl is FAR and tr is LS THEN Sa is TLS

Case 4

In this case, there are two vertical obstacles as shown in Figure 5-11 and the target is above the left obstacle.

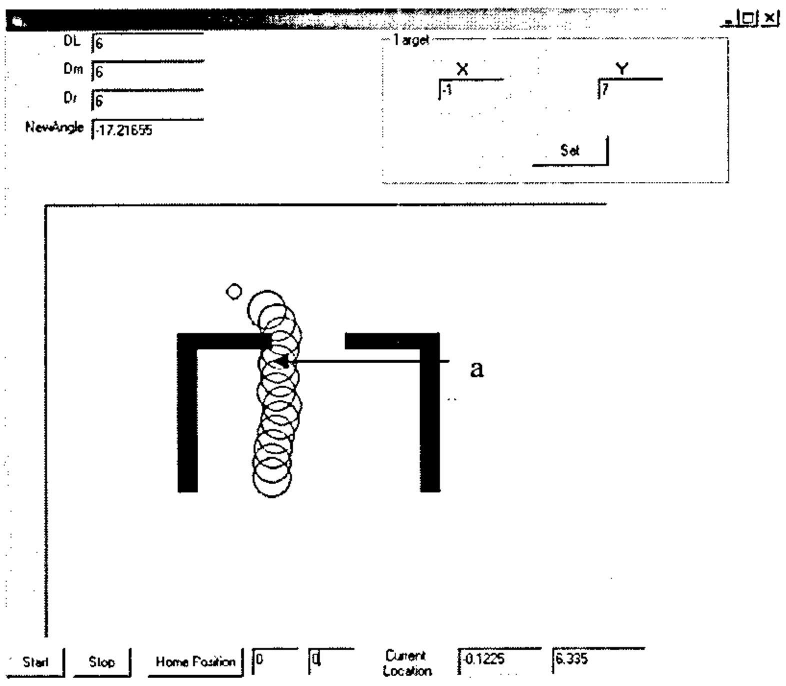

DL 6
Dm 6
Dr 6
NewAngle -17.21655
Target
X -1
Y 7
Home Position 0 0
Current Location -0.1225 6.335

Figure 5-11 FLC40, Case 4

When applying the fuzzy logic controller, the following notes appear.

- This case is one of the most difficult cases, as the robot tries to pass through the two obstacles. When the robot is in the home position, it detects the left obstacle whereas the other obstacles are far away. According to these inputs the activation rule is:

  IF dr is FAR and dc is FAR and dl is NEAR and tr is LS THEN Sa is TZ

  The robot moves in a straight line until it reaches point (a).

- In point (a) the left sensor detects the obstacle, so the robot still in the same direction. Accordingly the same rule is activated:

  IF dr is FAR and dc is FAR and dl is NEAR and tr is LB THEN Sa is TZ

  As a result, the robot touches the obstacle.

The small fuzzy inference engine has some problems at the edge of the obstacles where this is due to the limited reference of two sets (FAR, NEAR) in the fuzzy input distance. The

CPU time is very small since it has a limited number of rules. The CPU time results from the small inference engine as follows:

Fuzzufication time processing: 527 μS

Fuzzufication and Inference time processing: 659 μS

Inference time processing: 132 μS

The fuzzification time that translates the crisp value into fuzzy linguistic labels is small since the fuzzy sets for the distance are only two. The inference time in this inference engine is small as it consists of only fourty rules.

5.3.2 Improvement Fuzzy Navigation System

The FLC625 is implemented in the same previous case, and the results for each case are as follows:

Case 1

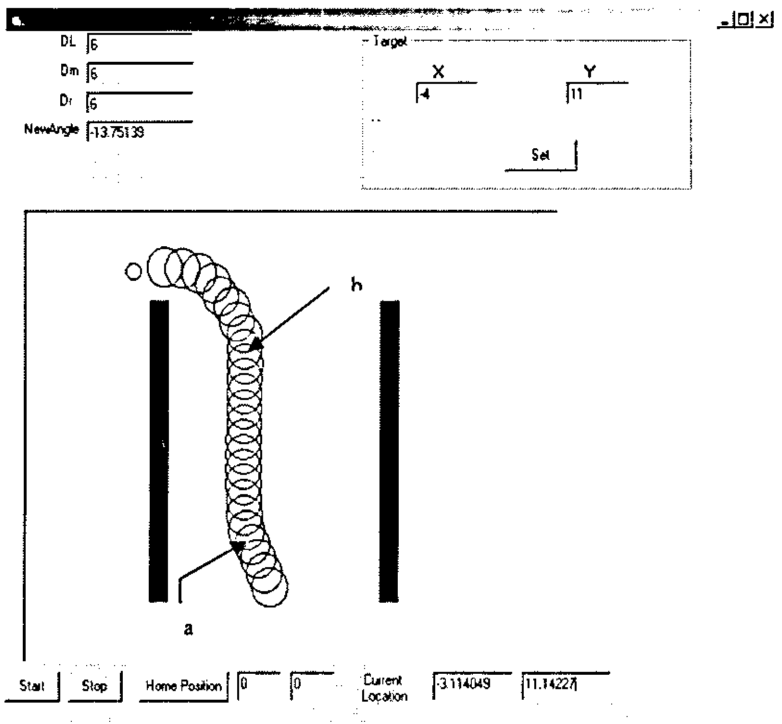
Applying this fuzzy controller to this case, the robot can avoid collision with the obstacles as shown in Figure 5- 12. The robot movements are discussed in the following points according to this controller:

- Two parallel obstacles and the target is on the left hand side. The sided sensors detected the distance between large and medium far from the robot, the robot detects the target position, using these data as input signals to the fuzzy controller, the following rules are activated:

  IF dr is L and dc is VL and dl is L and tr is LS THEN Sa is TLS

  IF dr is M and dc is VL and dl is M and tr is LS THEN Sa is TLS

  When defuzzification of these rules, the robot turns to the left towards the target of the zero point.



DL 6
Dm 6
Dr 6
NewAngle -13.75139

Target
X -4
Y 11

Current Location -3.114049 11.14227

Figure 5-12 FLC625, Case 1

- In point (a) the detected distance of the left sensor indicates that the distance between the robot and the left obstacle is between medium and small distance. As a result, the robot approximately turns to the zero since the activation rules are:

  IF dr is L and dc is VL and dl is M and tr is LS THEN Sa is TLS

  IF dr is M and dc is VL and dl is S and tr is LS THEN Sa is TZ

  And the robot moves in the same direction as the conditions are not changed.

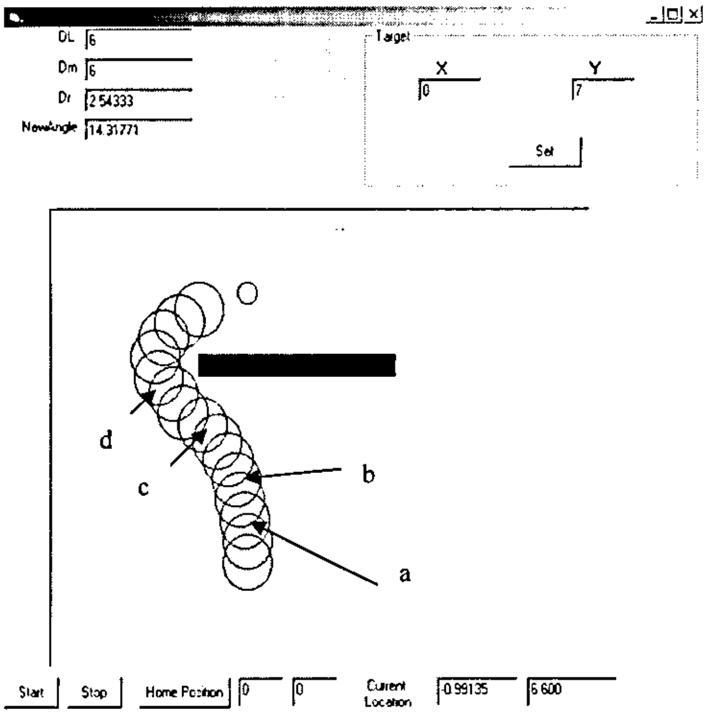
- At point (b) the robot sensors are free (detect Very Large distance), so the robot turns sharply to the left since the activation rule is

  IF dr is VL and dc is VL and dl is VL and tr is LB THEN Sa is TLB

  The robot moves until it reaches the target.

Case 2

Figure 5- 13 shows the result of the robot movement when the fuzzy logic controller is used. These movements are discussed in the following points



DL 6
Dm 6
Dr 2.54333
NewAngle 14.31771

Target
X 0
Y 7

Current Location -0.99135 6.600

Home Position 0 0

Figure 5 -13 FLC625, Case 2

- The first two steps in this case, the robot moves toward the target since the obstacle is (Large) far from the robot, and the activation rule is:

  IF dr is VL and dc is VL and dl is VL and tr is Z THEN Sa is TZ

- In point (a) the robot detects that the obstacle is too close than before, so the front sensor detects the distance of the obstacle between large and medium far from the robot, while the sided sensors of the robot detect Large far distance of the obstacles. According to these input signals the following rules are activated:

  IF dr is L and dc is L and dl is L and tr is Z THEN Sa is TZ

  IF dr is L and dc is M and dl is L and tr is Z THEN Sa is TLS

  As a result, the robot turns to the left slightly and then moves until it reaches point (b).

- In point (b) the right sensor of the robot detects the obstacle and the other sensors detect Very Large distance. So the activation rules are:

  IF dr is S and dc is VL and dl is VL and tr is RS THEN Sa is TZ

IF dr is VS and dc is VL and dl is VL and tr is RS THEN Sa is TLS

After defuzzification, the output action is to turn the robot to the left slightly again and the robot turns to the left until it reaches the critical point (c).

- In point (c) the right sensor is very close to the obstacle and the other sensors indicate that there are no obstacles and the target becomes on the right hand side of the robot. When entering these input signals to the fuzzy controller the following rules are activated

  IF dr is VS and dc is VL and dl is VL and tr is RS THEN Sa is TLS

  This time the robot turns to the left sharply since there is one set activated and keep going until it passes the obstacle.

- After passing the obstacle, all the robot sensors in point (d) indicate that there are no other obstacles between the robot and the target, so the robot turns sharply to the right since the activation rule is:

  IF dr is VL and dc is VL and dl is VL and tr is RB THEN Sa is TRB

Case 3

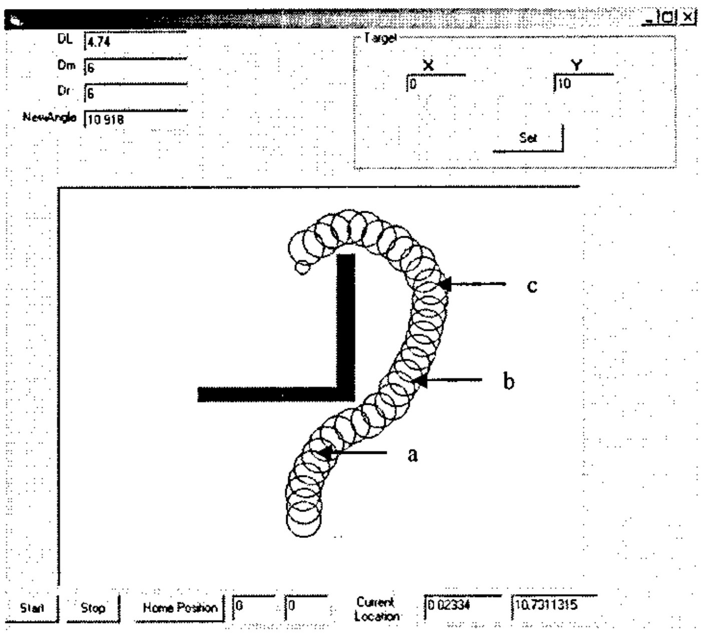
In this case as shown in Figure 5- 14, the fuzzy controller is applied, and the following results appear:

- From the first step, the robot turns to the right since the input signals are very large distance from the sensor and the target is approximately on the right hand side. So the activation rules are:

  IF dr is VL and dc is VL and dl is VL and tr is RS THEN Sa is TRS

  IF dr is VL and dc is VL and dl is VL and tr is Z THEN Sa is TZ

  As a result, the robot turns to the right slightly as two sets are activated. The robot continues to turns to the left until it reaches point (a).



DL 4.74
Dm 6
Dr 6
NewAngle 10 918
Target
X 0
Y 10
Current Location 0 02334 10.7311315
Home Position 0 0

Figure 5- 14 FLC625, Case 3

- In point (a), the left sensor of the robot indicates that the obstacle is close to the robot and the front sensor indicates that the obstacle is not too much close to the robot so the activation rules according to these data are:

  IF dr is VL and dc is L and dl is VS and tr is LS THEN Sa is TRS

  IF dr is VL and dc is M and dl is VS and tr is LS THEN Sa is TRS

  Then the robot turns to the right to avoid collision with the obstacle depending on the defuzzification of these rules.

- In point (b) the robot tries to avoid collision with the vertical obstacle, the left sensor detects the obstacle so the robot turns slightly to the left since the activation rules are:

  IF dr is VL and dc is VL and dl is M and tr is LB THEN Sa is TLS

  IF dr is VL and dc is VL and dl is S and tr is LB THEN Sa is TZ

- In point (c), the obstacle is far from the left sensor so the robot turns to the left slightly towards the target since the activation rules are

  IF dr is VL and dc is VL and dl is L and tr is LB THEN Sa is TLB

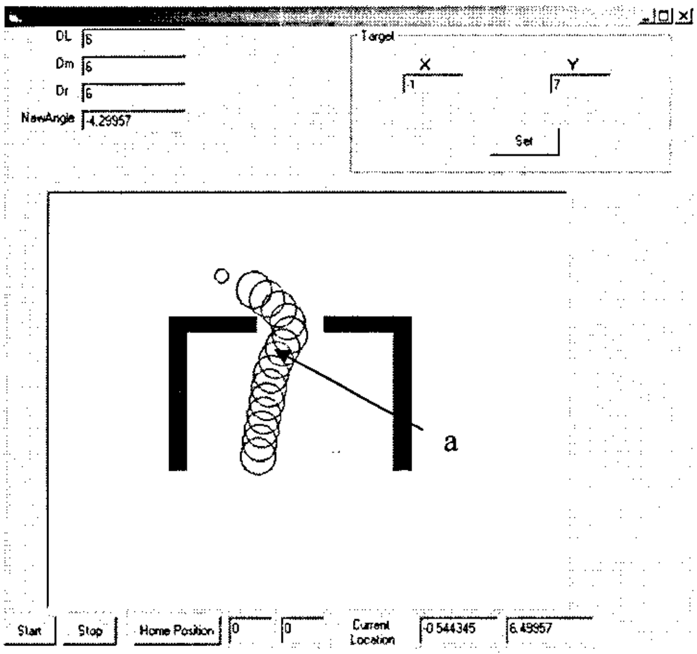
IF dr is VL and dc is VL and dl is M and tr is LB THEN Sa is TLS

- Finally, the robot moves above the vertical edge of the obstacle and all the robot sensors are free (Very Large distance detected) as it passes the obstacle. The robot again turns towards the target through the activation rule

  IF dr is VL and dc is VL and dl is VL and tr is LB THEN Sa is TLB

Case 4

This case is one of the difficult cases where the robot tries to avoid collision with the obstacles, as the robot moves through the narrow gap to reach the target as shown in Figure 5-15.


DL 6
Dm 6
Dr 6
NewAngle -4.29957
Target
X -1
Y 7

0 0
Current Location -0.544345 6.49957

Figure 5- 15 FLC625, Case 4

When implementing the fuzzy controller in this case the following results appear:

- In the home position, the left sensor of the robot detects the left obstacle, which is close to the robot, and the other sensors indicate that the other obstacles are far. According to these signals, the activation rules are

IF dr is VL and dc is VL and dl is S and tr is LS THEN Sa is TZ

IF dr is VL and dc is VL and dl is S and tr is Z THEN Sa is TZ

IF dr is VL and dc is VL and dl is M and tr is LS THEN Sa is TRS

IF dr is VL and dc is VL and dl is VL and tr is Z THEN Sa is TZ

After defuzzification of these rules, the robot turns to the right slightly and moves until it reaches point (a)

- In point (a) the left sensor is very close to the obstacle and the right sensor indicates that the obstacle is not too much far. So the activation rules are:

  IF dr is S and dc is VL and dl is VS and tr is LB THEN Sa is TRS

  IF dr is M and dc is VL and dl is VS and tr is LB THEN Sa is TRS

  The robot turns to the right to avoid collision with the obstacle and keep moving until it passes the obstacle.

- Once it passes the obstacle, the robot turns to the left towards the target as obstacles are not detected, according to the following rule:

  IF dr is VL and dc is VL and dl is VL and tr is LB THEN Sa is TLB

The performance of the fuzzy logic improvement is good since the robot doesn't touch any obstacle and it avoids collision with any obstacles as shown in the above cases. The main problem in this controller is the huge number of rules, which are reflected on the CPU time. The CPU time on this controller is:

Fuzzufication time processing: 1038 μS

Fuzzufication and Inference time processing: 2757 μS

Inference time processing: 1736 μS

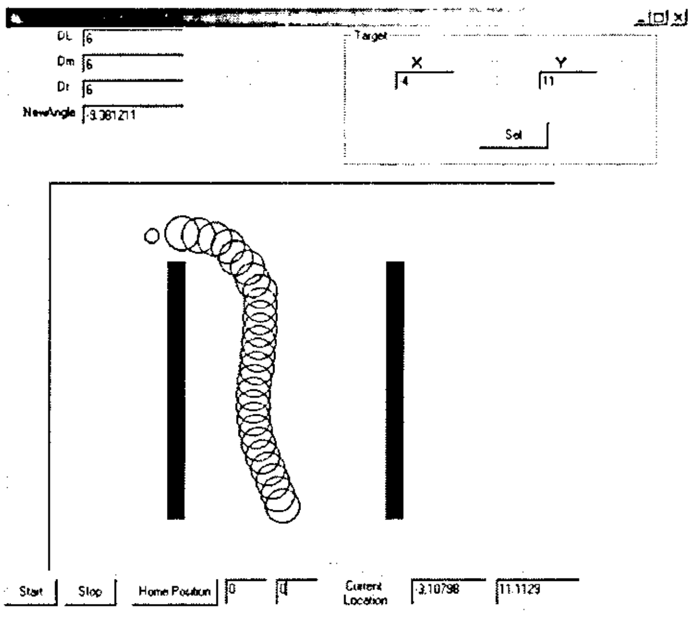
The fuzzification time required to translate the crisp input into the five sets is 1038μS, which is greater than the small fuzzy engine. The inference time in this controller is very much greater than the small inference engine since this controller processes a huge number of rules (625 activation rules) compared with small inference engine (40 activation rules).

5.3.3 Neuro- Fuzzy Controller

Considering the pervious cases, the performance of the Hybrid neuro- fuzzy controller is the same as the improvement fuzzy logic controller. The neural network in this controller is trained to do the same action as the inference engine in the improvement fuzzy logic controller. So the performance of this controller is a mirror to the improvement fuzzy performance as in the following cases.

__Case 1__

In this case, the robot moves exactly as the improvement fuzzy logic controller as shown in Figure 5-16.



DL 6
Dm 6
Dr 6
NewAngle -9.381271

X -4
Y 11

0 0
Current Location -3.10798 11.1129

Figure 5- 16 NFC, Case 1

__Case 2__

In this case, the robot moves towards the target then turns to the left sharply as shown in Figure 5-17. The robot avoids colliding with the obstacle since the neural network is well trained to keep the robot moving safely between the obstacles.

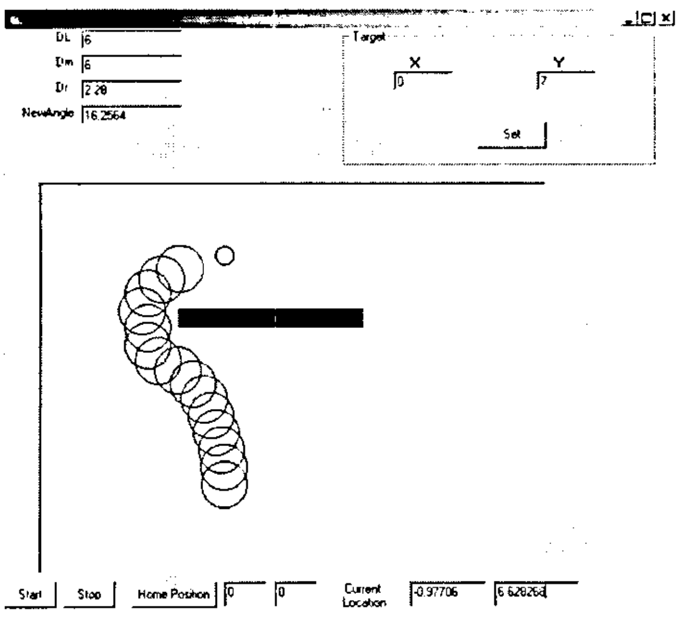
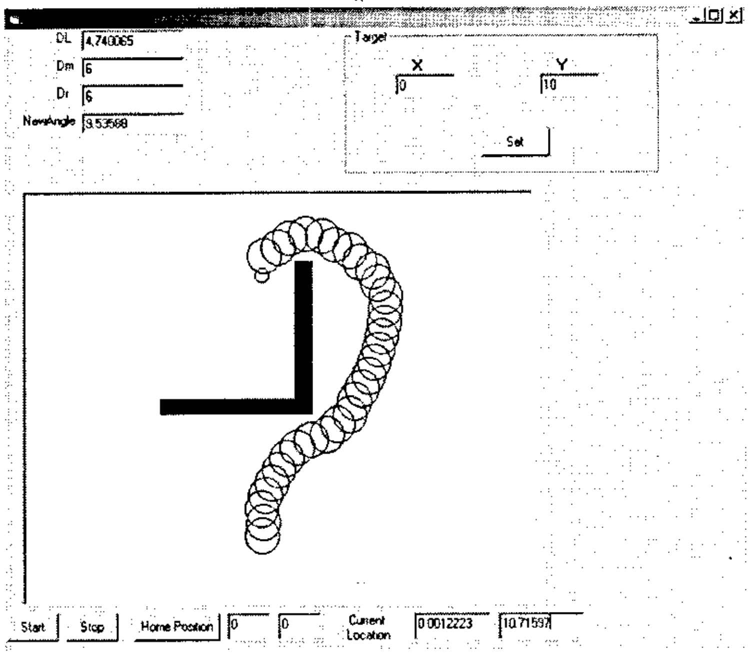

DL 6
Dm 6
Dr 2.28
NewAngle 16.2564
Target X 0 Y 7
Home Position 0 0
Current Location -0.97706 6.628268

Figure 5-17 NFC, Case 2

__Case 3__

The robot in this case has the same performance as the improvement fuzzy logic controller as shown in Figure 5-18. It turns to the left towards the target then turns to the left again sharply. After it passes the obstacle it turns to the right towards the target.

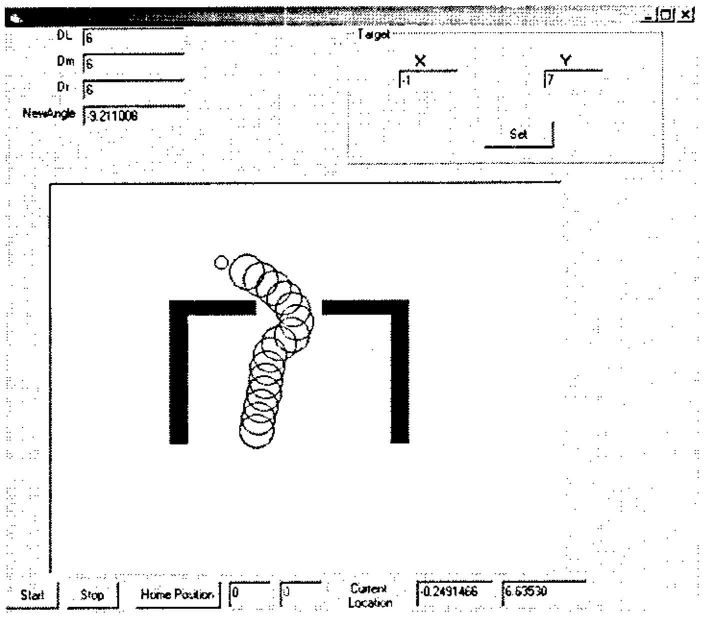

DL 4.740065
Dm 6
Dr 6
NewAngle 9.53568
Target
X 0
Y 10

0 0
Current Location 0.0012223 10.71597

Figure 5-18 NFC, Case 3

Case 4

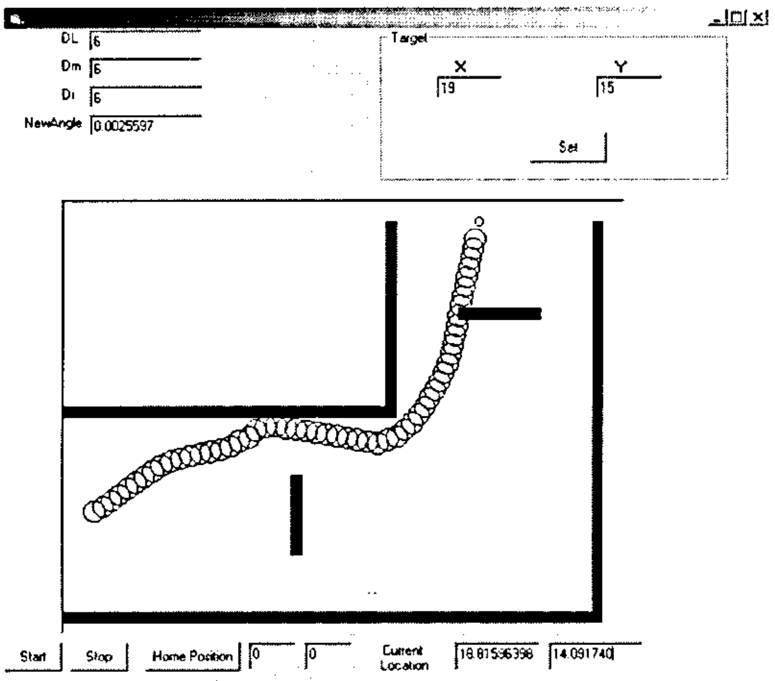
Also in this case, the robot moves towards the target with the same path as the FLC625 as shown in (Figure 5-19).


DL 6
Dm 6
Dr 6
NewAngle -9.211006
Target
X -1
Y 7

0 0
Current Location -0.2491466 6.63530

Figure 5- 19 NFC, Case 4

To test the performance of these three controllers in a close environment the outcome will be stated below. When implementing the FLC40 in this map, the robot collides with the wall first and after that collides with an obstacle as is shown in Figure 5- 20.



DL 6
Dm 6
Di 6
NewAngle 0.0025597

X 19
Y 15

0 0

Current Location 18.81596398 14.091740

Figure 5- 20 Implementing FLC40 to the First Map

The improved fuzzy navigator has a very good performance, the robot can avoid collision with any walls or obstacles as shown in Figure 5-21.

Then the hybrid neuro- fuzzy controller is implemented in the same map, it functions with a good performance as FLC625 and avoids collision with any obstacles or walls as shown in Figure 5-22.

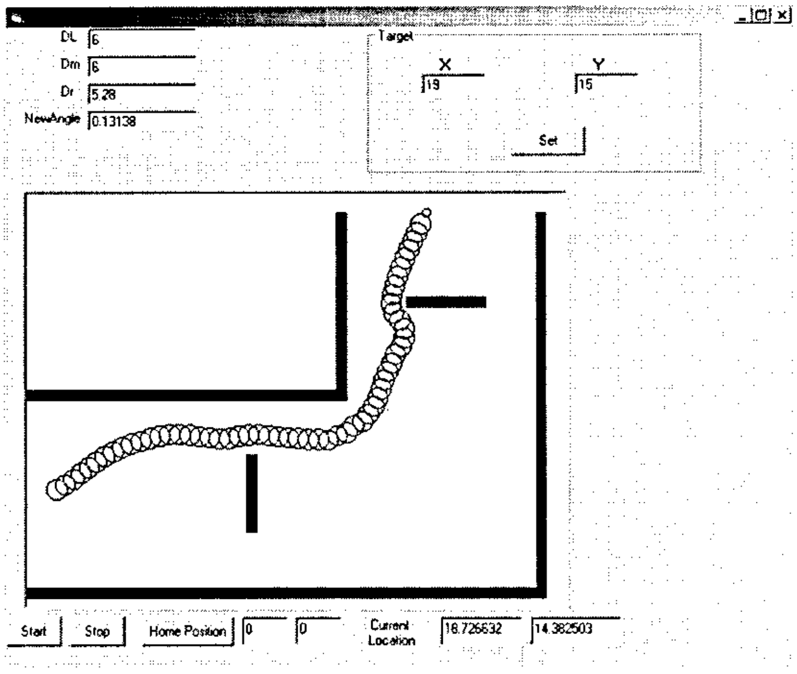

DL 6
Dm 6
Dr 5.28
NewAngle 0.13138
Target
X 19
Y 15
Current Location 18.726632 14.382503

Figure 5-21 Implementing FLC625 to the First Map.

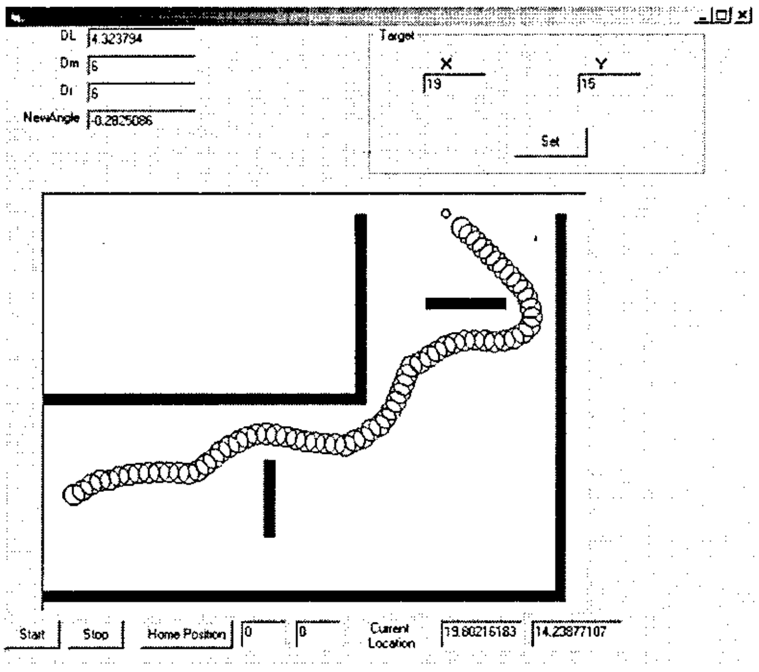
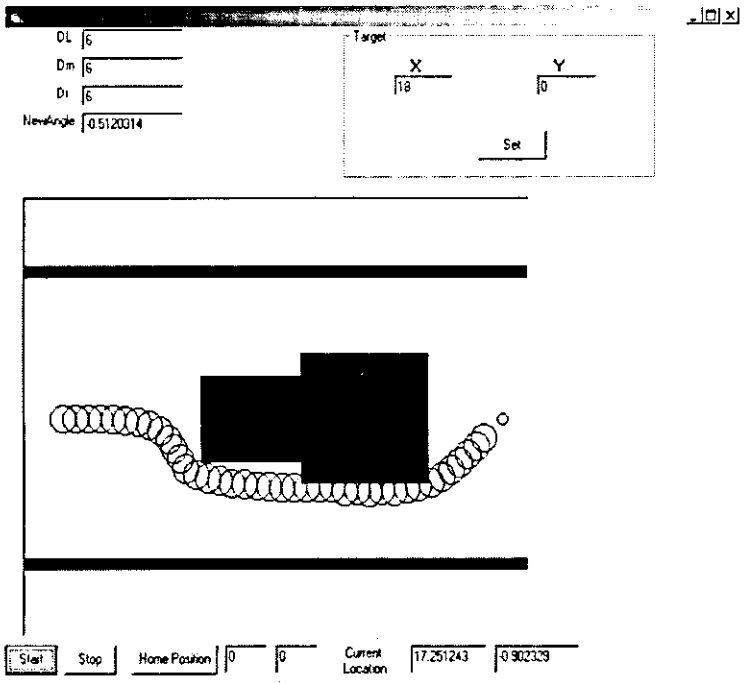

DL 4.323794
Dm 6
Dr 6
NewAngle -0.2825086
Target
X 19
Y 15
Current Location 19.60216183 14.23877107

Figure 5-22 Implementing NFC to the First Map

On the other hand, another map is designed, which is more difficult than before since the obstacles are continuous in two stages.

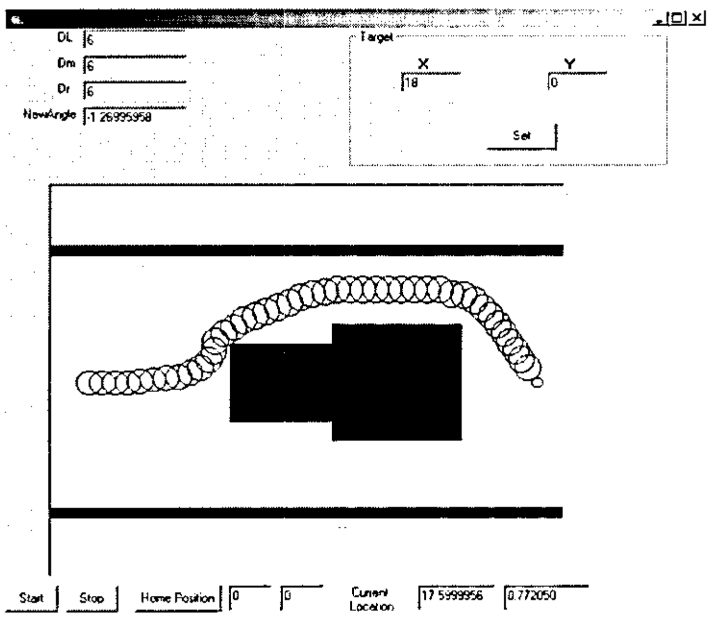
At the beginning the FLC40 is implemented in this map, the result is shown in Figure 5- 23 then the robot destroys the second stage of the obstacle.



DL 6
Dm 6
Dr 6
NewAngle -0.5120314
Target X 18 Y 0
Current Location 17.251243 -0.902329

Figure 5-23 Implementing FLC40 to the Second Map

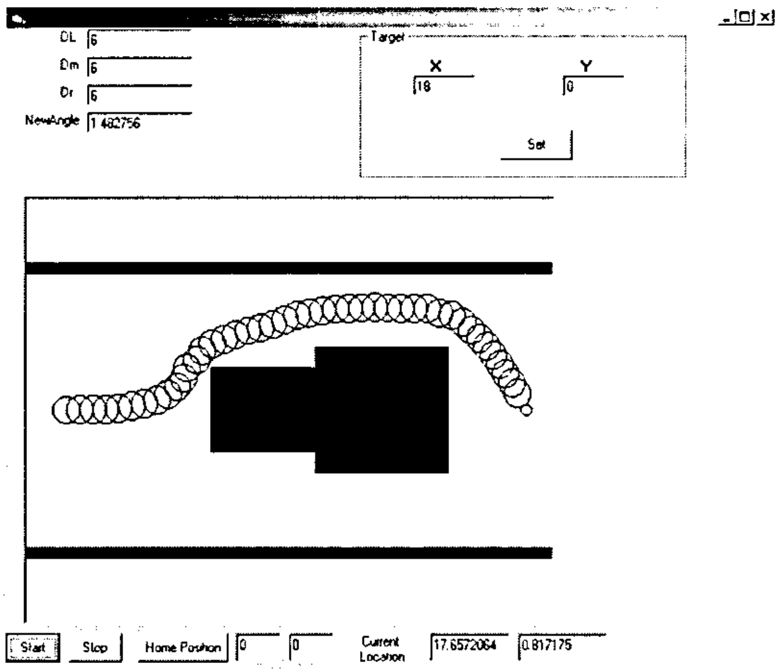
Using the FLC625 improve the performance of the robot, the robot doesn't touch the obstacle at the two stages and reaches the target safely as shown in Figure 5-24.



DL 6
Dm 6
Dr 6
NewAngle -1.26995958
Target X 18 Y 0
Current Location 17.5999956 0.772050

Figure 5- 24 Implementing FLC625 to the Second Map

Finally the hybrid neuro-fuzzy controller is applied to this map. The performance and the response of the robot are very good and the robot moves smoothly to its destination as shown in Figure 5-25.



DL 6
Dm 6
Dr 6
NewAngle 1.482756
Target X 18 Y 0

Home Position 0 0
Current Location 17.6572064 0.817175

Figure 5-25 Implementing NFC to the Second Map

As a result, the comparison between the three controllers, Forty rules navigator, improved fuzzy navigator system and hybrid (Neuro-Fuzzy) controller are illustrated in the following tables according to the performance for each controller and the CPU time.

The performance of the FLC625 is good compared to the FLC40 since the robot doesn't touch any obstacle and the robot avoids collision with any obstacles as shown in the above cases. But the inference time is much more than the FLC40. Implementing the hybrid neuro-fuzzy controller solves this problem by decreasing the inference time from 1736 μS to 456 μS which increases the response of the controller and improves the performance of the robot.

Table 5.1: Performance Evaluation of FLC40, FLC625, and NFC.

	Performance	CPU Time		
		Fuzzification Processing	Inference Processing	Total CPU Time
FLC 40	Slightly or severely colliding with the obstacles	527 μS	132 μS	659 μS
FLC 625	Avoid collision with the obstacles and smoothly reaches the target	1038 μS	1736 μS	2757 μS
NFC with Single Node	No collision at all with the obstacles & has a good response	1038 μS	456 μS	1494 μS
NFC with 5 Nodes	No collision at all with the obstacles & has a good response	1038 μS	2230 μS	3268 μS

Practically, simulation-using PC doesn't show the differences in the CPU time for the three controllers since the PC is very fast and the response of the hardware is slow. The CPU time for the three controllers will be noticed when using micro controller chip to control the robot motion and download the program in it. In the FLC40 the controller response time will be faster than both controllers, but the performance is limited. On the other hand, the FLC625 work well but with low response, which introduced a deficiency in the robot motion (create a dead point in the robot controller). The neural network if programmed in five chips, the data of the main micro-controller will enter to the five parallel NNT and this will increase the response of the whole controller and improve the performance of the robot motion.

5.4 Concluding remarks

Simulation program is discussed in this chapter where this program is divided into three major categories; simulation, hardware control and Neuro- Fuzzy control. The results of the different cases that are used to test the performance of the three controllers- FLC40, FLC625 and NFC have been discussed in detail. As a result, the hybrid neuro-fuzzy controller has got the advantages of both controller (neural networks and Fuzzy logic), this leads to the reduction of the processing the time of the rules to a lower limit after the neural network training (decision as the inference engine).

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

In today's environments, major advances in fuzzy logic arise primarily in conjunction with knowledge-engineering research. In this area, the power of Hybrid (Neuro- Fuzzy) systems derives primarily from the knowledge of learning process. The primary bottlenecks in the construction of Hybrid (Neuro- Fuzzy) are formulating knowledge to programming, converting the knowledge into effective procedures, iteratively evaluating programs behavior, modifying knowledge and re-implementing the corresponding program code.

A simple fuzzy inference system that models the navigation problem with four input variables, three for the distances and one for the relative direction between the robot position and the target, has been investigated first in this thesis. In this system each distance was treated with two fuzzy values, while the fourth input was classified into five fuzzy values, hence the rule base of this FIS consists of only 40 rules. Due to the low number of sets that describe the three fuzzy distance variables and the small rule base, it was found that the behavior of this system results in total failure when facing certain types of environments. To overcome these limitations, a new fuzzy system was developed with five fuzzy values for each distance input variable. This led to create an inference engine that consists of 625 rules. The behavior of this system when subjected to the same environments compared with the first system showed its superiority in avoiding all the obstacles that the first system failed to avoid. However, the processing time of the new system is found to be 13 times slower than the first system, and this is the main disadvantage in utilizing fuzzy systems with large inference engine.

To solve the inference engine processing time, a neural network is used to replace the classical implementation of the inference engine, a marriage between the fuzzy inference system and neural networks which is classified as a Hybrid (Neuro- Fuzzy) controller. In this controller the NN is trained to do the same action as the inference engine, with the input of the NN is fuzzy linguistic labels and the output is fuzzy too. A single NN was first utilized to replace the FLC625 inference engine. To do the same task, the NN was not capable due to problems in preparing the necessary data for its training. A sample was selected from the training data to train the single NN, the results were not acceptable due to high error rate, a large data was selected to do the same task as above the NN cease to operate. Therefore, five parallel neural networks with 20 input node and one output node were designed for this purpose. The parallel NN were capable to handle the large data training with minimum error. The behavior of the new neuro-fuzzy system was almost similar to that FLC625 when tested under the same environments. In addition to that, the processing time of such a neuro-fuzzy system- if a hardware implementation of the system is presumed- will cut the processing time by 38% less compared to that required for the FLC625.

Recommendations for Future Works

It is recommended that the proposed research for NFC adopted widely. It focuses on a set of fuzzy rules that are considerably different from those addressed in most previous researches in fuzzy controller.

- Deep study in virtual target techniques (switching theory) to improve the NFC to get access out of traps.

- Improvement in motion movement by introducing new variable such as variable speed.

- Utilization of NNT to improve the performance of FLC40 by discovering the faulty rule(s) and modify it, or adding a new rule(s), or by changing membership function and or number of sets of the input variables.

- A real life implementation of autonomous robot system by adding the robot dynamic parameters in the simulation program.

- Utilization of multi purpose flooring characteristics.

- Usage of micro-vision techniques to differentiate the environment.

REFERENCES

Aguirre E. and González A. 2000. Fuzzy behaviors for mobile robot navigation: design, coordination and fusion. *International Journal of Approximate Reasoning*, 25(3): 255-289.

Callan R. 1999. *The Essence of Neural Networks*. Prentice-Hall, Europe.

Cao J., Liao X. and Hall E. 2000. Reactive Navigation for Autonomous Guided Vehicle Using the Neuro-fuzzy Techniques. *Engineering Application of artificial intelligent,* 6(4): 320- 329.

Carpenter G., Grossberg S., Markuzon N., Reynold H. J. and Rosen D. B. 1992. Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analogue Multi-dimensional Maps. *IEEE Tran on Neural Networks,* 3(5): 698-713.

Fausett L. 1994. *Fundamentals of Neural Networks*. Prentice-Hall, Europe.

Hagras H. and Sobh T. 1998. Intelligent Learning and Control of Autonomous Robotic Agents Operating in Unstructured Environments. *IEEE Transaction on Systems, Man, and Cybernetics-Part C: Application and Reviews,* 26(4).

Jang J. S., Sun C.T. and Mizutani E. 1997. *Neuro-Fuzzy and Soft Computing.* Prentice-Hall, Europe.

Lee C. C. 1990. Fuzzy logic in control systems. Fuzzy logic controller- part 1 and part 2. *IEEE Trans. on systems, Man and Cybernetics,* 20 (2): 404-435.

Medsker A. R. 1995. *Hybrid Intelligent Systems.* Kluwer Academic Publishers, Europe.

Mustafa D.1999. *Robot trajectory planning using neural networks*. M.Sc. Thesis, University of Jordan, Jordan.

Nauck D., Klawonn F. and Kruse R. 1997. *Neuro-Fuzzy Systems.* John Wiley and Sons Ltd, New York.

Ng K.C. and Trivedi M. M. 1998. A Neuro-Fuzzy Controller for Mobile Robot Navigation and Multirobot Convoying. *IEEE Transaction on Systems, Man, and Cybernetics-Part B: Application and Reviews*, 28(6).

Pandya A. S. and Macy R. B. 1996. *Pattern Recognition with Neural Networks in C++.* IEEE Press. A CRC Book Published in cooperation with IEEE press, New York.

Proychev T. and Petrov M. 1996. Fuzzy Navigator of Mobile Robots in Unstructured Environment. *IEEE Transaction on Systems, Man, and Cybernetics-Part B: Application and Reviews*, 26(5).

Ross T.J. 1998. *Fuzzy Logic in Engineering Applications.* Mc Graw-Hill Inc, New York.

Samhouri M. 2000. *Genetic algorithm for robot trajectory planning.* M.Sc. Thesis, University of Jordan, Jordan.

Saffiotti A. 1997. Fuzzy Logic in Autonomous Robotics: behavior coordination. *IEEE International Conference of Fuzzy Systems*: 573-578.

Takagi H., Kouda T., Kojima Y. 1990. Neural network designed on approximate reasoning architecture and its application to the pattern recognition. *Proc. of the International Conf. on Fuzzy Logic &Neural Networks, Iizuka, Japan*: 671-674.

Tanaka K. 1996. *An Introduction to Fuzzy Logic for Practical Application.* 1st Edition. Spring-Verlag, New York.

Thongchai S. 1999. Behavior-Based Learning Fuzzy Rules for Mobile Robots. *Robotics and Autonomous systems*, 18: 100- 108.

Thongchai S., Suksakulchai S., Wilkes D. M., and Sarkar N. 1999. Sonar Behavior-Based Fuzzy Control for a Mobile Robot. *IEEE Transaction on Systems, Man, and Cybernetics-Part B: Application and Reviews,* 23(4).

Tsao E. C. K., Bezdek J. C. and Pal N. R.1992. Image Segmentation Using Fuzzy Clustering Networks. *North American Fuzzy Information Processing*: 98-107.

Tsoukalas L. and Uhrig R. 1997. *Fuzzy and Neural Approaches in Engineering.* John Wiley and Sons Inc, New York.

Tsukamoto. 1979. Advances in Fuzzy Set Theory and Applications. *An approach to fuzzy reasoning method, in: Madan Gupta, R. K. Ragade and R. R. Yager, editors. North Holland, Amsterdam*: 137-149.

Uhr L. and Honavar V. 1994. *Introduction in: Artificial Intelligence and Neural Networks — Steps Toward Principled Integration.* Academic Press, Europe.

Wang L. X. 1992. *Training Fuzzy Logic Systems Using Nearest Neighborhood Clustering, Manuscript.* Prentice-Hall, Europe.

Xu W.L., Tso S.K. 1996. Real time Self–Reaction of a Mobile Robot in Unstructured Environment Using Fuzzy Reasoning. *Engineering Application of artificial intelligent,* 9(5): 475- 485.

Xu W.L., Tso S.K. and Fung Y.H. 1998. Fuzzy reactive Control of a Mobile Robot Incorporating a real/ Virtual Target Switching Strategy. *Robotics and Autonomous systems,* 23: 171- 186.

Yung N. H. C. and Ye C. 1999. An Intelligent Mobile Vehicle Navigator Based on Fuzzy Logic and Reinforcement Learning. *IEEE Transaction on Systems, Man, and Cybernetics-Part C: Application and Reviews*, 29(2).

Zadeh L. A. 1996.The roles of fuzzy logic and soft computing in the conception, design and deployment of intelligent systems. *BT Technology,* 14(4): 32-36.

Zadeh L. A. 1965. Fuzzy sets. *Information and Control,* 8: 338-353.

Appendix 1

The Forty- Rules Fuzzy Navigator System

No	Dm	Dl	Dr	Tr	*Sa*
1	FAR	FAR	FAR	RB	***TRB***
2	FAR	FAR	FAR	RS	***TRS***
3	FAR	FAR	FAR	Z	***TZ***
4	FAR	FAR	FAR	LS	***TLS***
5	FAR	FAR	FAR	LB	***TLB***
6	FAR	FAR	NEAR	RB	***TZ***
7	FAR	FAR	NEAR	RS	***TZ***
8	FAR	FAR	NEAR	Z	***TZ***
9	FAR	FAR	NEAR	LS	***TLS***
10	FAR	FAR	NEAR	LB	***TLB***
11	FAR	NEAR	FAR	RB	***TRB***
12	FAR	NEAR	FAR	RS	***TRS***
13	FAR	NEAR	FAR	Z	***TZ***
14	FAR	NEAR	FAR	LS	***TZ***
15	FAR	NEAR	FAR	LB	***TZ***
16	FAR	NEAR	NEAR	RB	***TRB***
17	FAR	NEAR	NEAR	RS	***TRS***
18	FAR	NEAR	NEAR	Z	***TZ***
19	FAR	NEAR	NEAR	LS	***TLS***
20	FAR	NEAR	NEAR	LB	***TLB***
21	NEAR	FAR	FAR	RB	***TRB***
22	NEAR	FAR	FAR	RS	***TRS***
23	NEAR	FAR	FAR	Z	***TRS or TLS***
24	NEAR	FAR	FAR	LS	***TLS***
25	NEAR	FAR	FAR	LB	***TLB***
26	NEAR	FAR	NEAR	RB	***TLS***
27	NEAR	FAR	NEAR	RS	***TLS***
28	NEAR	FAR	NEAR	Z	***TLS***
29	NEAR	FAR	NEAR	LS	***TLB***
30	NEAR	FAR	NEAR	LB	***TLB***
31	NEAR	NEAR	FAR	RB	***TRB***
32	NEAR	NEAR	FAR	RS	***TRB***
33	NEAR	NEAR	FAR	Z	***TRS***
34	NEAR	NEAR	FAR	LS	***TRS***
35	NEAR	NEAR	FAR	LB	***TRS***
36	NEAR	NEAR	NEAR	RB	***TRB***
37	NEAR	NEAR	NEAR	RS	***TRB***
38	NEAR	NEAR	NEAR	Z	***TRB or TLB***
39	NEAR	NEAR	NEAR	LS	***TLB***
40	NEAR	NEAR	NEAR	LB	***TLB***

Appendix 2

Improvement Fuzzy Navigator Rules

No	Dm	Dl	Dr	Tr	*Sa*
1	VL	VL	VL	RB	***TRB***
2	VL	VL	VL	RS	***TRS***
3	VL	VL	VL	Z	***TZ***
4	VL	VL	VL	LS	***TLS***
5	VL	VL	VL	LB	***TLB***
6	VL	VL	L	RB	***TRB***
7	VL	VL	L	RS	***TRS***
8	VL	VL	L	Z	***TZ***
9	VL	VL	L	LS	***TLS***
10	VL	VL	L	LB	***TLB***
11	VL	VL	M	RB	***TRS***
12	VL	VL	M	RS	***TRS***
13	VL	VL	M	Z	***TZ***
14	VL	VL	M	LS	***TLS***
15	VL	VL	M	LB	***TLB***
16	VL	VL	S	RB	***TZ***
17	VL	VL	S	RS	***TZ***
18	VL	VL	S	Z	***TZ***
19	VL	VL	S	LS	***TLS***
20	VL	VL	S	LB	***TLB***
21	VL	VL	VS	RB	***TLS***
22	VL	VL	VS	RS	***TLS***
23	VL	VL	VS	Z	***TLS***
24	VL	VL	VS	LS	***TLS***
25	VL	VL	VS	LB	***TLB***
26	VL	L	VL	RB	***TRB***
27	VL	L	VL	RS	***TRS***
28	VL	L	VL	Z	***TZ***
29	VL	L	VL	LS	***TLS***
30	VL	L	VL	LB	***TLB***
31	VL	L	L	RB	***TRB***
32	VL	L	L	RS	***TRS***
33	VL	L	L	Z	***TZ***
34	VL	L	L	LS	***TLS***
35	VL	L	L	LB	***TLB***
36	VL	L	M	RB	***TRS***
37	VL	L	M	RS	***TRS***
38	VL	L	M	Z	***TZ***
39	VL	L	M	LS	***TLS***
40	VL	L	M	LB	***TLB***
41	VL	L	S	RB	***TZ***
42	VL	L	S	RS	***TZ***
43	VL	L	S	Z	***TZ***
44	VL	L	S	LS	***TLS***
45	VL	L	S	LB	***TLB***
46	VL	L	VS	RB	***TLS***
47	VL	L	VS	RS	***TLS***
48	VL	L	VS	Z	***TLS***
49	VL	L	VS	LS	***TLS***
50	VL	L	VS	LB	***TLB***
51	VL	M	VL	RB	***TRB***

Appendix 2

52	VL	M	VL	RS	***TRS***
53	VL	M	VL	Z	***TZ***
54	VL	M	VL	LS	***TLS***
55	VL	M	VL	LB	***TLS***
56	VL	M	L	RB	***TRB***
57	VL	M	L	RS	***TRS***
58	VL	M	L	Z	***TZ***
59	VL	M	L	LS	***TLS***
60	VL	M	L	LB	***TLS***
61	VL	M	M	RB	***TRS***
62	VL	M	M	RS	***TRS***
63	VL	M	M	Z	***TZ***
64	VL	M	M	LS	***TLS***
65	VL	M	M	LB	***TLS***
66	VL	M	S	RB	***TZ***
67	VL	M	S	RS	***TZ***
68	VL	M	S	Z	***TZ***
69	VL	M	S	LS	***TLS***
70	VL	M	S	LB	***TLS***
71	VL	M	VS	RB	***TLS***
72	VL	M	VS	RS	***TLS***
73	VL	M	VS	Z	***TLS***
74	VL	M	VS	LS	***TLS***
75	VL	M	VS	LB	***TLS***
76	VL	S	VL	RB	***TRB***
77	VL	S	VL	RS	***TRS***
78	VL	S	VL	Z	***TZ***
79	VL	S	VL	LS	***TZ***
80	VL	S	VL	LB	***TZ***
81	VL	S	L	RB	***TRB***
82	VL	S	L	RS	***TRS***
83	VL	S	L	Z	***TZ***
84	VL	S	L	LS	***TZ***
85	VL	S	L	LB	***TZ***
86	VL	S	M	RB	***TRS***
87	VL	S	M	RS	***TRS***
88	VL	S	M	Z	***TZ***
89	VL	S	M	LS	***TZ***
90	VL	S	M	LB	***TZ***
91	VL	S	S	RB	***TZ***
92	VL	S	S	RS	***TZ***
93	VL	S	S	Z	***TZ***
94	VL	S	S	LS	***TZ***
95	VL	S	S	LB	***TZ***
96	VL	S	VS	RB	***TLS***
97	VL	S	VS	RS	***TLS***
98	VL	S	VS	Z	***TLS***
99	VL	S	VS	LS	***TLS***
100	VL	S	VS	LB	***TLS***
101	VL	VS	VL	RB	***TRB***
102	VL	VS	VL	RS	***TRS***
103	VL	VS	VL	Z	***TZ***
104	VL	VS	VL	LS	***TRS***
105	VL	VS	VL	LB	***TRS***
106	VL	VS	L	RB	***TRB***
107	VL	VS	L	RS	***TRS***

Appendix 2

108	VL	VS	L	Z	***TZ***
109	VL	VS	L	LS	***TRS***
110	VL	VS	L	LB	***TRS***
111	VL	VS	M	RB	***TRB***
112	VL	VS	M	RS	***TRS***
113	VL	VS	M	Z	***TZ***
114	VL	VS	M	LS	***TRS***
115	VL	VS	M	LB	***TRS***
116	VL	VS	S	RB	***TRS***
117	VL	VS	S	RS	***TRS***
118	VL	VS	S	Z	***TRS***
119	VL	VS	S	LS	***TRS***
120	VL	VS	S	LB	***TRS***
121	VL	VS	VS	RB	***TZ***
122	VL	VS	VS	RS	***TZ***
123	VL	VS	VS	Z	***TZ***
124	VL	VS	VS	LS	***TZ***
125	VL	VS	VS	LB	***TZ***
126	L	VL	VL	RB	***TRB***
127	L	VL	VL	RS	***TRS***
128	L	VL	VL	Z	***TZ***
129	L	VL	VL	LS	***TLS***
130	L	VL	VL	LB	***TLB***
131	L	VL	L	RB	***TRB***
132	L	VL	L	RS	***TRS***
133	L	VL	L	Z	***TZ***
134	L	VL	L	LS	***TLS***
135	L	VL	L	LB	***TLB***
136	L	VL	M	RB	***TRS***
137	L	VL	M	RS	***TRS***
138	L	VL	M	Z	***TZ***
139	L	VL	M	LS	***TLS***
140	L	VL	M	LB	***TLB***
141	L	VL	S	RB	***TZ***
142	L	VL	S	RS	***TZ***
143	L	VL	S	Z	***TZ***
144	L	VL	S	LS	***TLS***
145	L	VL	S	LB	***TLB***
146	L	VL	VS	RB	***TLS***
147	L	VL	VS	RS	***TLS***
148	L	VL	VS	Z	***TLS***
149	L	VL	VS	LS	***TLS***
150	L	VL	VS	LB	***TLB***
151	L	L	VL	RB	***TRB***
152	L	L	VL	RS	***TRS***
153	L	L	VL	Z	***TZ***
154	L	L	VL	LS	***TLS***
155	L	L	VL	LB	***TLB***
156	L	L	L	RB	***TRB***
157	L	L	L	RS	***TRS***
158	L	L	L	Z	***TZ***
159	L	L	L	LS	***TLS***
160	L	L	L	LB	***TLB***
161	L	L	M	RB	***TRS***
162	L	L	M	RS	***TRS***
163	L	L	M	Z	***TZ***

Appendix 2

164	L	L	M	LS	***TLS***
165	L	L	M	LB	***TLB***
166	L	L	S	RB	***TZ***
167	L	L	S	RS	***TZ***
168	L	L	S	Z	***TZ***
169	L	L	S	LS	***TLS***
170	L	L	S	LB	***TLB***
171	L	L	VS	RB	***TLS***
172	L	L	VS	RS	***TLS***
173	L	L	VS	Z	***TLS***
174	L	L	VS	LS	***TLS***
175	L	L	VS	LB	***TLB***
176	L	M	VL	RB	***TRB***
177	L	M	VL	RS	***TRS***
178	L	M	VL	Z	***TZ***
179	L	M	VL	LS	***TLS***
180	L	M	VL	LB	***TLS***
181	L	M	L	RB	***TRB***
182	L	M	L	RS	***TRS***
183	L	M	L	Z	***TZ***
184	L	M	L	LS	***TLS***
185	L	M	L	LB	***TLS***
186	L	M	M	RB	***TRS***
187	L	M	M	RS	***TRS***
188	L	M	M	Z	***TZ***
189	L	M	M	LS	***TLS***
190	L	M	M	LB	***TLS***
191	L	M	S	RB	***TZ***
192	L	M	S	RS	***TZ***
193	L	M	S	Z	***TZ***
194	L	M	S	LS	***TLS***
195	L	M	S	LB	***TLS***
196	L	M	VS	RB	***TLS***
197	L	M	VS	RS	***TLS***
198	L	M	VS	Z	***TLS***
199	L	M	VS	LS	***TLS***
200	L	M	VS	LB	***TLS***
201	L	S	VL	RB	***TRB***
202	L	S	VL	RS	***TRS***
203	L	S	VL	Z	***TZ***
204	L	S	VL	LS	***TZ***
205	L	S	VL	LB	***TZ***
206	L	S	L	RB	***TRB***
207	L	S	L	RS	***TRS***
208	L	S	L	Z	***TZ***
209	L	S	L	LS	***TZ***
210	L	S	L	LB	***TZ***
211	L	S	M	RB	***TRS***
212	L	S	M	RS	***TRS***
213	L	S	M	Z	***TZ***
214	L	S	M	LS	***TZ***
215	L	S	M	LB	***TZ***
216	L	S	S	RB	***TZ***
217	L	S	S	RS	***TZ***
218	L	S	S	Z	***TZ***
219	L	S	S	LS	***TZ***

Appendix 2

220	L	S	S	LB	***TZ***
221	L	S	VS	RB	***TLS***
222	L	S	VS	RS	***TLS***
223	L	S	VS	Z	***TLS***
224	L	S	VS	LS	***TLS***
225	L	S	VS	LB	***TLS***
226	L	VS	VL	RB	***TRB***
227	L	VS	VL	RS	***TRS***
228	L	VS	VL	Z	***TZ***
229	L	VS	VL	LS	***TRS***
230	L	VS	VL	LB	***TRS***
231	L	VS	L	RB	***TRB***
232	L	VS	L	RS	***TRS***
233	L	VS	L	Z	***TZ***
234	L	VS	L	LS	***TRS***
235	L	VS	L	LB	***TRS***
236	L	VS	M	RB	***TRB***
237	L	VS	M	RS	***TRS***
238	L	VS	M	Z	***TZ***
239	L	VS	M	LS	***TRS***
240	L	VS	M	LB	***TRS***
241	L	VS	S	RB	***TRS***
242	L	VS	S	RS	***TRS***
243	L	VS	S	Z	***TRS***
244	L	VS	S	LS	***TRS***
245	L	VS	S	LB	***TRS***
246	L	VS	VS	RB	***TZ***
247	L	VS	VS	RS	***TZ***
248	L	VS	VS	Z	***TZ***
249	L	VS	VS	LS	***TZ***
250	L	VS	VS	LB	***TZ***
251	M	VL	VL	RB	***TRB***
252	M	VL	VL	RS	***TRS***
253	M	VL	VL	Z	***TRS or TLS***
254	M	VL	VL	LS	***TLS***
255	M	VL	VL	LB	***TLB***
256	M	VL	L	RB	***TRB***
257	M	VL	L	RS	***TRS***
258	M	VL	L	Z	***TRS or TLS***
259	M	VL	L	LS	***TLS***
260	M	VL	L	LB	***TLB***
261	M	VL	M	RB	***TRS***
262	M	VL	M	RS	***TRS***
263	M	VL	M	Z	***TLS***
264	M	VL	M	LS	***TLS***
265	M	VL	M	LB	***TLB***
266	M	VL	S	RB	***TZ***
267	M	VL	S	RS	***TZ***
268	M	VL	S	Z	***TLS***
269	M	VL	S	LS	***TLS***
270	M	VL	S	LB	***TLB***
271	M	VL	VS	RB	***TLS***
272	M	VL	VS	RS	***TLS***
273	M	VL	VS	Z	***TLS***
274	M	VL	VS	LS	***TLS***
275	M	VL	VS	LB	***TLB***

Appendix 2

276	M	L	VL	RB	***TRB***
277	M	L	VL	RS	***TRS***
278	M	L	VL	Z	***TRS***
279	M	L	VL	LS	***TLS***
280	M	L	VL	LB	***TLB***
281	M	L	L	RB	***TRB***
282	M	L	L	RS	***TRS***
283	M	L	L	Z	***TRS or TLS***
284	M	L	L	LS	***TLS***
285	M	L	L	LB	***TLB***
286	M	L	M	RB	***TRS***
287	M	L	M	RS	***TRS***
288	M	L	M	Z	***TLS***
289	M	L	M	LS	***TLS***
290	M	L	M	LB	***TLB***
291	M	L	S	RB	***TZ***
292	M	L	S	RS	***TZ***
293	M	L	S	Z	***TLS***
294	M	L	S	LS	***TLS***
295	M	L	S	LB	***TLB***
296	M	L	VS	RB	***TLS***
297	M	L	VS	RS	***TLS***
298	M	L	VS	Z	***TLS***
299	M	L	VS	LS	***TLS***
300	M	L	VS	LB	***TLB***
301	M	M	VL	RB	***TRB***
302	M	M	VL	RS	***TRS***
303	M	M	VL	Z	***TRS***
304	M	M	VL	LS	***TLS***
305	M	M	VL	LB	***TLS***
306	M	M	L	RB	***TRB***
307	M	M	L	RS	***TRS***
308	M	M	L	Z	***TRS***
309	M	M	L	LS	***TLS***
310	M	M	L	LB	***TLS***
311	M	M	M	RB	***TRS***
312	M	M	M	RS	***TRS***
313	M	M	M	Z	***TRS or TLS***
314	M	M	M	LS	***TLS***
315	M	M	M	LB	***TLS***
316	M	M	S	RB	***TZ***
317	M	M	S	RS	***TZ***
318	M	M	S	Z	***TLS***
319	M	M	S	LS	***TLS***
320	M	M	S	LB	***TLS***
321	M	M	VS	RB	***TLS***
322	M	M	VS	RS	***TLS***
323	M	M	VS	Z	***TLS***
324	M	M	VS	LS	***TLS***
325	M	M	VS	LB	***TLS***
326	M	S	VL	RB	***TRB***
327	M	S	VL	RS	***TRS***
328	M	S	VL	Z	***TRS***
329	M	S	VL	LS	***TZ***
330	M	S	VL	LB	***TZ***
331	M	S	L	RB	***TRB***

Appendix 2

332	M	S	L	RS	***TRS***
333	M	S	L	Z	***TRS***
334	M	S	L	LS	***TZ***
335	M	S	L	LB	***TZ***
336	M	S	M	RB	***TRS***
337	M	S	M	RS	***TRS***
338	M	S	M	Z	***TRS***
339	M	S	M	LS	***TZ***
340	M	S	M	LB	***TZ***
341	M	S	S	RB	***TZ***
342	M	S	S	RS	***TZ***
343	M	S	S	Z	***TZ***
344	M	S	S	LS	***TZ***
345	M	S	S	LB	***TZ***
346	M	S	VS	RB	***TLS***
347	M	S	VS	RS	***TLS***
348	M	S	VS	Z	***TLS***
349	M	S	VS	LS	***TLS***
350	M	S	VS	LB	***TLS***
351	M	VS	VL	RB	***TRB***
352	M	VS	VL	RS	***TRS***
353	M	VS	VL	Z	***TRS***
354	M	VS	VL	LS	***TRS***
355	M	VS	VL	LB	***TRS***
356	M	VS	L	RB	***TRB***
357	M	VS	L	RS	***TRS***
358	M	VS	L	Z	***TRS***
359	M	VS	L	LS	***TRS***
360	M	VS	L	LB	***TRS***
361	M	VS	M	RB	***TRB***
362	M	VS	M	RS	***TRS***
363	M	VS	M	Z	***TRS***
364	M	VS	M	LS	***TRS***
365	M	VS	M	LB	***TRS***
366	M	VS	S	RB	***TRS***
367	M	VS	S	RS	***TRS***
368	M	VS	S	Z	***TRS***
369	M	VS	S	LS	***TRS***
370	M	VS	S	LB	***TRS***
371	M	VS	VS	RB	***TZ***
372	M	VS	VS	RS	***TZ***
373	M	VS	VS	Z	***TZ***
374	M	VS	VS	LS	***TZ***
375	M	VS	VS	LB	***TZ***
376	S	VL	VL	RB	***TRB***
377	S	VL	VL	RS	***TRS***
378	S	VL	VL	Z	***TRS or TLS***
379	S	VL	VL	LS	***TLS***
380	S	VL	VL	LB	***TLB***
381	S	VL	L	RB	***TRB***
382	S	VL	L	RS	***TRS***
383	S	VL	L	Z	***TLS***
384	S	VL	L	LS	***TLS***
385	S	VL	L	LB	***TLB***
386	S	VL	M	RB	***TRS***
387	S	VL	M	RS	***TRS***

Appendix 2

388	S	VL	M	Z	***TLS***
389	S	VL	M	LS	***TLS***
390	S	VL	M	LB	***TLB***
391	S	VL	S	RB	***TZ***
392	S	VL	S	RS	***TZ***
393	S	VL	S	Z	***TLS***
394	S	VL	S	LS	***TLS***
395	S	VL	S	LB	***TLB***
396	S	VL	VS	RB	***TLS***
397	S	VL	VS	RS	***TLS***
398	S	VL	VS	Z	***TLS***
399	S	VL	VS	LS	***TLS***
400	S	VL	VS	LB	***TLB***
401	S	L	VL	RB	***TRB***
402	S	L	VL	RS	***TRS***
403	S	L	VL	Z	***TRS or TLS***
404	S	L	VL	LS	***TLS***
405	S	L	VL	LB	***TLB***
406	S	L	L	RB	***TRB***
407	S	L	L	RS	***TRS***
408	S	L	L	Z	***TRS or TLS***
409	S	L	L	LS	***TLS***
410	S	L	L	LB	***TLB***
411	S	L	M	RB	***TRS***
412	S	L	M	RS	***TRS***
413	S	L	M	Z	***TLS***
414	S	L	M	LS	***TLS***
415	S	L	M	LB	***TLB***
416	S	L	S	RB	***TZ***
417	S	L	S	RS	***TZ***
418	S	L	S	Z	***TLB***
419	S	L	S	LS	***TLS***
420	S	L	S	LB	***TLB***
421	S	L	VS	RB	***TLS***
422	S	L	VS	RS	***TLS***
423	S	L	VS	Z	***TLB***
424	S	L	VS	LS	***TLS***
425	S	L	VS	LB	***TLB***
426	S	M	VL	RB	***TRB***
427	S	M	VL	RS	***TRS***
428	S	M	VL	Z	***TRS or TRB***
429	S	M	VL	LS	***TLS***
430	S	M	VL	LB	***TLS***
431	S	M	L	RB	***TRB***
432	S	M	L	RS	***TRS***
433	S	M	L	Z	***TRB***
434	S	M	L	LS	***TLS***
435	S	M	L	LB	***TLS***
436	S	M	M	RB	***TRS***
437	S	M	M	RS	***TRS***
438	S	M	M	Z	***TRB or TLB***
439	S	M	M	LS	***TLS***
440	S	M	M	LB	***TLS***
441	S	M	S	RB	***TZ***
442	S	M	S	RS	***TZ***
443	S	M	S	Z	***TLB***

Appendix 2

444	S	M	S	LS	***TLS***
445	S	M	S	LB	***TLS***
446	S	M	VS	RB	***TLS***
447	S	M	VS	RS	***TLS***
448	S	M	VS	Z	***TLB***
449	S	M	VS	LS	***TLS***
450	S	M	VS	LB	***TLS***
451	S	S	VL	RB	***TRB***
452	S	S	VL	RS	***TRS***
453	S	S	VL	Z	***TRB***
454	S	S	VL	LS	***TZ***
455	S	S	VL	LB	***TZ***
456	S	S	L	RB	***TRB***
457	S	S	L	RS	***TRS***
458	S	S	L	Z	***TRB***
459	S	S	L	LS	***TZ***
460	S	S	L	LB	***TZ***
461	S	S	M	RB	***TRS***
462	S	S	M	RS	***TRS***
463	S	S	M	Z	***TRB***
464	S	S	M	LS	***TZ***
465	S	S	M	LB	***TZ***
466	S	S	S	RB	***TZ***
467	S	S	S	RS	***TZ***
468	S	S	S	Z	***TRB or TLB***
469	S	S	S	LS	***TZ***
470	S	S	S	LB	***TZ***
471	S	S	VS	RB	***TLS***
472	S	S	VS	RS	***TLS***
473	S	S	VS	Z	***TRS***
474	S	S	VS	LS	***TLS***
475	S	S	VS	LB	***TLS***
476	S	VS	VL	RB	***TRB***
477	S	VS	VL	RS	***TRS***
478	S	VS	VL	Z	***TRB***
479	S	VS	VL	LS	***TRS***
480	S	VS	VL	LB	***TRS***
481	S	VS	L	RB	***TRB***
482	S	VS	L	RS	***TRS***
483	S	VS	L	Z	***TRB***
484	S	VS	L	LS	***TRS***
485	S	VS	L	LB	***TRS***
486	S	VS	M	RB	***TRB***
487	S	VS	M	RS	***TRS***
488	S	VS	M	Z	***TRB***
489	S	VS	M	LS	***TRS***
490	S	VS	M	LB	***TRS***
491	S	VS	S	RB	***TRS***
492	S	VS	S	RS	***TRS***
493	S	VS	S	Z	***TRS***
494	S	VS	S	LS	***TRS***
495	S	VS	S	LB	***TRS***
496	S	VS	VS	RB	***TZ***
497	S	VS	VS	RS	***TZ***
498	S	VS	VS	Z	***TZ***
499	S	VS	VS	LS	***TZ***

__Appendix 2__

500	S	VS	VS	LB	***TZ***
501	VS	VL	VL	RB	***TRB***
502	VS	VL	VL	RS	***TRS***
503	VS	VL	VL	Z	***TRB or TLB***
504	VS	VL	VL	LS	***TLS***
505	VS	VL	VL	LB	***TLB***
506	VS	VL	L	RB	***TRB***
507	VS	VL	L	RS	***TRS***
508	VS	VL	L	Z	***TRB or TLB***
509	VS	VL	L	LS	***TLS***
510	VS	VL	L	LB	***TLB***
511	VS	VL	M	RB	***TRS***
512	VS	VL	M	RS	***TRS***
513	VS	VL	M	Z	***TLB***
514	VS	VL	M	LS	***TLS***
515	VS	VL	M	LB	***TLB***
516	VS	VL	S	RB	***TZ***
517	VS	VL	S	RS	***TZ***
518	VS	VL	S	Z	***TLB***
519	VS	VL	S	LS	***TLS***
520	VS	VL	S	LB	***TLB***
521	VS	VL	VS	RB	***TLS***
522	VS	VL	VS	RS	***TLS***
523	VS	VL	VS	Z	***TLB***
524	VS	VL	VS	LS	***TLS***
525	VS	VL	VS	LB	***TLB***
526	VS	L	VL	RB	***TRB***
527	VS	L	VL	RS	***TRS***
528	VS	L	VL	Z	***TRB or TLB***
529	VS	L	VL	LS	***TLS***
530	VS	L	VL	LB	***TLB***
531	VS	L	L	RB	***TRB***
532	VS	L	L	RS	***TRS***
533	VS	L	L	Z	***TRB or TLB***
534	VS	L	L	LS	***TLS***
535	VS	L	L	LB	***TLB***
536	VS	L	M	RB	***TRS***
537	VS	L	M	RS	***TRS***
538	VS	L	M	Z	***TLB***
539	VS	L	M	LS	***TLS***
540	VS	L	M	LB	***TLB***
541	VS	L	S	RB	***TZ***
542	VS	L	S	RS	***TZ***
543	VS	L	S	Z	***TLB***
544	VS	L	S	LS	***TLS***
545	VS	L	S	LB	***TLB***
546	VS	L	VS	RB	***TLS***
547	VS	L	VS	RS	***TLS***
548	VS	L	VS	Z	***TLB***
549	VS	L	VS	LS	***TLS***
550	VS	L	VS	LB	***TLB***
551	VS	M	VL	RB	***TRB***
552	VS	M	VL	RS	***TRS***
553	VS	M	VL	Z	***TRB***
554	VS	M	VL	LS	***TLS***
555	VS	M	VL	LB	***TLS***

Appendix 2

556	VS	M	L	RB	***TRB***
557	VS	M	L	RS	***TRS***
558	VS	M	L	Z	***TRB***
559	VS	M	L	LS	***TLS***
560	VS	M	L	LB	***TLS***
561	VS	M	M	RB	***TRS***
562	VS	M	M	RS	***TRS***
563	VS	M	M	Z	***TRB or TLB***
564	VS	M	M	LS	***TLS***
565	VS	M	M	LB	***TLS***
566	VS	M	S	RB	***TZ***
567	VS	M	S	RS	***TZ***
568	VS	M	S	Z	***TLB***
569	VS	M	S	LS	***TLS***
570	VS	M	S	LB	***TLS***
571	VS	M	VS	RB	***TLS***
572	VS	M	VS	RS	***TLS***
573	VS	M	VS	Z	***TLB***
574	VS	M	VS	LS	***TLS***
575	VS	M	VS	LB	***TLS***
576	VS	S	VL	RB	***TRB***
577	VS	S	VL	RS	***TRS***
578	VS	S	VL	Z	***TRB***
579	VS	S	VL	LS	***TZ***
580	VS	S	VL	LB	***TZ***
581	VS	S	L	RB	***TRB***
582	VS	S	L	RS	***TRS***
583	VS	S	L	Z	***TRB***
584	VS	S	L	LS	***TZ***
585	VS	S	L	LB	***TZ***
586	VS	S	M	RB	***TRS***
587	VS	S	M	RS	***TRS***
588	VS	S	M	Z	***TRB***
589	VS	S	M	LS	***TZ***
590	VS	S	M	LB	***TZ***
591	VS	S	S	RB	***TZ***
592	VS	S	S	RS	***TZ***
593	VS	S	S	Z	***TRB or TLB***
594	VS	S	S	LS	***TZ***
595	VS	S	S	LB	***TZ***
596	VS	S	VS	RB	***TLS***
597	VS	S	VS	RS	***TLS***
598	VS	S	VS	Z	***TLS***
599	VS	S	VS	LS	***TLS***
600	VS	S	VS	LB	***TLS***
601	VS	VS	VL	RB	***TRB***
602	VS	VS	VL	RS	***TRS***
603	VS	VS	VL	Z	***TRB***
604	VS	VS	VL	LS	***TRS***
605	VS	VS	VL	LB	***TRS***
606	VS	VS	L	RB	***TRB***
607	VS	VS	L	RS	***TRS***
608	VS	VS	L	Z	***TRB***
609	VS	VS	L	LS	***TRS***
610	VS	VS	L	LB	***TRS***
611	VS	VS	M	RB	***TRB***

Appendix 2

612	VS	VS	M	RS	***TRS***
613	VS	VS	M	Z	***TRB***
614	VS	VS	M	LS	***TRS***
615	VS	VS	M	LB	***TRS***
616	VS	VS	S	RB	***TRS***
617	VS	VS	S	RS	***TRS***
618	VS	VS	S	Z	***TRB***
619	VS	VS	S	LS	***TRS***
620	VS	VS	S	LB	***TRS***
621	VS	VS	VS	RB	***TZ***
622	VS	VS	VS	RS	***TZ***
623	VS	VS	VS	Z	***TZ***
624	VS	VS	VS	LS	***TZ***
625	VS	VS	VS	LB	***TZ***

Where:
THE DISTANCES:

VL: VERY LARGE Distance	***L: LARGE Distance***
M: MEDUIM Distance	***S: SMALL Distance***
VS: VERY SMALL Distance	

THE ANGLE ORIENTATION

RB: RIGHT BIG	***RS: RIGHT SMALL***	***Z: ZERO***
LS: LEFT SMALL	***LB: LEFT BIG***	

TURNING ANGLE

TRB: TURN RIGHT BIG	***TRS: TURN RIGHT SMALL***
TZ: TURN ZERO	***TLS: TURN LEFT SMALL***
TLB: TURN LEFT BIG	

متحكم ذكي لربوت متحرك باستخدام أساليب الشبكات العصبية والمنطق المحير

إعداد
أيمن عبد الفتاح أبو بكر

إشراف
د. خلدون طهبوب

المشرف المشارك
د. مناف سالم نجم الدين

الملخص

الــربوت الآلــي المــتحرك هو الربوت القادر على التحرك من نقطة البداية الى نقطة النهاية (الهدف) دون الاصطدام بأي من العوائق التي تواجهه في مسيره. لذلك انتشر استخدام هذا النوع من الربوتات في المصانع والنواحــي العسكرية والمجالات الأخرى. والأبحاث في هذا المجال وضعت جل اهتمامها في التحكم بحركة الربوت واستكشافه للعوائق والابتعاد عنها والوصول الى الهدف سالما.

هــذه الرســالة عنيت بحل مشكلة الربوت الآلي المتحرك وكان هدفها الأساسي هو تحريك الربوت الآلي من نقطـة الـبداية وحـتى نقطـة النهاية (الهدف) في بيئة غير معروفة بالنسبة له متحاشيا جميع العوائق التي تواجهه. وبالتالي تم تحقيق هذا الهدف عن طريق إستخدام أساليب الشبكات العصبية والمنطق المحير.

فــي بداية هذه الرسالة تم تصميم متحكم مبني على المنطق المحير والذي اعتمد على عدد كبير من القوانين. وهــذا بــدوره عمــل على تحسين عمل الربوت بشكل كبير ولكن المشكلة كانت في كثرة القوانين. لذلك تم اســتبدال مصــدر القوانيــن في المنطق المحير بالشبكات العصبية وذلك للتقليل من العمليات التي كان عليها سابقا.

لقد تم تصميم برنامج يقوم برسم حركة الربوت وذلك لقياس أداء الربوت في كل متحكم مستخدم. وفي النهاية تــم تطبيق هذا المتحكم على ربوت تم صناعته بدعم من عمادة البحث العلمي في الجامعة الاردنية. وقد أثبت أداء هذا الربوت نجاح المتحكم الأخير بالسيطرة والتحكم بالربوت في بيئه غير معلومة له.