A BEHAVIOR BASED ROBOT CONTROL SYSTEM USING NEURO-FUZZY APPROACH

A THESIS SUBMITED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES OF

THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

DEMET ÖĞÜT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF COMPUTER ENGINEERING

DECEMBER 2003

Approval of the Graduate School of Natural and Applied Sciences,

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Ayşe Kiper
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Ferda Nur Alpaslan
Supervisor

Examining Committee Members

Prof. Dr. Uğur Halıcı

Prof. Dr. Muslim Bozyiğit

Assoc. Prof. Dr. Ferda Nur Alpaslan

Assoc. Prof. Dr. Nihan Çiçekli

Assist. Prof. Dr. Bilge Say

ABSTRACT

A BEHAVIOR BASED ROBOT CONTROL SYSTEM USING

NEURO-FUZZY APPROACH

Öğüt, Demet

M.Sc., Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Ferda Nur Alpaslan

December 2003, pages 58

In autonomous navigation of mobile robots the dynamic environment is a source of problems. Because it is not possible to model all the possible conditions, the key point in the robot control is to design a system that is adaptable to different conditions and robust in dynamic environments.

This study presents a reactive control system for a Khepera robot with the ability to navigate in a dynamic environment for reaching goal objects. The main motivation of this research is to design a robot control, which is robust to sensor errors and sudden changes and adaptable to different environments and conditions. Behavior based approach is used with taking the advantage of fuzzy reasoning in design. Experiments are made on Webots, which is a simulation environment for Khepera robot.

Keywords : Behavior-cased robotics, neuro-fuzzy system, reactive control, autonomous navigation.

ÖZ

BULANIK MANTIK KULLANAN DAVRANIŞ-TABANLI

BİR ROBOT KONTROL SİSTEMİ

Öğüt, Demet

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi : Doc. Dr. Ferda Nur Alpaslan

December 2003, 58 sayfa

Devingen robotların özerk gezinimlerinde dinamik ortam bir sorun kaynağıdır. Tüm ihtimal durumların modellenmesi mümkün olmadığı için, robot kontrolünde anahtar nokta, farklı durumlara uyarlanabilir ve dinamik ortamlarda dirençli sistemler tasarlamaktır.

Bu çalışma, hedef noktalara ulaşmak için dinamik ortamlarda özerk gezinim yapma kabiliyetine sahip bir Khepera robotu için tepkisel kontrol sistemi sunuyor. Bu araştırmanın temel içgüdüsü, sensor hatalarına ve ani değişikliklere karşı dayanıklı ve değişik ortam ve durumlara uyarlanabilir bir robot kontrol sistemi tasarlamaktır. Tasarımda bulanık muhakemeden faydalanan davranış-tabanlı bir yaklaşım kullanılmıştır. Deneyler Khepera robotu için simülasyon ortamı olan Webots üzerinde yapılmıştır.

Anahtar Kelimeler : Davranış tabanlı robotbilim, nöral-bulanık system, tepkisel kontrol, özerk gezinim

To my family and my love

ACKNOWLEDGMENTS

I appreciate to Ass. Prof. Dr. Ferda Nur Alpaslan for her guidance and insight throughout my research. Thanks for Berrin Balcı for her valuable discussions and assistance. And special thanks to Hüsnü Demir for his encouragements and endurance with me to all difficulties.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Over the past few years, the research in autonomous mobile robots gained an extensive interest. This is due to the wish to replace the humans with robots in dangerous tasks, and to use the robots in classical everyday tasks and in industry.

Many methods for robot control have been developed which can be generally grouped into two categories: deliberative and reactive.

In deliberative approach global planning method is used in a completely known environment. These methods build the paths for reaching the target without any collision. A global optimum solution can be achieved in this approach. However, this scheme has well-known drawbacks. Exact model of the world is needed which is very difficult and modifications in this environment after modeling cannot be handled.

In reactive approach model of the world is not needed, actions are determined according to information gathered from sensors. The robot has to react to its sensor data by a set of stimuli-response mechanism. The drawback of these systems is limited and uncertain sensor data because of the limited range, poor observation conditions, and environmental effects.

A set of qualitative reasoning methods has been proposed to overcome the uncertainties in the systems. These approaches try to incorporate the uncertainty to gathering and reasoning steps, instead of trying to identify them with numerical methods. And these methods try to capture the human behavior. Mostly used qualitative reasoning method is fuzzy inference.

The fuzzy logic systems are inspired from human reasoning, which is based on perception. Fuzzy logic provides a methodology for representing human expert knowledge and perception-based actions without needing analytical model of the system.

Neuro-fuzzy systems add the advantages of fuzzy reasoning to neural networks, which learn fast without needing symbolic representation of the system.

Behavior based systems tries to model the reactive abilities of humans, animals, insects etc. to sensed environment. In behavior based approach goals are achieved by subdividing the overall task into smaller independent behaviors that focus on execution of specific tasks. For example a behavior can focus on traversing from start to target place, while another behavior focus on obstacle avoidance.

In this study, a reactive behavior based robot control system is modeled and implemented. The goal of the robot is to navigate and reach the goal points with avoiding obstacles. The system uses the neuro-fuzzy approach with online learning. No supervision and global world modeling is used.

Adaptive Network Fuzzy Inference System (ANFIS) is applied for a mobile robotics problem with Tsukamoto inference system. Each rule output is defined as a crisp value and overall output of system is calculated by weighted average of each rule’s output. Since each rule gives a crisp output and the aggregation is made by weighted average, this model avoids the time-consuming process of defuzzification.

The thesis is organized as follows. Chapter 2 introduces behavior based systems and fuzzy control. Chapter 3 gives information about the adaptive networks and specifically introduces ANFIS. Chapter 4 covers the design of robot control system. Chapter 5 concludes this thesis and indicates future work.

CHAPTER 2

ROBOT CONTROL

2.1 Behavior Based Robotics

It is obvious that a robot must have some form of intelligence to navigate in real world for a purpose without crashing into things. Intelligence can be defined as capacity to acquire and apply knowledge. The robot must be able to acquire information from its sensors, and use it to control the movement in an effective manner. It may also use a map, keep track of its position, and decide where it should go in what order, all of which requires usage of acquired information.

The problems of a mobile robot may seem same as classical AI programs at first glance. For a given goal, the robot must find a solution based on available paths. In top-down design methodology the operation of a robot could be described as a cycle of world model construction, path generation, actuator output, sensor readings, and world model revision. The control of this approach is given in Figure 2-1.

Traditional sense-model-plan-act approach work well for industrial robots, for which the environment they work and the task they accomplish is well defined. Most established robotics and artificial intelligence techniques are successful because of the finite, knowledge-based nature of the applications to which they have been applied. If the world model is complete enough for the computer, it can effectively consider relevant options and the consequences of its decisions.

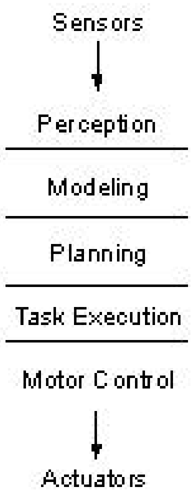

Figure 2-1 Traditional sense-model-plan-act approach (adopted from [1])

But unfortunately the environment of a mobile robot cannot be described as a complete world model. This is the difference of mobile robots from the stationary robots in the industry. And also the industrial robots’ work place is designed to limit variation, and structured so that any variation can easily be sensed and compensated for, but a mobile robot must deal with any number of unexpected circumstances. The robot must be able to sense and respond to anything that could affect its ability to accomplish its mission.

Mobile robots are different from ordinary computer applications. Programmers perform the functions of selecting relevant information from the world, translating it into symbolic data for entry, and interpreting the results. The function that the computer performs is actually simple by comparison, but requires so many tedious steps that it is better suited to a machine than a human being. The question of appropriate representation has been a more difficult problem for the AI community to deal with than the strategies associated with decision making. For this reason, the most successful and widespread AI applications have concerned with finite state spaces [2].

There are difficulties with using the symbolic system approach to mobile robots. First, robot needs the complete representation of the world to plan and make decisions and this world model must be detailed enough for the needs of the robot. But accurate construction of this model is extremely difficult. Second, maintaining the model and making plans according to this model is very time consuming. This usually makes the robot's performance extremely slow. Third, because of the limitations of the model, sensors, and planning algorithms, there is always a risk in the real world, which is not possible to incorporate into the model and can result the failure of the system.

The traditional AI methodology has two important characteristics [3]: the ability to represent hierarchical structure by abstraction and the use of strong knowledge that employs explicit symbolic representational assertions about the world. AI point of view for robotics applications was in the following way: the important thing is knowledge and its representation and robotics is not an exception. Behavior based robotics systems reacted against these idea and the sensing and acting in an environment dominated the interest on knowledge representation and planning [4].

Successful navigation of small animals, like insects in the world, show that deliberative reasoning has small effect on reliable autonomous behavior and opposes the idea of human like reasoning can solve the problems of mobile robotics. New trends focus on decentralized intelligence and reactive control schemes, which is inspired by observation of simple animals’ behaviors.

Simple animals make decisions to survive like finding food sources, running away from predators, etc. although they have limited brains. Because an insect does not deliberate the decisions then it experiences some primitive forms of fear and hunger and have a brain capable of acting upon these motivations, than making rational choices. Motivations are more important than logic to autonomous agents.

While biologists have been developing models and simulations of animal behavior, roboticists have been searching for the means to distribute the control tasks faced by autonomous machines, and make their performance more reliable. The fusion of these disciplines was inevitable. In an experiment aimed toward both the AI community and biologists, Beer developed a computer simulation of an artificial cockroach [4,5]. His control scheme used a model of living neurons as a building block for simple behaviors. Despite the simplicity of its components, the simulated insect could coordinate six legs for walking, avoid obstacles, and search for food in its virtual world.

Parallel architectures improve the performance, reliability, and scalability of autonomous machines. Minsky [6] describes a mind as being composed of individual entities called agents. Each agent performs a small part of a task and larger, more complex functions are achieved by groups of agents with coordinated and concentrated interaction between them. In behavior-based robotics individual behaviors can be viewed as agents.

Brooks [1] introduced the layered control approach to the robotics community. Rather than decomposing the autonomous control problem into separate layers in a sense/model/plan/act scheme, Brooks proposed an architecture where each layer has direct effect on the action. Behaviors are represented as layers in subsumption architecture and multiple behaviors are coordinated in parallel. This architecture allows rapid and robust interaction with the environment, and offers incremental development of capabilities. Brooks' subsumption architecture is implemented on a wide variety of mobile robots. In the subsumption architecture, high-priority behaviors such as collision avoidance subsume other behaviors, such as goal seeking. The arbitration scheme is based on hierarchical switching. Brooks architecture is shown in the Figure 2-2:

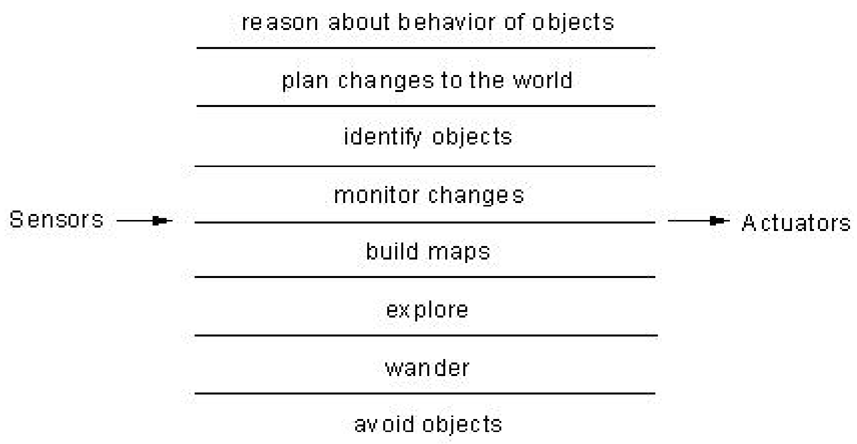

Figure 2-2 Layered Control Architecture (adopted from [1])

Brooks' work has inspired a variety of behavior-based control schemes. Arkin[7] developed the schema theory. Schemas are the basic primitives of the behavior, and they form a complex action with acting in a distributed, parallel manner. Output of each schema is a vector that defines the way that the robot must move, which is then combined by weighted summation. In each motor schema a perceptual schema is embedded which provides the environmental information specific to that behavior. Figure 2-3 presents the perception-action schema relations:

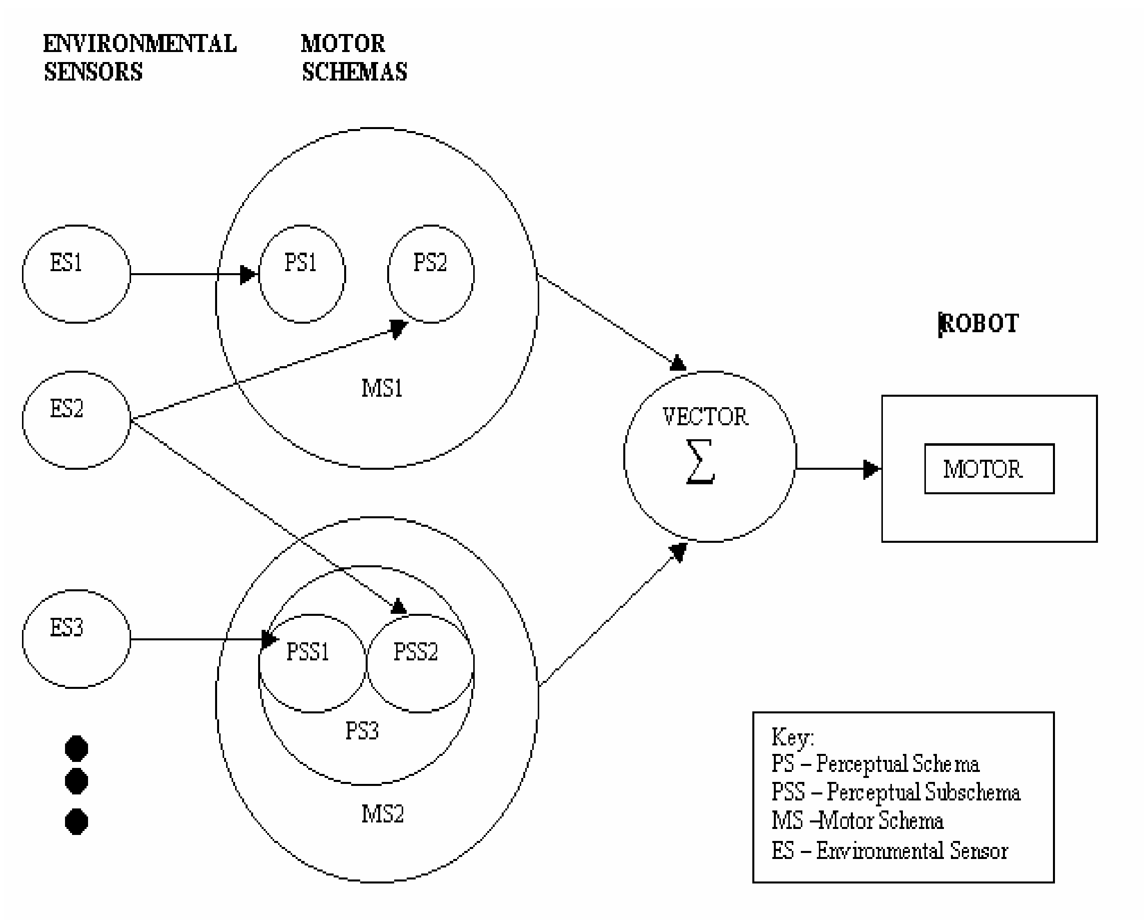

Figure 2-3 Perception-action schema relations (adopted from [8])

Minsky, Brooks, and Arkin proposed that a complex problem could be solved by breaking down it into multiple small specialized agents [2]. Although they have different ideas for individual agents, in general, an agent defines a mapping between inputs and outputs, which can be connected to sensors and actuators. Each agent performs its function continuously and operates in parallel with the others.

Behavior based approaches are evaluated similar in terms of their parallelism, modularity, ease of development, flexibility and performance. These architectures are different in the granularity of behavioral decomposition, coordination of behaviors and response encoding technique [8].

2.2 Fuzzy Control in Mobile Robots

The goal of a mobile robot is moving purposefully in unmodified environments i.e. not specific environments designed for robots, without human intervention. Robot needs the model of the world as it navigates. Because the environment is dynamic, model of the world cannot be obtained at the start of execution of the robot. One way to overcome this problem is, modeling the environment in runtime by using the sensors. But the information that can be gathered from the sensors is not enough because of the limited range, poor observation conditions, and environmental effects.

These problems has been tried to be solved by the careful design of robot mechanics and sensors [9]. But this increases costs, reduces robots autonomy and cannot be applied to all environments. The main challenge in current robotics is to build robust control programs that can perform complex tasks in dynamic environments.

Qualitative nature of fuzzy logic makes it a useful tool for dealing with these problems, which sourced from uncertainty. Fuzzy set theory is developed by Lotfi Zadeh [10] in the 1960's but it has become a popular tool for control applications in recent years. In fuzzy set theory the information is classified into sets that do not have crisp boundaries. And control functions are described in linguistic terms such as "large", "small", "fast," etc. One of the main benefits of fuzzy systems is that an expert human knowledge about the control of the system can be incorporated, without using the mathematical description of the problem.

Fuzzy logic is different from predicate logic. In predicate logic assertions about the world are true or false which are referred as crisp values and have exact meaning. In fuzzy logic inputs have values according to how much they belong to a fuzzy set. Fuzzy sets are defined with membership functions, which measure the degree of similarity of an instance to the set as a numerical value.

Fuzzy if-then rules or fuzzy conditional statements are expressions of the form IF A THEN B, where A and B are labels of fuzzy sets characterized by appropriate membership functions. An example that describes a simple fact is:

If pressure is high, then volume is small.

where pressure and volume are linguistic variables, high and small are linguistic values or labels that are characterized by membership functions.

Another form of fuzzy if-then rule, proposed by Takagi and Sugeno [11], has fuzzy sets involved only in the premise part. By using Takagi and Sugeno's fuzzy if-then rule, we can describe the resistant force on a moving object as follows:

If velocity is high, then force = $velocity^2$ * k

where, again, high in the premise part is a linguistic label characterized by an appropriate membership function. However, the consequent part is described by a non-fuzzy equation of the input variable, velocity.

Both types of fuzzy if-then rules have been used extensively in modeling and control. With the use of linguistic labels and membership functions, a fuzzy if-then rule can easily capture the knowledge of a human expert.

Fuzzy control systems produce actions according to fuzzy rules based on fuzzy logic. The basic units of the fuzzy logic controller are: fuzzifier, fuzzy rule base, fuzzy inference engine, and defuzzifier [8]. In fuzzifier, crisp input values are mapped to fuzzy sets using membership functions. Fuzzy rule base contains the IF-THEN rules which specifies the behavior of the system. Fuzzy inference engine maps input fuzzy sets to output fuzzy sets using rule base. Defuzzifier maps the fuzzy output sets to crisp output value. The structure of a fuzzy control system is given in the Figure 2-4:

In a fuzzy control system of a mobile autonomous robot, firstly crisp sensor values is translated into linguistic classes in the fuzzifier, and then appropriate rules are fired in the fuzzy inference engine which generates a fuzzy output value, finally this value is translated into crisp turning angle or speed in the defuzzifier. This final result is passed to motor as a command.

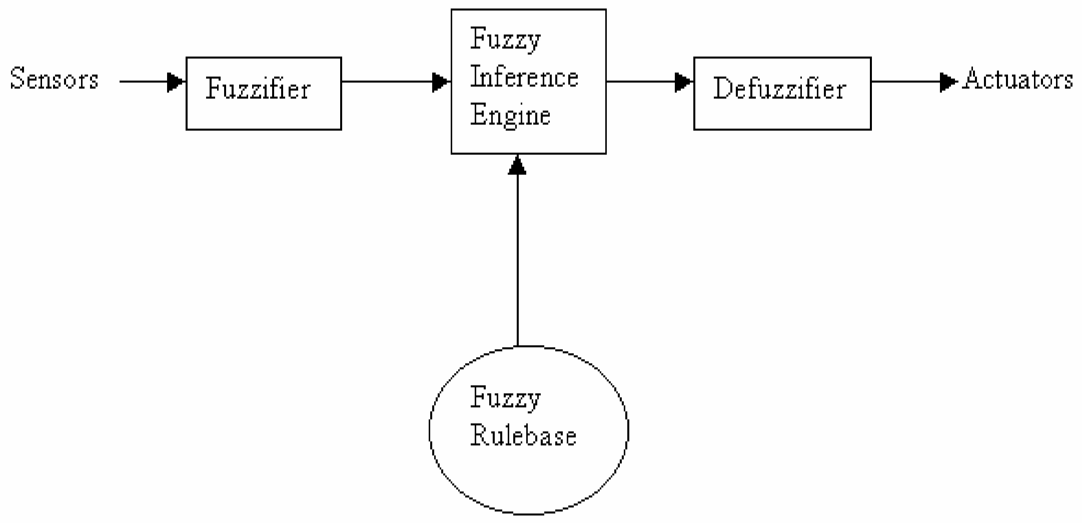

Figure 2-4 Fuzzy Logic Control System Architecture (adopted from [8])

The first and most common application of fuzzy logic techniques in the domain of autonomous robotics is the use of fuzzy control to implement individual behavior units. Fuzzy logic controllers incorporate heuristic control knowledge in the form of if-then rules, and are a convenient choice when a precise linear model of the system cannot be easily obtained. They have also shown a good degree of robustness in the case of variability and uncertainty in the parameters. The key to this robustness is in the interpolation mechanism implemented by fuzzy controllers, which embodies the idea that similar inputs should produce similar actions.

These characteristics fit the needs of autonomous robot navigation where: mathematical model of the environment is not available, sensor data is uncertain and imprecise and real-time operations are fundamental.

The advantages of fuzzy control can be summarized in three headings. Firstly, the fuzzy rule format makes it easy to write simple and effective behaviors for a variety of tasks, without having to use complex mathematical models. Secondly, because of their qualitative nature, fuzzy behaviors are prone to be transferred from one platform to another with few modifications. Finally, the interpolative nature of fuzzy control results in smooth movement of the robot, and in graceful degradation in face of errors and fluctuations in sensor's data [12].

2.3 Behavior Design

2.3.1 Proprioceptive Behaviors

The classic behavior used in mobile robotics is path tracking. In path tracking controller that is given a path in the form of coordinates, follows the path as close as possible. The controller needs its position relative to the path to accomplish this task. The position is usually gathered from wheel encoders or other sensors of the robot. These kinds of behaviors are called proprioceptive behaviors [12].

Because the kinematics and dynamics of the robot may be complex and nonlinear and interaction of the robot and the terrain is hard to model, path tracking can be a difficult task. These problems led the roboticists to use fuzzy control techniques for path tracking. Fuzzy control is used in several applications: pass-through-way-points problem by Benreguieg[13], spline curves by Zhang[14], following a straight line Tanaka and Sano[15], tracking arbitrary continuous points by Ollero[16,17].

Instead of generating a path from the plan to reach a global goal, another popular technique used is to generate a potential field that has maxima around the obstacles, and minima at the goal location. A potential field based on the model of the environment is generated, and the robot then navigates through this field by using gradient descent algorithm, which can be implemented by fuzzy rules. Makita [18] has proposed a system based on this idea and which has been tested in simulation.

2.3.2 Sensor-Based Behaviors

Tracking a precomputed path is an effective way to reach a target position when the assumptions used during the computation of the path are still valid at execution time (the environment was correctly modeled, and it has not changed afterwards); and the robot is able to determine its position with respect to the path. Because these conditions are rarely met in real world environments, the robots are based on sensor-based behaviors. A sensor-based behavior implements a control policy based on the sensing of environment in execution and navigating accordingly, rather than with respect to a path. Typical examples include moving along a wall or a contour, reaching a light source, and avoiding obstacles.

In 1985 Sugeno and Nishida developed a fuzzy controller for a model car that moves in a track delimited by two walls [19]. A single rotating sonar sensor was used to measure the distances from the walls. The fuzzy controller was developed according to heuristic rules derived from the observation of a human driver. The results were encouraging, but the controller was not robust to sensor's errors, and the speed of robot was very slow 1.7 cm/sec.

Shortly after the publication of Sugeno and Nishida, Takeuchi [20] developed a fuzzy controller for obstacle avoidance. The controller was obtaining occupied and free areas in front of the robot from a video camera. The rules were experimentally derived with the help of a simulator. The experimental results were satisfying, although perceptual errors in the vision system sometimes led to failures.

In the subsequent years, an increasing number of fuzzy sensor-based behaviors had implemented in robots but the attention was mostly focused on the same two fundamental behaviors: following environmental features (walls, road edges, white lines on the floor, etc), and avoiding obstacles.

More extensive autonomous robots have been developed in following years, which have different behaviors. Examples for these autonomous robots are Flakey [21, 22], Marge [2], Moria [23], and Lobot [24]. These robots include fuzzy behaviors for going to a given position, for orienting towards a target, for docking to an object, for crossing a door, and so on.

In Flakey [21, 22], a hybrid architecture is implemented; fuzzy control is used in conjunction with modeling and planning techniques to provide reactive guidance by Saffotti, Ruspini and Konolige in 1993. Sonar is used by Flakey to construct a cellular map of its environment. A high-level supervisory process then extracts the locations of landmarks and features such as walls from this map. These values are passed to selected fuzzy control behaviors to generate the proper wheel velocities. The behaviors are implemented as schemas that have three components: context (behavior relevancy to the current situation), desirability function (set of rules defining control) and descriptor set (objects in the world that are significant for the robot).

For modeling behaviors Marge used distributed fuzzy agents, which are all independent and concurrent. Fuzzy behaviors implemented in Marge include goal seeking, obstacle avoidance, wall following and docking. In behavior fusion part, these behaviors are given gains and final command is gathered by vector summation.

2.3.3 Complex Behaviors

The behaviors like obstacle avoidance, wall following, path tracking, etc. are basic behaviors. They consider only one goal. If consideration of two or more goal is needed, there are two options. Complex rules can be written whose antecedents consider all goals or different rules can be written for each goal and outputs of these can be combined.

Whether a complex behavior is better implemented by a monolithic or a partitioned set of rules is a difficult question. The monolithic solution can take better care of the interactions between the goals, and should be preferred when these interactions are important. Unfortunately, the monolithic solution can easily become intractable, as the number of rules tends to grow exponentially in the size of the input space. When input space is large, a partitioned solution is likely to be easier.

2.4 Behavior Coordination

Coordination of the several simultaneous independent behavior-producing units to obtain an overall behavior that achieves the intended task is behavior coordination. The simplest example is the coordination of an obstacle avoidance behavior and a goal reaching behavior in order to safely reach the target in an environment with obstacles. Today behavior coordination is still a major problem. Behavior coordination problem can be evaluated in two categories: behavior arbitration or command fusion.

2.4.1 Arbitration

The arbitration policy determines which behavior should influence the operation of the robot at each moment, and determines the task performed by the robot. Early solutions based on fixed arbitration policy. An example for this approach is the famous subsumption architecture proposed by Brooks [1], which is based on a hard-wired network of suppression and inhibition links. (Figure 2-5) This rigid organization contrasts with the requirement that an autonomous robot can be

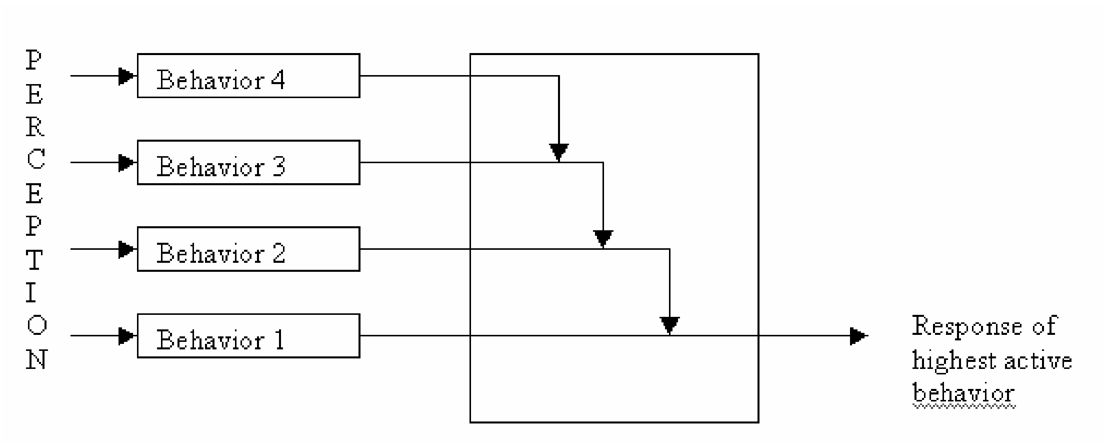

Figure 2-5 Arbitration by Suppression (adopted from [8])

programmed to perform a variety of different tasks in a variety of different environments. In fact, Brooks' robots were usually built to perform one single task.

One of the arbitration methods is action-selection. In this method the activation level of each behavior is determined according to agents goals and incoming sensor data. The behavior with the highest activation level is carried out at run time. No hierarchy exists between the behaviors. (Figure 2-6)

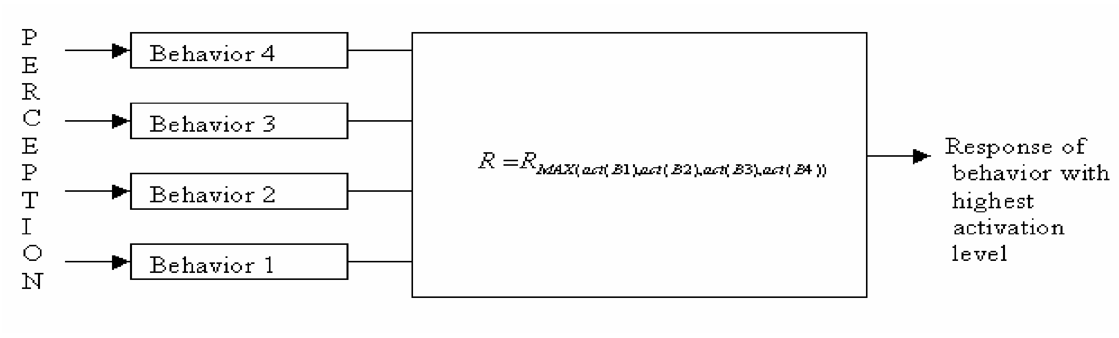
P E R C E P T I O N
Behavior 4
Behavior 3
Behavior 2
Behavior 1
$R = R_{MAX(act(B1),act(B2),act(B3),act(B4))}$
Response of behavior with highest activation level

Figure 2-6 Arbitration by Action Selection (adopted from [8])

Behavior coordination with voting is another arbitration method. In this architecture, behaviors vote for a predefined set of motor actions and the action receiving the highest vote is accomplished. (Figure 2-7)

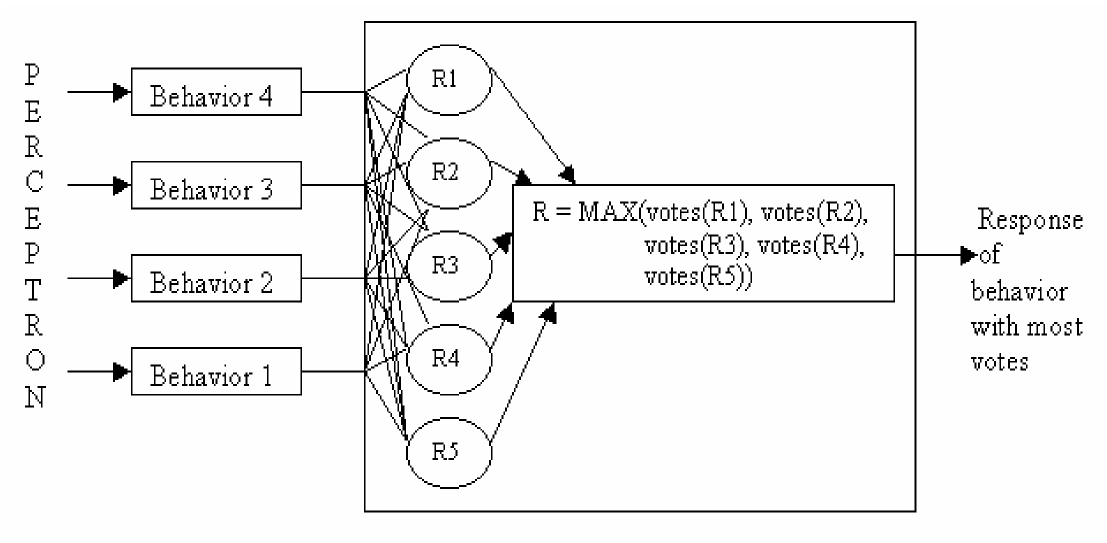
P E R C E P T R O N
Behavior 4
Behavior 3
Behavior 2
Behavior 1
R1
R2
R3
R4
R5
R = MAX(votes(R1), votes(R2), votes(R3), votes(R4), votes(R5))
Response of behavior with most votes

Figure 2-7 Arbitration by Voting (adopted from [8])

2.4.2 Command Fusion

In arbitration one of the behaviors is selected and accomplished according to arbitration method. This scheme may be inadequate in situations where several criteria should be taken into account. For example, consider a robot that encounters an obstacle while following a path and arbitration policy selects the obstacle avoidance behavior. Going around the obstacle from left or right is unimportant for the obstacle avoidance behavior. But, from the point of view of the path-following behavior, one choice might be dramatically better than the other.

To overcome this problem, different behaviors are executed parallel and outputs of these behaviors are combined. The most popular approaches for this type are based on vector summation scheme: a force vector represents each command, and commands produced by different behaviors are combined by vector summation. The robot carries out the resulting action. (Figure 2-8)

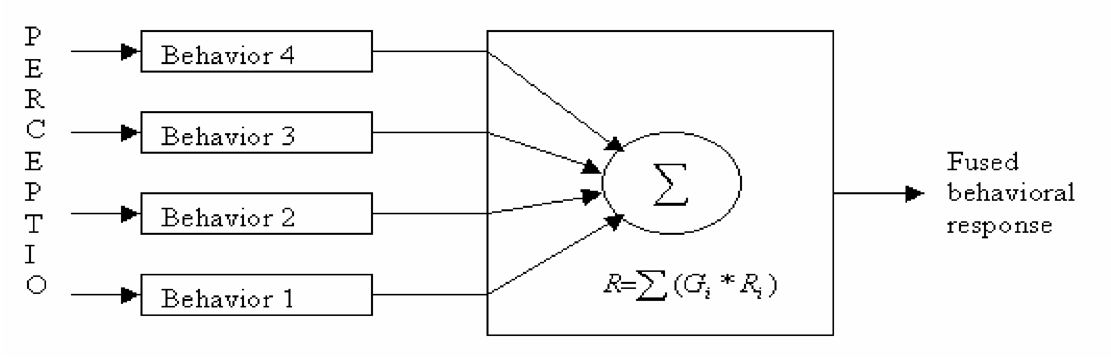

Figure 2-8 Behavior Coordination by Vector Summation (adopted from [8])

When the output of a behavior is represented by a fuzzy set, fuzzy operators can be used to combine the output of different behaviors into a collective result, and finally choose a command according to this result. Fuzzy logic offers many different operators to perform combination, and many defuzzification functions to perform decision. It is important to note that the decision taken according to the collective output can be different from the result of combining the decisions taken from the individual outputs. Each individual decision issued by a behavior gives the preferred command, but does not tell anything about the desirability of alternatives. Preferences contain more information, as they give a measure of desirability for each possible command: combining preferences thus uses more information than combining vectors, and can produce a different final decision. Figure 2-9 graphically illustrates this point in the case of two behaviors. This argument explains why fuzzy command fusion is fundamentally different from vector summation.

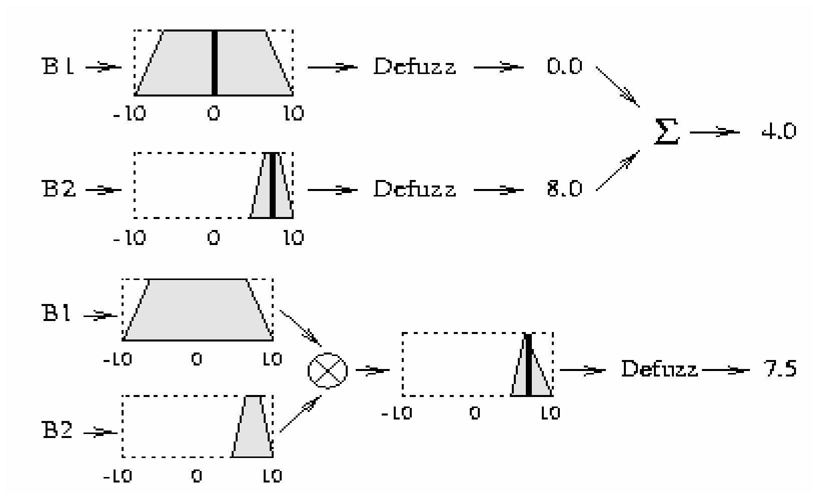

Figure 2-9 Fuzzy Behavior Coordination (adopted from [12])

Another form of behavior combination that can be realized using fuzzy logic is obtained by using both (i) fuzzy meta-rules to express an arbitration policy, and (ii) fuzzy combination to perform command fusion. This form of combination, was initially suggested by Ruspini [25], and fully spelled out by Saffiotti [21, 26] under the name of context-dependent blending of behaviors, or CDB.

CDB can be implemented in a hierarchical fuzzy controller as shown in Figure 2-10. In CDB, it is essential that the defuzzification step must be performed after the combination. Although in Figure 2-10 all the context-rules are grouped in one module, each context-rule can be put inside the corresponding behavior and this solution would be more appropriate for distributed implementations. This architecture can be iterated to implement individual behaviors, and combine them using a second layer of context-rules. Defuzzification should still be the last step.

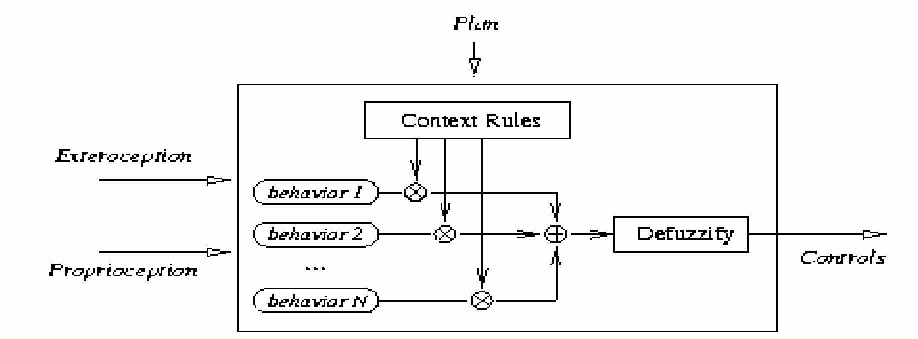

Figure 2-10 Hierarchical Fuzzy Controller (adopted from [12])

CHAPTER 3

ADAPTIVE-NETWORK-BASED FUZZY INFERENCE SYSTEM (ANFIS)

3.1 Adaptive Networks

An adaptive network is a kind of multi-layer feedforward network. Some or all of the nodes in this network are adaptive, which means that each output of these nodes depends on the parameter(s) held in these nodes, and the learning rule specifies how these parameters should be updated to minimize the error. Each node performs a particular function *(node functio*n) on incoming signals and parameters. The function of the node may vary from node to node, and the choice of each node function depends on the overall input-output function, which the adaptive network is required to carry out.

The links in an adaptive network only shows the flow direction of signals between nodes; no weights are associated with these links. For indicating different capabilities in an adaptive network, circles and squares are used. A square node (adaptive node) has parameters while a circle node (fixed node) has no parameters. In order to achieve a desired input-output mapping, these parameters are updated according to given training data and a learning procedure. An example for adaptive networks is given in Figure 3-1.

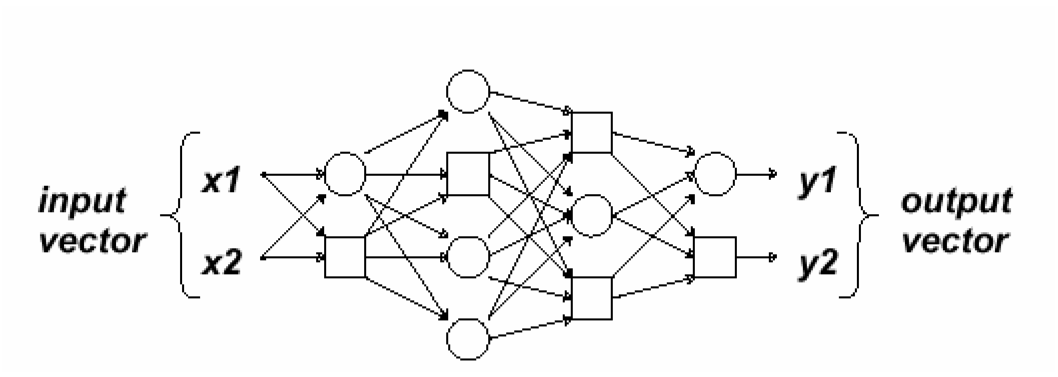

Figure 3-1 Adaptive Network (adopted from [27])

The basic learning rule of adaptive networks is based on the gradient descent and the chain rule, which was proposed by Werbos [28] in the 1970's. But the gradient method is slow and has tendency to become trapped in local minima.

3.2 Fuzzy Reasoning Mechanisms

Several types of fuzzy reasoning have been proposed in the literature. Depending on the types of fuzzy reasoning and fuzzy if-then rules employed, most fuzzy inference systems can be classified into three types as shown in Figure 3-2.

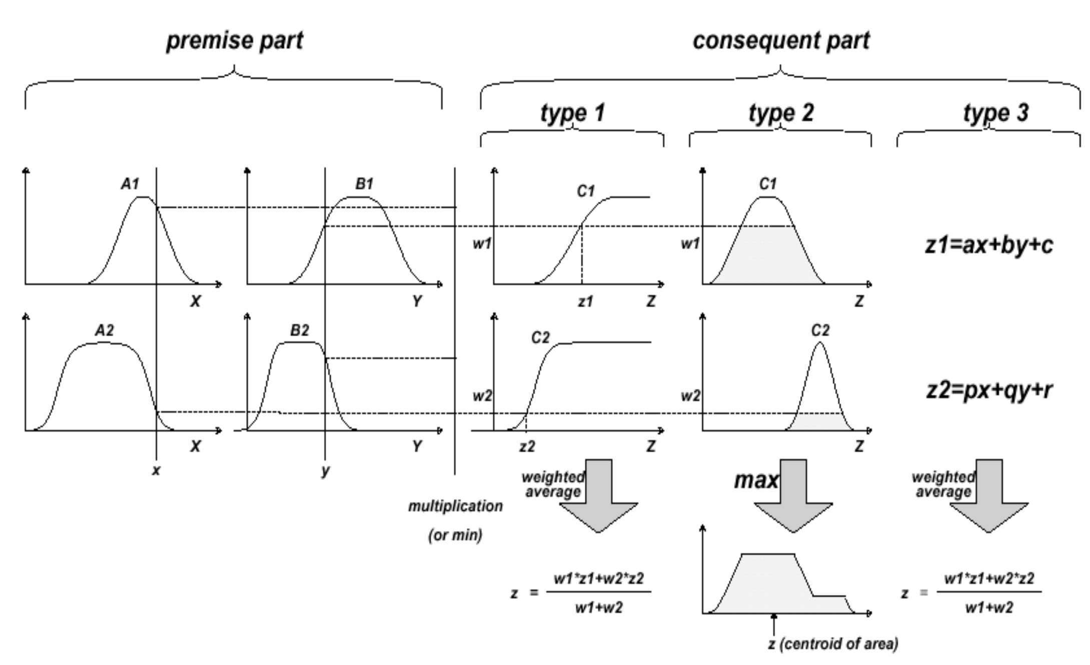

Figure 3-2 Commonly used fuzzy reasoning mechanisms (adopted from [27])

The differences between them lie in the consequents of their fuzzy rules, and thus their aggregation and defuzzification procedures differ accordingly.

In type 1 systems, the overall output is calculated as weighted average of each rule’s crisp output, which is represented by the rule’s firing strength and output membership functions. The rules firing strength can be determined by the product or minimum of the output of input membership function(s). The output membership functions used must be monotonically non-decreasing [29]. This model is called Tsukamoto model.

In type 2 systems, which is called *Mamdani model,* standard fuzzy sets are used in both the antecedents and consequents of the rules, and a defuzzification procedure is utilized to obtain a crisp value from the output. The overall output is calculated by applying “max” function to the fuzzy outputs of each rule. For final crisp output several approaches might be used like: center of area, bisector of area, mean of maxima, maximum criterion, etc. [30,31]

In type 3 systems Takagi and Sugeno’s fuzzy if-then rules are used [11]. The output of each rule is a linear combination of input variables plus a constant term, and the final output is the weighted average of each rule’s output.

3.3 ANFIS

ANFIS is a fuzzy inference system implemented in the framework of adaptive networks by Jang [27]. By using a hybrid learning procedure, ANFIS constructs an input-output mapping based on both human knowledge (in the form of fuzzy if-then rules) and input-output data pairs.

3.3.1 Architecture

Example ANFIS architecture for type 3 reasoning system is given in Figure 3-3:

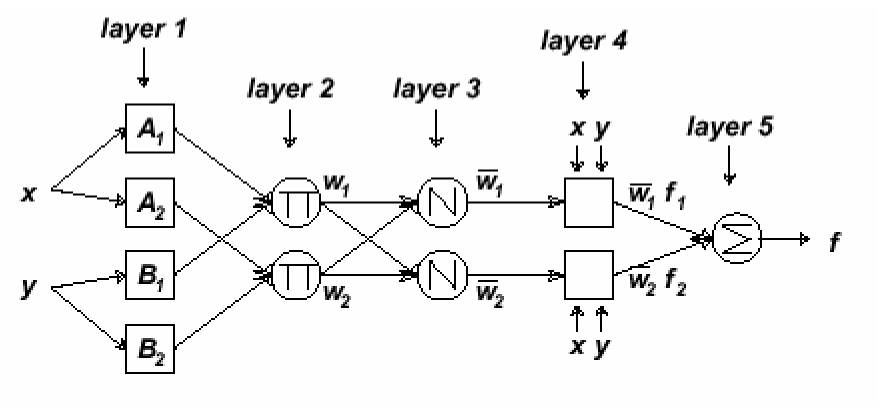

Figure 3-3 Type-3 ANFIS Architecture (adopted from [27])

ANFIS layers are defined as follows:

Layer 1 This layer represents the membership functions. Every node in this layer is a square node with a node function:

$$O_i^1 = \mu A_i(x) \qquad (1)$$

where x is the input node i and A_i is the linguistic label (high, small, far, near, etc.) associated with this node function. O_i^1 is the membership function of the A_i and it specifies the degree which the given x satisfies the A_i. Usually the $\mu A_i(x)$ is chosen as bell-shaped function with maximum at 1 and minimum at 0 like:

$$\mu A_i(x) = \frac{1}{1 + \left[\left(\frac{x - c_i}{a_i}\right)^2\right]^{b_i}} \qquad (2)$$

In this case {a, b, c} are parameters of this node. As these parameters change the numerous membership functions can be represented. Any continuous and piecewise differentiable function, such as commonly used trapezoidal or triangular-shaped membership functions, might be used in this layer as node functions. Parameters in this layer are referred to as premise parameters.

Layer 2 This layer represents the rules of the system. Every node in this layer is a circle node (has no parameters). In these nodes incoming inputs are multiplied and result is sent as output. Nodes in this layer are labeled as ∏. Each node output represents the firing strength of a rule. Output of the nodes in this layer is:

$$w_i = \mu A_i(x) * \mu B_i(y), \qquad i = 1,2 \tag{3}$$

Layer 3 This layer calculates the ratio of a rule’s firing strength to sum of all rules firing strengths. Every node in this layer is a circle node labeled as N. The output of this layer is called normalized firing strengths calculated with:

$$\overline{w_i} = \frac{w_i}{w_1 + w_2}, \qquad i = 1,2 \tag{4}$$

Layer 4 This layer calculates the output of each rule. Every node in this layer is a square node with the following function:

$$O_i^4 = \overline{w_i} f_i = \overline{w_i}(p_i x + q_i y + r_i) \tag{5}$$

where $\overline{w_i}$ is the output of third layer and { p_i, q_i, r_i } is the parameter set of the node. Parameters in this layer are called consequent parameters.

Layer 5 Node in this layer combines the output of each rule with vector summation. This single node is a circle node with the output:

$$O_1^5 = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \tag{6}$$

For type-1 fuzzy inference systems, the extension for type-3 ANFIS system is quite straightforward (the type-1 ANFIS is shown in Figure 3-4). In this architecture the output of each rule is induced jointly by the output membership function and the firing strength. For type-2 fuzzy inference systems, if we replace the centroid defuzzification operator with a discrete version, which calculates the approximate centroid of area, then type-3 ANFIS can still be constructed accordingly. However, it will be more complicated than its type-3 and type-1 versions.

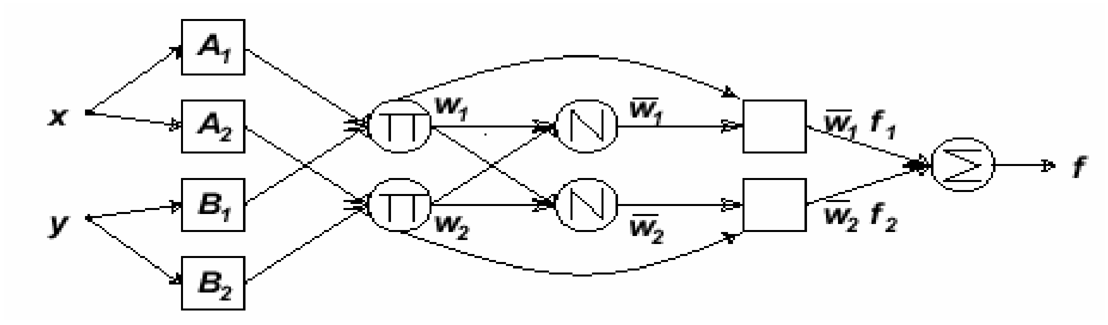

Figure 3-4 Type-1 ANFIS Architecture (adopted from [27])

3.3.2 Learning Algorithm

In ANFIS hybrid learning which combines the gradient descent method and least square estimate (LSE) is used to identify parameters.

The ANFIS network has only one output that can be represented as:

$$\text{Output} = F(\vec{I}, S) \tag{7}$$

where $\vec{I}$ represents the input and S represents the parameters of the network. S is the total parameters in premise and consequent parts. If there is a function H such that composite function H $\circ$ F is linear in some elements of S, that it can be said that elements of S can be found by using least squares method. If the parameter set S can be divided into two sets S_1 and S_2 such that H $\circ$ F is linear in S_2, then after applying H to Equation 7, we get the following, which is linear in S_2:

$$\text{H(output)} = \text{H} \circ F(\vec{I}, S) \tag{8}$$

Given the values of S_1, training data P can be added to the equation and the following matrix equation is gathered:

$$\text{AX} = \text{B} \tag{9}$$

X is an unknown vector whose elements are in S_2. If dimension of S_2 is M then dimension of A is PxM, dimension of B is Px1 and dimension of X is Mx1. Since number of training data (P) is greater than number of linear parameters (M), this is an overdetermined problem and there is no exact solution. Instead, least square estimate of X, X^*, can be found. The most well known formula for X^* is:

$$X^* = (A^T A)^{-1} A^T B \qquad (10)$$

where A^T is transpose of A and $(A^T A)^{-1} A^T$ is pseudo inverse of A if $A^T A$ is non-singular. Because the Equation 10 is expensive in computation and it becomes a problem when $A^T A$ is singular, sequential method of Least Square Estimate is used given in the following equation:

$$\begin{aligned} X_{i+1} &= X_i + S_{i+1} a_{i+1} (b_{i+1}^T - a_{i+1}^T X_i) \\ S_{i+1} &= S_i - \frac{S_i a_{i+1} a_{i+1}^T S_i}{1 + a_{i+1}^T S_i a_{i+1}}, \quad i = 0, 1, \cdots, P-1 \end{aligned} \qquad (11)$$

where a_i^T is i-th row vector of matrix A, b_i^T is the i-th element of matrix B. S_i is covariance matrix and least square estimate X^* is X_P. Initially X_0 = 0 and S_0 = γI where I is identity matrix with dimension MxM and γ is a positive large number.

In ANFIS hybrid-learning procedure is composed of a forward pass and a backward pass. In the forward pass, input data is given to network and functional signals go forward to calculate each node output until the A and B given in the Equation 9 is obtained and then the parameters in S_2 are identified by the sequential least squares formulas in Equation 11. After parameters in S_2 are identified, the functional signals go forward and the output is calculated.

With using the actual and target output values the error is calculated. In the backward pass the error is propagated from output end to input end and parameters in S_1 is updated by gradient descent method.

Assuming the training data set has P entries, the error measure can be defined for the p-th entry as in equation 12:

$$E_p = \sum (T_p - O_p)^2 , \qquad (12)$$

where T_p is the target output and O_p is the actual output. Then the overall error measure is E = $\sum_{p=1}^{P} E_p$.

To implement gradient descent in E over parameter space, firstly the error rate $\frac{\partial E_p}{\partial O}$ for the p-th training data and for each node output must be calculated. The error rate for the output node at layer L can be calculated from equation 13:

$$\frac{\partial E_p}{\partial O} = -2\,(T_p - O_p^L)\,. \tag{13}$$

For the internal node at (k,i) layer k index i, error rate can be derived by the chain rule:

$$\frac{\partial E_p}{\partial O_{i,p}^k} = \sum_{m=1}^{\#(k+1)} \frac{\partial E_p}{\partial O_{m,p}^{k+1}} \frac{\partial O_{m,p}^{k+1}}{\partial O_p^k}, \tag{14}$$

where $1 \le k \le L-1$. That is the error rate of an internal node can be expressed as a linear combination of the error rates of the nodes in the next layer. Therefore for all $1 \le k \le L$ and $1 \le i \le \#(\mathrm{k})$, $\frac{\partial E_p}{\partial O_{i,p}^k}$ can be found by using equation 13 and 14.

If α is a parameter of the given adaptive network,

$$\frac{\partial E_p}{\partial \alpha} = \sum_{O^* \in S} \frac{\partial E_p}{\partial O^*} \frac{\partial O^*}{\partial \alpha}, \tag{15}$$

where S is the set of nodes whose output depends on α. Then the derivative of the overall measure E with respect to α is,

$$\frac{\partial E}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \alpha}\,. \tag{16}$$

Accordingly the update formula for the parameter α is

$$\Delta\alpha = -\eta \frac{\partial E}{\partial \alpha}, \qquad (17)$$

in which η is a learning rate which can be expressed as:

$$\eta = \frac{k}{\sqrt{\sum_{\alpha} (\frac{\partial E}{\partial \alpha})^2}}, \qquad (18)$$

where k is step size, the length of each gradient transition in the parameter space. The value of k change to vary speed of convergence.

CHAPTER 4

IMPLEMENTATION

4.1 Problem Definition and Motivation

Several studies are made on reactive autonomous mobile robots. Until recent years, in most of the studies numerical methods are used to control the robot, but the usage of fuzzy logic is gaining interest in mobile robotics. Fuzzy logic provides tools that are of potential interest to mobile robot control [12].

In this study fuzzy control is used in a neural network by which both the advantages of numerical systems and fuzzy systems are obtained. Numerical systems learn fast without dealing with symbolic representations and fuzzy systems gets the profit of human reasoning in the modeled system.

An online learning system for a dynamic environment is modeled and implemented in this work. The goals are to navigate the robot without striking to the obstacle, to find and reach the target objects. Environment is cluttered with different shaped obstacles. Experiments are performed in a simulation environment named Webots 2.0 of the Khepera robot [32]. This robot has eight short-ranged infrared sensors. A matrix vision turret is used to get the image of the world seen by robot.

Main motivation of this study is to design a robot control system, which is robust to sensor errors and sudden changes, adaptable to different objectives and reactive in dynamic environments. No global world model and deliberative reasoning is used.

4.2 Khepera and Webots

Khepera is a small robot especially used for research and education. It is 5 cm in diameter. It has 8 infrared sensors with the capability to detect obstacles, which are maximum 5 cm far away. Light sources also can be detected with these infrared sensors that is in the 15 cm range.

It has two independent DC step motor wheels with encoders. Each wheel can move with the speed in the range ± 20, where unit of speed is 8mm/s. Maximum speed of Khepera is 16 cm/s.

Khepera has a matrix vision turret, which is used to gather the image of the environment. It supplies 24-bit color depth with a 60x80 pixel resolution.

Webots is a simulator, which is specifically designed for Khepera, and it adds %10 error to sensors and actuator to simulate the real world conditions.

4.3 Control Architecture

4.3.1 Behaviors

A behavior-based architecture is used for the control system. Different methods are used for behaviors. There are four basic behaviors of the robot, which are: Avoid Obstacle, Go to Goal, Wall Following and Generate Random Action. Each behavior is described as follows.

Avoid Obstacle and Go to Goal: An online learning is applied for the avoid obstacle and go to goal behavior. These two behaviors are implemented in a neuro-fuzzy architecture, ANFIS, which is defined in chapter 3. The membership functions and rules can be configurable, so structure of the network is not fixed. It can be adapted to different conditions easily.

Goal objects are identified with red color. The output of the network is used as the direction of the robot. Negative values are used as left turn; positive values are used as right turn.

There are four inputs of the network:

Density of Goal: Image of the world is taken as input from matrix vision turret and red color density in the image is calculated for density of goal.

Direction of the Goal: This input is also calculated from the image gathered from color sensor. Image is divided into two parts, left and right, and according to the red color density in both parts direction of the goal is determined.

Obstacle Distance: Infrared sensor values are used for distance to obstacles. Sensor values are grouped into three categories: left, right and front. The maximum of these three groups is used as obstacle distance.

Obstacle Direction: Again infrared sensor values are used to determine obstacle direction.

This behavior is designed in a generic approach. Number of membership functions, parameters for these functions and rules of the system are adaptable.

Wall Following: When the robot is faced with a wall this behavior makes the robot follow it.

Search Goal: This behavior is activated when no goal is seen for a number of predefined steps (about 100 simulation step). In this behavior robot turns around and at every 20 degree gets the image of the environment, calculates the goal density. At the end of rotation, robot decides to go in the direction where the goal density is greatest.

Control architecture is given in the Figure 4-1:

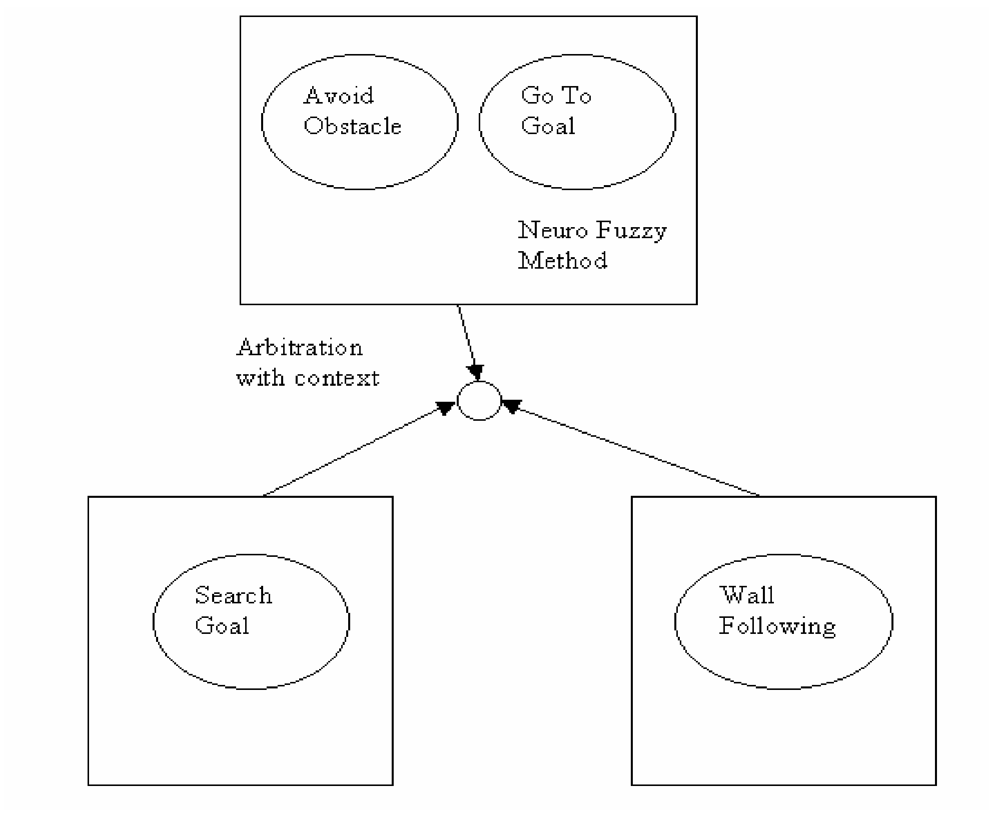

Figure 4-1 Control Architecture

Avoid obstacle and go to goal behaviors are implemented as cooperative behaviors. Coordination of these activities is supplied with the use of ANFIS architecture.

The overall coordination of behaviors is performed by arbitration with context. According to the situation of the world a priority is given for each behavior and the behavior with the highest priority is carried out.

4.3.2 Learning Mechanism

ANFIS is built according to a configuration file which defines the number of rules, number of membership functions and relationship of nodes. Initial parameters for membership and output functions are taken from a parameter file. Khepera is enabled and it begins to run. At each step according to context a behaviour is selected and robot is moved according to output of the this behavior. This execution is shown in Figure 4-2.

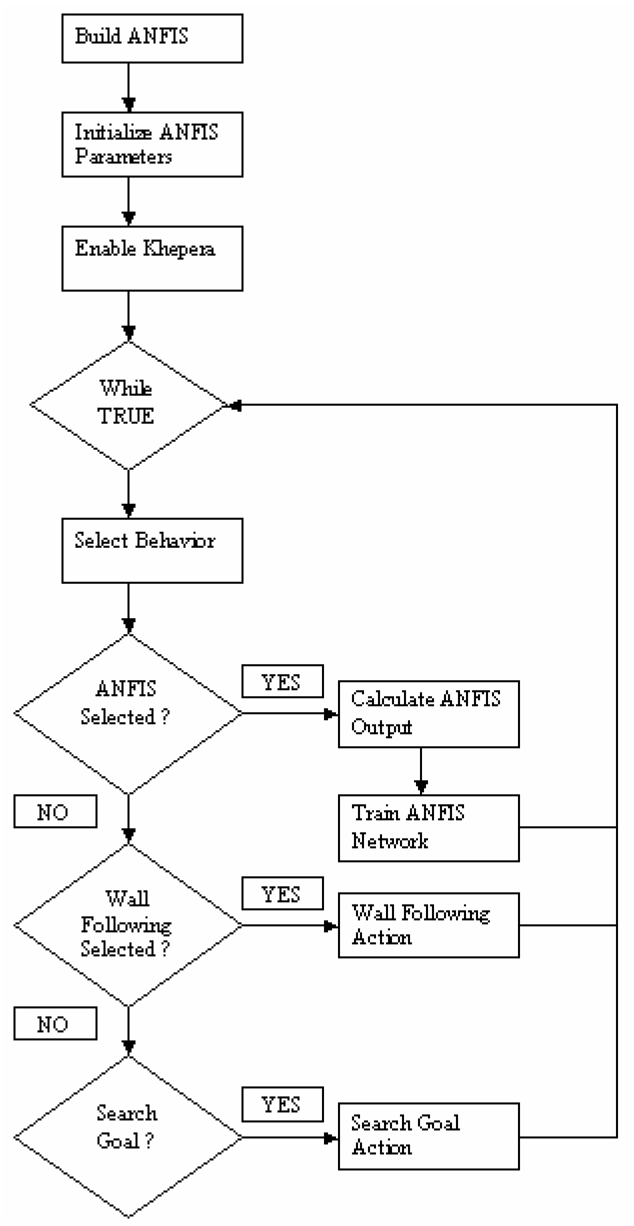

Figure 4-2 Main Execution Cycle

In behavior selection, current image of world seen by Khepera is taken from camera. Image is analyzed and if any goal object is observed ANFIS behavior is selected for the robot to move toward the goal. After goal is reached, for a number of steps again ANFIS behavior is activated although the goal can not be seen. By this way, with avoid obstacle behavior in ANFIS, it enables the robot go away from the goal.

If goal was not seen for a determined number of steps, Search Goal behavior is selected. If the robot was following a wall, the Wall Following behavior is selected again. Otherwise sensor values are gathered and analyzed to see if there exist a new wall in the left or right of the robot. If so, Wall Following behavior is executed. Robot moves according to ANFIS behavior, otherwise. Behavior selection is shown in Figure 4-3.

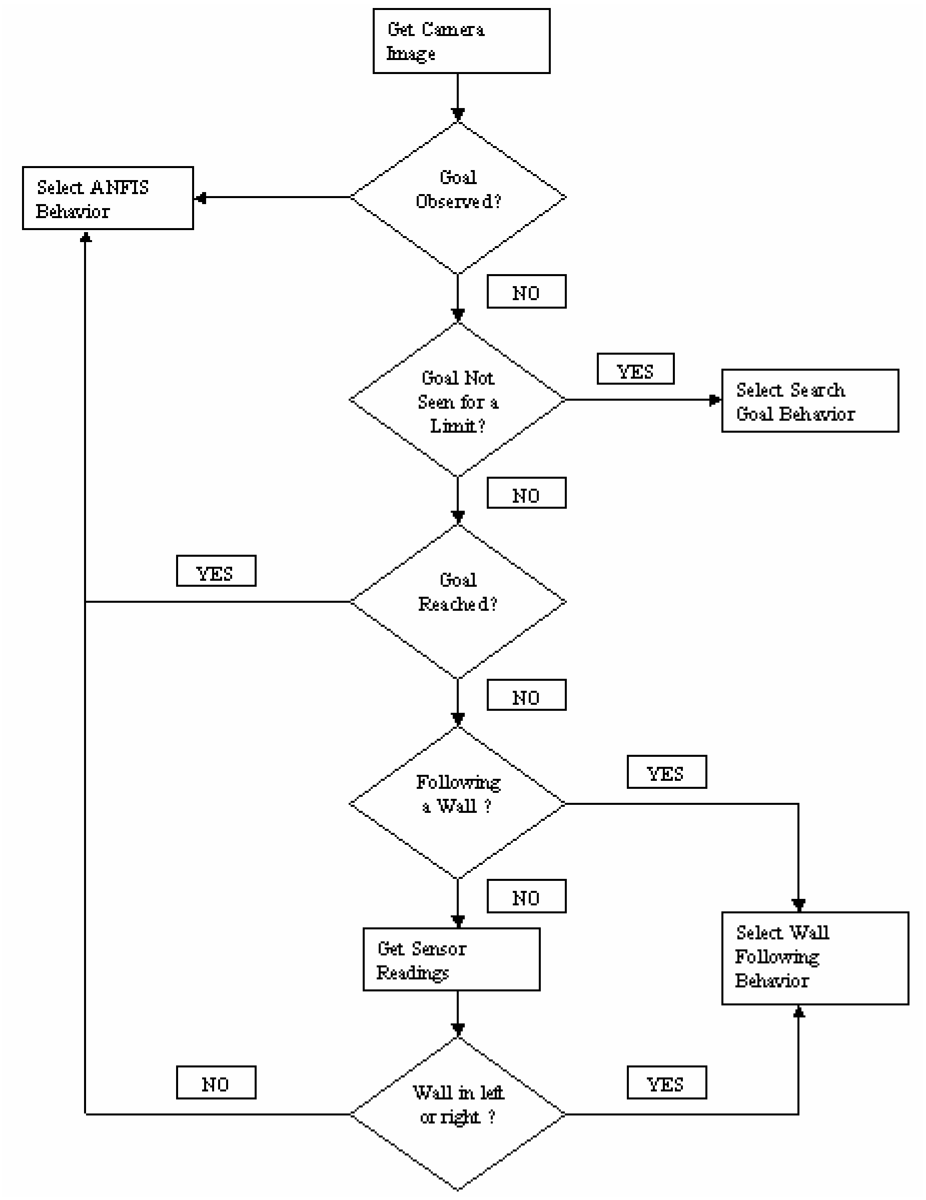

Figure 4-3 Behavior Selection

In ANFIS behavior, inputs of the network are calculated and given to ANFIS network to determine the output. If output is smaller than 0, robot turns left with the speed value as the output of network. If output is larger than 0, robot turns right with the speed value as the output of network. Otherwise robot moves forward. ANFIS behavior is shown in Figure 4-4.

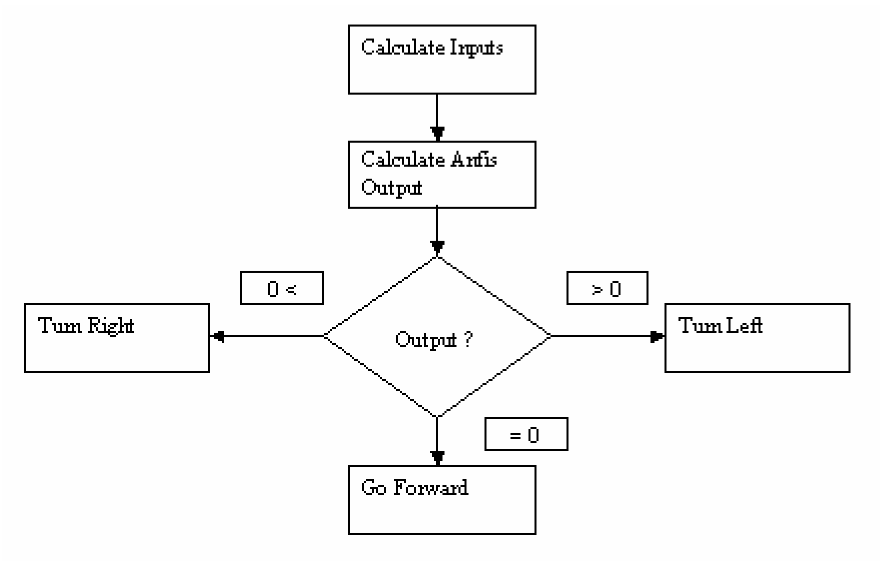

Figure 4-4 ANFIS Behavior

After robot action, ANFIS network is trained. The target output value for the network is determined using the situation of the world perceived by the robot before and after its action. A negative reinforcement arises when the robot is so close to an obstacle or goal density observed is decreased. Positive reinforcement arises when the goal density is increased. If there exist an obstacle close to robot, target value must be to turn opposite direction of the obstacle. If observed goal density is decreased after the move, target value is selected as the opposite direction that the robot was moved. But if goal density is increased target value is selected same value as the robot was moved. If the change in the goal density is small, it is assumed to be not changed. In this condition, goal direction is used to determine the target value. If goal seen, the direction of the goal is selected as the target value. In other conditions target value is selected as the same value that the robot was moved. Target value determination algorithm is given in Figure 4-5.

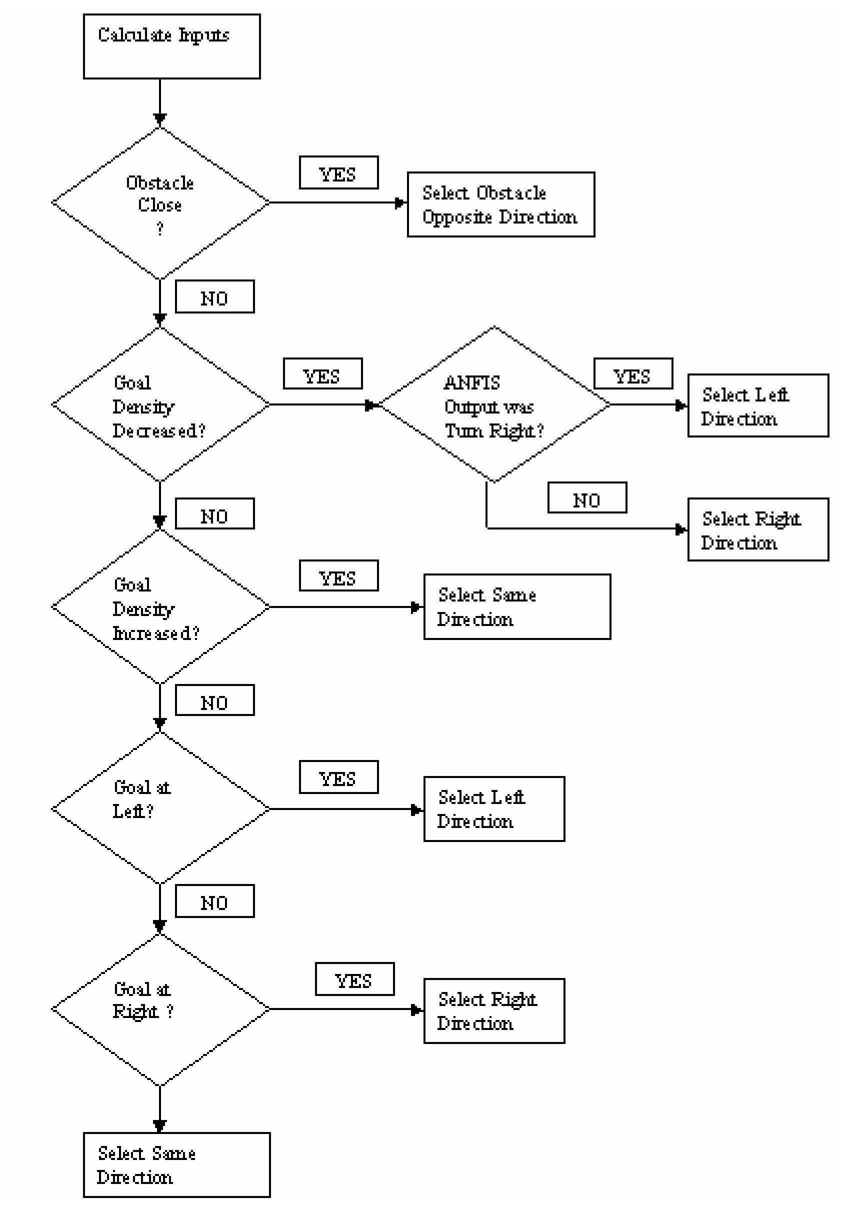

Figure 4-5 Find Network Target Value

Network is trained according to determined target value. Network is selected as type-1 ANFIS. Membership functions in 2nd layer of the network are choosen as bell-shaped functions as shown in Figure 4-6 . Parameters a, b and c are used to determine the membership function.

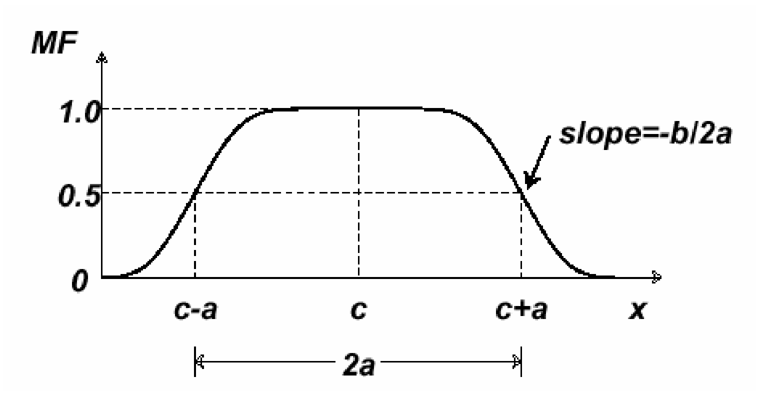

Figure 4-6 Membership Functions

Consequent functions in 4th layer must be monotonically non-decreasing for type-1 ANFIS. The consequent functions are also determined according to parameters p and q, as shown in Figure 4-7 .

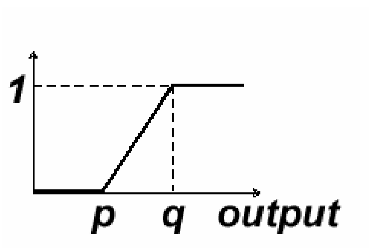

Figure 4-7 Consequent Functions

Input values are given to network and output of nodes in layer 1 to 3 is calculated. For identifying parameters of consequent function in 4th layer Least Square Estimate is used which is given in Equation 11. According to identified parameters, actual output of network is gathered. Error is calculated with actual and target output values. With gradient descent method error is propagated backwards and in 2nd layer membership functions parameters are updated. Network training is shown in Figure 4-8.

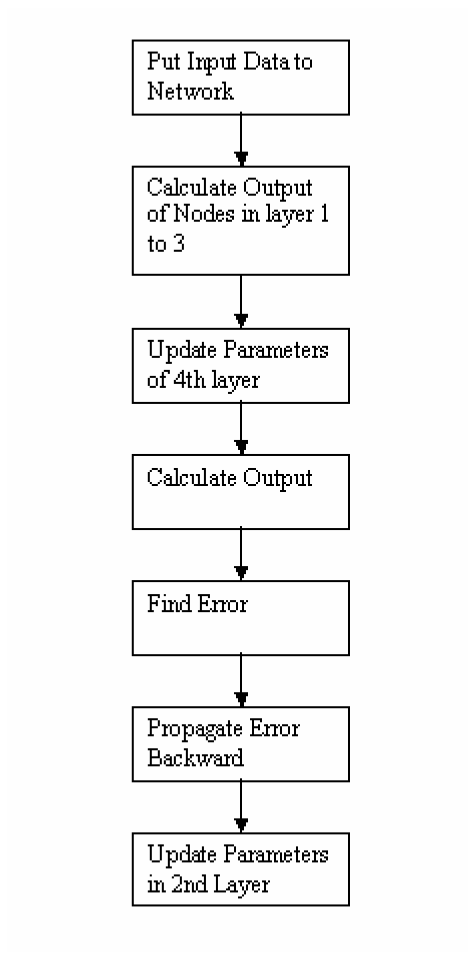
Put Input Data to Network
Calculate Output of Nodes in layer 1 to 3
Update Parameters of 4th layer
Calculate Output
Find Error
Propagate Error Backward
Update Parameters in 2nd Layer

Figure 4-8 ANFIS Network Training

When goal is not seen for a number of steps (about 100 simulation steps), robot stops and begins to turn its around. After each 20 degree turning, it stops and image of the world is gathered from camera and goal density is calculated. When robot completed a turn around itself, it stops. From calculated goal densities the direction where goal density was maximum is selected. Robot turns to this direction and begins to move in this direction. Search goal behavior execution is given in Figure 4-9.

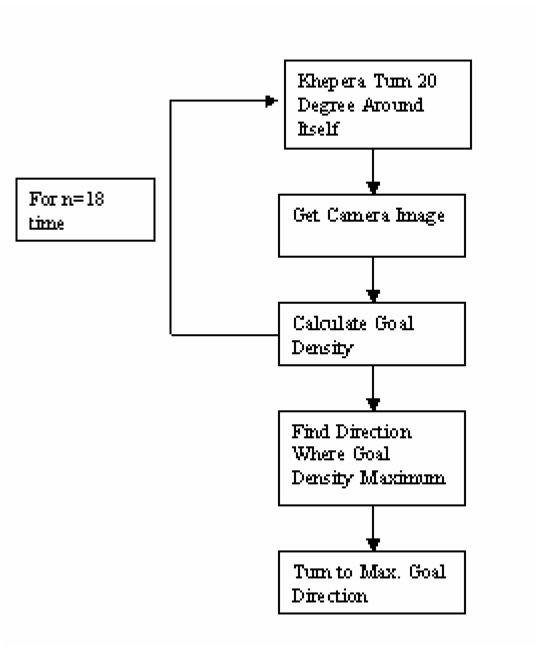
Khepera Turn 20 Degree Around Itself
For n=18 time
Get Camera Image
Calculate Goal Density
Find Direction Where Goal Density Maximum
Turn to Max. Goal Direction

Figure 4-9 Search Goal Behavior

In Wall Following behavior, direction of the wall is determined from sensor readings. Speed for motors are calculated according to distance and direction to wall. Robot makes movement according to determined motor values. If the wall is not seen after this action, wall following behavior is stopped. Then again behavior selection given in Figure 4-3 is executed. Wall Following behavior execution is shown in Figure 4-10.

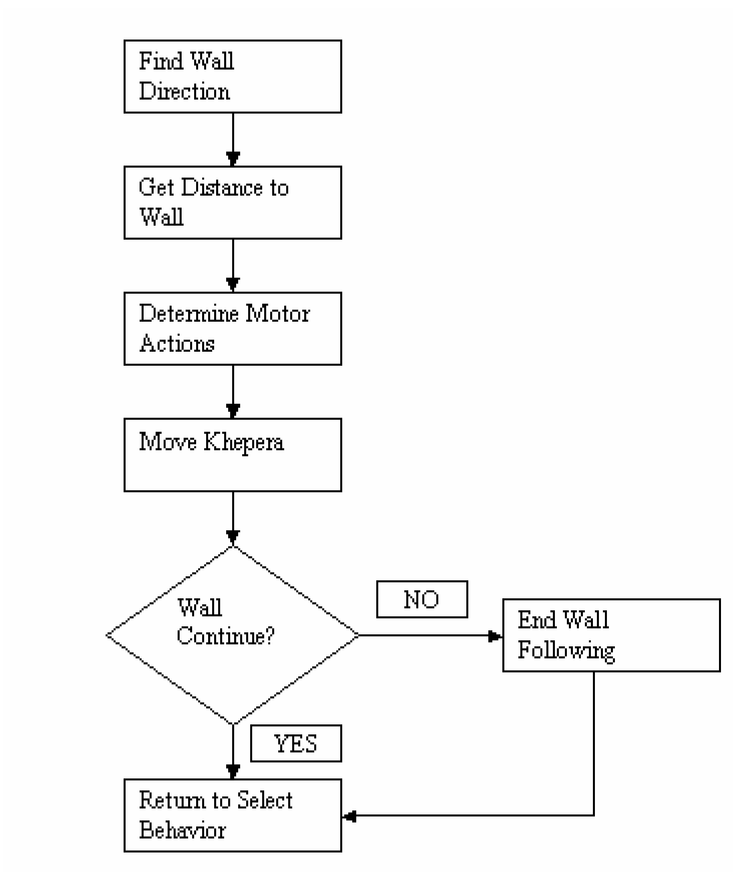

Figure 4-10 Wall Following Behavior

4.4 Experimental Results

Experiments are carried out at Webots. Fixed and dynamic environments are used as test environment. In environments there were obstacles and goal objects placed randomly. In dynamic environments goal object were dynamic. In both environments the robot was able to find and reach the goal objects. When the objects are moving, the robot followed the goal objects.

The complexity of the environment is determined according to number of obstacles, number of goal objects and wall orientations. As the number of obstacles increases and the walls prevents the robot to see and reach the goal objects, the complexity of the environment increases.

When in the environment number of obstacles and goal objects are increased, the robot learned the behavior faster. Because the data used in the training of the network is collected as the robot moves around, if there exist more objects in the environment, then richer data will be available. With rich data the network trained faster.

But crowded environments were designed such that the robot can see the goal objects easily. When the robot cannot see the goal object, because of walls or obstacles it can get stuck at same positions.

Experiments can be grouped under four headings that can be categorized according to learning algorithm used, rules employed, world models and static/dynamic goal objects.

1. In ANFIS network training, both batch and online learning is tested. While the robot navigates in the world, training data is collected. In batch learning, after each 1000 training data is collected, the network is trained. These 1000 training data used in training are not same. The training of network with each 1000 data is called an epoch. One epoch is divided into two phases, a forward and backward phase. In first phase consequent function parameters is identified with Least Square Estimate (LSE) method. In this phase membership function parameters are not updated. In second phase, gradient descent algorithm is used to update membership function parameters.

In online learning, for each training data firstly Least Square Estimate is used to update consequent function parameters and then with using these identified parameters output of network is calculated. Error is calculated for training data and membership parameters are updated using this error in Gradient Descent algorithm.

In batch processing, total error collected for all training data in one epoch is used in Gradient Descent. But in online learning only the current training data error is used.

Experiments are made in the world model shown in Figure 4-11.

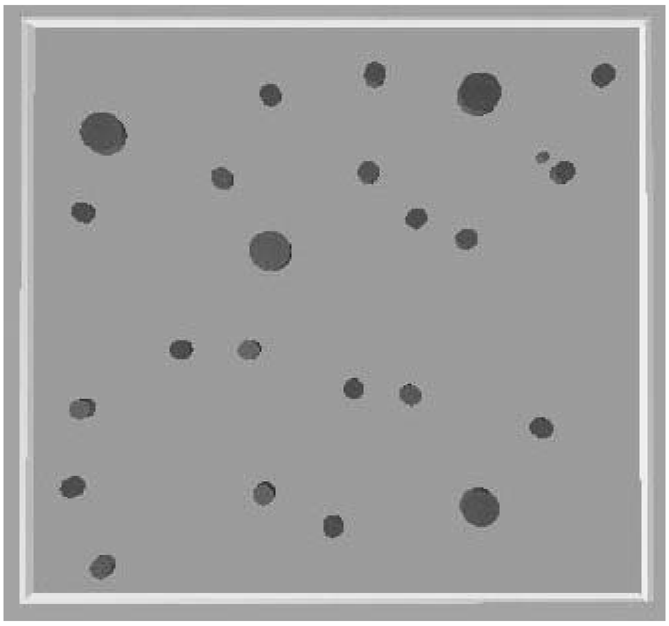

Figure 4-11 An Example World

In Figure 4-12 analysis for batch and online learning is shown:

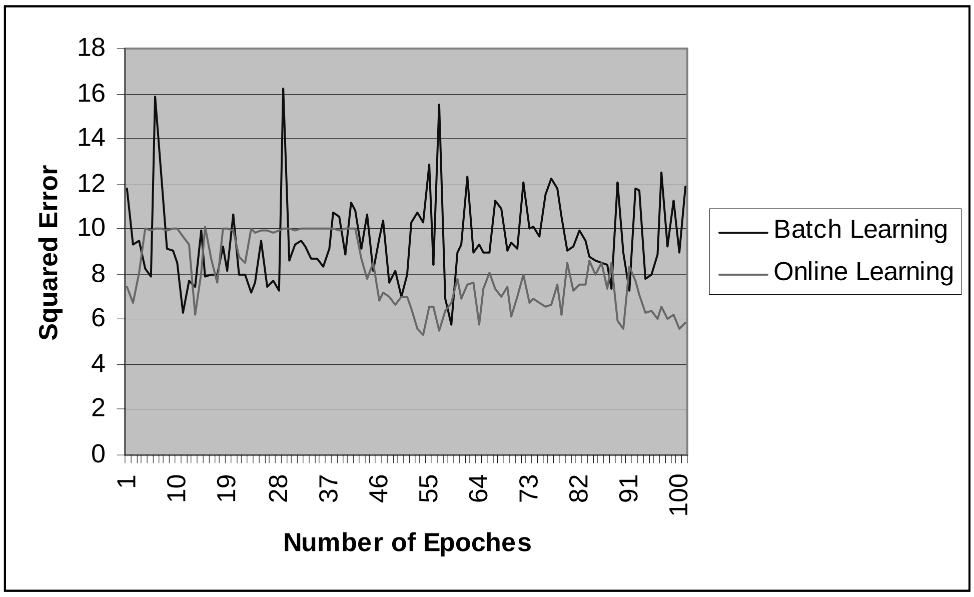

Figure 4-12 Error Analysis for Batch and Online Learning

Online learning gives better results as compared to batch learning. System is adapted to current conditions in online learning easily. Because in batch learning, the training occurs in epochs and between epochs the robot acts according to last results obtained and does not adapt to current conditions. But in online learning the training occurs after each action of the robot and robot uses these updated results that reflects the current situation better.

It is difficult to learn escape when the robot stuck near a wall in batch learning, the picks seen in graphics is caused by these situations. The robot can stuck near a wall when there exist a wall corner at one side of it, and sees target objects in same direction. It oscillates between the Avoid Obstacle and Go to Goal behaviors. But Avoid Obstacle has higher priority and robot learns to escape from this condition. In batch processing because learning is made within epochs, robot learns to escape from this condition one epoch later.

Online learning is also tested using forgetting factor in Least Square Estimate method. With this factor squared error measure is formulated to give higher weighting factors to more recent data pairs. By this way the effects of old data pairs are decayed as new data pairs become available. The updated formulas for LSE is given in Equation 19 with forgetting factor λ:

$$\begin{aligned} X_{i+1} &= X_i + S_{i+1} a_{i+1} (b_{i+1}^T - a_{i+1}^T X_i) \\ S_{i+1} &= \frac{1}{\lambda}\left[S_i - \frac{S_i a_{i+1} a_{i+1}^T S_i}{\lambda + a_{i+1}^T S_i a_{i+1}}\right] \end{aligned} \tag{19}$$

In Figure 4-13 comparison with and without forgetting factor in online learning is shown. With forgetting factor used in LSE, system gives better results.

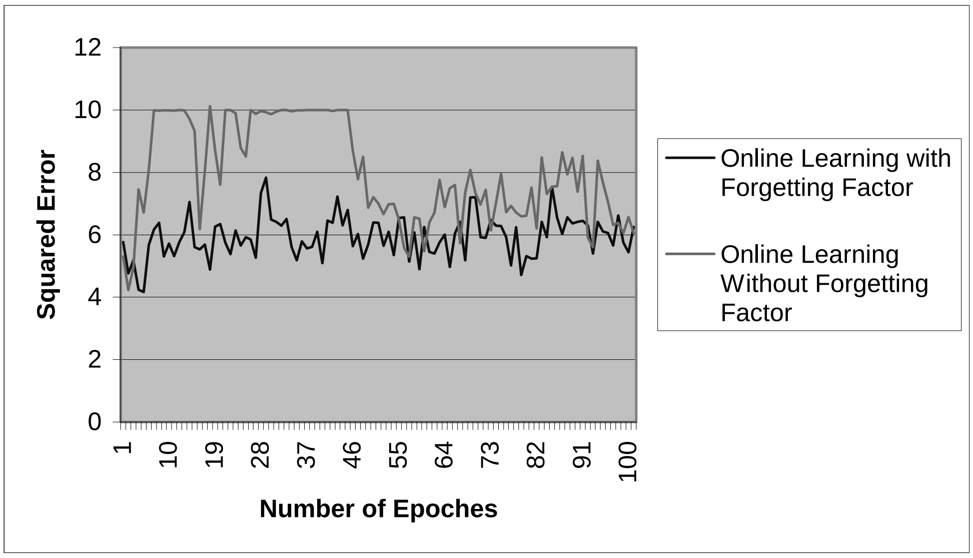

Figure 4-13 Error Analysis for Online Learning with Forgetting Factor

2. Robot is tested with two different rule sets and membership functions for ANFIS network within same world model (Figure 4-11).

In both sets the inputs are same: Goal Density, Goal Direction, Obstacle Distance and Obstacle Direction.

In first set,

There are 2 membership functions for each input:

- Goal Direction: Left, Right
- Goal Density: Low, High
- Obstacle Distance: Near, Far
- Obstacle Direction: Left, Right

And 6 rules:

1. If Goal Direction is Right AND Goal Density is Low TURN RIGHT

2. If Goal Direction is Right AND Goal Density is High TURN RIGHT SHARP

3. If Goal Direction is Left AND Goal Density is Low TURN LEFT

4. If Goal Direction is Left AND Goal Density is High TURN LEFT SHARP

5. If Obstacle Distance is Near AND Obstacle Direction is Right TURN LEFT

6. If Obstacle Distance is Near AND Obstacle Direction is Left TURN RIGHT

The network structure according to this rule set is given in Figure 4-14. In this set, there are no middle values for membership functions. The inputs related with direction are classified as left or right and front is discarded and there is no action for going forward. When the network is trained with these rules, going forward action is simulated by oscillating left and right directions.

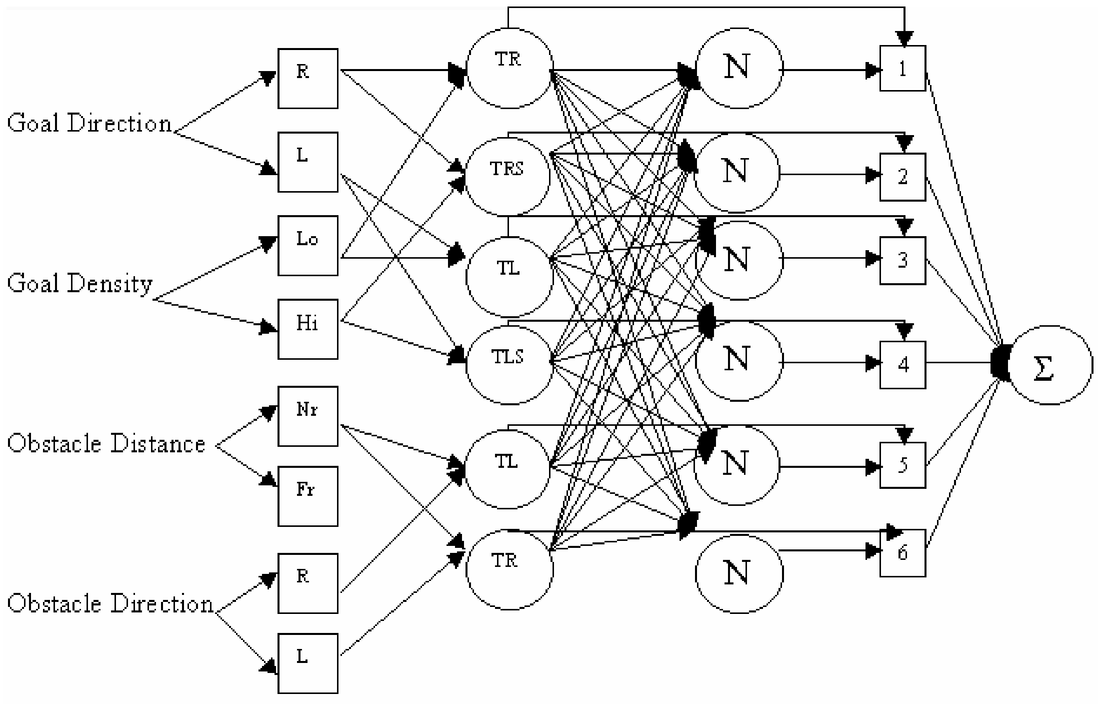

Figure 4-14 Network Structure for First Rule Set

In the second set middle values for membership functions and a rule for going forward is added.

In second set,

Now there are 3 membership functions for direction inputs:

- Goal Direction: Left, Front, Right
- Goal Density: Low, High
- Obstacle Distance: Near, Far
- Obstacle Direction: Left, Front, Right

And 8 rules (2 rules added to first set):

1. If Goal Direction is Right AND Goal Density is Low TURN RIGHT
2. If Goal Direction is Right AND Goal Density is High TURN RIGHT SHARP
3. If Goal Direction is Left AND Goal Density is Low TURN LEFT
4. If Goal Direction is Left AND Goal Density is High TURN LEFT SHARP
5. If Goal Direction is Front GO FORWARD
6. If Obstacle Distance is Near AND Obstacle Direction is Right TURN LEFT
7. If Obstacle Distance is Near AND Obstacle Direction is Left TURN RIGHT

7. If Obstacle Distance is Near AND Obstacle Direction is Front TURN RIGHT

The network structure for second rule set is given in Figure 4-15. Because second rule set is more complete and it handles more situations than first set, robot learns faster and moves smoother with this set. When there is an obstacle in front, robot tends to turn right because of the last rule.

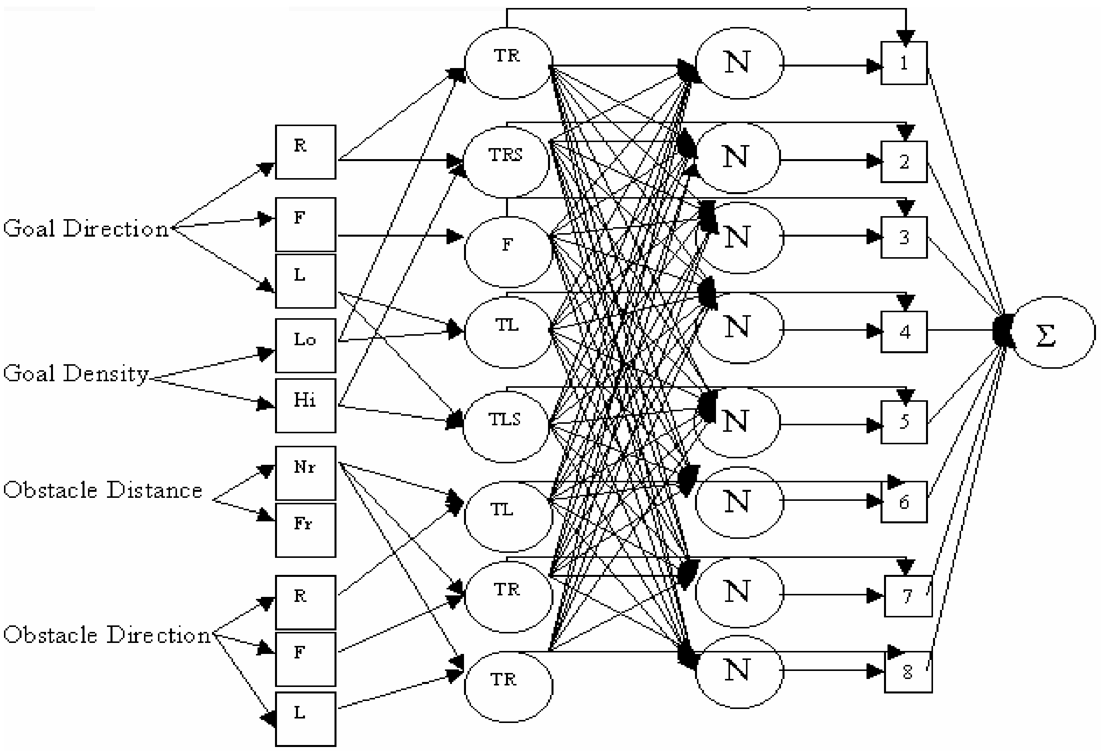

Figure 4-15 Network Structure for Second Rule Set

The error analyses of two sets are given in Figure 4-16. Second rule set gives better results according to first set.

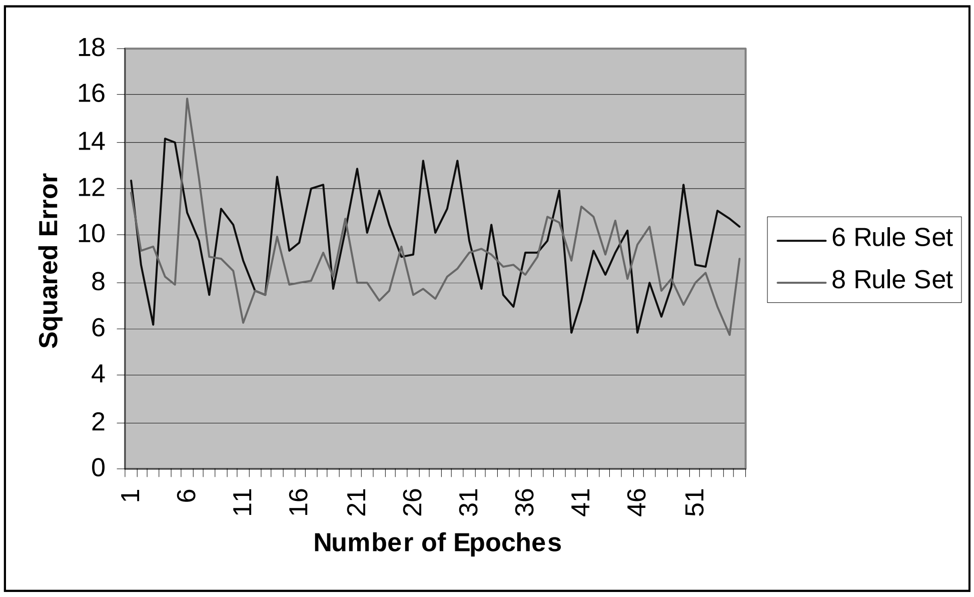

Figure 4-16 Error Analysis with Two Rule Set

If the complexity of the network increases the performance of the robot decreases. Learning of behaviors becomes more difficult as the number of connections between the nodes increases. The complexity of the network is determined by the number of inputs, number of membership/consequent functions, number of rules and membership/consequent functions' complexities. As the number of nodes and connections between them increases, the complexity of the network increases. Furthermore, if complex algorithms are used in the membership/consequent functions, the complexity of the overall network increases.

3. Robot is tested with the world that has walls which is shown in Figure 4-17

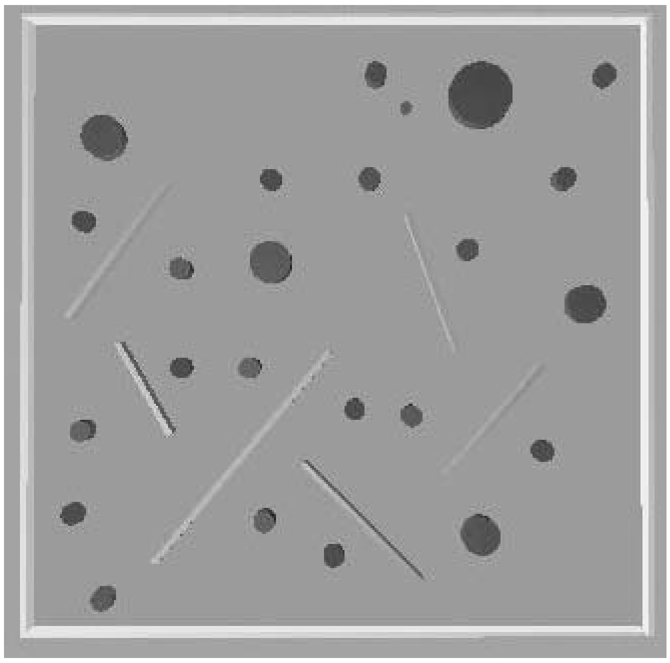

Figure 4-17 World Model With Walls

In this world model robot learns faster according to world model without walls, because by following walls robot reaches the goal objects easily. But if the walls prevent the robot to see and reach the goal objects, the performance of robot decreases. Comparison is shown in Figure 4-18 with the two world models. World model without walls is given in Figure 4-11.

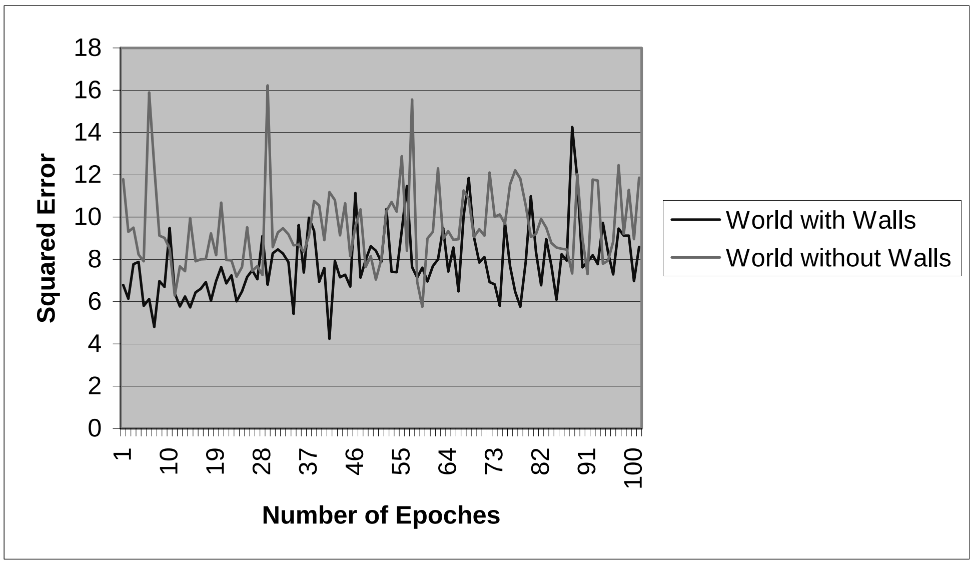

Figure 4-18 Error Analysis for World Model With Walls

4. The robot is tested in dynamic environment, where the goal objects are not stationary, after training. When there are moving goal objects in the world, the robot followed them successfully. The error analysis for a world model with dynamic goal objects are given in Figure 4-19:

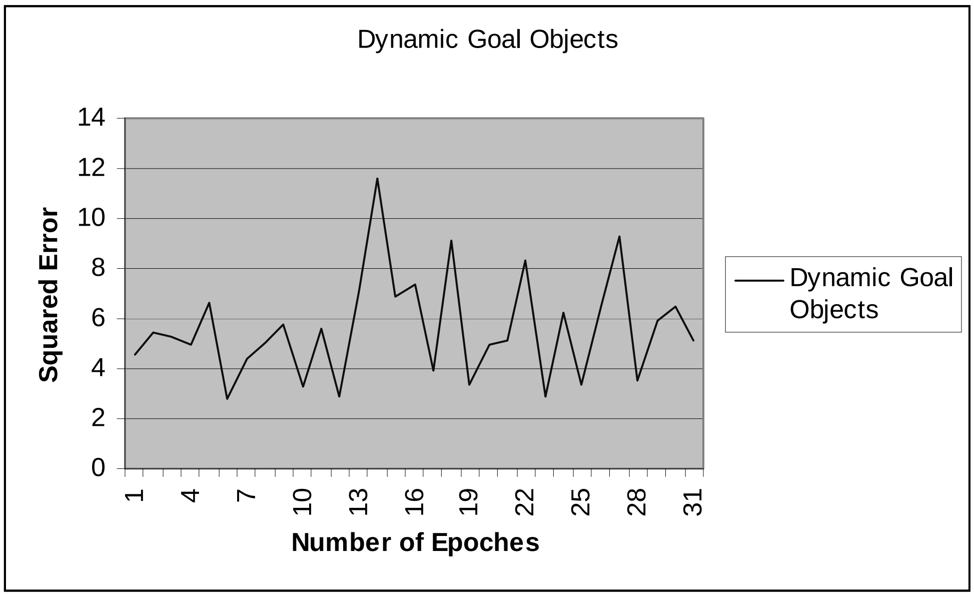

Figure 4-19 Error Analysis for Dynamic Environment

4.5 Comparison With Previous Studies

Construction of a well performing fuzzy system is not always easy. The problem of finding appropriate membership functions and fuzzy rules is often a tiring process of trial and error. But usage of learning algorithms with fuzzy systems helps to overcome this problem. The learning capabilities of neural networks made them prime target for a combination with fuzzy systems in order to automate or support the process of developing a fuzzy system for a given task. And in this study neural network is used for learning parameters of membership and consequent parameters of fuzzy system

In a similar study to this study was made by Zavlangos and Tzafestas [33] which is an intelligent navigation system for omnidirectional mobile robots based on fuzzy logic. The objective is finding collision free trajectories for a robot, in static or dynamic environments containing some obstacles between start and goal configuration. The fuzzy rule base of the system combines the repelling influence, which is related with distance and the angle between the robot and nearby obstacles, with attracting influence produced by the angular distance between the actual direction and position of the robot and final configuration, to generate a new actuating command. The disadvantage of this system is membership functions are fixed and determined by the developer. Finding the optimum parameters for membership function is a problem in fuzzy systems without using learning algorithms. These membership functions are peculiar to the worked environment and robot characteristics and are not adaptable to different conditions, like different sensors. As an advantage of our study, membership and consequent function parameters are not fixed. They are adapted with online learning using ANFIS for Go to Goal and Avoid Obstacle Behaviors.

In their study J.Cao and X. Liao [34] proposed a reactive navigation system using neuro-fuzzy techniques. The purpose of the robot is to follow a path identified by two dashed lines. The robot current angle with respect to the line direction and the robot offset with respect to centerline is used as inputs and output is defined with three actions: turn right, turn left and go straight. In this system, the parameters of membership/consequent functions and connection weights between nodes in the network are learned. When the weight of a rule or membership function is close to zero, they are deleted and new ones are added.

Rule generation is very difficult in complex systems especially when the number of inputs is high. Pruning techniques from neural networks can be used to reduce the number of rules in neuro-fuzzy systems. It is also possible to use fuzzy clustering methods to find fuzzy rules and initialize system with them. One possibility to learn rules is to start with a system that contains all possible fuzzy rules that can be defined according to partitioning of inputs. Thus the system begins with an inconsistent rule base, which is made consistent by learning. The idea of decremental rule learning is to try out existing rules and to evaluate them. Rule units that do not pass this test are eliminated from the network. This learning algorithm becomes very expensive, if there are a lot of fuzzy sets defined for a lot of variables.

For this reason, it is useful to use knowledge about the system to generate initial rules and avoid generating all possible rules. Incremental rule goes in the opposite way, creates a rule base from scratch by adding rule by rule. NEFCON [35,36] system supplies both incremental and decremental rule-learning techniques. Disadvantage of ANFIS is that there is no learning procedure defined for a given neuro-fuzzy model and structure of network is fixed.

Another fuzzy navigation study was developed by Benreguieg [37], which is also tested on Khepera. This system has a planner and navigator divisions. Planner part generates paths for the robot to achieve goal points with using A* algorithm and visibility graph. The navigator handles goal-seeking and avoid obstacle behavior with using fuzzy logic. In this study complete world knowledge is needed for the planner and the world must be static. In our study no deliberative planning is made, it is a reactive system that acts according to local data. The advantage of our system is to adapt changing environments without using a world model.

There is a trade off in neuro-fuzzy approaches. To obtain high performance, complex training algorithms based on gradient descent algorithms which demand Sugeno or Tsukamoto (Section 3.2) type fuzzy systems is needed. When Mamdani type fuzzy systems are used which are easier to interpret, fast heuristic for training can be used but achieve lower performance[36]. In ANFIS hybrid learning of gradient descent and LSE is used which causes high computational complexity but good performance.

In this study, ANFIS is applied with Tsukamoto inference system. The output of each rule is defined as a crisp value induced by the rule’s firing strength. The overall output is taken as the weighted average of each rule’s output. Since each rule gives a crisp output and the aggregation is made by weighted average, this model avoids the time-consuming process of defuzzification.

CHAPTER 5

CONCLUSION

In robotics most applications of fuzzy logic concern the use of fuzzy control techniques to implement individual behaviors. Fuzzy controllers are a convenient choice when an analytical model of the system to be controlled cannot be easily obtained, and there is variability and uncertainty in the parameters. These characteristics fit well for reactive behaviors because a mathematical model of the environment is usually not available, sensor data is uncertain and imprecise, and real time operation is essential. And also fuzzy logic provides a methodology for representing human expert knowledge and perception-based actions without needing analytical model of the system.

However, identifying optimum fuzzy membership and consequent functions is a tedious task. To solve this problem learning algorithms are used. Neural networks have an important application area in the field of mobile robotics because of their adaptive and learning properties. Several studies are made using this hybrid model of fuzzy logic and neural networks [38-40].

In this study a behavior-based navigation strategy using neuro-fuzzy approach is presented. ANFIS is applied in a mobile robotics problem with Tsukamoto style inference system. The resulting architecture is adaptable to different objectives and robust to environmental changes. The robot successfully navigates in a dynamic environment, find and reach to target objects. When there are walls in the environment that lead the robot to reach goals, the performance of robot increases.

This structure has advantages over analytical methods because of fuzzy logic. With using fuzzy logic rules the robot behaviors are defined as simple and understandable and human expert knowledge is added to system easily. The usages of rules to define the behaviors give the flexibility to define different behaviors and objectives with update of these rules. So the system is adaptable to different objectives. Second, because of the interpolative nature of fuzzy control the movement of the robot is smooth without abrupt changes that may be caused by sensor values.

The system is adaptable to different conditions with the usage of neural network for learning the fuzzy membership and consequent functions. Because the membership functions are not fixed when the input values range change because of different environmental conditions or usage of different sensors, the system adapts to these conditions by updating membership function parameters.

In our study, because of behavior-based strategy is used; the system has a modular structure that can be extended easily to add new behaviors. In this architecture multiple behaviors are combined into a unified action without abrupt transitions.

As a future work, the capabilities of the robot control system can be extended by adding new behaviors. A behavior remembering previously visited goal object positions and encouraging the robot to go unvisited sections of the world can be added. And also different characteristics can be used other than color to identify goal objects in the world. Rule generation functionality can be added by modifying ANFIS system.

In this study, a matrix vision turret is used to get the image of the world seen by the robot. And this instrument is an extension for Khepera robot, which does not exists in robot at hand. So it was not possible to apply the method to the real robot. As a future work, after this extension turret is gathered the proposed method can be tested on real robot.

With enhancing this study target object identification may be changed and this system can be used in real world problems when the goal is finding target objects in an unknown environment.

REFERENCES

[1] Rodney A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, Vol. RA-2, No.1, pp 14-23, March 1986.

[2] S. G. Goodridge and R. C. Luo. Fuzzy behavior fusion for reactive control of an autonomous mobile robot*: MARGE*. *In Procs. of the IEEE Int. Conf. on Robotics and Automation*, pp 1622-1627, San Diego, CA, 1994.

[3] M. Boden. AI's Half Century, *AI Magazine*, Vol. 16, No.2, pp 96-99, 1995.

[4]R. Beer. Intelligence as Adaptive Behavior: An Experiment in Computational Neuroethology. *Academic Press*, New York, NY.1990.

[5] R. Beer, H. Chiel and L. Sterling. A Biological Perspective on Autonomous Agent Design. *Robotics and Autonomous Systems*, Vol. 6. pp.169-86, 1990.

[6] Marvin Minsky. *The Society of Mind*. Simon and Schuster, New York, 1985.

[7] Ronald C. Arkin. Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior. *In Procs. of the IEEE Int. Conf. on Robotics and Automation*, pp 264-271, 1987.

[8] R. C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, Ma, 1998.

[9] A. Saffiotti. *The use of Fuzzy Logic for Autonomous Robot Navigatio*n. Soft Computing, Vol. 1(4), pp 180-197, 1997.

[10] L. A. Zadeh. Fuzzy Sets. *Information and Control*, No 8, pp. 338-353, June 1965.

[11] T. Takagi and M. Sugeno. Derivation of fuzzy control rules from human operator’s control actions. *Procs. of the IFAC Symp. on Fuzzy Information, Knowledge Representation and Decision Analysis*, pp 55–60, July1983.

[12] A. Saffiotti, E. Ruspini, K. Konolige. *Using Fuzzy Logic for Mobile Robot Control in International Handbook of Fuzzy Sets*. Kluwer Academic Publishing, 1999.

[13] M. Benreguieg, H. Maaref, and C. Barret. Fuzzy helps to control a motion of a mobile robot in a partially-known environment. *In Procs. of the European Congress on Fuzzy and Intelligent Technologies (EUFIT),* pp 905-909, Aachen, DE, 1995.

[14] J. Zhang and J. Raczkowsky an A. Herp. Emulation of spline curves and its applications in robot motion control. *In Procs. of the IEEE Int. Conf. on Fuzzy Systems,* pp 831-836, Orlando, FL, 1994.

[15] K. Tanaka and M. Sano. *Trajectory stabilization of a model car via fuzzy control. Fuzzy Sets and Systems*, Vol. 70, pp 155-170, 1995.

[16] J. L. Mart'inez, A. Ollero, and A. Garc'ia-Cerezo. Fuzzy strategies for path tracking of autonomous vehicles. *In Procs. of the European Congress on Fuzzy and Intelligent Technologies (EUFIT),* pp 24-30, Aachen, DE, 1993.

[17] A. Ollero, A. Garc'ia-Cerezo, J. L. Mart'inez, and A. Mandow. *Fuzzy tracking methods for mobile robots*. In M. Jamshidi, A. Titli, L. Zadeh, and S. Boverie, editors, *Applications of fuzzy logic: Towards high machine intelligence quotient systems*, chapter 17. Prentice-Hall, New Jersey, 1997.

[18] Y. Makita, M. Hagiwara, and M. Nakagawa. A simple path planning system using fuzzy rules and a potential field. *In Procs. of the IEEE Int. Conf. on Fuzzy Systems*, pp 994-999, Orlando, FL, 1994.

[19] M. Sugeno and M. Nishida. *Fuzzy control of model car*. Fuzzy Sets and Systems, Vol. 16, pp 103-113, 1985.

[20] T. Takeuchi, Y. Nagai, and N. Enomoto. Fuzzy control of a mobile robot for obstacle avoidance. *Information Sciences*, Vol. 43, pp 231-248, 1988.

[21] A. Saffiotti, E. H. Ruspini, and K. Konolige. Blending reactivity and goal directedness in a fuzzy controller. *In Procs. of the IEEE Int. Conf. on Fuzzy Systems*, pp 134-139, San Francisco, California, 1993.

[22] A. Saffiotti. Fuzzy logic in the autonomous mobile robot Flakey: on-line bibliography. URL: http://iridia.ulb.ac.be/saffiotti/flakeybib.html

[23] H. Surmann, J. Huser, and L. Peters. A fuzzy system for indoor mobile robot navigation. *In Procs. of the IEEE Int. Conf. on Fuzzy Systems*, pp 83-86, Yokohama, JP, 1995.

[24] E. Tunstel, H. Danny, T. Lippincott, and M. Jamshidi. Autonomous navigation using an adaptive hierarchy of multiple fuzzy behaviors. *In Procs. of the IEEE Int. Sym. on Computational Intelligence in Robotics and Automation*, Monterey, CA, 1997.

[25] E. H. Ruspini. Fuzzy logic in the Flakey robot. *In Procs. of the Int. Conf. on Fuzzy Logic and Neural Networks (IIZUKA)*, pp 767-770, Iizuka, JP, 1990.

[26] A. Saffiotti, K. Konolige, and E. H. Ruspini. A multivalued-logic approach to integrating planning and control. *Artificial Intelligence*, Vol. 76(1-2), pp 481-526, 1995.

[27] Jyh-Shing R. Jang. ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Trans. Systems, Man & Cybernetics*, Vol. 23, pp 665-685, 1993.

[28] P. Werbos. *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. PhD Thesis,Harvard University, 1974.

[29] Y. Tsukamoto. An approach to fuzzy reasoning method. *Advances in Fuzzy Set Theory and Applications.* pp 137–149. North-Holland, Amsterdam, 1979.

[30] C.C. Lee. Fuzzy logic in control systems: fuzzy logic controller-part 1. *IEEE Trans. on Systems, Man, and Cybernetic*s, Vol. 20(2), pp 404–418, 1990.

[31] C.C. Lee. Fuzzy logic in control systems: fuzzy logic controller-part 2. *IEEE Trans. on Systems, Man, and Cybernetic*s, Vol. 20(2), pp 419–435, 1990.

[32] F. Mondada, E. Franzi and P. Lenne. Mobile Robot Miniaturisation : A Tool for Investigation in Control Algorithms. *3rd Int. Symposium on Experimental Robotics.* pp 336-341, Kyoto, Japan, 1993.

[33] P. G. Zavlangas, Prof. S. G. Tzafestas, Dr. K. Althoefer. Fuzzy Obstacle Avoidance and Navigation for Omnidirectional Mobile Robots. *ESIT 2000,* Aachen, Germany, 14-15 September 2000.

[34] J. Cao, X. Liao and E. Hall. Reactive Navigation for Autonomous Guided Vehicle Using the Neuro-fuzzy Techniques. *Procs of the SPIE Intelligent Robots and Computer Vision Conference XVIII*, Boston. MA, 19-22 September 1999.

[35] D. Nauck and R. Kruse. NEFCON-I: An X-Window based simulator for neural fuzzy controllers. *In Procs. IEEE Int. Conf. Neural Networks 1994 at IEE WCCCI'94*, pp 1638-1643, Orlando, FL, June 1994.

[36] D. Nauck Neuro-Fuzzy Systems: Reviews and Prospects. *Fifth European Congress on Intelligent Techniques and Soft Computing (EUFIT'97)*, Aechen, 8-11 September 1997.

[37] M.Benreguieg, P.Hoppenot, H.Maaref, E. Colle, C.Barret. Fuzzy Navigation Strategy: Application to Two Distinct Autonomous Mobile Robots. *Robotica*, Vol. 15, pp 609-615, 1997.

[38] C. Wharehinga, W. L. Xu. Behaviour Based Fuzzy Navigation of A Mobile Robot, *Projects*, Vol. 9, 2002

[39] H. Seraji, A. Howard. Behavior-Based Robot Navigation on Challenging Terrain : A Fuzzy Logic Approach. *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 3, June 2002

[40] C. Moraga. Neuro-Fuzzy Modeling with Standard Feedforward Neural Networks, *Proc. European Syposium on Intelligent Techniques ESIT 2000,* (CD) Aachen, 2000