University of Alberta

Situation Based Navigation of Autonomous Mobile Robots using Reinforcement Learning

by

Rhea Lynn Guanlao 

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**

Department of Electrical and Computer Engineering

Edmonton, Alberta
Fall 2004

Your file Votre référence
ISBN: 0-612-95758-6
Our file Notre référence
ISBN: 0-612-95758-6

"This is an exercise in science, or science fiction, if you like that better. Not for amusement: science fiction in the service of science. Or just science, if you agree that fiction is a part of it, always was, and always will be as long as our brains are only miniscule fragments of the universe, much too small to hold all the facts of the world but not too idle to speculate about them."

- Valentino Braitenberg

This thesis is dedicated to my parents,
Remigio David Guanlao and Rosalyn Fabella Yaneza Guanlao

Acknowledgements

First and foremost, I wish to express my sincere gratitude to my supervisor, Dr. Petr Musilek, for the support, guidance, and valuable opportunities he provided me with. His encouragement has helped me and I have learned a lot from him throughout my graduate experience. I will take this with me always.

I would also like to thank Dr. Michael Bowling and Dr. Marek Reformat for reviewing this thesis and providing invaluable and insightful feedback. I also thank Dr. Richard Sutton for his comments on this research.

In the FACIA lab, I've been surrounded by friends whose knowledge, advice and company have helped me often. I'd like to thank Armita and Farooq for their contributions to this research. Our scattered ideas grew into what is now this thesis. Also, thanks to Nitin for his opinions and endless conversations and arguments, on things both meaningless and meaningful. Most of all, I'd like to thank these three for their friendship and stimulating forms of comic relief. I'd also like to acknowledge Yifan for his advice and interesting discussions. To the rest of the FACIA lab including Loren, our Spanish visitors, our summer students, and our close neighbours, you have all made grad school an unforgettable and amusingly enjoyable experience.

Finally, I have to thank and remember those closest to me, who have been supportive, constantly reminding me of the most important things in life. To Mariam, who instilled a passion of music in me – a gift I will always appreciate. To my friends, who never cease to bring a sense of balance, escape and laughter into my life. Thank you for always being there. To Marco, who has brought me happiness. His strength and patience meant a lot to me and helped through many difficult moments. To my family, for always being a constant source of friendship, love and encouragement. To Ryan, for always being a caring big brother and friend. And especially to my parents, they have raised me and set a foundation of expectations, while still being

compassionate and trusting in me and my decisions. It amazes me how parents unconditionally give of themselves so their children may prosper – I thank them. And of course, I am forever grateful to the Lord who always sustains and lives in me...

Contents

List of Tables

List of Figures

1 Introduction

In his historical study on the progress of robots since the earliest existence of electric computers, Moravec preconceives that "Artificial minds for mechanical bodies capable of autonomous manual work finally seem within reach" [57]. The prospects of this statement rely on the challenging and ongoing research in the field of autonomous mobile robots.

1.1 Preface

Along with hardware and processing limitations, the challenges in the field of mobile robotics stem from the large and complex spaces encompassed by the sensors and actuators, making it difficult to draw meaningful information and to create intuitive sensor-actuator mappings. The research in this thesis attempts to overcome these challenges by developing a controller capable of satisfying the following two

autonomous objectives:

1. Self interpretation of the input space
2. Self-determination of the mapping between the interpretation derived from Objective 1 and the actuator space

Since it is one of the most important features required by a functional mobile robot, *navigation* is the particular application field of mobile robotics chosen in which to implement these objectives. The navigation task involves purposefully traversing unfamiliar environments while dealing with noisy, imperfect sensor information. While several control schemes have been developed to accomplish the navigation task, the behaviour based control paradigm continues to be an essential element of autonomous mobile robot navigation. Behaviour based controllers often incorporate reactive control schemes that use range sensors such as sonar and laser to provide responsive control in dynamic and uncertain environments.

Fuzzy logic control is well suited for behaviour based approaches towards navigation as it can be made forgiving against sensor noise and imprecise or ambiguous inputs that inherently exist in real world environments. Fuzzy logic controllers (FLCs) have been implemented in many successful autonomous navigation systems [68]. However, while navigational behaviours based on fuzzy logic designs can produce good performance, the designs often require human intuition and an *ad hoc* trial and error approach that is inflexible and time consuming to implement. In recent research, the adoption of learning strategies, such as evolutionary and reinforcement learning, have attempted to address this problem by automating the design of low level behaviours and behaviour integration [30], [79]. However, the large state space involved in mobile robot navigation often slows the learning of navigation tasks. Attempts to lower the dimensionality of the state space by grouping input data using expert knowledge only shift the element of input interpretation from one design stage

to another. An effective way to reduce the state space dimensionality is to cluster input data using a fuzzy clustering algorithm, so that the resulting fuzzy clusters can then be considered as fuzzy states or situations by an algorithm that can learn the association between situations and actuator values. Sonar data from sample indoor environments are partitioned with a fuzzy clustering algorithm. The resulting partitions may not correspond to the state space partitions that a human expert would use, but they would provide a good description of typical situations the robot will encounter in its environment.

Although the fuzzy clustering of robot data into situations occurs prior to the execution of the fuzzy control system, rule learning is online and is applied to navigational behaviours such as obstacle avoidance and wall following. Each fuzzy situation is associated with an action to be taken in that state; and a reinforcement learning algorithm is used to associate punishments generated from a reward function to the state-action pair that caused the collision so that over several iterations an optimal policy is learned. This proposed system is referred to as an *autonomous situation-based fuzzy control system* for the purpose of navigational mobile robots. As today's field of mobile robots is still far from developing completely autonomous robots that are both useful and effective, this research attempts only to incorporate features into the fuzzy control system (FCS) to increase its autonomy in accomplishing navigational tasks.

1.2 Thesis Organization

This thesis is organized in seven sections. Chapter 2 provides a literature review discussing how the following related challenges have been dealt with in the past: robot navigation, input interpretation and rule base learning. Chapter 3 summarizes background knowledge of topics which deserve a thorough introduction with respect to this thesis. These topics include mobile robots, the navigation problem, fuzzy rule

based systems, clustering, and reinforcement learning. The proposed autonomous situation-based FCS for robot navigation is described in Chapter 4. In this chapter, the proposed design is applied to a single low-level navigation behaviour: *obstacle avoidance.* The results are observed and analyzed in Chapter 5. With successful results, Chapter 6 investigates whether a similar system can be used towards learning other navigational behaviours by applying it towards the wall following task. As well, this chapter studies the need to combine and fuse behaviours in order to create more purposeful autonomous robots. The two behaviours fused together are obstacle avoidance and goal finding. Finally, Chapter 7 brings main conclusions and indicates possible directions of future work.

"Really we create nothing. We merely plagiarize nature."

– J. Baitaillon

2 Literature Review

This literature review examines the previous work related to the research described in this thesis. The general challenge of this research deals with the problem of mobile robot design and navigation. Therefore, first and foremost, the history of mobile and navigational robots is discussed. Secondly, several approaches that have been used to induce autonomy and intelligence in mobile robots are reviewed.

In this thesis, a FCS is the approach taken towards inducing intelligence in a mobile navigational robot. Furthermore, in this thesis, two main objectives in the realm of the FCS will be actualized in attempts to increase its autonomy. These two objectives are input self-interpretation and the autonomous design of the fuzzy rule base. Thus, the final section of this Chapter details previous work toward fulfilling these two objectives.

2.1 History of Mobile Robots

The study of behavioural mobile robots began in the 1950s when W. Grey Walter [78] experimented with robots that possessed simple reactive behaviours (i.e., ELSIE and ELMER). With an almost direct relationship between its sensors and actuators, minimal computation was involved. He studied how these simple, collision-retracting phototropic robots interestingly conducted themselves with response to their environment. Thus, from the demonstrations, he concluded that complex behaviours can emerge from simple systems. Following Walter's observations, Braitenberg [15] continued to build vehicles with simple electrical connections between two light sensors and two motors. Reminiscent of Walter's ELSIE and ELMER, Braitenberg's vehicles were either inhibitory or excitatory towards light, depending on the configuration of its interconnections. Observing the behaviours of these vehicles without any prior knowledge of its design would lead one to believe that they encompassed a rather sophisticated level of intelligence. This was the fundamental idea of Braitenberg's research, which furthered the robotics field of *thinking* vehicles.

Since the beginning of Walter's experiments, however, traditional artificial intelligence (AI), a hierarchical approach, dominated the field of robot control for about 30 years starting from its emergence in the latter half of the 20th century. It was based on a *sense-plan-act* (or *sense-plan-execute*) schema in which internal world representations, and the manipulation of these rich internal representations using deliberative reasoning methods, were used to plan the course of action [19]. The *sense* and *execute* components were considered to be low-level aspects while the *plan* component was considered as a high-level aspect [69]. In 1971, one of the first mobile navigational robots, SRI International's Shakey [59], was built on this overtly deliberative and logical planning architecture. Experiments followed in which Shakey navigated in a set of predetermined environments to complete tasks. Its

logical foundations and high-level cognitive skills appeared attainable and attractive, however several issues arose.

The main problem that arose (in Shakey, and in other mobile robots based on traditional AI) was the intensive computation required by the logical reasoning and deliberations in the planning stage. This resulted in a failure of classical methods with its struggle of performing and processing in real time. Complimenting this problem was the fact that world models and symbolic representations were stored prior to execution. The drawback lied in the basis that planning is performed in the world model rather than the real world. Thus, the resulting action is executed blindly of the real world, ignoring unexpected characteristics or new changes that might occur in the environment. This resulted in the difficulty of integrating world models and the inability to adapt to new and dynamic environments.

With the discouraging prospects of traditional AI, researchers shifted from desiring sophisticated cognitive robots to desiring more practical and reactive behaviours in the decades following Shakey. Therefore, the need was to abandon representation and extensive planning, and tighten the coupling between the sensing and execution stages so that the interaction between the robot and the environment was more intimate. Brooks argued against the traditional artificial intelligence in navigational robotics [18] stating that the reliance on representation is lost when intelligence is approached in an incremental manner with a strict reliance on interfacing to the real world through perception and action. The real world is so complex that a world model rarely exists that could fully represent it. With mobile navigational robotics, "the world is its own best model" [16], thus all information about the world is derived from its sensors and there is no need for an internal model of the environment. This eliminated the *plan* component and favored a *sense-execute* sequence. Several other researchers agreed with this criticism of traditional AI based robots (e.g., [2], [50], [40]). Subsequently, Brooks helped to revive the study of behavioural robotics since Walter

and Braitenberg, and proposed the subsumption architecture – a modular top down approach to behavioural robotics. In the subsumption architecture, the execution layer is divided into simpler behaviours that run in parallel. Each behaviour has a separate task and each is closely reactive to the environment. Since the introduction of this architecture, behavioural based mobile robots became more accepted.

Despite the popularity of behavioural and reactive robotics, traditional AI, or hierarchical, methods have not been abandoned. In fact, representational models and planning are still widely studied in their application towards robotics (i.e., partially observable Markov decision processes, Bayesian Nets, hidden Markov models, linear dynamical systems, particle filters, etc). Through the growing research in both subfields of robotics, a hybrid approach has emerged combining both behavioural and hierarchical methods. Arkin suggested that "The false dichotomy that exists between hierarchical control and reactive systems should be dropped" [3]. The strong intentionality of hierarchical approaches were useful in guiding reactivity at a higher level (e.g., [4], [29], [48]).

2.2 Navigation and Control for Autonomous Mobile Robots

Following the movement against hierarchical AI robotics, the study of behavioural based mobile robots that exhibited more intelligence and autonomy increased in the 1980s. Researchers desired their robots to *learn* its behaviours from the environment, rather than explicitly telling their robots how to react. For example, Brooks and Matarić developed mobile robots that successfully learned how to navigate online from maps that it would build as it wanders the environment [53]. In the task of designing behavioural mobile robots that navigate intelligently and autonomously, as opposed to

the simple sense-react robots constructed by Walter and Braitenberg, several machine learning and computational intelligence techniques have been applied for both control and optimization schemes. The following sections include a brief survey of methods commonly utilized in mobile robotics for their sophisticated sensory-actuator mapping approaches.

2.2.1 Fuzzy Control Systems

FCSs have been often used as a solution to controller designs for mobile robots. Their rule based system assists in good reasoning and decision making skills, fulfilling an important aspect of *intelligence*. They are also beneficial because their IF-THEN rule structure allows an easy mapping between sensory and actuator information. Equally important, they are able to forgivingly process the uncertain and ambiguous data that naturally exists in real environments. Saffiotti, Ruspini, and Konolige designed one of the first notable fuzzy logic controllers for a navigational mobile robot using SRI's International mobile robot Flakey as a testbed [66]. Flakey had 12 behaviours and each behaviour had 4-10 predesigned rules. The behaviours were successful, however, the control rules were written based on the designers' intuition of the desired performance and constant debugging and tuning of the parameters occurred through several trial and error experiments. Other examples that incorporate fuzzy methods as controllers for mobile robots are found in [71], [49], [74]. These and other FCSs are successful in utilizing linguistic rules and performing smooth transitions between them. However, there is still a high reliance on the human designer to specify the rules, functions and parameters of the knowledge base. Thus, the performance of the FCS also depends on the humans perception of the real-time environment before execution. This comes as a disadvantage as no real *learning*, nor adaptiveness to unexpected environments, is incorporated in a robot that is based solely on fuzzy

logic. Rather, an FCS incorporates sophisticated processing and reasoning of real world data.

2.2.2 Neural Controllers

Artificial neural networks have been suggested as a promising method for controlling mobile robots due to their resemblance to natural biological systems with adaptive characteristics. They are well suited for learning numerical data pairs, such as sensor-actuator pairs for mobile robots. In their review on neural approaches for navigation, Kröse and van Dam [44] emphasize the advantages of neural control: no world models, as in traditional AI, or predefined rules, as in fuzzy logic controllers, are required *a priori.* They can be well trained to learn reactive behaviours for mobile robots due to the strong relationship they provide between the sensors and actuators. Another advantage is that self-supervised learning is possible with the training data being provided from the robot's inputs, outputs, and their corresponding performance with the environment. Thus, training can continue as the environments change, allowing the emergence of an adaptive characteristic towards dynamic worlds. With the high-level information and symbolic knowledge that are stored in a neural network's intricate internal structure, neural networks have been successful in learning and adapting complex behaviours. Kröse and van Dam describe a controller that learns to avoid obstacles from negative feedback through the network whenever a collision occurs. However, after overcoming the issue of input interpretation via self-organization schemes, they still encountered problems of misclassification of *safe* and *dangerous* states. The network of another similar example of a neural controller, ANNIE, improves on Kröse and van Dam's controller because it further classifies the strength of the feedback value to varying degrees, rather than just indicating the presence or non-presence of a collision [42]. Although ANNIE was successful,

the main disadvantage of its controller, and neural controllers in general, is that an optimal network structure is unknown before training and training takes a long time since the learning begins from scratch. As well, the fusion of two or more behaviours becomes problematic, particularly when the behaviours are contradicting. This is due to the strong reflexive nature of neural networks. Their proficiency in learning numerical data pairs makes it difficult for neural networks to learn and distinguish multiple behaviours. This is exemplified in [26] in which a neural-based robot oscillates back and forth in response to the obstacles in its path while it is trying to reach a goal

2.2.3 Reinforcement Learning

Reinforcement learning [72] has its advantages in a diverse range of applications, including mobile robots, because of the generality of the problem specification. The theory that behaviours can be learned by reinforcing positive and negative behaviours has been applied to several robot navigation applications (e.g., [6], [32], [51], [54], [63]).

Millán [54] lead the design of reinforcement learning robots and built TESEO, a navigational robot that was able to learn efficient obstacle avoiding and goal finding behaviours through reinforcement learning. TESEO generalized its input situations and output reflexes to further reduce the possible mappings between the two spaces. By continually punishing unsuccessful situation-reflex pairs, and rewarding successful pairs, TESEO also learns incrementally, however the rate of learning is greater since the search space is significantly reduced through the generalized situations. Furthermore, with reinforcement learning's emphasis on predicting which pairs receive the most cumulative rewards, TESEO learns how to avoid encounters with undesirable and dangerous situations, even when its sensors cannot detect obstacles. This is an

advantageous characteristic of reinforcement learning – to foresee which situation-reflex pairs are most beneficial in the long run. In TESEO, however, perceived situations are added incrementally and discrete classification of the state to the situations is rather ambiguous. TESEO, and other reinforcement learning based controllers, still have the challenge of efficiently classifying the situations while, at the same time, they are learning how to map the situations and reflexes. As well, with reinforcement learning's focus to learn situation-action pairs, it would also undergo the same difficulties in learning multiple behaviours as neural networks. Nevertheless, reinforcement learning is adaptive and continuous and, with an efficient generalization scheme for its input and output spaces, it is quite suitable for online learning.

Acquiring navigation behaviours through reinforcement learning and neural networks have several overlapping characteristics. Both learn sensor-actuator pairs, both can effectively learn through interaction with the environment and both incrementally update its weights or utility values towards reaching an optimal stable system. Many neural networks actually exploit the temporal-difference methods and reward functions of reinforcement learning algorithms. A difference lies, however, in that reinforcement learning focuses on *evaluating* the actions taken rather than *instructing* which actions to take. In addition, reinforcement learning with a minimized rule space is advantageous over pure neural controllers with complex architectures. The same simple sensory-actuator mapping developed in reinforcement learning is distributed throughout the complex set of weights of the network making learning less generalized and more complicated.

2.2.4 Hybrid Methods

Evidently, machine learning and computationally intelligent methods used singlehandedly have their advantages and drawbacks. Many attempts to combine

these methods are beneficial to capitalize on each method's strengths and to compensate for each method's weakness. For example, evolutionary neural networks have been used in designing robot navigational controllers and have been demonstrated to perform properly as classifier systems [38] that exhibit adaptive behaviour. Such hybrid classifiers benefit from the genetic algorithm's optimization capabilities and explorative direct search, particularly of large spaces. From the neural network, they benefit from the ability of multi-layer perceptrons to encode sophisticated numerical behaviours and from the network's ability to learn through back propagation methods. Some examples of applying evolutionary neural networks can be found in [23], [58]. Among the growing number of hybrid architectures, a few notable ones are genetic fuzzy systems [12], evolvable neural networks [7], uniform coevolution [9], fuzzy neurons [62], [35], reinforced fuzzy systems [11], reinforced neural networks [47] and reinforced evolutionary algorithms [25], [43]. Section 2.3 further discusses the combination of fuzzy systems with other machine learning methods in learning its parameters or rules.

2.3 Fuzzy Systems

Following the notion that the combination of two or more learning methods can benefit from the individual advantages of each method and compensate for each other's weaknesses, the situation-based FCS proposed in this thesis exploits the benefits from clustering, reinforcement learning and fuzzy systems. Fuzzy systems suffer from the fact that a human expert is required to design its rules and membership functions (input interpretation). The input interpretation for the fuzzy system is handled by a clustering technique that generalizes the environment into situations. The design of rules is handled by the reinforcement learning technique. These two enhancing methods are combined with the reasoning and decision making skills of the fuzzy

system. In the following two sections, alternative methods that handle the issues of input interpretation and rule learning for fuzzy systems are discussed.

2.3.1 Input Interpretation for Fuzzy Logic Controllers

Sensory input interpretation deals with interpreting the environment information from sensors. This is difficult because even a simple robot can have several sensors and, with the wide range of possible values, this can create an enormous number of possible input combinations that the robot controller must understand. As a result, the input space of a robot's sensory values is large and complex and the ability to draw meaningful information from the sensors is a challenging task. Ideally, input interpretation involves perceiving the environment through extracting specifically relevant information and understanding or recognizing its significance.

As mentioned in Section 2.2.1, a disadvantage of fuzzy logic controllers is the need for an expert to design the knowledge base functions and parameters. These components direct the fuzzy logic controller in interpreting the potentially large and complex input space. As a result, generalization or classification schemes should be applied to the input space.

Several classification methods have been used to fuse and interpret multiple sensory inputs into situational data. For example, TESEO incorporated incremental classifications of situations as it explored the environment. However, this was problematic because situational classes were created even before the environment was fully explored. Hoffmann used feed-forward neural networks to classify different environment situations given the robot's sensor data and steering angles. From the neural network, the controller is able to distinguish the sonar data into eight different basic situations (e.g., dead end, corners, walls) [36]. Similarly, Provost *et al.* used self organizing maps [41] to develop a set of higher level perceptual features from the low level sensory inputs [64]. Researchers who have used evolutionary

techniques towards learning the parameters of the membership functions in genetic-fuzzy systems (e.g., [21], [52]) encode the directions for input interpretation within its chromosomes. Although it provides great encoding flexibility and a diverse search for input membership functions and parameters, the range of possible values is typically so large that convergence is difficult. Finally, researchers have also used clustering techniques as a method of reducing the dimensionality of the input sensory information [60]. Clustering measures the similarity between sensory data and forms groups of data that can represent higher level situations. In Brooks' and Matarićs' experiments in which robots learned by building maps of its environment [53], clustering techniques were used to classify objects such as walls, corridors, etc. This provided a successful generalization of the robots surrounding environment. Furthermore, by calculating the belongingness of the state to each cluster, the input space is heavily reduced from generalizing several sensors, and their respective ranges, to single situational clusters. However, some precision of the robot's state is inherently lost with this generalization scheme.

2.3.2 Fuzzy Rule Base Learning for Fuzzy Logic Controllers

Recently, much research has been done towards the use of fuzzy controls in directing autonomous robots. The problem in fuzzy logic controllers lies in the need for the experience and intuition of an expert in designing the controller's rule base. The ability of the fuzzy logic controller to learn and self-adapt its own rules would increase the level of its autonomy.

The neuro-fuzzy approach has been gaining popularity in recent years. Because of their complex nature, fuzzy neural networks are able to embed the insightful symbolic knowledge of the fuzzy model. As well, neural learning capabilities aid in adapting to dynamic environments. Some background and applications of neuro-fuzzy methods can be found in [55], [20], [73], [79]. Although several researchers proved successful

in learning navigational behaviours via the neuro-fuzzy approach, many issues arose such as the difficulty in determining the network architecture, long training times, and difficulty in providing *learnable* training data (in supervised and unsupervised neural learning).

The application of evolutionary algorithms towards the learning of fuzzy rules, as well as parameters (i.e., genetic-fuzzy systems), has also been widely investigated. With the large and complex search space involved in determining the optimal rules and the optimal parameters of the membership functions, evolutionary algorithms benefit from their flexible encoding schemes and evolutionary operators (i.e., mutation and crossover). A well defined fitness function and appropriate selection strategies facilitate population convergence to individuals representing optimal fuzzy models. In the Michigan Approach, individuals in the population represent separate rules, thus the fuzzy rule base is comprised of all the individuals in the population [35]. The problem arises in the credit assignment of the rules. It is difficult to evaluate competing rules when they *all* combine their properties to generate the actions of the system. Thus, a compromise between cooperation and competition [12] is needed to find an entire rule base whose individual rules successfully collaborate to meet the desired performance. This problem might be solved with the Pittsburgh approach in which each individual in the population encodes the entire fuzzy rule base and/or knowledge base parameters [35]. Evaluation of each individual is more straightforward such that each individual can be evaluated in the environment separately from others. The disadvantage, however, is that with the large population sizes and high number of generations generally required with evolutionary algorithms, the evaluation of each individual would be computationally intensive and time consuming. Several genetic-fuzzy systems, particularly those applied to navigational mobile robots, may be found in [36], [37], [31], [46].

Finally, reinforcement learning can be used for learning the fuzzy rule base of

a navigational robot controller (e.g., [11], [8]). Bonarini has studied the learning of fuzzy rules through both evolutionary algorithms and reinforcement learning [14]. He developed a robot, ELF, which demonstrated an agent that can learn reactive behaviours, strategic behaviours, behaviour coordination and multi-agent coordination through reinforcement learning of the fuzzy rule base [13]. The system was successful and was able to adapt to unknown environments. Thongchai experimented with the similar method in an FLC that learned two navigational tasks: obstacle avoidance and landmark detecting [75]. Although his robots were successful in learning both behaviours, as with Bonarini, he had several inputs and several fuzzy sets for each input. This created a large number of possible input-output mappings, thus, learning was time consuming and difficult.

"There is no practical reason to create machine intelligences indistinguishable from human ones. People are in plentiful supply. Should a shortage arise, there are proven and popular methods for making more."

– Anon

3 Background

The word *robot* derived its meaning from the Czech word *robota*, meaning *forced labor* – after a play R.U.R.: Rossum's Universal Robots (1920) by playwright K. Capek in which mechanical human clones were created to be slave workers. Brooks defined a *robot being* as a robot which 'lives' in the world, carrying out its own agenda of on-going projects, while maintaining the necessary balance with its environment to ensure its continued successful operation [17]. More generally, robots are defined as machines that are able to extract information from their environment and use knowledge about their world to act safely in a meaningful and purposive manner [5]. The interaction with their surrounding environment may be strictly controlled and predefined or it may be more independent and autonomous.

3.1 Mobile Robots

Mobile robots have the following major components:

Sensors. Robots are equipped with sensors that allow them to perceive information about themselves and the environment. There are generally two classes of sensors. Proprioceptive sensors are internal sensors that give information about the robots internal status. These sensors provide information of the robots odometry, speed, acceleration, position, etc. Exteroceptive sensors are sensors that provide information of the robots external environment. These sensors provide information of the objects surrounding the robot through range finders (i.e., sonar, infrared, laser), cameras, tactile sensors, etc.

Actuators or Effectors. In order to carry out tasks and to respond to the environment, robots are also equipped with actuators. There are a wide range of actuators, but among the most common are limbs, grippers and motion actuators that control the robots velocity, heading and acceleration (i.e., motors, wheels, legs, etc.).

Controllers. Controllers are needed to determine the most appropriate actuator values for the given situation of the robot and its environment.

3.2 Navigation Problem

Autonomous robot navigation is a necessary characteristic in mobile robots which entails moving purposefully through environments while executing behaviours that maintain a desired performance or complete particular tasks. In doing so, robots must combine and integrate multiple sensors, have many degrees of freedom and make control decisions in order to meet real-time deadlines. The navigational behaviours

could include reaching a goal, following walls or objects, keeping track of its position relative to the environment (self-localization), object recognition and manipulation, creating a map of its surroundings (mapping), path planning, etc.

Three navigational tasks which will be further studied in this thesis are obstacle avoidance, wall following and goal finding. The safety aspect of navigation can be handled by various methods such as obstacle avoidance. Obstacle avoidance typically uses information about the robot's immediate surroundings to choose an action (i.e. heading) appropriate to avoid obstacles in the environment. Such information is provided through means of the robot's sensory subsystem, equipped with proximity sensing devices. Wall following is another behaviour which could also be considered as a safety component in some environments. Otherwise, it is a navigational behaviour in which a particular distance between the robot and its adjacent wall must be maintained. The challenge arises in determining the heading adjustments it must make in response to the various positions it could have with respect to the wall. The problem of goal seeking is that of performing actions to react from a current position towards the desired location despite any obstacles or barriers that may exist in its path.

3.3 Clustering

Clustering, or cluster analysis, is a technique for partitioning and finding structures in data [39]. By partitioning a data set into clusters, similar data are assigned to the same cluster whereas different data should belong to different clusters. The underlying motive for clustering is to be able to draw structure and natural groupings, and to prospectively extract significant conclusions about the data as a whole. While it is possible to assign each data-point strictly to only one cluster, as in conventional clustering techniques such as K-means clustering [34], such crisp assignment rarely captures the actual relationship and similarity among the data, i.e., data-points from

the real world can simultaneously belong to multiple clusters to varying degrees.

A variant to conventional discrete clustering methods leads to the formulation of fuzzy clustering [10], where membership degrees between zero and one are used instead of crisp assignment of the data to clusters. In addition to better conformity, fuzzy clustering provides a simple interpolation between the cluster prototypes and an accurate representation of the relationships among the data. This latter property is very important when considering clustering in connection with rule-based systems as described in Chapter 4.

The fuzzy c-means clustering algorithm [10] is a commonly used method of fuzzy clustering. It is based on the minimization of an objective function J with respect to U, a fuzzy partition of the data set, and V, a set of c prototypes. Alternatively, U is the membership matrix of the data points and cluster centers and V is the cluster centers matrix. The following equation is the objective function J of the order m

$$J_m(U,V) = \sum_{j=1}^{n}\sum_{i=1}^{c} u_{ij}^m \|X_j - V_i\|^2 \qquad \{j = 1 \ldots n, i = 1 \ldots c, m \geqslant 1\} \tag{3.1}$$

where $\|*\|$ is any norm expressing the similarity between the data points and cluster centers (e.g., Euclidean distance). This similarity measure between two points, i and j, can also be expressed as D_{ij}. Although the Euclidean distance is one of the most common measures of similarity between two points used in clustering techniques, other such similarity measures, such as Canberra, Squared Chord, or Squared Chi-squared, are further discussed in [70].

The membership of data-point j to cluster i, denoted by u_{ij}, is determined by

$$u_{ij} = [\sum_{k=1}^{c} (\frac{D_{ij}}{D_{kj}})^{\frac{2}{m-1}}]^{-1} \tag{3.2}$$

The partition order, m, affects the degree of fuzziness. If $m = 1$, then the partition is hard or discrete as in K-means clustering. As m increases, the fuzziness of the

partition increases and becomes maximally fuzzy ($u_{ik} = \frac{1}{c}$) as $m \longrightarrow \infty$.

The number of cluster prototypes, c, must be specified prior to clustering in most clustering algorithms. However, if the optimal number of clusters is not properly selected prior to execution, it is likely that the resulting clusters will not closely match the natural structure of the data. Thus, different numbers of clusters should be experimented with. The optimal number of clusters for a set of data is such that distances between the data and their respective clusters (weighted by their memberships for fuzzy clustering) is minimal while the distances between the clusters is maximal. There are some techniques supported by this principle, such as *subtractive clustering* [22], which estimate an appropriate number of clusters for a given data set.

The fuzzy partition is eventually obtained through the iterative optimization of the objective function in Equation 3.1 with the gradual update of memberships u_{ij} and cluster centers V_i following Algorithm 1. Updating the cluster centers such that they move towards more centroidal positions within their respective partitions ensures minimization of the objective function since it enforces distinctive high and low memberships.

3.4 Fuzzy Control Systems

A fuzzy control system (FCS) is a controller that performs mapping from the space of situations that the robot can encounter to the space of actions the robot can perform through the use of fuzzy logic. FCSs can be used to extend conventional approaches to mobile robot navigation using the notion of fuzzy behaviours [79]. In the case of navigational mobile robots, fuzzy logic can be used in deciding which actions to take (e.g. velocity, direction) based on the crisp values from its surrounding environment. The advantage is that it allows the intuitive nature of navigation to be easily modelled

Algorithm 1 Fuzzy C-Means Clustering algorithm

1: Choose T as the maximum number of iterations
2: Choose ε as the maximum change in cluster positions to indicate the convergence of cluster centers.
3: Choose c as the number of the cluster centers for cluster center vector V
4: Guess the initial positions of the cluster vector randomly V
5: **repeat**
6: **for all** $u_{ij} \in U$ **do**
7: Calculate membership u_{ij} of each data point j to each cluster i using Equation 3.2
8: **end for**
9: **for all** $v_i \in V$ **do**
10: Update center clusters v_i using the following averaging equation such that the cluster centers are placed in more centroidal positions with respect to its current position within the data. Thus, the objective function in Equation 3.1 is minimized.

$$v_i = \frac{\sum_{j=1}^{n}(u_{ij}x_j)}{\sum_{j=1}^{n} u_{ij}}$$

11: **end for**
12: **until** $t = T$ or $|V_t - V_{t+1}| < \varepsilon$

using linguistic terminology. And with its intelligent reasoning, it allows for smooth uninterrupted robot motion [76]. In order to further understand how FCSs operate, a review in fuzzy logic is deserved.

The notion of a *set* is implied by the encapsulation of objects into a collection in which the members of the collection all share some general feature or property [62]. Zadeh claimed that many *sets* in the world that surround us are defined by non-distinct boundaries [81]. With discrete belongingness to sets, semantic value or meaning is surely lost. Fuzzy belongingness, through *fuzzy sets*, encompass a more accurate and representational description of the compatibility between an element and a set. Fuzzy systems convert real-world input values into their respective degrees of memberships to these fuzzy sets so that approximate reasoning can be used in decision making.

Membership functions are used to determine the degree of belongingness of an element or value to a set. This process of evaluating the membership degrees of crisp data to each corresponding fuzzy set is called fuzzification and the resulting membership grades are the fuzzified inputs. The membership of the element x to fuzzy set A given the universe of discourse X is between 0 and 1, inclusively, and is denoted by

$$\mu_A(x) : X \rightarrow [0, 1].$$

In fuzzy systems, although not limited, these membership functions often take on the following various shapes: triangular, trapezoidal, gaussian, singleton, sloped ramp, etc. Each membership function represents a linguistic label, in which a linguistic variable has one or more linguistic labels to describe it. For example, *Temperature* is a linguistic variable. Its linguistic labels could include *hot*, *warm* and *cold*. Membership degrees of an element x to *hot*, *warm* and *cold* can be evaluated to describe the temperature of x.

Fuzzy behaviours then associate the fuzzified inputs with an appropriate fuzzy action through fuzzy rules. The fuzzy rules together form the inference engine, or the rule base, of the fuzzy logic controller. The fuzzy rule base is a set of IF-THEN or IF-THEN-ELSE rules in which the antecedents and consequents of the rules are made up of the linguistic values pertaining to given membership functions [45]. Generally in mobile robotics, the antecedents refer to sensory data while the consequents refer to actuator control signals. The structure of a fuzzy rule with two antecedents and one consequent is illustrated below

$$\text{IF } x \text{ is } A \text{ and } y \text{ is } B \text{ THEN } z \text{ is } C,$$

where x and y are elements in the universes of discourses X and Y, respectively, and A and B represent the linguistic labels of their corresponding fuzzy sets.

According to the *Min-Max* fuzzy logic inference method, the minimum numeric value among the antecedents is the strength of the rule. For each rule, the linguistic label(s) of the consequents accordingly take on numeric value of its rule strength. Then under composition, each linguistic label of the system takes on its maximum numeric value determined among all the rules in the rule base. The numerical values assigned to each consequent linguistic label are the fuzzified outputs. More information on the Min-Max inference method and other fuzzy logic inference methods can be found in [24]. Finally, defuzzification is used to convert the fuzzified outputs into a single crisp output. In the Center of Gravity [56], the output variable is determined by finding the centroidal position among of the membership functions for that fuzzy linguistic variable according to Equation 3.3. More information on the Center of Gravity method, and other defuzzification methods can be found in [62].

$$y = \frac{\sum_{k=1}^{n}(w_k \cdot x_k)}{\sum_{k=1}^{n}(w_k)} \tag{3.3}$$

Figure 3.1 depicts the workings of a FCS with two inputs and one output via the fuzzy logic components: fuzzification, rule evaluation, and defuzzification. Triangular membership functions are used for the input fuzzy sets, and singleton membership functions are used for the output fuzzy sets. There are 4 membership functions for each input and 4 membership functions for the output. Thus, there are 8 input fuzzy sets and 4 output fuzzy sets. The components combined generate intelligent reasoning and inference processes for decision making. As a result, FCSs provide a smooth interpolation between a set of rules.

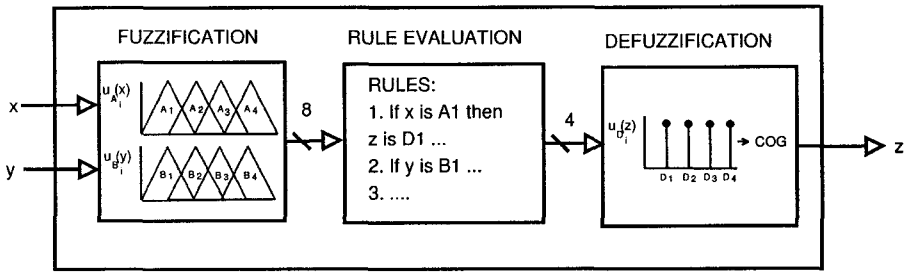

Figure 3.1: A high level diagram of a typical FCS

3.5 Reinforcement Learning

Reinforcement learning problems exhibit online learning and dynamic adjustments in order to search for optimal methods and actions to maximize cumulative rewards over time. Some examples include adaptive control systems, game playing machines, and autonomous robot navigation and exploration. Evolutionary heuristics such as genetic algorithms or genetic programming have been previously applied in solving such problems. However, they lack in the sense that they require time consuming stochastic searches of large and complex spaces. Instead, a more general machine learning technique, that is closely dependent on interactions with the environment, would

be more efficient. Reinforcement learning is favored for its generality and similarity to how humans think and learn on a higher level – through experience. Sutton describes reinforcement learning as a computational approach to understanding and automating goal-directed learning and decision-making [72]. The learning agent determines which actions yield the most rewards in a particular environment by exploring and performing several different actions repeatedly thus learning which actions to exploit.

The underlying theories of reinforcement learning include the following: trial and error, optimal control and temporal-difference (TD) learning. Trial and error is essentially performing actions and determining which are beneficial by observing the error they acquire from the environment. This is an important aspect of reinforcement learning as experience is gained and exploited from the rewards and penalties received. Optimal control is controlling a system, through developing control policies, such that its dynamical behaviour over time is optimized. Dynamic programming (DP) and Markovian decision processes (MDPs) are used in solving optimal control problems. However, although MDPs and DP require complete knowledge of the system, reinforcement learning problems do not always fulfill this requirement, nor do they always fulfill the Markov property which states that the future of the system is independent of its path. Therefore, reinforcement learning often deals with approximating MDPs, in particular, finite MDPs (i.e., MDPs in which the states and actions are finite). Finally, TD learning is driven by the difference between temporally successive elements. It uses sampling from the environment, an idea derived from Monte Carlo methods. Consequently, the incorporation of TD-learning allows reinforcement learning systems to utilize the ideas of DP and approximate MDPs without having complete knowledge of the system – thus, a model is not required. DP is important in reinforcement learning particularly because of the *bootstrapping* method in which estimates are learned from previous

estimates. This allows a system to gradually reach an optimal function with successive approximations. Eventually, the system converges deterministically to a single answer as long as the learning rate remains small.

A reinforcement learning system consists of a set of defined attributes:

- The **environment**, or the **model** of the environment, provides the perceived **situations** (or **states**) and stimuli for the robot to interact and learn from.
- **Actions** are the possible executions the learning agent may perform with response to the perceived situation.
- **Utility values** are assigned to each situation-action pair to indicate how beneficial it is to perform that action in the perceived state. The utility values are assigned through the action-value function.
- **Rewards** are the immediate reinforcements the agent receives to specify the desirability of the performed action in the perceived state. The rewards are defined by the reward function.
- The **reward function** defines and enforces the desired goal of the learning problem by assigning the rewards to situation-action pairs depending on how they perform in the environment.
- The **behaviour policy** determines how the learning agent responds to the environment by indicating which actions to execute in the perceived state. It instructs the agent to either explore the actions and expand its learning experience, or to exploit the actions it already knows will gain positive rewards. Behaviour policies may either be stochastic allowing an explorative search (i.e., soft policies), exploitive through repeatedly choosing the actions with maximal utility values (i.e., greedy policies), or a combination of both (i.e., soft-max policies).

Reinforcement learning relies on a continual combination of exploration and exploitation. Exploration facilitates visiting new parts of the input space to expand the learning experience. Exploitation utilizes the knowledge gained from exploration to increase the rewards gained from the environment. Ideally, a progressive combination of the two should be performed to continually evaluate and improve the agent.

- The **state-value function** ($V^{\pi}(s)$) is the expected return when starting in the current situation and following the given policy π. Therefore, it gives an indication of the desirability of the state s.

- The **action-value function** ($Q^{\pi}(s,a)$) is the expected cumulative return when taking the particular action a in state s and following the given policy. This function learns which actions produce the highest rewards in the long run or after several states, as opposed to which actions produce the highest rewards in the immediate state. For example, a particular action may receive a low immediate reward, but it may produce higher cumulative rewards in the states to follow than another action that has a higher immediate reward.

- **Optimal value functions** (π^*) determine the policy which receives the most possible rewards over the long run (i.e., Greedy policy). The optimal state-value function is given as $V^*(s) = maxV^{\pi}(s)$ and the optimal action-value function is given as $Q^*(s,a) = maxQ^{\pi}(s,a)$.

Algorithm 2 describes how the above attributes of reinforcement learning are combined.

From the constant interaction with the environment and from the concept of *delayed rewards* integrated into the value function, actions affect the immediate reward, as well as the subsequent rewards afterwards. This allows the learning agent to effectively maximize rewards and optimize performance over time.

Algorithm 2 Reinforcement Learning

1: Define the conditions of a *terminal* **state**
2: Define the possible **actions** and perceived **states**
3: Initialize the **utility values** of the **state-action** pairs
4: **repeat**
5: Observe perceived **state** from the *environment*
6: From the **behaviour policy**, determine an **action**
7: Perform the determined **action**
8: From the **reward function**, observe the **reward** resulting from the **action**
9: From the **value function**, update the **utility value** of the perceived **state** and the **action** performed
10: **until State** is terminal

"Third Law of Robotics: A robot must protect its own existence as long as such protection does not conflict with the First of Second Law."

– Isaac Asimov

4

Situation-Based Fuzzy Control System for Autonomous Mobile Robots

---

FCSs are advantageous for their good reasoning capabilities and their ability to handle ambiguous and vague inputs. The navigational system proposed in this chapter exploits these benefits of FCSs. Furthermore, the proposed system attempts to increase the autonomy of a typical FCS (Figure 3.1) through the modification of the fuzzification and rule evaluation components.

4.1 Overall Structure

The proposed navigational system described in this chapter is referred to as an *autonomous situation-based FCS*. It attempts to meet two objectives: self-interpretation of the input space and self-determination of its fuzzy rule base. It

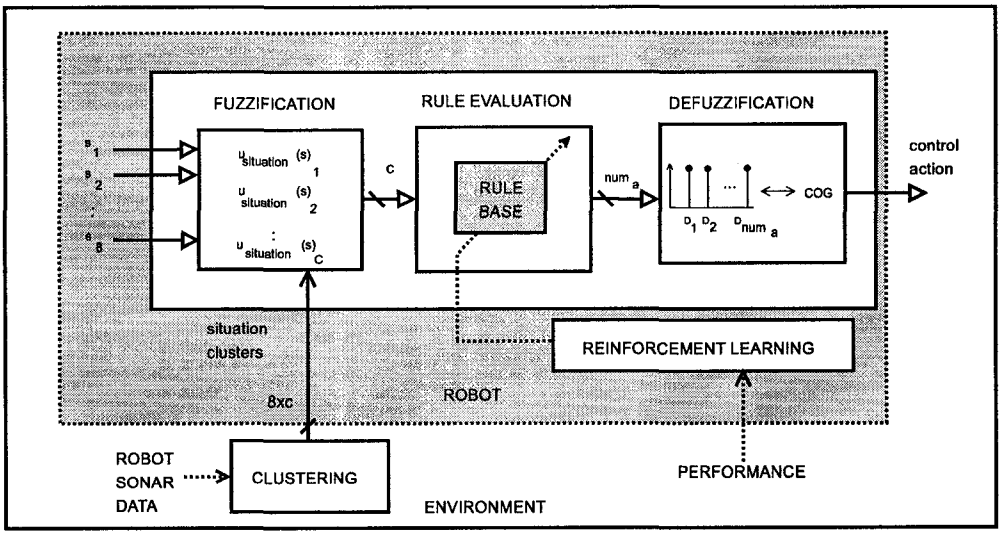

Figure 4.1: Diagram of the proposed situation-based FCS

incorporates two features to meet these two objectives:

- The **situation-based** component is used to interpret the robot's input space. In the fuzzification step, a membership function determines how closely the robot's current state belongs to each fuzzy set, such that each fuzzy set represents a particular situation the robot could encounter with respect to its environment. These situations are determined prior to execution of the FCS through the off-line fuzzy clustering of the robot's sensory space.

- The **autonomous** component is based on reinforcement learning and it used to acquire and adapt the FCS's fuzzy rule base. From clustering of the robot's sensory space, the resulting situational clusters are the appropriate antecedents of the fuzzy rules in the rule base, while the consequents express the actions the robot can execute in response to the situations. For the rule evaluation step, the rules themselves are initially random and unknown. They are learned on-line through the process of reinforcement learning.

The overall structure of the proposed situation-based FCS is illustrated in Figure 4.1. It is a modified structure of the typical FCS shown in Figure 3.1 where the inputs are the 8 sonar range values (s_i) and the output is a control action (i.e. heading). c is the number of clusters and num_a is the number of possible actions for the single control action. The membership function $u_{situation_i}(s)$ determines the belongingness of the sonar data s to the fuzzy set $situation_i$.

The proposed system is used to learn navigational behaviours for mobile robots, where a *behaviour*, intuitively, is a particular control regime that focuses on achieving one specific, predetermined goal [67]. The predetermined goal used for these experiments is *obstacle avoiding*. Therefore, the consequents of the fuzzy rule base is the turning angle or the relative change of the robot's current heading. In expanding the performance of the FCS, velocity optimization is then incorporated as a second consequent in Section 4.7.

As seen from Chapter 2, fuzzy systems, reinforcement learning and fuzzy clustering have been previously used and experimented with in their application towards robot navigation. The novelty of the approach in this thesis, however, lies in the hybridized combination of these three methods. Reinforcement learning has been shown to be useful in learning beneficial long term actions in an online environment (i.e., Section 2.2.3). Subsequently, it is also efficient in responding to changing and unexpected environments. Reinforcement learning, however, suffers from the *curse of dimensionality* – the amount of computation will increase with an increase in the number of inputs or state variables. Thus, clustering of the robot's sensory data serves to achieve two aspects: it interprets and draws meaningful information from the crisp input data and it overcomes the curse of dimensionality by reducing the state variables. The resulting reinforcement learning system is not a finite MDP – states and actions are fuzzy. Therefore, the application of reinforcement learning onto a FCS requires modification of the value function to account for the varying degrees

of membership to states and actions. The intent is to take advantage of FCSs and incorporate efficient fuzzification schemes and a simple rule structure in which the rules are learned autonomously through interacting with the environment.

4.2 Robot Simulation and Environment

The simulations model the Pioneer 2DX mobile robot from Activmedia [1]. For sensing, it has 16 proximity range sonars placed around its body. For control purposes, it has a left and right motor which, in combination with its 3 wheels, are used to control the robot's velocity, acceleration, displacement and rotational movements. Two large wheels are placed in parallel on each of the robot's sides. Each of these side wheels is connected to a motor, while the main functionality of the back wheel is to keep the robot steady (Figure 4.3b). In this research, only 8 front sonars are used. They are spaced nearly evenly from 90 degrees at its left side, and around its front to -90 degrees at its right side (Figure 4.3a). The maximum range of the proximity sonars is 3000mm. The experiments in this thesis model the Pioneer 2DX robot in a simulated environment.

During the simulations, the velocity and heading are controlled by the FCS. The velocity is either constant at 150 mm/s, or it varies between 100 mm/s and 300 mm/s. The maximum possible velocity of the Pioneer 2DX is 300 mm/s. The relative heading varies between -90 left to 90 right. The FCS determines the heading and/or velocity values after each iteration and commands the robot to move in the acquired direction and speed. For each iteration, the robot maintains its command by continuing its current motion for 1500ms. If a collision occurs, the robot is programmed to retract slightly back and forth, adjusting it's heading relative to the position of the obstacle it collided with, to help it recover from stalled positions.

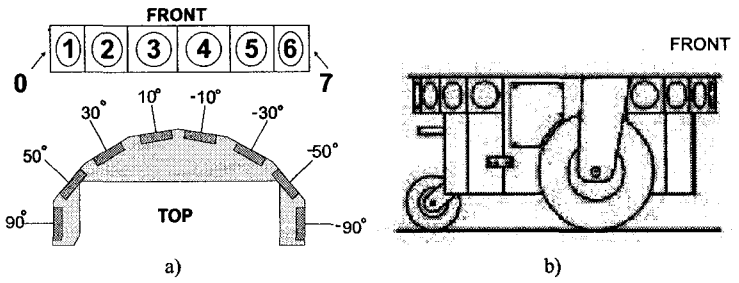

Figure 4.2: Pioneer 2DX a) Sonar placement b) Wheel placement

The robot is placed in two-dimensional rooms equipped with walls, corridors and polygon obstacles placed at varying angles. The robot is represented by a small circle with a line stemming from its center to an edge of the circle. The position of this line determines the direction in which it is facing. The rooms are approximately 10m x 10m, while the robot itself is about 30cm in diameter. Figure 4.4 is a sample 13m x 8m environment used for the training of obstacle avoidance. The placement of its walls and objects should create an environment that provides sufficient availability of the possible situations. From exploring such an environment, the robot will encounter as many diverse situations as possible and will thus increase its experience.

4.3 Data Collection

In order to collect the sonar data required for clustering, experiments have been performed by placing the robot into simulated 3m x 6m environments. Figure 4.4 represents the five room environments created for the collection of data. The robot was allowed to traverse the trajectories in theses rooms indicated by the arrows. The numbers indicate the robot's path by representing the order in which it followed the lines. The environments and forced trajectories attempt to encompass all the possible

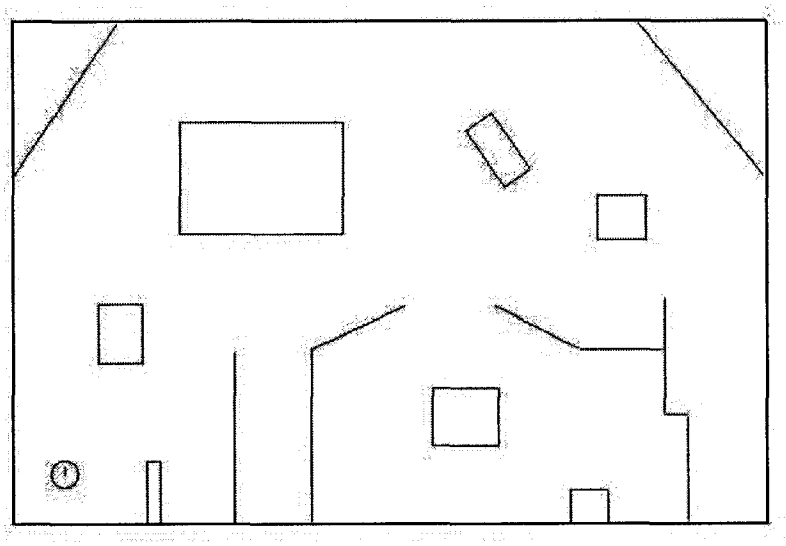

Figure 4.3: Training environment for learning the obstacle avoidance behaviour.

situations the robot may encounter in its environment. Two 8 dimensional data points of the robot's sonar range values were recorded every second, running at a velocity of 150mm/s. From the data collection scheme described, about 1500 data points were collected.

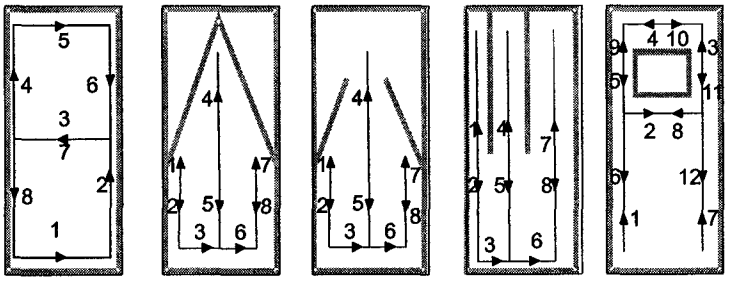

Figure 4.4: Data collection environments and trajectories

4.4 Fuzzy C-Means Clustering

The sonar data in Section 4.3 was acquired to obtain a partition of the 8 dimensional sonar space to the space of fuzzy situations (e.g., *open space*, *right corner*, etc.). Although the situations can be labelled and interpreted this way, the labels only serve to gain insight into the results of clustering and are not relevant for the robot and its navigation system.

Clustering was performed using the *fuzzy c-means* clustering algorithm discussed in Section 3.3, using the Euclidean distance as the similarity measure D_{ij}. The Euclidean distance measure was used since it was experimentally found to be effective in determining the similarity between two situations in an 8 dimensional space. In practical applications, there are several parameters of the clustering algorithm to be chosen that have a profound impact on quality of the resulting partition, particularly the number of clusters c and the partition order m.

The number of clusters, c, should be large enough such that an overgeneralization of the data does not occur. As well, it should be particularly large enough so that the situation-based FCS is able to distinguish important situations from one another, significantly affecting the potential learning capabilities of the robot. On the contrary, having a large number of clusters would result in a time consuming and computationally intensive learning period of the FCS due to the large number of state-action pairs. This would defeat the initial purpose of the input generalization. Thus, a compromise must be made for c. By examining the environment used for data collection, and the desired behaviour of obstacle avoidance, 8 clusters were selected to represent 8 potentially significant situations. This number is somewhat flexible and can be increased to allow for greater resolution of the situational space, if needed. However, depending on the desired navigational behaviour, more clusters may be required for some behaviours due to the need for some behaviours to have a more precise interpretation of the environment. A separate parallel experiment is also

demonstrated with 12 clusters to compare the learning process and performance of the two systems with a different number of clusters.

The partition order, m, should be greater than 1 since $m = 1$ constitutes a hard partition. Additionally, m should be a relatively low number since large values of m (i.e. $m \rightarrow \infty$) cause maximal fuzziness. This means that any situation instance of the robot would belong to each cluster by equal degrees of $m \simeq \frac{1}{c}$. This makes it very difficult to distinguish situations from one another. Typical values of m in several fuzzy clustering experiments were chosen to be $m = 2$. Experimentally, this value still caused a high degree of fuzziness and situational distinction was difficult. Thus, the value of $m = 1.2$ was determined as a good compromise.

There are also several cluster validity measures [39], [61], [77], [65] that can be used to evaluate the quality of the partition and may also aid in the process of selecting particular values of c. In fuzzy c-means clustering, the objective function (Equation 3.2) is minimized in order to maximize the memberships of each data point to its closest cluster. Likewise, depending on its definition, each validity measure should either be maximized or minimized so that distinct clusters are formed and good inter-cluster separability is achieved without forming a hard partition. Since discrete clustering is also not desirable, setting the value of m as $1 < m \ll \infty$ ensures that a hard partition is not formed despite the tendencies induced by these measures. Along with the objective function in Equation 3.2, there have been seven validity measures used to evaluate the quality of the partition. They are used in determining parameters such as c and m. Also, varying partitions form with each clustering trial due to the different initial guess of the cluster prototypes. Thus, these validity measures can also help determine suitable partitions.

1. Partition coefficient V_{PC}

$$V_{PC}(U) = \frac{\sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik}^2}{n} \tag{4.1}$$

2. Partition entropy V_{PE} where

$$V_{PE}(U) = -\frac{1}{n}\{\sum_{k=1}^{n}\sum_{i=1}^{c}[u_{ik}log_a(u_{ik})]\}, a \in (1,\infty) \tag{4.2}$$

Bezdek [10] defined the partition coefficient V_{PC} and the entropy coefficient V_{PE}. V_{PC} validates the partition among the data since it is proportional to the squared sum of the data-cluster memberships. The maximum value is $V_{PC} = 1$ with a hard partition and its minimum value is $V_{PC} = \frac{1}{c}$ for the fuzziest partition possible (i.e. $u_{ik} = \frac{1}{c}\forall i,k$). V_{PE} is appropriate in validating the number of clusters since $0 \leq V_{PE} \leq log(c)$, where $V_{PE} = 0$ with a hard partition and $V_{PE} = log(c)$ with its fuzziest partition. Therefore, to ensure good inter-cluster separation, V_{PC} should be maximized and V_{PE} should be minimized.

3. Non-fuzzy index

$$NFI(c) = \frac{c[\sum_{k=1}^{K}\sum_{i=1}^{c}u_{ik}^2] - n}{n(c-1)} \tag{4.3}$$

The non-fuzzy index [65] provides another measure of how fuzzy a c partition is: it takes its maximum value for crisp partitions and its lowest value in the case of fuzziest clustering. Analogously to V_{PC}, NFI should be maximized.

4. Minimum hard tendency

$$MinHT = max\{-log_{10}(T_s)\}, 1 \leq s \leq c, \tag{4.4}$$

5. Mean hard tendency

$$MeanHT = \frac{1}{c}\sum_{s=1}^{c} -log(T_s) \tag{4.5}$$

where

$$T_s = \frac{\sum_{x_i \in Y_s} r_{i_s}}{card(Y_s)} \tag{4.6}$$

and

$$r_{i_s} = \frac{u_{ki}}{u_{ji}} \tag{4.7}$$

Minimum and Mean Hard Tendency measure the tendency of the data to hard clustering [77]. u_{ki} is the strongest membership value of a data point x_i to cluster s for all i in which x_i exists in cluster s. Likewise, u_{ji} is the second strongest membership value of another data point to cluster s among all x_i. Therefore, r_{i_s} compares the two strongest membership values in cluster s. T_s is inversely proportional to the tendency towards hardness for each cluster, s. $card(Y_s)$ is the number of data points x_i that belong to cluster s. Thus, from equation 4.6, T_s is minimized when the closest data points to the cluster s are distant from other clusters and vice versa. In other words, T_s has its lowest value with a hard partition and its maximum value with the fuzziest partition. Therefore, according to Equations 4.4 and 4.5, $MinHT$ extracts the least favorable hard tendency of the set of clusters while $MeanHT$ determines the average of the hard tendencies of all clusters. $MinHT$ and $MeanHT$ should both be maximized to yield distinct clusters.

6. Minimum Cardinality

$$MinCard = min(\sharp Members_s) \tag{4.8}$$

The minimum cardinality of the input data set is defined in the Equation 4.8 where $\sharp Members_s$ is the number of data points in which cluster s has the strongest membership to among all the clusters. If there exists a cluster in which $\sharp Members_s = 0$, then this is an *empty* cluster with no data points with a strong belongingness to it. This can often occur when clusters are very close to one another. Empty clusters are not desirable as a cluster prototype, thus the minimum cardinality, or the minimum value of $\sharp Members_s$ for all s , should be greater than 0 ($MinCard > 0$).

7. Relative fuzziness

$$Rf = \frac{\sharp fuzzymembers}{\sharp non\_fuzzymembers} \tag{4.9}$$

Relative fuzziness Rf is defined as the ratio of fuzzy members to non-fuzzy members for all u_{ij} in U, where u_{ij} is a fuzzy member if its value is less than a threshold value. This threshold value is usually defined as

$$Threshold = \frac{1+c}{2c}$$

Since Rf is minimized for hard partitions, Rf should also be minimized in fuzzy c-means clustering to maintain distinct clusters with strong memberships to the data.

Figure 4.5 shows graphical representations of the functionals, with respect to increasing number of clusters, after fuzzy c-means clustering was performed on the robot sonar data gathered in Section 4.3. It is evident that most functionals are satisfied with increasing number of clusters from 1 to 20. However, the partition entropy is notably satisfied with a lower number of clusters. This is expected since the entropy requirement favours a lower number of clusters (Equation 4.2). The cardinality value, although satisfied (i.e. $MinCard > 0$), value gradually gets closer to 0 with increasing number of clusters. This is also expected because the probability of empty clusters increases with the number of clusters. These contradicting tendencies among the functionals between low and high number of clusters further emphasize the need for a compromise in choosing c.

4.5 Fuzzy Control System

For the purpose of the robot navigation problem, the crisp fuzzy inputs are derived from the sensory information of the robot's front eight proximity sonars. The fuzzified inputs provide an indication of the robot's position relative to the walls and obstacles in the environment. The defuzzified output should give constructive directions for the robot to effectively stay clear from collisions while moving at a constant velocity.

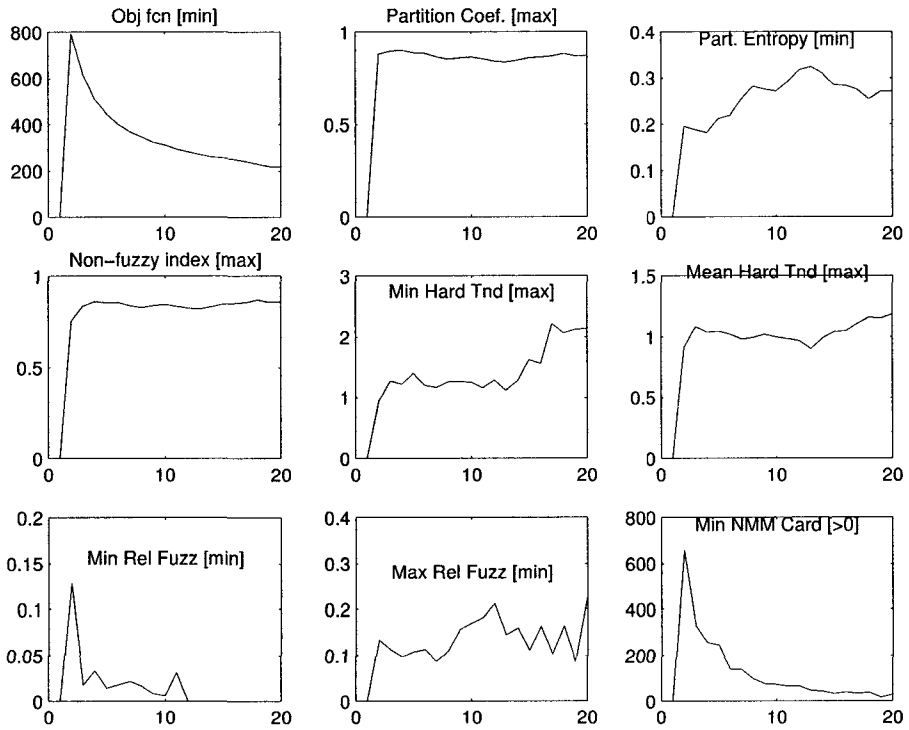

Figure 4.5: Functionals with respect to increasing number of clusters

4.5.1 Fuzzification

The number of input linguistic variables and their corresponding fuzzy sets should be minimized to reduce the complexity of the search space. Thus, the complexity of fuzzy rule base learning will also be reduced. In order to minimize the input state space, input generalization is performed using fuzzy clustering as described in the previous section.

A single linguistic variable, given as *situation*, is used to represent the situational environment of the robot. Its describing linguistic labels are derived from the results of fuzzy c-means clustering on the sonar data. Thus, each cluster center is considered as a *fuzzy set* for the *situation* variable. Since only the numerical values of the cluster centers are known, the degrees of membership to the *situation* fuzzy sets are determined by comparing the similarity distances between the crisp inputs, the current 8 sonar range values, and the cluster centers. This is accomplished with the same expression used in calculating the U membership matrix in Equation 3.2,

$$u_{lk} = [\sum_{j=1}^{c} (\frac{D_{lkA}}{D_{jkA}})^{\frac{2}{m-1}}]^{-1}$$

where u_{lk} is the degree of membership of the current crisp input vector k to cluster l, c is the number of clusters, and D_{lkA} is the similarity distance of the data point k to cluster l using the Euclidean distance measure, A. It is important to note that m is the same value as in the clustering process ($m = 1.2$).

4.5.2 Rule Structure and Evaluation

There is one crisp output of the obstacle avoiding FCS which dictates the relative heading of the robot. The linguistic variable for this output is *heading*. The fuzzy sets representing the linguistic labels for *heading* are singleton membership functions. The respective angle heading values for each singleton membership function used for

the experiments in this Chapter are denoted in Table 4.1. The number of fuzzy sets for *heading* is another design parameter which should be chosen by taking into consideration that the increase in the number of fuzzy sets

- increases the precision of the control actions
- increases the computation required in fuzzy rule base learning since the rules search space is increased.

Thus, given the scope and definition of the problem, a range of about 5-10 fuzzy sets would be a suitable choice as the number of output fuzzy sets. In this chapter, 7 fuzzy sets are used for experimental purposes.

Singleton Function ♯	*Value*	*Description*
1	90	Complete left
2	60	Very left
3	30	Left
4	0	Straight
5	-30	Right
6	-60	Very right
7	-90	Complete right

Table 4.1: Singleton membership functions and their corresponding values for the output variable *heading*.

The surface structure of a fuzzy linguistic rule showing the relationship between the input and output linguistic variables takes the following form

IF situation is *left corner* **THEN** turn amount is *complete right*

The next step in the fuzzy inference model [39] is rule evaluation. Since there is only one input variable, there is always exactly one antecedent in each

rule. Thus, according to the Min-Max inference method [28], the rule strength, which is the numerical minimum value of the antecedents in a rule, is strictly the membership degree of the current sonar ranges to the rule's single *situation* antecedent. Consequently, the numerical value of each consequent is equal to its rule strength. Finally, the fuzzified output, or the numerical strength for each output fuzzy label, is the maximum value of its consequents derived among all the rules in the rule base.

4.5.3 Defuzzification

Finally, defuzzification is performed to convert the fuzzified outputs into crisp system outputs. Using the singleton membership functions defined in Table 4.1, the crisp output is determined via the commonly used Center of Gravity Algorithm for defuzzification (COG) [62]. COG defuzzification is determined by the formula

$$y = \frac{\sum_{k=1}^{n}(w_k \cdot x_k)}{\sum_{k=1}^{n}(w_k)}$$

where y is the crisp output, w_k is the strength of the fuzzy output k, x_k is the position of the singleton k in output domain, and n is the number of output fuzzy sets.

4.6 Reinforcement Learning

Initially, the rule base of the FCS for the obstacle avoiding behaviour is not known. Reinforcement learning is used to determine the optimal policy or mapping between situations and actions (or antecedents and consequents) in order to maximize rewards over time and thus, in this case, to maintain repeatable obstacle avoiding performance. The main challenge that arises in applying reinforcement learning to robot navigation lies in assigning appropriate immediate and delayed credit to actions. Reinforcement,

induced by the robot's immediate and/or past actions, is very particular to locality and closely dependent on how the robot interacts with the environment. Therefore, optimal long term actions are difficult to learn. Q-learning, an off-policy [80] control based on a temporal-difference (TD) algorithm, is used to address this challenge. Q-learning instructs how the utility values for the situation-action pairs are updated such that the more beneficial they are, the greater their utility value. Consequently, it estimates the action-value function [72].

Algorithm 3 outlines the combination of clustering and reinforcement learning method applied in the proposed situation-based FCS. It is important to note that

Algorithm 3 Fuzzy c-means clustering and reinforcement learning applied in the FCS

1: Perform clustering on a set of robot sonar data for c clusters (Section 4.4). Each cluster is referred to as a *situation.*
2: Define the output *actions* with singleton membership functions by evaluating FCS
3: State the termination condition for reinforcement learning
4: Initialize utility values for each *situation-action* pair randomly or with constant values
5: Initialize the FCS's rule base with random *situation-action* pairs so that each situation has an associated action (i.e., The number of rules is equal to the number of situations).
6: **repeat**
7: Get crisp inputs from the range sonars as inputs to the FCS and perform fuzzification to get fuzzified inputs
8: Calculate a crisp output action using the current fuzzy rule base
9: Perform the resulting crisp action
10: Assess the performance of the resulting crisp action using a reward function
11: Update the utility values of all the rules currently in the rule base using the Q-learning update rule
12: Update the rule base with new rules chosen using the behaviour policy
13: **until** Termination condition satisfied

the search space for the rules is given as the number of action-situation combinations.

$$Total_{Rules} = Num_{Situations} * Num_{Actions}$$

Thus, in the case of 8 or 12 situations and 7 actions, the rule search space is 56 or 84, respectively. A simple fuzzy logic controller for navigation, without its input space generalized into situations, would have a significantly larger search space. For example, if each sonar input had three membership functions such that each fuzzy set represented the relative distance to its closest obstacle (e.g. *very far*, *medium far*, *near*), then the total rule search space would be

$$Total_{Rules} = 3^8 * 7 = 45927$$

Compared to 56 or 84 rules, generalizing the robot's surrounding state into situations drastically reduces the input search space. Thus, it makes reinforcement learning easier to explore the rule space.

4.6.1 Reward Function

The *reward function* is the central mechanism which defines and enforces the goal of the reinforcement learning component and, inadvertently, of the FCS. It provides the positive or negative reinforcements (or rewards) to each situation-action pair depending on the assessment of its performance. Positive performance will be given positive reinforcement, and vice versa. Thus the reward function facilitates determining the intrinsic desirability of each situation-action pair's rule within the FCS's rule base.

The reward function is typically simple, leaving the complexity of learning for the action-value function which states *how* the reward affects the utility values. For obstacle avoidance, the goal to be enforced is to prevent collisions with the walls and obstacles in the environment. The reward function is described in Algorithm 4.

Algorithm 4 Reward function for obstacle avoidance

1: **if** Robot is stalled (collision occured) **then**
2: $Reward = -30$
3: Recover from stalled position
4: **else**
5: $Reward = 30$
6: **end if**
7: **if** $Reward > 0$ **then**
8: $Reward = Reward - Crisp\_Output\_Heading * 0.1$
9: **end if**

If the robot had multiple possible actions which produced equal rewards, it would be beneficial to choose the action which required less energy. Thus, this defines a subgoal of obstacle avoidance: to derive an *efficient* obstacle avoiding behaviour. A deduction, which is proportional to the crisp output performed in that time, is applied to positive reinforcements iteration (Lines 7-8) in Algorithm 4.

Note that the absolute values of the reward were chosen experimentally by studying how varying these values would affect the rate of convergence. Small values would result in long simulations with little effect on utility values after each iteration. Larger values resulted in a rapid increase or decrease in utility values before the robot is given enough time for exploration. A compromise was required and reasonable absolute values for the reward were between 10 and 50.

4.6.2 Q-Learning

With on-policy reinforcement learning (i.e., Sarsa [72]), the system is learning the value of the policy π which is used to make the decisions. Thus, as in Equation 4.10, the utility value is dependant on the reward received from the situation-action pair, and the utility value of the following situation-action pair which is chosen by π. The

resulting optimal policy Q^* is given by $Q^*(s,a) = maxQ^{\pi}(s,a)$.

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha[r(s_t,a_t) + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)] \tag{4.10}$$

Q-learning [80], on the other hand, is an off-policy since it learns the value of a policy other than the one used to make the decisions. The policy used to make the decisions is the behaviour policy π. In Equation 4.11, the utility value, or Q-value, is dependant on the reward received from the situation-action pair, and the maximum utility value possible in the next state. Thus, it is learning the value of the maximum policy and is directly estimating Q^*. Therefore, similarly to the value function, a Q-value can be defined as a prediction of the sum of the reinforcements the agent will receive when performing the associated action and following the given policy thereafter (Section 3.5, [72]). For Sarsa on-policy learning, this given policy is the same as the behaviour policy, π. For off-policy Q-learning, the given policy is the estimated optimal policy π^*, or the estimation policy.

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha[r(s_t,a_t) + \gamma max\{Q(s_{t+1},a)\} - Q(s_t,a_t)] \tag{4.11}$$

In Equations 4.10 and 4.11, $Q(s_t,a_t)$ is the Q-value of situation s and action a pair at time t, and $r(s,a)$ is the reinforcement determined by the reward function after performing the action a in situation s.

The discount factor γ is chosen between 0 and 1 ($0 \leq \gamma \leq 1$). It defines the weight of the future possible rewards $max\{Q(s_{t+1},a)\}$ or $Q(s_{t+1},a_{t+1})$ on new Q-values. This helps to predict which actions are more beneficial with the subsequent states. Since it is desirable to learn which rules will return the most rewards over time, a high emphasis on the discount factor was made ($\gamma = 0.9$). Note that if $\gamma = 0$, the agent is only concerned in maximizing its current immediate rewards without considering the future states.

The other parameter in Q-learning is the learning rate α, which is also bounded between 0 and 1 ($0 \leq \alpha \leq 1$). It affects the rate of convergence of the reinforcement learning system. If it is set too high (i.e., $\alpha \simeq 1$), then more rapid and drastic changes of the utility values will occur. Thus, oscillations will constantly exist between utility values greater and less than the expected value from positive and negative rewards, respectively. This instability makes it difficult, if not impossible, to reach the expected value without slow and incremental approximations. On the other hand, if the learning rate is too small (i.e., $\alpha \simeq 0$), then changes in utility values are insignificant and no learning occurs.

In non-dynamic environments, it is beneficial to have the learning rate slowly decay with time (e.g., $\alpha_t = \frac{1}{t}$) to satisfy two requirements important in assuring convergence [72]:

$$\sum_{t=1}^{\infty} \alpha(t) = \infty \tag{4.12}$$

and

$$\sum_{t=1}^{\infty} \alpha(t)^2 < \infty \tag{4.13}$$

However, since the robot should react to dynamic environments, the learning rate should be constant (i.e., $\alpha(t) = \alpha$) so that effective learning will still occur when the system is faced with new situations in changing environments. Since a constant learning rate does not satisfy the second condition (Equation 4.13), indicating that *complete* convergence of the utility values will never be reached, the convergence of the rule base is used in determining the termination condition rather than the convergence of the utility values (Section 4.6.4).

In summarizing these desirable characteristics of the learning system, it is important to choose α to avoid large oscillations in the utility values, to have more gradual changes in successive utility values, and to be able to perform in dynamic

environments. As a result, the learning rate was set to relatively low positive constant value of $\alpha = 0.3$.

In this case, the dimensions of the lookup table holding the Q-value predictions correspond to the number of clusters by the number of possible fuzzy sets for the *heading* output. Thus, each situation-action pair has a Q-value associated with it. With each iteration, immediate reinforcements to the Q-values of responsible situation-action pairs are applied. Additionally, after several iterations, the knowledge gained from updating the Q-values is propagated backwards through the lookup table from later states to earlier states in time until it eventually predicts the optimal action-value function.

Algorithm 5 outlines the Q-learning procedure.

Algorithm 5 Q-Learning algorithm

1: Initialize Q-values, $Q(s, a)$
2: Observe the state s
3: **repeat**
4: Choose an action a using behaviour policy derived from Q
5: Perform action a
6: Observe the reward, r
7: Observe the next state s'
8: Update utility values using Q-learning formula (Equation 4.11) which is dependant on the r and the max utility in s'
9: $s \leftarrow s'$
10: **until** Termination condition satisfied

Q-Learning Modifications towards the Incorporation of Fuzzy Logic

Each situation-action pair in the lookup table can be regarded as a possible rule in the fuzzy rule base. A rule base is formed from all the possible rules in the fuzzy knowledge base. Each rule in the rule base provides an action for each situation.

Thus, the number of rules in the rule base at any time is equal to the number of clusters. Initially, a rule is added for each situation and the action associated with that situation is chosen randomly. With each iteration, the system observes the belongingness to each fuzzy state and uses the current set of rules in the FCS's rule base to produce the final (crisp) change of heading direction. Therefore, instead of identifying discrete situations and actions as in traditional reinforcement learning, fuzzified combinations of situations and actions are considered. In order to account for this, a modification to Equation 4.11 must be made such that *all* the rules which were activated in the rule evaluation step are updated and their Q-values are updated such that the change in Q-value for each responsible situation-action pair is rightfully deserved. For example, if a positive reward was received from the current set of rules in the rule base, then the rule(s) that were most responsible will have a greater change in Q-value, and vice versa.

The change in Q-value is mainly dependant on two variables which are received in response to the FCS's resulting output action:

1. The reward $R(s, a)$
2. The maximum utility in the next state $max\{Q(s_{t+1}, a)\}$

Since each rule in the rule base is partially responsible for the formulation of the crisp output heading action, then in Equation 4.11, these two factors must be weighted proportionally by the responsibility of each corresponding situation-action pair. Thus, these two factors are weighted by the strength of the rule. This leads to the reformulation of Equation 4.11 to

$$Q(s_t, a_t)_i = Q(s_t, a_t)_i + \alpha[w_{i_{sa}} r(s_t, a_t) + \gamma w_{i_{sa}} max\{Q(s_{t+1}, a)\} - Q(s_t, a_t)_i] \quad (4.14)$$

where $w_{i_{sa}}$ is the strength of rule i that corresponds to the situation-action pair sa.

The maximum utility in the next state, $max\{Q(s_{t+1}, a)\}$, can be regarded as that action's *potential* in the current state. It can also be regarded as the *desirability* of the next state. The maximum utility in the next state is used to incorporate the discounted rewards over time. Typically, it is evaluated by observing the next state s_{t+1} and determining the single action in that state which holds the maximum Q-value. This maximum Q-value is the maximum utility in the next state $max\{Q(s_{t+1}, a)\}$. In the situation-based FCS, since the observed next state is not a single discrete state, $max\{Q(s_{t+1}, a)\}$ is determined by considering the maximum Q-value found in all of the *next* fuzzy states, similarly weighted by the membership strength of the *next state*. Thus, it is calculated as the weighted average of the membership degrees to each situation and the maximum Q-value for that fuzzy situation:

$$max\{Q(s_{t+1}, a)\} = \sum_{j=1}^{c} [u_{js_{t+1}} max\{Q(j, a)\}] \tag{4.15}$$

where $u_{js_{t+1}}$ is the membership degree of the next state s_{t+1} to the fuzzy situation j and $max\{Q(j, a)\}$ is the maximum Q-value of the fuzzy situation j among all the actions a.

4.6.3 Exploration

In order for Q-learning to converge to the optimal policy the robot must visit every situation and execute each possible action in the situation several times. To ensure a maximum exploration of the rules search space, a *behaviour policy* must be assigned to facilitate an even distribution of the actions so that all the situation-action pairs continue to be updated, given that all the situations are available in the environment.

The *soft-max* policy [33] has been chosen, which assigns a probability to each action proportional to the situation-action pair's Q-value in the lookup table. This probability indicates the chance that the rule will be added to the fuzzy rule base: the higher the Q-value an action has in the relevant situation, the more likely that

the situation-action pair will be added.

Other such policies are the ϵ-*greedy*, which selects the actions with the greatest utility value, and ϵ-*soft*, which selects the actions on a random basis [72]. ϵ-greedy favors the exploitation of the rules with maximum estimated value, whereas ϵ-soft favors the exploration of the rule space. The soft-max policy overcomes the individual drawbacks of the ϵ-greedy and ϵ-soft policies by still selecting action stochastically but with favoritism of situation-action pairs with high estimated values. Ultimately, soft-max provides a simple compromise between the exploration and exploitation of the rule space.

The soft-max behaviour policy defines *what* to choose as the next actions in the rule base. For exploration, it must also be decided *when* to change the rule base. A rule base must be implemented long enough to be able to evaluate its performance in the environment. It must also be changed frequently to assist exploration of the remaining rules in the search space. For these experiments, the rule base is changed when one of the following occurs

- a collision
- b iterations have passed without any collisions

4.6.4 Termination Condition

The robot continues to explore and the lookup table continues to update itself until the current state is terminal. A terminal state is defined with the following two conditions:

1. The rule base has converged to a set of rules. Since *complete* convergence of the utility values will never occur due to the positive constant value of the learning rate α (Section 4.6.2), convergence is measured through the consistency of the rule base. A complete set of rules have converged when the estimation policy, or

the actions with the maximum utility value for each situation, remains constant for a period of p iterations.

2. A maximum number of iterations l has occurred without convergence of the rule base.

The fulfillment of the first terminal condition is verified by the Q-values in the lookup table. As reinforcement learning progresses, and even after sufficient learning has been achieved, the Q-values of the situation action pairs do not cease to change. Either they may continue to increase, decrease, or modulate. However, as convergence progresses, the situation-action pairs should reach a particular equilibrium such that the *relative* Q values among the actions for each situation cease to change. Consequently, an increasing gap forges between positive and negative actions as the probability of choosing the positive actions increases (and vice versa) until finally the estimation policy (the actions with the greatest utility value in each situation) ceases to change.

The second terminal condition is set to a relatively high number of iterations so that it can be presumed that the first terminal condition may never be reached as $t \rightarrow \infty$. Thus, there is no need to continue the experiment, and learning can be interrupted and terminated at l iterations. If learning continued for this long without fulfilling the first terminal condition, then either of the following assumptions could be made:

- Learning of the navigational behaviour could not be accomplished;
- The learned rule base keeps oscillating between similar sets of rules. These multiple sets of rules could all be sufficient in maintaining the desired behaviour. Thus, the estimation policy may continue to change even when learning is achieved.

4.7 Improvements: Optimize Velocity

The experiment described in Section 4.6 used a constant velocity setting to solve the obstacle avoidance problem. With the maximum velocity being 300 mm/s, a relatively slower velocity of 150 mm/s was used to give the robot enough time to observe the presence of obstacles and to react to them. If a velocity of 300 mm/s were to be used, and the robot was, for example, in a left corner situation, the high speed would not give the robot enough time to turn away from the corner. Thus, the velocity factor must be taken into consideration for optimizing the obstacle avoiding behaviour in a more realistic navigational controller.

The reinforcement learning method for obstacle avoidance was used again for the purpose of learning the optimal speed in which it should travel with respect to the situations it encounters. The learned obstacle avoiding rule base with the *heading* control action is incorporated and is kept constant. However, now there are two output variables: *heading* and *velocity*. Thus, the obstacle avoiding rule base with a *velocity* control action is incorporated and learned using the same reinforcement learning method with respect to the same clustered situations. The output singleton membership functions have the values shown in Table 4.2.

Singleton Function ♯	*Velocity Value*	*Description*
1	100 mm/s	Slow
2	200 mm/s	Medium
3	300 mm/s	Fast

Table 4.2: Singleton membership functions and their corresponding values for the output *velocity*.

The following limitations were enforced by the reward function of the reinforcement learning component to describe the desired behaviour:

Reward

- If we didn't hit an obstacle, reward proportionally to the numerical velocity value. This is to encourage high speeds and to take advantage of the maximum speed of the robot. Otherwise, without this factor, the tendency would be for the robot to always choose the safest action: to use the slowest velocity possible.

Penalize

- If an obstacle collision occurs
- If the previous closest distance cd_p to an obstacle is less than a distance $cd_{max} = 30cm$, and the current closest distance to an obstacle cd_c is less than the previous closest distance, the system is penalized proportionally to the difference between cd_p and cd_c. This penalty is to prevent the robot from approaching obstacles too quickly.

This reward function is described in Algorithm 6.

Algorithm 6 Reward function for optimal velocity

1: **if** Robot is stalled (collision occured) **then**
2: $Reward = -30$
3: Recover from stalled position
4: **else if** $(\{cd_p < cd_{max})$ and $(cd_c < cd_p)\}$ **then**
5: $Reward = cd_c - cd_p$
6: **else**
7: $Reward = 10 + 0.2 * CurrentVelocity$
8: **end if**

Table A.1 in the Appendix summarizes the chosen values of the parameters discussed in this Chapter.

"Part of the inhumanity of the computer is that, once it is competently programmed and working smoothly, it is completely honest."

– Isaac Asimov

5

Results and Analysis

The situation-based FCS collaborates the workings of a typical fuzzy logic controller with clustering and reinforcement algorithms. This chapter presents and analyzes the results obtained, first from the underlying components of clustering and reinforcement learning to the higher level performance of the situation-based FCS.

As stated in Section 4.4, 8 cluster and 12 cluster vectors were used in the fuzzy c-means clustering of the robot sonar data gathered in Section 4.3 with a partition order of $m = 1.2$. Identical parallel experiments were performed on the two cluster vectors to study the effect of the number of clusters on the resulting situation-based FCS.

5.1 Clustering

In using fuzzy c-means clustering on the robot sonar data, 50 trials were performed for both the 8 cluster and 12 cluster vector sets. Depending on the initial randomly guessed positions of the cluster centers, each trial of the fuzzy-c means algorithm would converge to varying sets of cluster centers. For 8 clusters, the following graphs in Figure 5.1 were produced for the objective function (Equation 3.1) and each of the functionals represented in Equations 4.1 - 4.9 with respect to the trial number. The functionals diagram for 12 clusters is shown in Figure A.1 in the Appendix. To determine which trial would be used in the following experiments, the functionals were optimized. For optimal fuzzy clustering, each functional should be either maximized or minimized as stated in Section 4.4. The trial which satisfied as many of these requirements as possible was chosen as the set of cluster centers used in the experiments that followed.

Figures 5.2 and 5.3 show the resulting cluster prototypes for the 8 and 12 cluster vectors among the 50 trials for each experiment. The cluster center prototypes are represented in the form of polar plots of the collective sonar ranges. Although the cluster vectors among all trials exhibited some similarities, the diversity of the clusters in each trial demonstrates the affect of this initial random guess of the cluster centers. Figures A.2 and A.3 in the Appendix are two separate trials for 8 clusters to show some notable similarities and subtle differences among the separate trials.

Each cluster represents a situation that the robot may encounter in its environment. Although it isn't required for the functionality of the situation-based FCS, labels were assigned to each cluster solely to gain some intuitive insight on the meaning of each situation. The labels are subjective descriptions from a human's perspective and several possible descriptions could be assigned. The descriptions in Tables 5.1 and 5.2 provide a reasonable interpretation of the situations or cluster vectors.

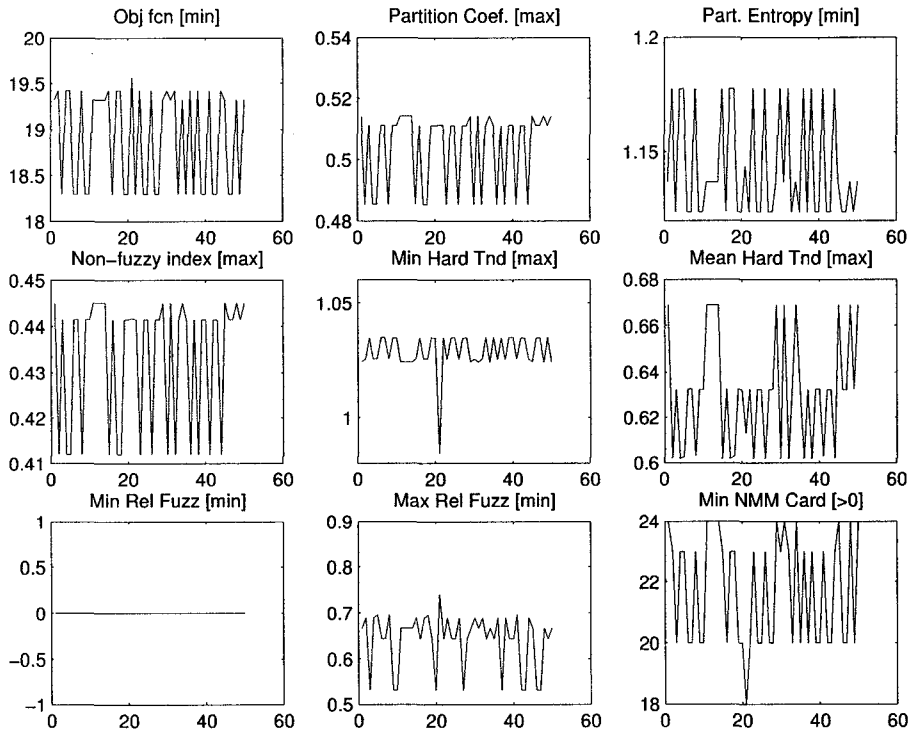

Figure 5.1: Functionals for 8 clusters with respect to trail number

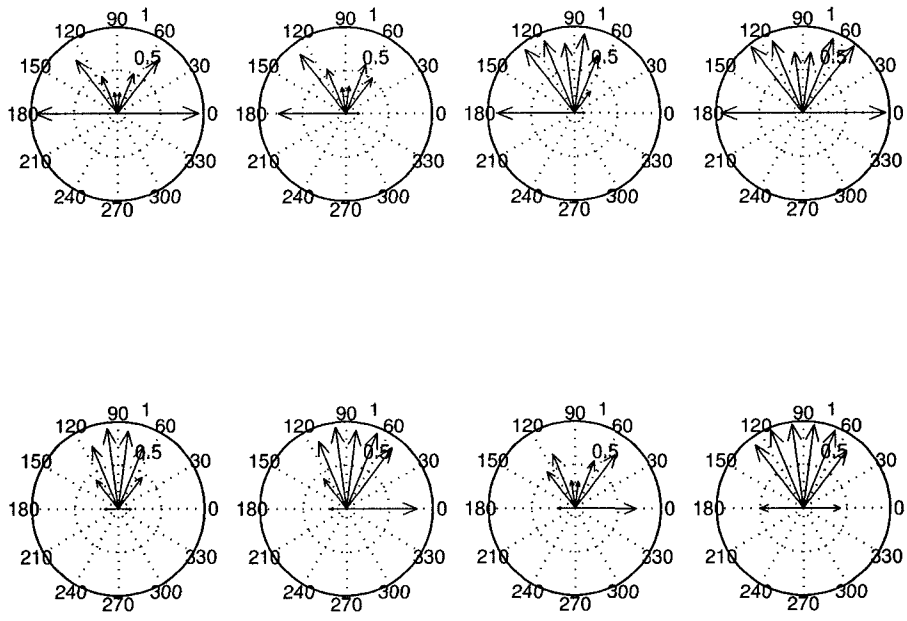

Figure 5.2: 8 Cluster prototypes of fuzzy situations

Descriptions for 8 Clusters			
1. Front Wall	2. Right Corner	3. Right Wall	4. Open Space
5. Corridor	6. Left Wall	7. Left Corner	8. Wide Corridor

Table 5.1: Descriptive labels corresponding to the 8 clusters in Figure 5.2

Clusters represented by the same situation (e.g., clusters 1 and 10 in Table 5.2 are both *left corner* situations) have similar sonar ranges, but may differ in how close the walls or objects are, or by the angle to which the robot is facing them. The clusters represented by *right wall* or *left wall* may also represent situations in which there is

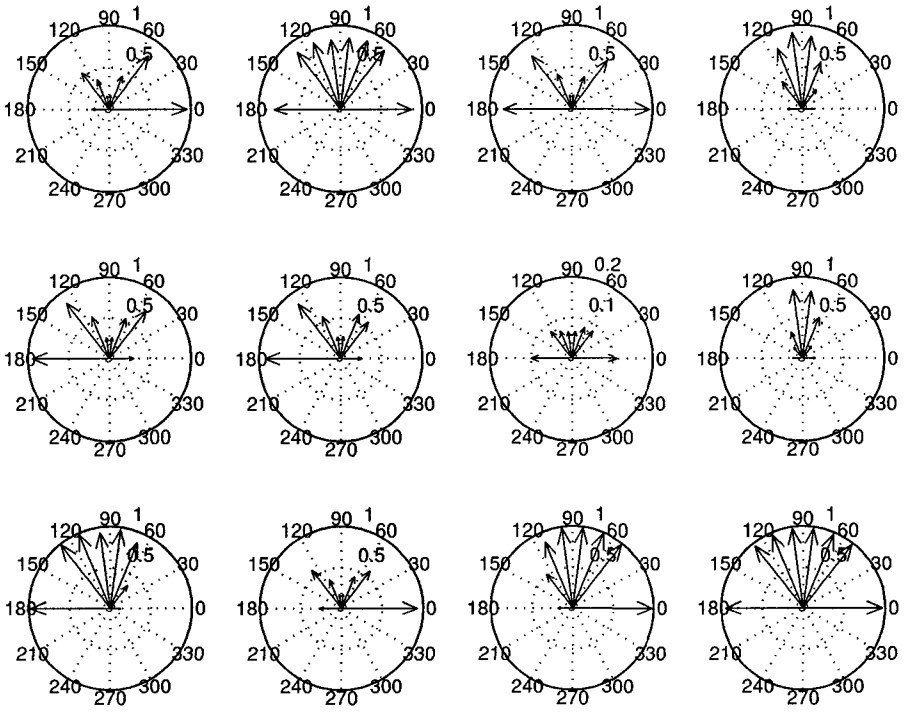

Figure 5.3: 12 Cluster prototypes of fuzzy situations

Descriptions for 12 Clusters			
1. Left Corner 1	2. Open Space 1	3. Front Wall	4. Corridor 1
5. Right Corner 1	6. Right Corner 2	7. Dead End	8. Corridor 2
9. Right Wall	10. Left Corner 2	11. Left Wall	12. Open Space 2

Table 5.2: Descriptive labels corresponding to the 12clusters in Figure 5.3.

some obstacle on the right and left sides, rather than walls. The objects indicated by the short sonar ranges in any situation may be either walls, obstacles, or any object in the environment. There are several descriptions that could represent a single cluster. Alternate example descriptions of the labels are given in Table 5.3 and can be used interchangeably with those in Tables 5.1 and 5.2.

Original Label	*Alternate Label*
Front Wall	Front Obstacle
Left Wall	Left Obstacle
Right Wall	Right Obstacle
Left Corner	Left and Front Obstacles
Right Corner	Right and Front Obstacles
Corridor	Right and Left Obstacles
Dead End	Left, Right and Front Obstacle
Open Space	No Obstacles

Table 5.3: Alternate descriptions to the labels in Tables 5.1 and 5.2.

These two sets of cluster centers were then used as the situation antecedents in the FCS. The results of using 8 cluster center and 12 cluster center vectors for the situation-based FCS are studied in the following sections.

5.2 Reinforcement Learning

The results presented in this chapter are based on the 8 cluster and the 12 cluster vectors derived in the previous section. The cluster prototypes are the premises or antecedents for reinforcement learning and for the fuzzy rules in the FCS. The results of using these prototypes to learn obstacle avoidance are studied.

5.2.1 Exploration

As learning continues, beneficial situation-action pairs repeatedly receive rewards and non-beneficial situation-action pairs repeatedly receive penalties. Occasionally, depending on the rule strengths, a situation-action pair that is usually rewarded could be penalized and vice versa. This happens when beneficial situation-pairs are combined with non-beneficial situation-action pairs in the fuzzy rule base at the same time. *All* the situation-actions pairs in the rule base are collectively either penalized or rewarded regardless of their rule strength. However, the situation-action pair(s) most responsible for the control action and outcome is given its rightful reward weighted by its rule strength. The Q-values of those situation-action pairs which are less responsible are less affected, regardless of how beneficial they are. With enough exploration of the environment and enough exploration of the possible situation-action pairs, a rightful and all-encompassing determination of beneficial and non-beneficial situation-action pairs should inevitably emerge.

Exploration is controlled by the behaviour policy, which states that in each situation, each action has a probability to be chosen as rule in the rule base. This probability is proportional to its Q-value and how it compares to the Q-values of the other actions. Proper exploration is necessary to ensure that the FCS will eventually converge to a set of fuzzy rules that satisfy the desired criterion, thus each situation-action pair must be visited several times. Figure 5.4 shows the number of times each situation action pair was dominant in deciding the control action for a total of 5000

iterations in the 8 cluster experiment. It shows that there is a relatively thorough distribution of the possible situation-action pairs, and it can therefore be presumed that proper exploration is achieved. It also demonstrates that the availability of the situation space was sufficiently provided in the training environment. Although some situations were significantly more common than others (e.g Open space vs. Narrow corridor), there was sufficient exploration exhibited in all the situations to learn the adequate actions for each situation. Also, as learning continues, beneficial situation-action pairs will increase in Q-values. Consequently, desirable actions will have a greater probability of being added to the fuzzy rule base in each situation due to the soft-max behaviour policy. Therefore, in Figure 5.4, these beneficial situation-action pairs have a slight tendency to have a greater occurrence. These same characteristics are evident in the 12 clusters situation-action pair distribution in Figure A.4 of the Appendix.

5.2.2 Derived Estimation Policy

Study of the Converged Lookup Table

The Q-values in the lookup table are predictions of the cumulative sum of reinforcements the agent will receive by performing the action in the current situation and following the optimal (Q-learning) policy thereafter [72]. Therefore, as learning continues, the lookup table should progress such that the most desired actions will positively increase in Q-values.

Table 5.4 is an example of the lookup table after convergence. By studying the Q-values for each situation, the action with the greatest Q-value is the estimated preferred heading value. From observing the Q-values among all the actions for each situation, the tendency is that as the Q-values decrease as the action for situation becomes increasingly dangerous. For example, for the Situation 1 (*front wall*), the

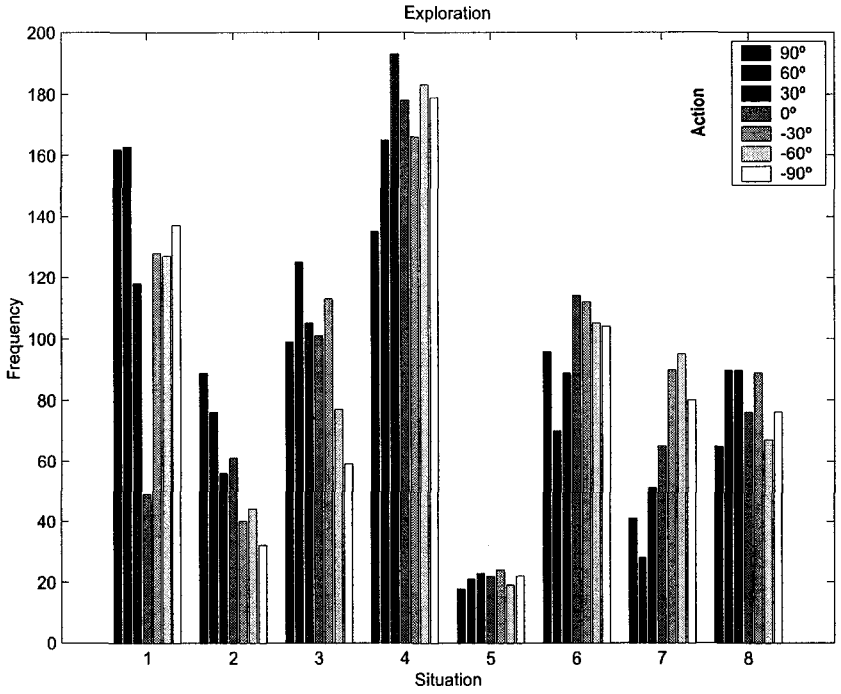

Figure 5.4: Exploration distribution for 8 clusters in obstacle avoidance experiment. (Sit.1-*Front wall*, Sit.2-*Right corner*, Sit.3-*Right wall*, Sit.3-*Open space* Sit.4-*Corridor*, Sit.5-*Left wall*, Sit.6-*Left corner*, Sit.7-*Wide corridor*)

action with the greatest utility value is to turn *complete left*. The other action with a relatively high utility is to turn *complete right*. Thus, these actions are considered the most preferred actions. Note that preferred actions are considered safe and less preferred actions are considered dangerous. In obstacle avoidance, *safe* actions prevent the robot from colliding with any object(s) in its path while *dangerous* actions would lead to the collision with obstacles.

Furthermore, in the front wall situations, the Q-values tend to decrease as the turning values go from *complete left* to *very left*, to *left*, then finally to *straight*. Likewise, the same pattern is reflected on the right side. From this pattern it can be further deduced that in *situation 1*, *complete left* and *complete right* turns are the preferred actions, while turns with a stronger inclination towards a straighter angle are less preferred. This tendency occurs repeatedly for the remaining situations: safe actions have the highest Q-values, and these Q-values tend to decrease towards more dangerous actions. The same patterns are also observed in the same experiment performed for 12 clusters as shown in Table 5.5.

The reinforcement learning method using the maximum utility (Q-value) of the next state to estimate the potential of a situation-action pair. If the next state has a low maximum utility, then the situation-action pair does not perform well in its subsequent states. Thus, it is important to note that situations with a high maximum utility value give a higher potential to its predecessor situation-action pairs, and vice versa. The maximum utility is also a measure of the desirability of the situation. Situations with a higher maximum utility value are more desirable, or, in other words, are more *safe* in the sense that the robot is less likely to hit an obstacle in that state. Likewise, the robot in *dangerous* situations is less desirable and more likely to hit an obstacle. This can be drawn from the lookup table since the desirable situations (i.e., *open space* and *wide corridor*) don't have as many close obstacles around it than the less desirable situations (i.e., *corridor*, *left corner*). Table 5.6 is a summary of the

maximum utility values in each situation in the order of decreasing desirability.

Clus.	*Label*	*Heading Value*						
		90^o	60^o	30^o	0^o	-30^o	-60^o	-90^o
1	Front Wall	88.5232	75.3621	75.3006	55.2341	68.4009	74.3626	88.499
2	Right Corner	102.933	94.1128	47.664	64.4788	57.2275	62.0681	46.1149
3	Right Wall	120.748	131.578	121.268	110.18	83.0621	76.4927	79.4707
4	Open Space	142.698	137.49	139.55	144.581	133.763	130.251	123.06
5	Corridor	40.0379	65.8052	66.7891	70.9823	57.6694	67.1838	38.9514
6	Left Wall	64.4424	104.558	106.079	111.84	129.185	136.504	132.74
7	Left Corner	51.7911	55.8056	70.2846	76.5532	83.4929	89.704	103.498
8	Wide Corridor	132.115	134.254	130.71	153.952	143.199	131.069	132.351

Table 5.4: An example lookup table after convergence for the obstacle avoidance experiment using 8 clusters

After convergence, the actions with the greatest utility value for each situation make up the estimation policy. The estimation policy subsequently comprises the rule base for the fuzzy logic controller. Tables 5.7 and 5.8 show the complete rule bases derived from their respective lookup tables (Tables 5.4 and 5.5). Other trials derived similar rule bases with some small variations (e.g. *front wall* situation was coupled with a turn of +90 instead of -90). The other actions that were often derived in the final rule base for each situation among all the trials are listed in Tables A.2 and A.3 in the Appendix. It would be ideal for the rule bases to converge to the same state at each trial, however, the small fluctuations occur and the final derived rule base is partially dependant on random characteristics of the initial Q-values, the soft-max behaviour, and the environment. Fundamentally, it is important that the converged rule base maintains the desired behaviour.

Clus.	*Label*	*Heading Value*						
		90^o	60^o	30^o	0^o	-30^o	-60^o	-90^o
1	Left Corner 1	27.9703	57.0372	34.4936	39.8101	60.6995	87.7947	84.5178
2	Open Space 1	149.34	158.868	141.379	174.779	164.066	151.513	159.67
3	Front Wall	139.383	123.961	132.325	91.1964	125.686	59.2321	147.626
4	Corridor 1	65.0343	68.0864	71.8306	74.0205	63.1513	49.9582	62.743
5	Right Corner 1	116.521	107.222	103.493	81.6878	109.622	108.595	91.0663
6	Right Corner 2	131.497	119.72	113.563	107.402	69.8178	103.206	102.701
7	Dead End	42.7199	9.40794	9.49937	16.1608	28.6717	24.5497	16.9143
8	Corridor 2	32.8932	31.2288	27.7333	39.2169	31.7463	31.1907	32.0261
9	Right Wall	144.735	153.5	176.768	134.076	117.491	89.9767	115.849
10	Left Corner 2	48.5934	25.8245	95.3795	70.3588	73.9922	109.089	74.4227
11	Left Wall	117.146	119.674	131.374	112.953	129.042	154.277	141.827
12	Open Space 2	171.123	175.827	176.003	189.002	183.428	187.346	185.257

Table 5.5: An example lookup table after convergence for the obstacle avoidance experiment using 12 clusters

5.2.3 Convergence Results

Convergence occurs when the estimation policy is constant for a certain number, p, of change intervals. The learning process is alternatively terminated if a predefined number of iterations, l is reached. The values of these criteria used in this study are $p = 50$ and $l = 5000$.

The parameter, l, is meant to cut off learning if convergence was taking too long. Although it was set relatively high, in the trials executed, the Q-values were able to converge before the system reached 5000 iterations. Table 5.9 depicts the number of iterations it took for convergence for the trials for 8 and 12 clusters. It is evident that the 8 cluster vector trials required less number of iterations to converge compared to the 12 cluster vector trials. This agrees with the assumption that a greater number of

Cluster	*Label*	*Max Q-Value*	*Desirability*
8	Wide Corridor	153.952	Most Desirable
4	Open Space	144.581	
6	Left Wall	136.504	.
3	Right Wall	131.578	.
7	Left Corner	103.498	.
2	Right Corner	102.933	.
1	Front Wall	88.5232	
5	Corridor	70.9823	Least Desirable

Table 5.6: Maximum utility values for each state in order of decreasing desirability in the obstacle avoidance experiment using 8 clusters

Cluster	*Label*	*Action*
1	Front Wall	90^o
2	Right Corner	90^o
3	Right Wall	60^o
4	Open Space	0^o
5	Narrow Corridor	0^o
6	Left Wall	-60^o
7	Left Corner	-90^o
8	Wide Corridor	0^o

Table 5.7: Learned rule base for obstacle avoidance after convergence for 8 clusters

clusters would require a longer learning time due to the increased rule search space.

5.3 Performance Evaluation

The situation-based FCS was able to converge to a learned rule base in each trial. The ultimate question, however, is how well is the converged fuzzy rule base able to maintain the desired performance. In this section, the ability of the derived fuzzy rule base to avoid obstacles is examined.

Cluster	*Label*	*Action*
1	Left Corner	-60^o
2	Open Space	0^o
3	Front Wall	-90^o
4	Wide Corridor	0^o
5	Close Right Corner	90^o
6	Right Corner	90^o
7	Dead End	90^o
8	Corridor	0^o
9	Right Wall	30^o
10	Left Corner	-60^o
11	Left Wall	-60^o
12	Stronger Open Space	0^o

Table 5.8: Learned rule base for obstacle avoidance after convergence for 12 clusters

Figures 5.5 - 5.8 show how learning progressed with time in typical simulation runs. Figures 5.5 and 5.6 show the number of iterations between collisions in 5000 iterations for the 8 and 12 cluster experiments, respectively. The converged rule base is kept constant once convergence is achieved. Since each iteration lasts for 1.5 seconds, it can also be considered as the the time between each collision. From these two figures, it can be concluded that the 12 cluster experiment took a longer time to converge, and the 8 cluster experiment had a more gradual increase in the number of iterations between clusters. However, once convergence was reached, the 12 cluster experiment performed better since there was more time in between the collisions.

Figures 5.7 and 5.8 show the average reinforcement in 50 iteration intervals for the two experiments. The reinforcement was assigned according to the reward function for obstacle avoidance described in Section 4.6. Initially, there is almost an equality between positive and negative rewards: positive and negative rewards keep

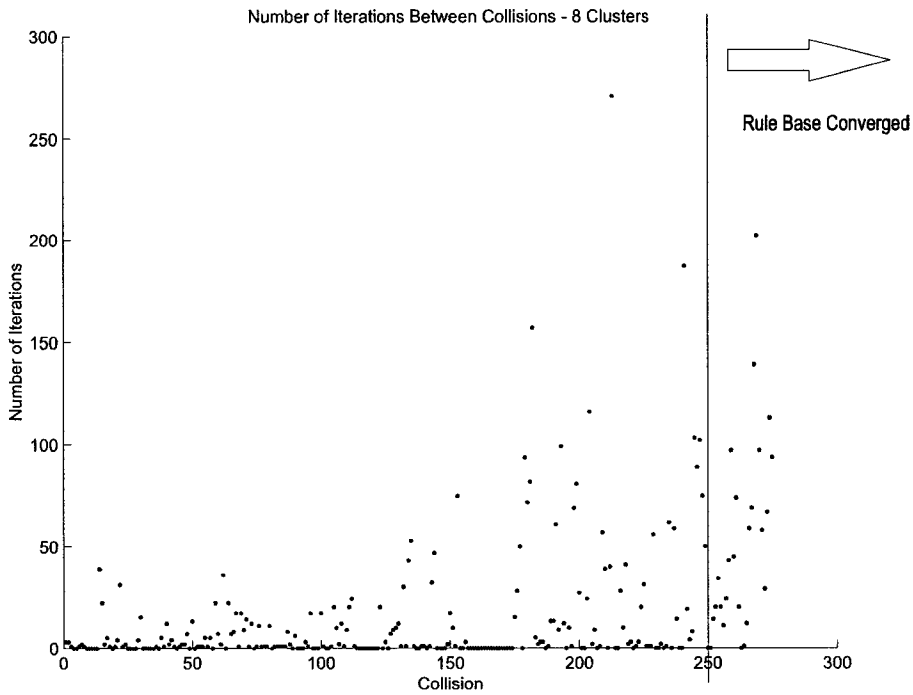

Figure 5.5: Number of iterations between collisions in learning obstacle avoidance with 8 clusters

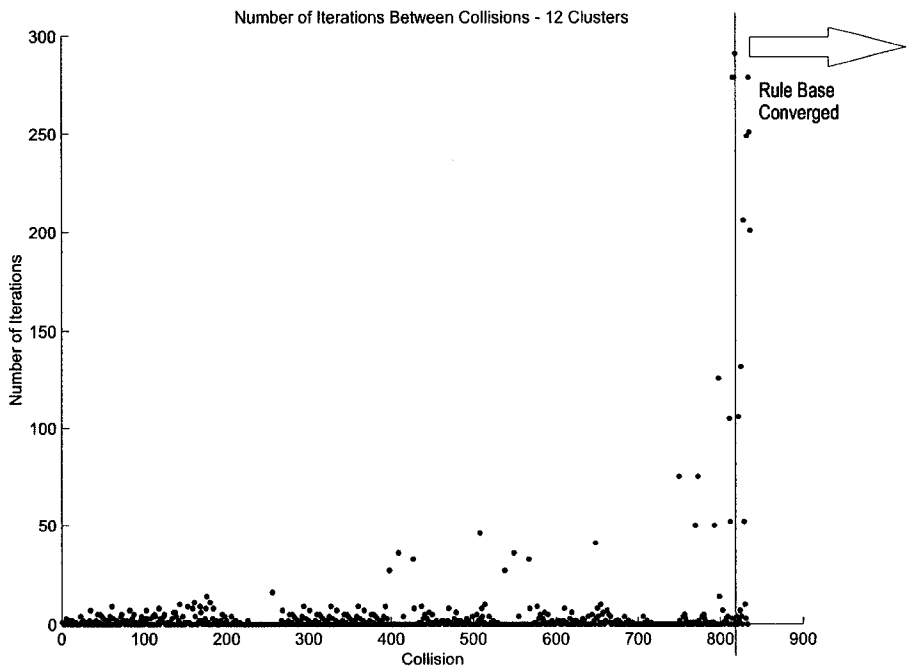

Figure 5.6: Number of iterations between collisions in learning obstacle avoidance with 12 clusters

Trial	8 Clusters	12 Clusters
1	2006	4805
2	2009	2337
3	1884	3387
4	1808	2813
5	1829	4157
6	403	3557
7	1114	3065
8	1305	2421
9	2244	2649
10	987	2651
11	1833	2926
Avg	1582	3160

Table 5.9: Number of iterations until convergence for 8 clusters and 12 clusters in the obstacle avoidance experiments.

alternating, thus there is no segregation between positive actions and negative actions. As learning continues, a segregation forms as the controller learns which actions perform as desired and the number of negative rewards decreases gradually. It can observed that the 8 cluster experiment's rewards had a more gradual improvement, however, the 12 cluster experiments performed slightly better (received more positive rewards) after convergence.

Collisions

In Figures 5.5 - 5.8, it can be seen that collisions occur throughout learning, and continue to occur even after convergence. Collisions will occur frequently during learning due to the exploration factor required to gain experience. However, even though the number of collisions significantly decreases after the rule base has

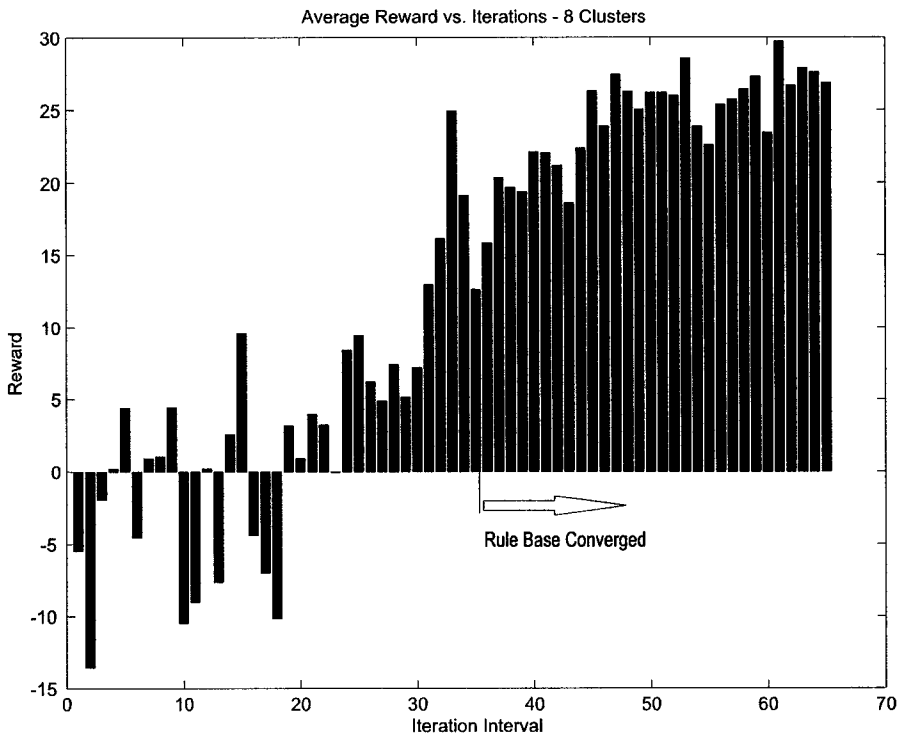

Figure 5.7: Reward values with respect to 50 iteration intervals in learning obstacle avoidance with 8 clusters

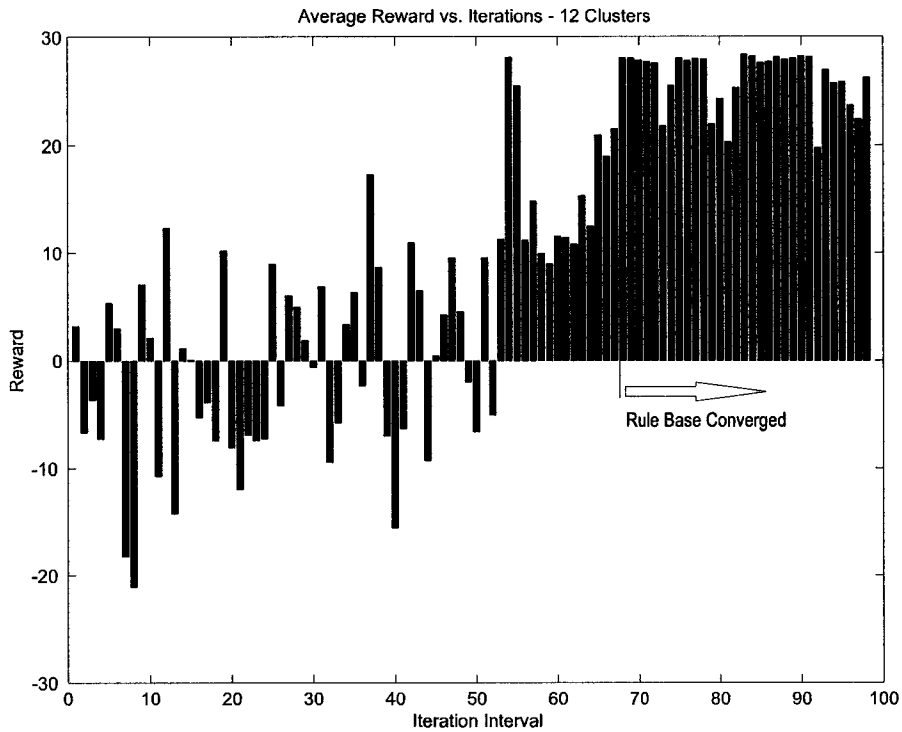

Figure 5.8: Reward values with respect to 50 iteration intervals in learning obstacle avoidance with 12 clusters

converged, collisions still occur intermittently. When observing the robot's movement after convergence and its interpretation of the search space, the cause of the collisions is the robot's inability to correctly assess its situation due to one of the three conditions:

Sonar Placement The Pioneer 2DX's sonar placement is almost evenly spaced with small gaps between each sonar (Figure 4.3). Depending on what angle the robot is approaching an obstacle, the sonars may not see the obstacle from one of its sonar ranges. Thus, the robot thinks there is nothing in its path at that particular angle. This could be solved by having more sonars around the robot.

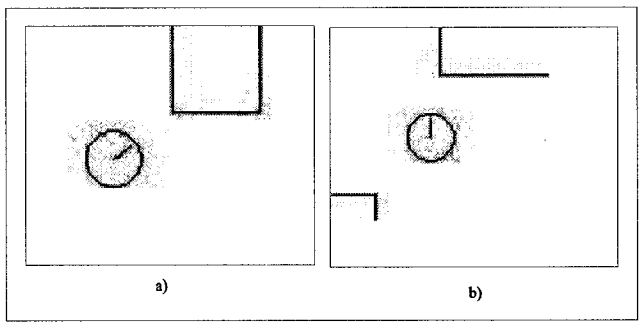

Figure 5.9: Misclassified situations a) Approaching a corner b) Approaching an edge

Generality of the Clusters As seen with the difference between 8 clusters and 12 cluster, the 8 cluster system has more collisions even after learning has been achieved. A smaller number of clusters generalizes many data points into the same situation. Thus, several actual robot situations will have the same action command, and the chosen action will be the one which is beneficial for most instances of the situation, not necessarily for all instances. The solution for over-generalized situations is simply to increase the number of clusters, as demonstrated in the performance comparison of the 8 and 12 cluster experiments.

Absence of the Situation in Robot Sonar Data The data collection in Section 4.3 was meant to capture all the possible situations a robot could encounter in an environment. Although it appeared that most situations were accounted for, some particular situations were overlooked. Thus, the robot didn't have a situational cluster available to learn what to do in this situation. The FCS had to associate these situations with the closest cluster prototype. Two particular situations occurred when it was approaching an obstacle's corner or edge, illustrated in Figures 5.9a) and b). All the sonar ranges are long except the two front ranges or the two front right ranges. These situations were thus misclassified as *open space* situations. This could be remedied by having the robot randomly wander in the data collection step so that it may encounter more unexpected situations. Another solution would be to increase the adaptability of the robot by clustering on-line. In this way, the robot would be able to handle any new situation that it has never seen before.

Figure 5.10 shows a simulated trajectory of the fuzzy controlled robot in the training environment. It follows after the convergence of a rule base in an experiment with 8 clusters. The robot with the same fuzzy rule base is placed in a novel environment to exemplify its ability to adapt to different environments (Figures 5.11). Figure A.5 in the Appendix shows the obstacle avoidance behaviour of a converged situation-based FCS with 12 clusters prototypes in another novel environment. It is evident that the robot's path is such that it avoids collisions with any of the obstacles present in both environments. However, since there is no goal directedness incorporated with the obstacle avoiding behaviour, at times, the robot tends to encircle a small area with obstacles surrounding it.

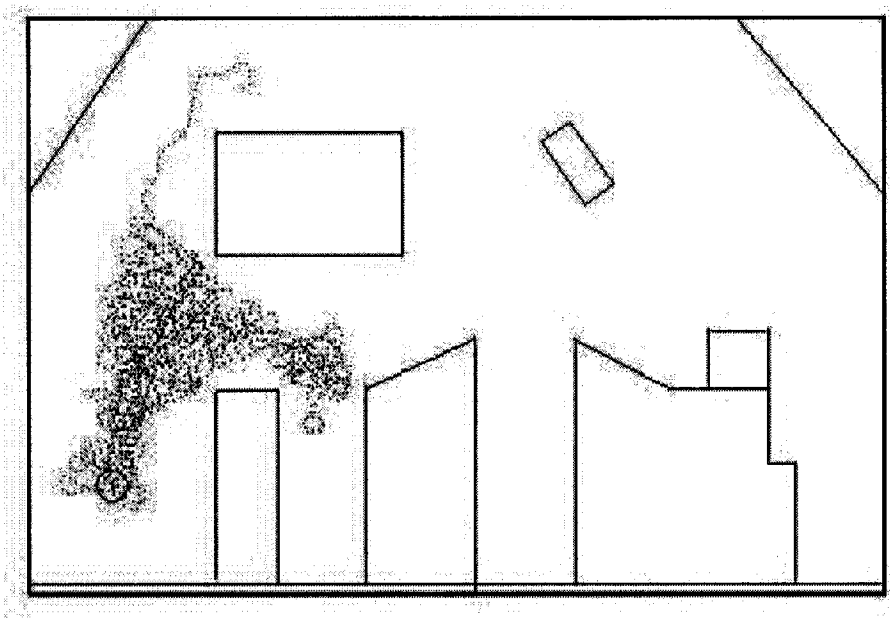

Figure 5.10: Simulated trajectory with obstacle avoidance behaviour in training environment

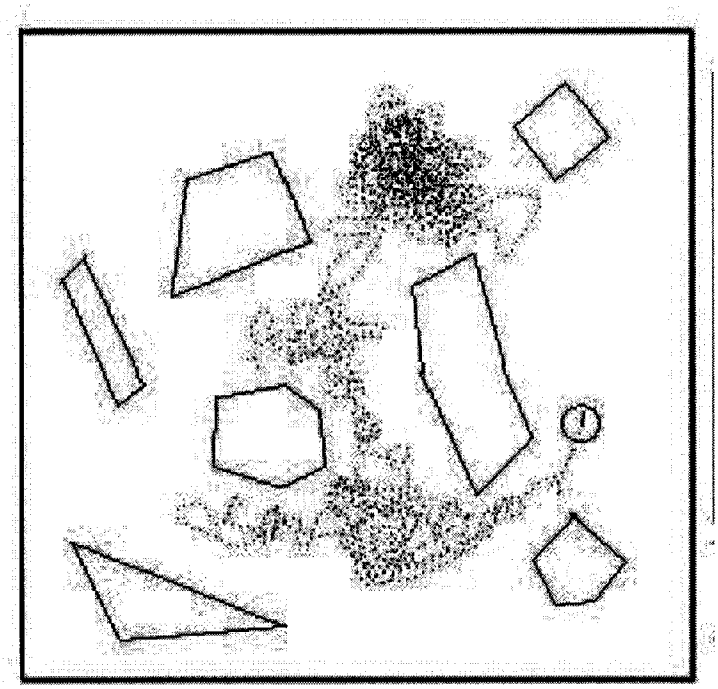

Figure 5.11: Simulated trajectory with obstacle avoidance behaviour in novel environment

5.4 Improvement (Velocity) Results

Section 4.7 describes the reinforcement learning method applied towards learning the optimal velocities with respect to the situations. The method is analogous to the method used in learning to avoid collisions. The only differences are that now there is are two outputs, velocity and heading (the rule base for deciding the *heading* action is constant) and the reward function is changed following Algorithm 6 in Section 4.7.

Table 5.10 is the lookup table of the Q-values after convergence and Table 5.11 is the corresponding derived fuzzy rule base of a trial which took 2132 iterations to converge. From Table 5.11, it is evident that the learned rule base, or policy, favours maximum speeds in the *open space* and *wide corridor* situations. Medium speeds are suggested for the *right wall*, *left wall* situations. Slowest speeds are suggested for *left corner*, *right corner*, *narrow corridor* and *front wall situations*. This can be interpreted that the slower speeds are suggested for the more dangerous situations and vice versa. Safe (desirable) and dangerous (less desirable) situations were classified in Table 5.6. In the Appendix, refer to Table A.4 for a list of convergence times and Figure A.6 for a summary of the robot's exploration of its situation-action space.

Figure 5.12 shows the average reward values, with increasing iteration intervals of 50, according to the reward function detailed in Algorithm 6. The reinforcement is generally high since an obstacle avoidance behaviour for the heading control is already implemented. The main purpose of the reward function is to optimize the robot's velocity. It is evident that the reward values are initially high, and they slightly increase continually until the optimization is learned.

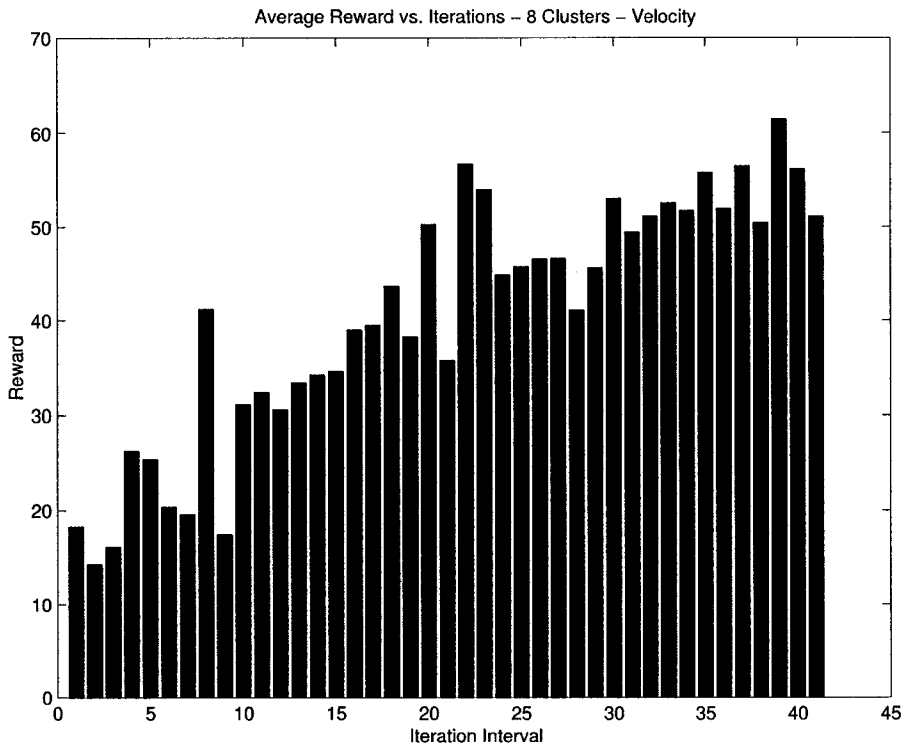

Figure 5.12: Reward values with respect to 50 iteration intervals for 8 clusters, learning optimal velocities

Cluster	*Label*	*Heading Value*		
		100 mm/s	200 mm/s	300 mm/s
1	Front Wall	256.187	214.836	222.199
2	Right Corner	234.351	223.433	220.652
3	Right Wall	217.566	244.973	242.778
4	Open Space	251.759	297.853	320.878
5	Corridor	195.344	199.148	195.554
6	Left Wall	255.582	274.673	284.981
7	Left Corner	232.087	223.566	211.141
8	Wide Corridor	226.75	249.217	255.856

Table 5.10: An example lookup table for learning optimal velocities after convergence using 8 clusters

Cluster	*Label*	*Action Velocity*
1	Front Wall	100mm/s
2	Right Corner	100mm/s
3	Right Wall	200mm/s
4	Open Space	300mm/s
5	Narrow Corridor	200mm/s
6	Left Wall	300mm/s
7	Left Corner	100mm/s
8	Wide Corridor	300mm/s

Table 5.11: Learned Rule Base after convergence for learning optimal velocities using 8 clusters.

"Properly educated, the resulting robots are likely to be intellectually formidable."

– H. Moravec

6 Extensions

The research described in the previous chapters explores the possibility of learning a single primitive behaviour using the techniques of fuzzy c-means clustering and reinforcement learning in a navigational fuzzy logic controller. This chapter extends this research by exploring the derivation of other navigational behaviours and the fusion of multiple low-level behaviours.

The positive results in learning the fuzzy rule base for obstacle avoidance through means of reinforcement learning and fuzzy c-means clustering suggests the prospects of learning other navigational behaviours. The changes made to the situation-based FCS to learn other behaviours exist in the reward function, which essentially defines the desired performance of the behaviour.

The non-directed and loopy behaviour of the robot with only the obstacle avoiding

behaviour (i.e. Figures 5.10 and 5.11) suggests that different techniques might be needed to establish the existence of a more practical and functional navigational robot. Although it is an essential component of navigation, the obstacle avoiding behaviour, single-handedly, does not have much purpose. Other low-level behaviours should be integrated to create more higher-level and complex controllers. This is typical of behavioural based robots since their controllers often consist of multiple low-level behaviours, each controlling a particular aspect of the robot's desired performance. This is an underlying aspect of Brook's subsumption architecture [18]. However, once the single behaviours are defined, the task of fusing and coordinating the behaviours is a non-trivial challenge. Behavioural cooperation and integration is essential in exploiting the single behaviours efficiently and correctly, particularly when the behaviours contradict each other.

In this chapter, the behaviour of wall following is learned using the same methodology and situation-based FCS that was applied to obstacle avoidance. Next, two behaviours, obstacle avoidance and goal finding are fused together in order to accomplish the ultimate goal of finding a goal without hitting any obstacles along the way.

6.1 Application to Other Behaviours

The purpose of the research in Chapter 4 was to demonstrate that the obstacle avoidance behaviour could be learned if the positive and negative traits of the behaviour are well defined in the reward function and proper exploration of clustered situations is maintained. Furthermore, a more useful application of this research is to demonstrate that this method could be used to learn other low level behaviours in which a reward function can be reasonably well defined. A well defined behaviour is such that it is clear from the current, and/or subsequent, states whether the behaviour

is being satisfied or not. For example, it is clear whether the robot has maintained the obstacle avoidance behaviour in the current state by examining if a collision occurred through the robot's internal and external sensors.

The alternative navigation behaviour chosen to demonstrate this assertion is the *wall following* behaviour. More specifically, the objective is to follow either the left wall or right wall within a specified allowable distance without hitting the wall. Thus, similar to the obstacle avoiding behaviour, a penalty is given if the robot hits a wall or obstacle. However, additional characteristics are essential in the definition of the wall following behaviour. The robot should not move too close to a wall, and if it is following a right or left wall within an allowable distance, it should not move too far away from the wall. If it is not following a wall, it should explore the environment, or approach a wall if it sees one in its field of view (from the sonars). The specified allowable distance from a wall chosen for the purpose of this experiment is 15-35cm. The parameters chosen for the wall following experiment are summarized in Table A.1 of the appendix. The corresponding reward function for the wall following behaviour is defined in Algorithm 7.

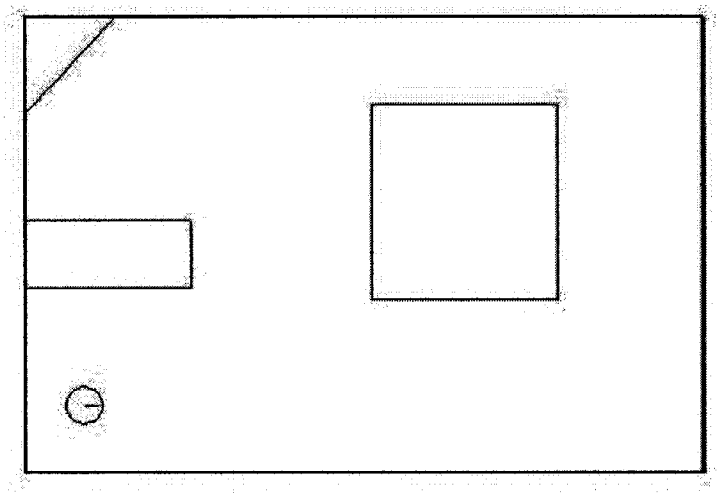

Figure 6.1: Training environment used for wall following

Algorithm 7 Reward function for wall following

```
1: if Robot is stalled (collision occured) then
2:     Reward = −30
3:     Recover from stalled position
4: else if Robot goes too close to the wall then
5:     Reward = −20
6: else if Robot goes too far from the wall then
7:     Reward = −20
8: else if Robot maintains allowable distance then
9:     Reward = 30
10: else if Robot is approaching a wall then
11:    Reward = 20
12: end if
13: if Reward > 0 then
14:    Reward = Reward − Crisp_Output_Heading ∗ 0.1
15: end if
```

The experiments were conducted using a simple enclosed room with narrow and wide corridors. There were few obstacles to prevent the robot from encircling the objects in the environment and to focus more on the wall following situations. Figure 6.1 displays the wall following training environment.

The same 8 cluster situations (Figure 5.2) and 7 output membership functions (Table 4.1) that were used in Chapter 4 were initially used for this wall following learning system. However, the fuzzy logic controller was unable to distinguish situations that should be separated for the purpose of wall following. For example, *left corner* and *approaching left wall at an angle* are two situations that have similar sonar ranges - short front and left sonar ranges. With 8 clusters, these two situations are classified as one situation (i.e. *left corner*). Also, the output singleton heading angles need to be more precise in order to facilitate small heading adjustments of the robot to better align itself with the wall. In summary, it was determined that the precision in determining the current situation and the precision of the heading values had to be increased due to the more strict requirements of the wall following problem. Therefore, the clustering procedure (Algorithm 1) was repeated for 16 clusters. Figure B.1 in the Appendix shows the corresponding functionals graphically with respect to trial number and Figure 6.2 shows the 16 cluster prototypes resulting from the clustering. The corresponding descriptions of the clusters are in Table 6.1.

Descriptions for 16 Clusters			
1. Right Wall 1	2. Open Space 1	3. Dead End 1	4. Right Corner 1
5. Front Wall 1	6. Front Wall 2	7. Left Corner 1	8. Left Wall
9. Corridor 1	10. Dead End 2	11. Right Wall 2	12. Open Space 2
13. Corridor 2	14. Left Corner 2	15. Right Corner 2	16. Wide Corridor

Table 6.1: Descriptive labels corresponding to the 16 clusters in Figure 6.2.

Although some labels are similar (e.g. *Right Corner 1*), and *Right Corner 2*, the

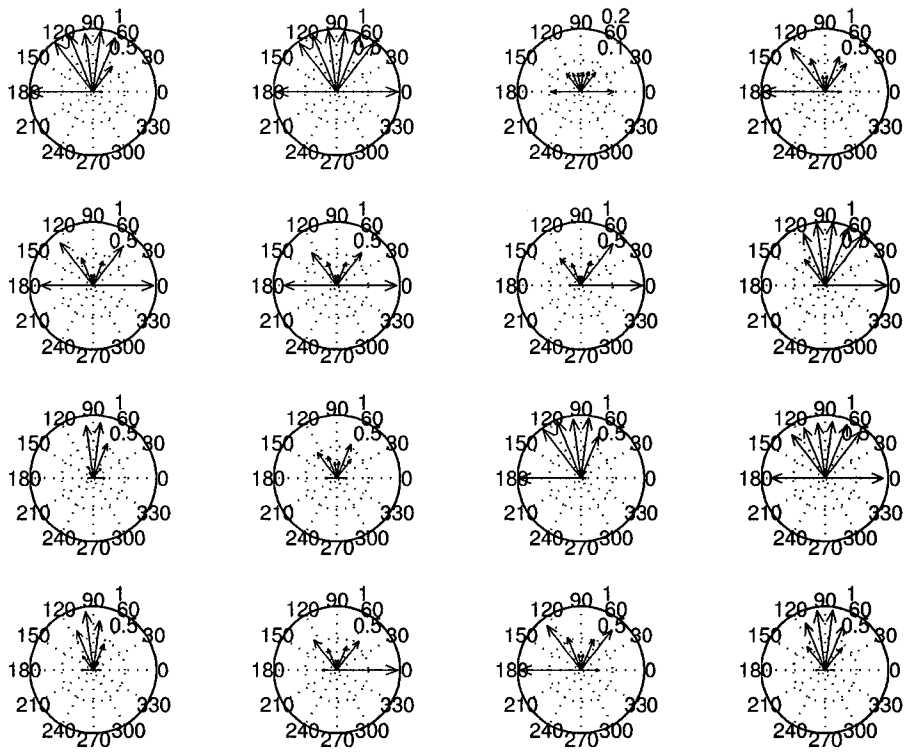

Figure 6.2: 16 Cluster prototypes of fuzzy situations

figure shows that the prototypes with the same labels are not identical. They have some variations which are helpful in distinguishing the situations from one another. For example, the left and right corner situations could represent varying positions and angles towards facing the corner, or perhaps a wall.

For the output control action, *heading*, 9 singleton membership functions, rather than 7 are used. Table 6.2 shows these membership functions and their respective values.

Singleton Function♯	*Value*	*Description*
1	90	Complete left
2	60	Very left
3	40	Left
4	20	Small Left
5	0	Straight
6	-20	Small Right
7	-40	Right
8	-60	Very right
9	-90	Complete right

Table 6.2: Singleton membership functions and their corresponding values for the output label *heading* in learning the Wall Follow behaviour.

The maximum number of iterations was increased to $l = 6000$ since the wall following behaviour required more time in order to converge. The reasons for the increased convergence time are the increased rule search space and the increased granularity of the desired behaviour. Wall following requires careful adjustments of the robot's heading to maintain a close and parallel position with the wall. In addition, the correlation between the situations and positive and negative rewards were much less consistent than that of obstacle avoidance. For example, depending

on the distance and angle the robot is towards a wall, the FCS could either be rewarded or penalized. This was a problem because different situation instances that received different reinforcement were often categorized into the same situation, whereas in obstacle avoidance, reinforcement was often consistent with the generalized categories. As with 8 clusters, the 16 cluster FCS still had difficulty in distinguishing important situations for wall following. Thus, the learning rate α was reduced to 0.2 to give less dependence on any single reinforcement value and more emphasis on small incremental adjustments in Q-values over a longer period of time. The design parameters chosen for the wall following experiment are summarized in Table B.1 of the Appendix.

Results

Table B.2 in the Appendix summarizes the robot's exploration of the situation-action space in the training environment. Table 6.3 is an example lookup table after convergence. The derived rule base from this lookup table, listed in Table 6.4, performed reasonably in wall following, however, the Q-values for each situation are much closer in value and are therefore less distinct from one another. Also, the trends described in Section 5.2.2 are not as clear as they are in the obstacle avoidance lookup Tables 5.4 and 5.5. Convergence was comparably difficult in learning the wall following behaviour; among 11 trials, 5 trials were not able to converge to a set of rules, and from inspection of the lookup tables, learning could not be accomplished since the resulting utility values did not describe a wall following behaviour. However, 6 trials were able to converge with an average of 5415 iterations. The convergence table for the converged trials is found in Table B.2 of the Appendix. As a measure of performance, the reward is displayed with respect to 50 interval iterations (Figure B.3 of the Appendix).

Figure 6.3 shows the resulting trajectory of the robot after convergence of the

situation-based FCS has been achieved in learning the wall following behaviour. This trajectory takes place in the robot's learning environment. Another trajectory of the robot's wall following behaviour in a novel environment is shown in Figure B.4 of the Appendix. It is evident that the robot follows the wall, however the trajectory of the robot is not smooth.

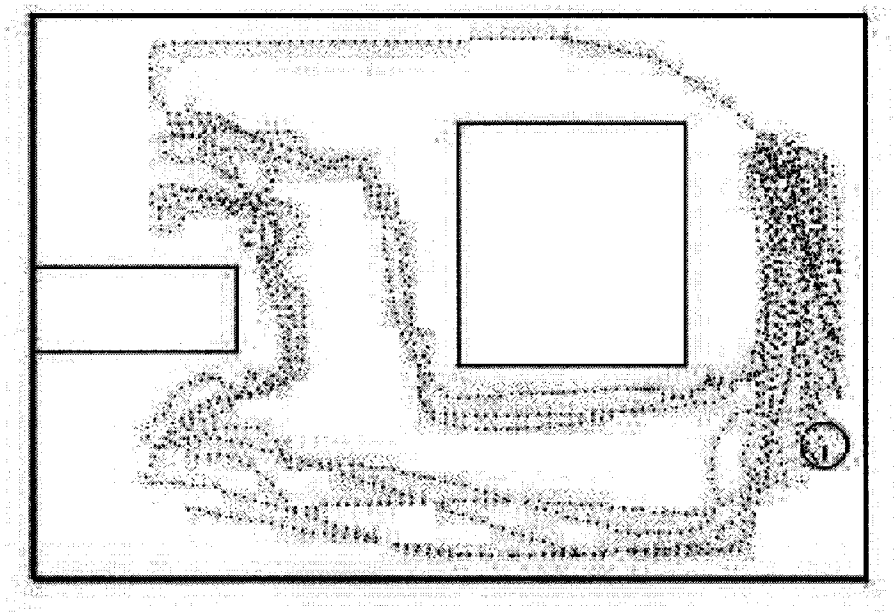

Figure 6.3: Simulated trajectory with the wall following behaviour in the training environment

6.2 Fusing Behaviours

The fact that most single behaviours lack any practical purpose in realistic robots introduces the need for behaviour fusion. For example, if there are obstacles in the path between a robot and its goal, the goal finding behaviour cannot function on its own without an obstacle avoidance behaviour. Likewise, a robot with an obstacle avoiding behaviour is almost stochastically random if not given some sort of goal or

Cluster	*Label*	*Heading Value*								
		90^o	60^o	40^o	20^o	0^o	-20^o	-40^o	-60^o	-90^o
1	Right Wall 1	27.8897	22.9001	28.603	20.1345	32.4299	26.4643	30.4934	16.9796	19.7044
2	Open Space 1	50.2647	61.3818	58.0298	51.6143	54.2206	46.7648	57.0035	53.938	49.6289
3	Dead End 1	91.812	93.6142	86.0746	69.1931	87.2259	90.0663	86.4325	91.6526	96.9195
4	Right Corner 1	61.8049	64.1147	57.7355	47.6291	50.2104	54.3856	55.053	55.0033	53.2401
5	Front Wall 1	1.66957	0	0	0	0	0	0	0	0
6	Front Wall 2	41.3929	28.8122	31.3356	24.9925	32.8465	24.4726	31.0862	31.6784	32.1446
7	Left Corner 1	71.9307	59.8252	70.491	64.2581	65.8838	65.5508	70.2877	74.6655	72.1682
8	Left Wall	48.6307	48.5053	58.0591	46.0609	60.5521	55.01	45.7778	51.1417	50.0563
9	Corridor 1	86.3302	77.3797	86.1895	92.1883	86.0332	71.9746	91.7118	82.736	89.7352
10	Dead End 2	87.2363	86.744	78.3902	76.6838	81.6978	82.431	83.0827	86.3719	82.0843
11	Right Wall 2	7.40642	19.0983	10.5893	13.5143	20.49704	11.48572	17.8905	16.9806	12.8326
12	Open Space 2	32.1137	25.1416	41.4446	29.0385	32.4487	32.1361	30.9956	34.1177	25.9387
13	Corridor 2	57.9797	54.1379	56.312	58.2101	61.1984	58.9031	61.1533	58.7782	56.3036
14	Left Corner 2	56.4678	66.9239	54.7208	59.7596	67.9234	65.6429	66.8447	62.741	70.3342
15	Right Corner 2	49.8177	44.0165	45.0761	41.8883	43.1813	42.7232	45.3957	46.5214	48.8518
16	Wide Corridor	24.5826	19.238	24.3606	22.2213	28.61694	12.0339	21.7737	27.6879	19.9675

Table 6.3: An example lookup table after convergence for 16 clusters to learn the wall following behaviour

Cluster	*Label*	*Action*
1	Right Wall 1	0^o
2	Open Space 1	60^o
3	Dead End 1	-90^o
4	Right Corner 1	60^o
5	Front Wall 1	90^o
6	Front Wall	90^o
7	Left Corner 1	-60^o
8	Left Wall	0^o
9	Corridor 1	20^o
10	Dead End 2	90^o
11	Right Wall 2	0^o
12	Open Space 2	40^o
13	Corridor 2	0^o
14	Left Corner 2	-90^o
15	Right Corner 2	90^o
16	Wide Corridor	0^o

Table 6.4: Learned rule base after convergence for 16 clusters in learning the wall following behaviour.

direction. Otherwise, the robot could just simply continue to rotate in a small circular orbit. This trajectory may appear to have no purpose, however it is still successful in avoiding obstacles.

For these reasons, behaviour based architectures should incorporate multiple simple behaviours. While the designer has the challenge of designing the individual functionalities of the behaviours prior to execution, they are also met with the challenge of coordinating them with limited knowledge of the real world environment. To give some goal directedness to the loopy behaviour in Figures 5.10 and 5.11, two low-level navigation behaviours are combined: obstacle avoidance and goal finding using a technique called *context dependent blending* (CDB). Each behaviour has its own separate fuzzy logic controller which produces separate preferred directions of

movement based on the same crisp inputs. Sometimes, the direction of movement for each fuzzy logic controller conflicts with each other and often, a fair combination (i.e., the average value) of the two movements will produce undesirable actions. The behaviours should be compromised and fused together in such a way that takes advantage of more flexible combinations of the preferred control actions. This is on the contrary to some popular arbitration schemes (i.e., [16], [27]) which switch or activate single behaviours on and off. The disadvantage of these arbitration schemes is evident when a movement from the activated behaviour is taken that does not defy any desired performance features, but perhaps is less efficient from a more global perspective.

CDB [67] is an improvement to these arbitration schemes because rather than turning behaviours on and off, preferences are weighted and fused together in a trade-off control manner. The weight placed on each behaviour's desired control action can only be decided during execution and in response to the current situation of the environment. To calculate the weight of each behaviour during runtime, the relationship between the *context* (conditions from the environment) and its preferred *objective* (behaviour) in that context must be determined prior to execution. In the case of goal finding and obstacle avoidance, the goal finding behaviour is most effective when it is in an open space. When it approaches obstacles in its path towards the goal, the obstacle avoidance behaviour should be preferred to enable reactive maneuvers around the obstacles.

For CDB, a separate higher level fuzzy logic controller is used to decide the weight of each behaviour relative to the distance to the closest obstacle. If there are no obstacles, the high-level fuzzy logic controller will favour the goal finding behaviour. Otherwise, the presence of close obstacles will emphasize the obstacle avoiding behaviour. Thus, the preferences of the goal-seeking behaviour are still considered during the reactive obstacle avoiding maneuvers. This biases the control

choices toward the achievement of the common goal.

CDB Fuzzy Logic Controller

The input label for the higher level fuzzy logic controller for CDB is *context* which, in this case, refers to the closest distance to an obstacle. Thus, the crisp input for *context* is determined by taking the minimum distance from the 8 front sonar ranges. The two fuzzy sets for *context* are *close* and *far*. The triangular membership functions shown in Figure 6.4 indicate how fuzzification is performed.

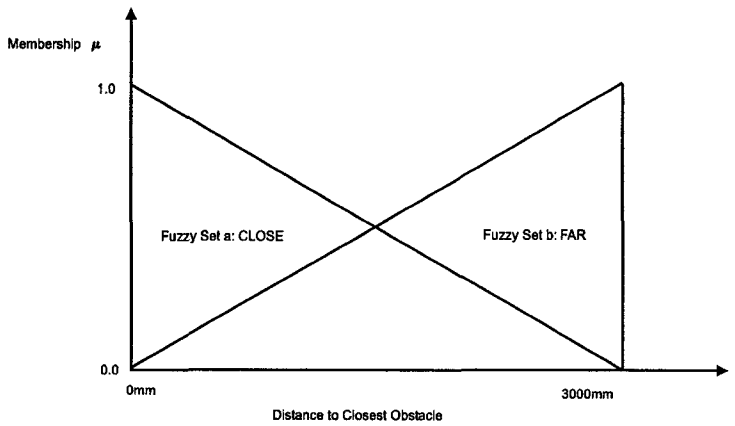

Figure 6.4: Membership functions for the *context* input for CDB

The rule structure of the fuzzy rule base has the form

If *context* is C_i, then *objective* is O_i

The two objectives are *avoid obstacles* and *go to goal*. Thus, the two rules in the CDB rule base are

If *context* is *close*, then *objective* is *avoid obstacles*

If *context* is *far*, then *objective* is *go to goal*

The output membership functions are singletons. Each objective has a desired control action, a heading value, which are the numerical values of each singleton function.

Thus, the control action for each behaviour, and similarly, the value of the two singleton functions, are dynamic and are determined by the two lower level controllers:

- The obstacle avoiding controller uses the learned situation-based FCS discussed in Chapter 4
- The goal finding controller compares the relative heading of the robot to the goal and the absolute position heading of the robot to determine the desired heading change to reach the goal.

The COG defuzzification method is used to calculate the final crisp output.

Figure 6.5 shows a trajectory of the resulting goal finding system with obstacle avoidance. It is evident that the robot avoids and tracks around obstacles that lie between itself and the destination. The transition between dominant behaviours is fluid and the robot's path towards the goal is smooth. Another trajectory of the goal finding system in a different environment is shown in Figure B.5 of the Appendix.

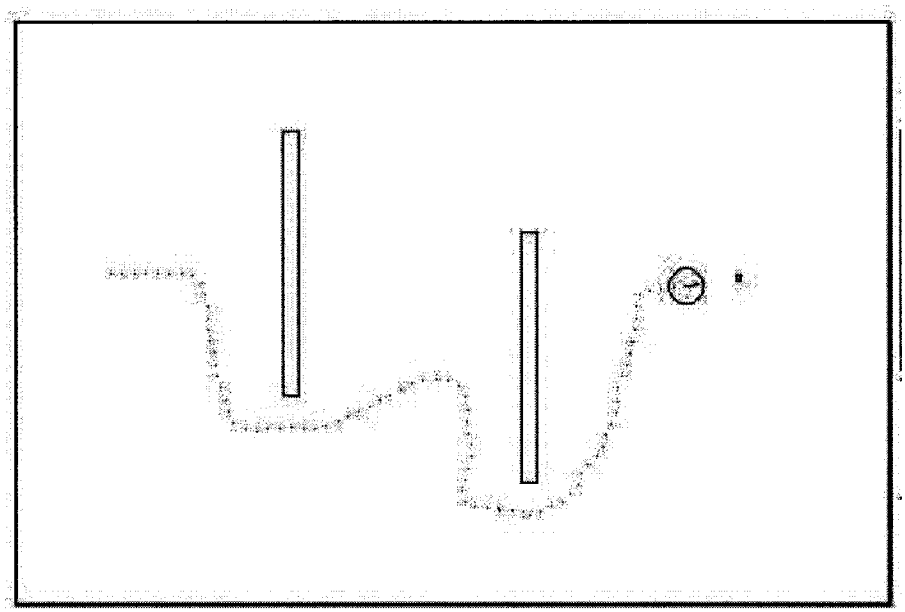

Figure 6.5: Simulated trajectory with fused behaviours of obstacle avoidance and goal finding.

"Man is still the most extraordinary computer of all."

– John F Kennedy

7 Conclusion

In this thesis, a new approach for robot navigation behaviours has been presented combining favorable properties of several existing technologies: fuzzy clustering for the classification of situations in robot's environment, fuzzy rule based systems for inferring actions suitable in different situations, and reinforcement learning to learn the rules of inference in an autonomous manner. The proposed system is referred to as an autonomous situation-based FCS.

Robot sonar data was collected by allowing the robot to follow predetermined trajectories in a set of known environments. About 1500 sonar data points were collected. Using fuzzy c-means clustering, these data points were translated into a set of situational cluster prototypes. Two parallel experiments were performed that differed in the number of fuzzy clusters used in the FCS: one experiment used 8 clusters while the other used 12 clusters. The resulting cluster vectors were used as the input situational fuzzy sets in the situation-based FCS's fuzzy rule base. The heading

output was determined by 7 possible actions represented as singletons. The FCS learned the situation-action mapping which maintained obstacle avoidance through means of reinforcement learning. In particular, a temporal-difference learning method called Q-learning was used for updating the utility values in reinforcement learning. The robot was allowed to roam in a training environment to learn a rule base which satisfies the obstacle avoidance requirements. It was determined that the FCSs with a greater number of cluster prototypes had longer learning periods: the FCS with 8 clusters required an average of 1582 iterations until convergence while the FCS with 12 clusters had an average of 3160 iterations. However, the performance of the system with the greater number of clusters was relatively better with respect to the number of collisions and rewards received after convergence. Several trials were run for both parallel experiments and all trials managed to converge to a rule base that maintained obstacle avoidance. The derived rule bases were similar for each trial. Moreover, the rule bases comprised of rules that might have been intuitively written by a human designer. After convergence, the trained robot was placed in a novel environment and obstacle avoidance was still maintained. Thus, with the reduction of the inputs space through fuzzy c-means clustering, reinforcement learning was successful in learning this low-level behaviour.

The main goal of this research was to learn the obstacle avoidance behaviour for mobile robots. To demonstrate the flexibility and generality of this reinforcement learning technique, the situation-based FCS was also applied towards other navigational behaviours. The *wall following* behaviour was used as the application for this experiment since its reward function could be well defined. The learning methodology of the situation-based FCS remained constant while the reward function was changed to define the new desired behaviour. Initially, the same 8 clusters and output singleton membership functions from the previous experiment were used. However, the wall following behaviour required more precision in terms of input

situations and output actions (heading values). Thus, the number of clusters and the number of output singleton functions were increased. Although the situation-based FCS was able to learn the wall following behaviour, learning was difficult due to the increased rule space and the complexity of the behaviour.

Robots with single behaviours typically do not serve any practical purpose in realistic robots. Behaviour based robots involve one or more low level behaviours arbitrated together to fulfill the high level goal of the robot. Thus, single behaviours may be combined to create more sophisticated mobile robots. The two behaviours used to demonstrate behaviour arbitration are goal finding and obstacle avoidance. The action desired by *goal finding* is generated through means of inspection. The action desired by *obstacle avoiding* is generated using the obstacle avoiding situation-based FCS. The two desired actions are arbitrated using a higher-level fuzzy logic controller which decides the strength of each action to send as the crisp output for the mobile robot based on the context. This method is called context dependant blending. The result is a smooth compromise between obstacle avoiding and goal finding towards the completion of a common goal.

The navigation methods described in this thesis are successful because they are inspired by intuitive sensing and decision making processes made when navigating through environments. Although navigational behaviours seem almost trivial, the general thought process would be to classify oneself to their current situation sensed in the immediate environment and react based on that situation. Negative reinforcements from the environment would encourage the search for more beneficial actions. Also, the natural curiosity or inquisitiveness may lead one to explore different alternatives. If there are multiple behaviours to incorporate, one will also reason to compromise between these behaviours to smoothly satisfy the main purpose.

Future Work

Navigational robots in general, and the ones used in this thesis, are still far from being completely autonomous – thus there is still a growing need for research in this field. Although the research in this thesis provides successful results, this study introduces the need for future work. The following are three areas of work that could significantly improve the autonomy of the situation-based FCS.

- To further increase the degree of autonomy of the situation-based navigation system, the current off-line clustering of situations could be replaced by an on-line incremental clustering system that would create models of new situations as they arise during the robot's interaction with the world. Situational clusters would be modified as more representational sets of cluster prototypes would form with the new data. Thus, the learning would be completely online and more adaptable to unexpected situations and dynamic environments.

- The issue of the number of clusters or situations should also be dealt with in a more autonomous and incremental manner. As of now, the number of clusters is a decision on the part of the human designer. The designer must be careful that the number of clusters chosen is able to adequately represent the important situations in the environment to learn the desired behaviour. However, the number of clusters should not be so high such that learning will be too computationally intensive. For example, with wall following, more clusters were required to represent all the situations that would allow the robot to properly align itself with the wall. The appropriate number of clusters is difficult to estimate without several trial and error experiments. Thus, an online learning method could be devised that learns the optimal number of clusters with respect to the environment and goal.

- A final component that could increase the autonomy of the situation-based system is to learn *how* different behaviours should be arbitrated. Currently,

CDB is used as a high-level controller to assess the environment and decide on the importance of each behaviour. This high-level fuzzy controller is pre-determined by the human designer. They must decide on the significant contexts and the relationship between contexts and behaviours. However, this relationship may also be learned using similar methods that were used to learned navigational behaviours in the situation-based FCS. Since the situations are already determined through fuzzy c-means clustering, the situation can be the context and the situational clusters can be the fuzzy sets of the context. The reinforcement learning method would be used to determine which behaviour is most important in each situation. Learning should be performed in a hierarchical order (i.e. first, learn the lower-level behaviours, then learn how to arbitrate them).

Indeed, the above suggestions would substantially require more time and computation. Along with learning the individuals behaviours, the FCS would be faced with the challenges of learning the incremental cluster centers, the number of clusters, or the arbitration of the learned low-level behaviours. Despite the challenges, these features would further diminish the need for human decisions, especially in more complex systems, and would create a more autonomous and adaptable controller.

Bibliography

[1] ActivMedia Robotics, *Pioneer 2/PeopleBot Operations Manual*, October 2001.

[2] P.E. Agre and D. Chapman, Abstract Reasoning as Emergent from Concrete Activity, In *Reasoning About Actions and Plans - Proceedings of the 1986 Workshop*, pp. 411–424, Los Altos, California, 1986.

[3] R.C. Arkin, Towards the Unification of Navigational Planning and Reactive Control, *AAAI Spring Symposium on Robot Navigation*, pp.1–5, Stanford University, March 1989.

[4] R.C. Arkin and R.B. Tucker, AuRA: Principles and Practice in Review, *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 9, Nos. 2-3, pp.175–188, April 1997.

[5] R. C. Arkin, Social Behaviour, In R.C. Arkin *Behavior-Based Robotics*, MIT Press, Cambridge, MA, 1998.

[6] M. Asada, M. Noda, S. Tawaratumida, Purposive Behaviour Acquisition for a Real Robot by Vision Based Reinforcement Learning, *Machine Learning*, Vol. 23, pp. 279–303, 1996.

[7] R.K. Belew, J. McInerney and N.N. Schraudolph, Evolving Neural Networks: Using the Genetic Algorithm with Connectionist Learning, In C.G. Langton, C.

Taylor, J.D. Farmer, S. Rasmussen (eds.) *Artificial Life II, SFI Studies in the Sciences of Complexity*, Addison-Wesley, Vol. 10, 1991.

[8] H.R. Beom and H. S. Cho, A Sensor-Based Navigation for a Mobile Robot Using Fuzzy Logic and Reinforcement Learning, *IEEE Transactions of Systems, Man, and Cybernetics*, Vol. 25, No. 3, 1995.

[9] A. Berlanga, A. Sanchis, P. Isasi and J. M. Molina, A General Coevolution Method to Generalize Autonomous Robot Navigation Behavior, *Proceedings of the 2000 Congress on Evolutionary Computation*, Vol. 1, Nos. 16-19, pp.769–776, 2000.

[10] J. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.

[11] A. Bonarini, Learning Dynamic Fuzzy Behaviours from Easy Missions, In *Proceedings of the 1996 International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, pp. 1223–1228, 1996.

[12] A. Bonarini, Evolutionary Learning of Fuzzy Rule: Competition and Cooperation, In W. Pedrycz (ed.), *Fuzzy Modelling: Paradigms and Practice*, pp. 265–284, Kluwer Academic Press, Norwell, MA, 1996.

[13] A. Bonarini, Reinforcement Distribution for Fuzzy Classifiers: A Methodology to Extend Crips Algorithms, In *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI) - Evolutionary Computation*, pp. 51–56, IEEE Computer Press, Piscataway, NJ, 1998.

[14] A. Bonarini, Evolutionary Learning, Reinforcement Learning, and Fuzzy Rules for Knowledge Acquisistion in Agent-Based Systems, In *Proceedings of the IEEE*, Vol. 89, No. 9, pp. 1334–1346, 2001.

[15] V. Braitenberg, *Vehicles: Experiments in Synthetic Psychology*, MIT Press, Cambridge, MA, 1984.

[16] R.A. Brooks, A Robust Layered Control System for a Mobile Robot, *IEEE Transactions on Robotics and Automation*, Vol. 2, No. 1, pp. 14–23, 1986.

[17] R.A. Brooks and A.M. Flynn, A Robot Beings, In proc. *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pp. 2–10, Tsukuba, Japan, 1989.

[18] R.A. Brooks, Intelligence without Representation, *Artificial Intelligence*, Vol. 47, pp. 139–159, 1991.

[19] R.A. Brooks, Artificial Life and Real Robots, In F.J. Vareal and P.Bourgine (eds.) *Towards a Practice of Autonomous Systems.* Proceedings of the First European Conference on Artificial Life, pp. 3–10, MIT Press/Bradford Books, Cambridge, Ma, 1992.

[20] J.J. Buckley and Y. Hayashi, Fuzzy Neural Networks: A Survey, *Fuzzy Sets and Systems*, Vol. 66, pp. 1–13, 1994.

[21] J. Casillas, O. Cordón, M.J. Del Jesus, and F. Herrera, Genetic Tuning of Fuzzy Rule Deep Structures for Linguistic Modelling, Technical Report DECSAI-01-01-02, Department of Computer Science and Artificial Intelligence, Universidad de Granada, January 2001.

[22] S. Chiu, Fuzzy Model Identification Based on Cluster Estimation, *Journal of Intelligent and Fuzzy Systems*, Vol. 2, No. 3, pp. 267–278, September 1994.

[23] D. Cliff, I. Harvey and P. Husbands, Artificial Evolution of Visual Control Systems for Robots, In M. Srinivisan and S. Venkatesh (eds.) *From Living Eyes to Seeing Machines*, Oxford University Press, 1996.

[24] E. Cox, *The Fuzzy Systems Handbook*, Academic Press, 1994.

[25] M. Dorigo, M. Colombetti, Robot Shaping: Developing Situated Agents Through Learning, *Artificial Intelligence*, Vol. 71, No. 2, pp. 321–370, 1994.

[26] A. Dubrawski, J.L. Crowley, Self-Supervised Neural System for Reactive Navigation, In proc. *IEEE International Conference on Robotics and Automation*, pp. 2076–2081, San Diego, May 1994.

[27] R. Firby, An Investigation into Reactive Planning in Complex Domains, In *Proceedings of the National Conference on Artificial Intelligence (AAAI 87)*, pp. 202-206, Seattle, WA, 1987.

[28] D. Franklin and N. Prince and M. Tsai and R. Yu, Fuzzy Logic, Technical Report, Department of Electrical and Computer Engineering, Michigan State University, April 1998.

[29] E. Gat, Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robots, In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pp. 809–815, 1992.

[30] D. Gu and H. Hu, Reinforcement Learning of Fuzzy Logic Controllers for Quadruped Walking Robots, In *Proceedings of 15th World Congress of the International Federation of Automatic Control (IFAC)*, Barcelona, Spain, July 2002.

[31] D. Gu and H. Hu and L. Spacek, Learning Fuzzy Logic Controller for Reactive Robot Behaviours, In *Proceedings of IEEE/ASME International Conference on Advanced Intellligent Mechatronics*, 2003.

[32] S.T. Hagen, D. l'Ecluse and B. Kröse, Q-Learning for Mobile Robot Control, In *Proceedings of the 11th Belgian-Dutch Conference on Artificial Intelligence (BNAIC 99)*, Vol. 3-4, pp. 203–210, 1999.

[33] M.E. Harmon, S.S. Harmon, Reinforcement Learning: A Tutorial, http://www-anw. cs.umass.edu./ mharmon/rltutorial/frames.html, 1996.

[34] J.A. Hartigan, *Clustering Algorithms*, John Wiley and Sons, New York, 1975.

[35] F. Herrera and L. Magdalena, Genetic Fuzzy Systems: A Tutorial, In R. Mesiar and B. Riecan (eds.) *Fuzzy Structures. Current Trends.* Proceeedings of the 7th International Fuzzy Systems Association World Congress, Vol. 13, pp. 93–121, Tatra Mountains Mathematical Publications, 1997.

[36] F. Hoffmann, Soft Computing Techniques for the Design of Mobile Robot Behaviours, *Journal of Information Sciences*, Vol. 122, Nos. 2-4, pp. 241–258, February 2000.

[37] F. Hoffmann, Evolutionary Algorithms for Fuzzy Control System Design, In *Proceedings of the IEEE Special Issue on Industrial Applications of Soft Computing*, Vol. 89, No. 9, pp. 1318–1333, September 2001.

[38] J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975.

[39] F. Hoppner, F. Klawonn, R. Kruse and T. Runkler, Fuzzy Cluster Analysis, John Wiley, Chichester, UK, 1999.

[40] L.P. Kaelbling and S.J. Rosenschein, Action and Planning in Embedded Agents, In P. Maes (ed.) *Designing Autonomous Agents*, MIT Press, pp. 35–48, 1990.

[41] T. Kohonen, *Self-Organizing Maps*, Springer-Verlag, 3rd Edition, Berlin, 1995.

[42] D.A. Kontoravdis, A. Likas, and A. Stafylopatis, Collision-Free Movement of an Autonomous Vehicle Using Reinforcement Learning, In B. Neumann (Ed.) *Proceedings of the European Conference on Artificial Intelligence (ECAI 92)*, John Wiley and Sons Ltd, pp. 666–670, 1992.

[43] T. Kovacs and S.I. Reynolds, A Proposal for Population-Based Reinforcement Learning, Technical Report CSTR-03-001, Department of Computer Science, University of Bristol, January 2003.

[44] B. Kröse and J. van Dam, Neural Vehicles, In P. van der Smagt and O. Omidvar (eds.) *Neural Systems for Robotics*, Academic Press, pp. 271-296, San Diego, 1997.

[45] C.C. Lee, Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part I, In *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 20, No. 2, pp. 408–418, 1990.

[46] S. Lee and S. Cho, Emergent Behaviours of a Fuzzy Sensory-Motor Controller Evolved by Genetic Algorithm, *IEEE Transactions on Systems, Man and Cybernetics, Part B*, Vol. 31, No. 6, pp. 919–929, 2001.

[47] L.J. Lin, Hierarchical Learning of Robot Skills by Reinforcement, In *Proceedings of the IEEE International Conference on Neural Networks*, Vol. 1, pp. 181–186, 1993.

[48] D.M. Lyons and A.J. Hendriks, A Practical Approach to Integrating Reaction and Deliberation, In J. Hendler (ed.) *Artificial Intelligence Planning Systems: Proceedings of the First International Conference*, pp. 153–162, 1992.

[49] M. Maeda, Y. Maeda and S. Murakami, Fuzzy Drive Control of an Autonomous Mobile Robot, *Fuzzy Sets and Systems*, Vol. 39, pp. 195-204, 1991.

[50] P. Maes, The Dynamics of Action Selection, In M. Kaufmann (ed.) *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI 89)*, pp. 991–997, Detroit, USA, 1989.

[51] S. Mahadevan and J. Connell, Scaling Reinforcement Learning to Robotics by Exploiting the Subsumption Architecture, In *Proceedings of the Eighth International Workshop on Machine Learning*, pp. 328–323, 1991.

[52] L. Magdalena and J.R. Velasco, Evolutionary Based Learning of Fuzzy Controllers, In W. Pedrycz (ed.) *Fuzzy Evolutionary Computation*, Kluwer Academic Press, pp. 249–268, 1997.

[53] M.J. Matarić and R. A. Brooks, Learning a Distributed Map Representation Based on Navigation Behaviors, In *Proceedings of the 1990 USA-Japan Symposium on Flexible Automation*, pp. 499–506, Kyoto, Japan, July 1990.

[54] J. Millán and C. Torras, Efficient Reinforcement Learning of Navigation Strategies in an Autonomous Robot, In *Proceedings of the International Conference on Intelligent Robotic Systems (IROS 94)*, pp. 15–22, Munchen, September 1994.

[55] S. Mitra and Y. Hayashi, Neuro-Fuzzy Rule Generation: Survey in Soft Computing Framework, In *IEEE Transactions on Neural Networks*, Vol. 11, No. 3, pp. 748–768, May 2000.

[56] M. Mizumoto, Improvement Methods of Fuzzy Controls, In *Proceedings of the 3rd International Fuzzy Systems Association World Congress (IFSA 89)*, pp. 60-62, Seattle, WA, 1989.

[57] H. Moravec, Robots After All, *Communications of the Association for Computing Machinery (ACM)*, Vol. 46, No. 10, pp. 90–97, October 2003.

[58] L.N. Moussi, R.R. Gudwin, F.J. Von Zuben and M.K. Madrid, Neural Networks in Classifier Systems (NNCS): An Application to Autonomous Navigation, In V.V. Kluev and N.E. Mastorakis (eds.) *Advances in Signal Processing, Robotics*

and Communications, Electrical and Computer Engineering Series, WSES Press, pp. 256–262, 2001.

[59] N. Nilsson, Shakey the Robot, Technical Report, AI Center Technical Note 323, SRI International, California, 1984.

[60] T. Oates, M.D. Schmill and P.R. Cohen, A Method for Clustering the Experiences of a Mobile Robot that Accords with Human Judgements, In *Proceedings of the 17th National Conference on Artificial Intellilgence (AAAI 2000)*, pp. 846–851, 2000.

[61] N.R. Pal and J. Bezdek, On Cluster Validity for the Fuzzy C-Means Model, In *IEEE Transactions on Fuzzy Systems*, Vol. 3, pp. 370–379, June 1995.

[62] W. Pedrycz and F. Gomide, An Introduction to Fuzzy Sets, MIT Press, 1998.

[63] T.J. Prescott, and J.E.W. Mayhew, Obstacle avoidance through Reinforcement Learning. In J.E. Moody, S.J. Hanson and R.P. Lippman (eds.) *Advances in Neural Information Processing Systems 4*, Morgan Kaufmann, New York, 1992.

[64] J. Provost, B.J. Kuipers and Risto Miikkulainen, Self-Organizing Perceptual and Temporal Abstraction for Robot Reinforcement Learning, In *Artificial Intelligence in Industry (AIII) Workshop on Learning and Planning in Markov Processes*, 2004.

[65] M. Roubens, Pattern Classification Problems and Fuzzy Sets, *Fuzzy Sets and Systems*, Vol. 1, pp. 239–253, 1978.

[66] A. Saffiotti, E.H. Ruspini and K. Konolige, A Fuzzy Controller for Flakey, an Autonomous Mobile Robot, Technical Report, AI Center Technical Note 529, SRI International, California, 1993.

[67] A. Saffiotti, E.H. Ruspini and K. Konolige, Blending Reactivity and Goal Directedness in a Fuzzy Controller, In *Proceedings of the 2nd IEEE Conference on Fuzzy Systems*, pp. 134-139, 1993.

[68] A. Saffiotti, The Uses of Fuzzy Logic in Autonomous Robot Navigation, *Soft Computing*, Vol. 1, No. 4, pp. 180–197, 1997.

[69] A. Saffiotti, Fuzzy Logic in Autonomous Robot Navigation, A Case Study, In E. Ruspini, P. Bonissone, and W. Pedrycz (eds.) *Handbook of Fuzzy Computation, Chapter G6: Autonomous Robot Navigation*, Oxford Univ. Press and IOP Press, 1998.

[70] G. Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison Wesley, 1989.

[71] M. Sugeno, M. Nishida, Fuzzy Control of Model Car, *Fuzzy Sets and Systems*, Vol. 16, pp. 103–113, 1985.

[72] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, 1998.

[73] H. Takagi, Fusion Technology of Fuzzy Theory and Neural Networks - Survey and Future Directions, In *Proceedings of the First International Conference on Fuzzy Logic and Neural Networks*, pp. 13–26, July 1990.

[74] T. Takeuchi, Y. Nagai and N. Enomoto, Fuzzy Control of a Mobile Robot for Obstacle Avoidance, *Information Sciences*, Vol. 43, pp. 231–248, 1998.

[75] S. Thongchai, Behaviour-Based Learning Fuzzy Rules for Mobile Robots, In *Proceedings of the 2002 American Control Conference*, Alaska, 2002.

[76] E. Tunstel and T. Lippincott and M. Jamshidi, Introduction to Fuzzy Logic Control With Application to Mobile Robots, In *Proceedings of the First National*

Students Conference of the National Alliance of NASA University Research Centers, North Carolina A&T State University, 1997.

[77] F.F. Rivera, E.L. Zapata, and J.M. Carazo, Cluster Validity Based on the Hard Tendency of the Fuzzy Classification, *Pattern Recognition Letters*, Vol. 11, pp. 7–12, 1990.

[78] W.G. Walter, *The Living Brain*, W.H. Norton, New York, 1953.

[79] K. Ward, A. Zelinsky and P. McKerrow, Learning Robot Behaviours by Extracting Fuzzy Rules From Demonstrated Actions, *AJIPS*, Vol. 6, No. 3, pp. 154–163, 2001.

[80] J. Watkins, Technical Note: Q-Learning, *Machine Learning*, Vol. 8, pp. 279-292, 1992.

[81] L.A. Zadeh, Fuzzy Sets, *Information and Control*, Vol. 12, pp. 338–353, 1965.

A

Obstacle Avoidance

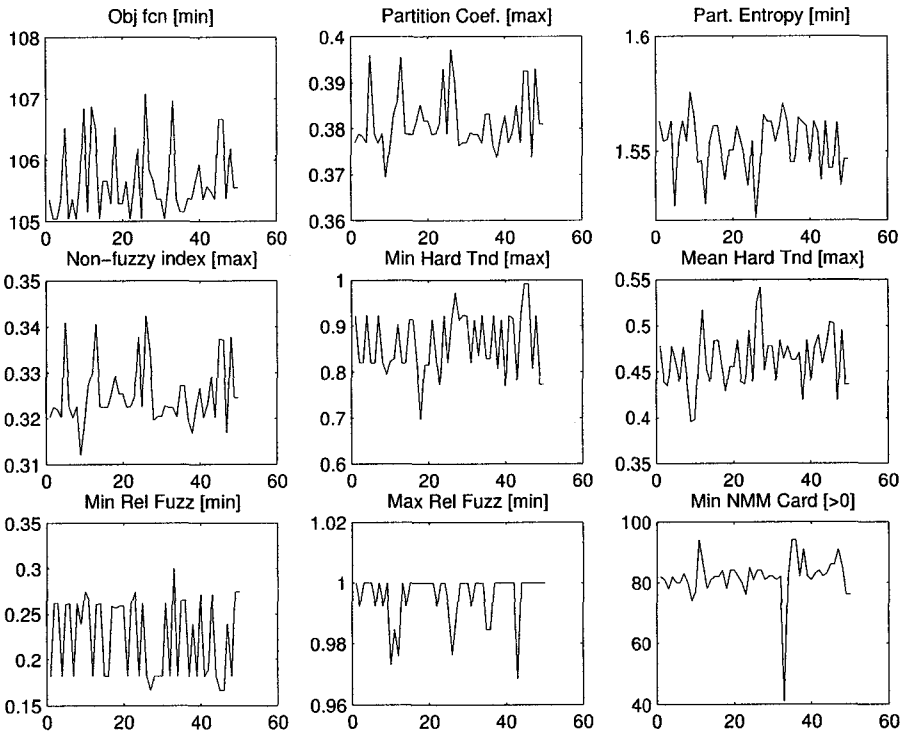

Figure A.1: Functionals with respect to trial number in fuzzy c-means clustering for 12 clusters

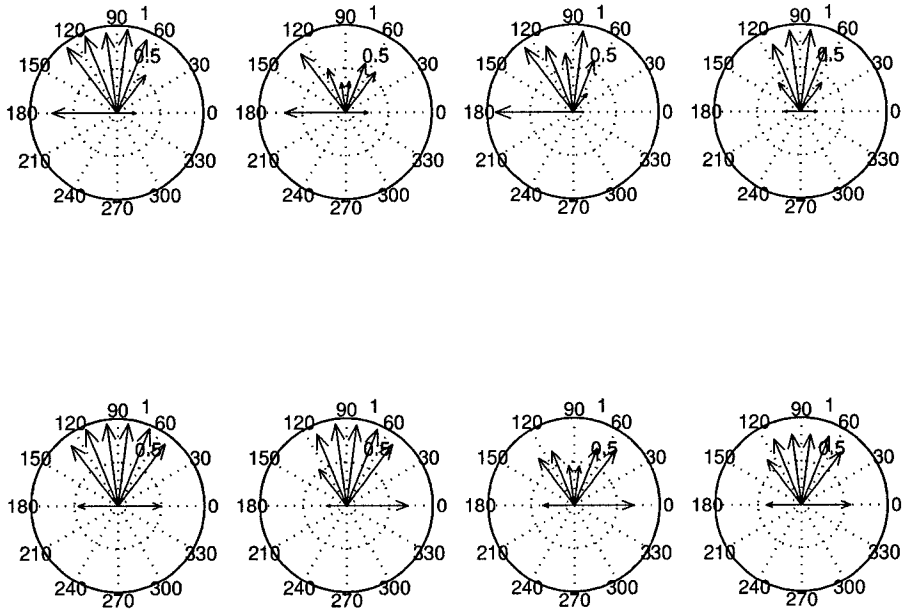

Figure A.2: Trial 1: 8 cluster prototypes of fuzzy situations from fuzzy c-means clustering

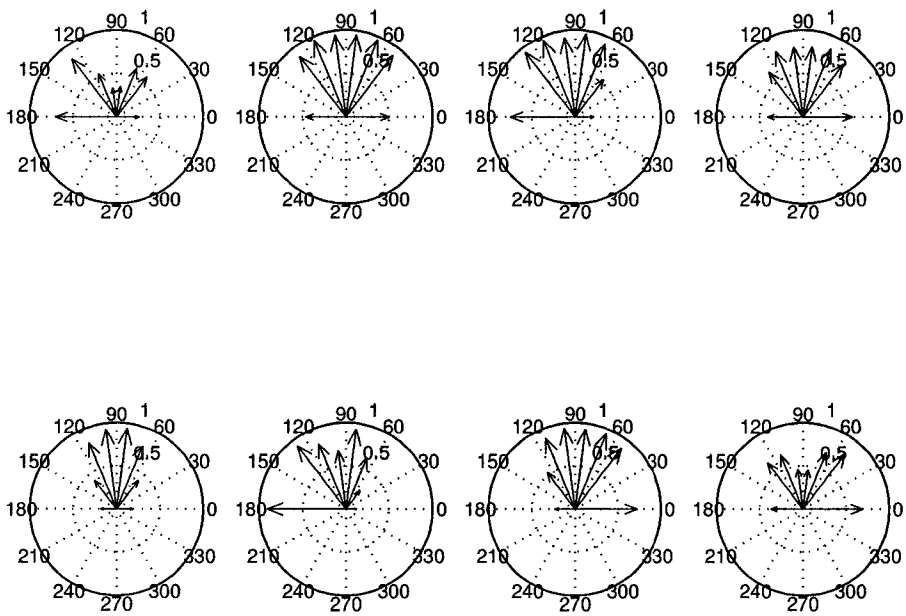

Figure A.3: Trial 2: 8 cluster prototypes of fuzzy situations from fuzzy c-means clustering

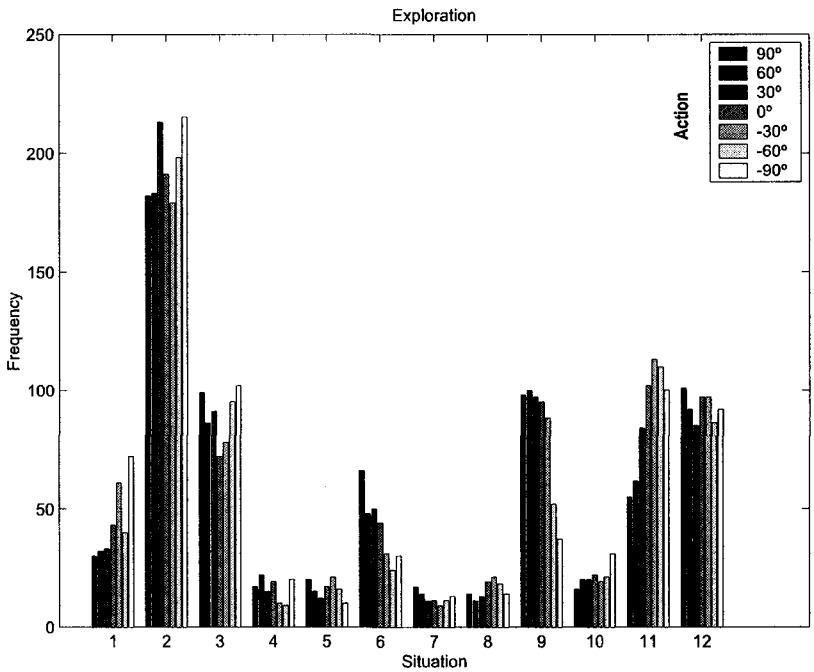

Figure A.4: Exploration distribution for 12 clusters in obstacle avoidance experiment

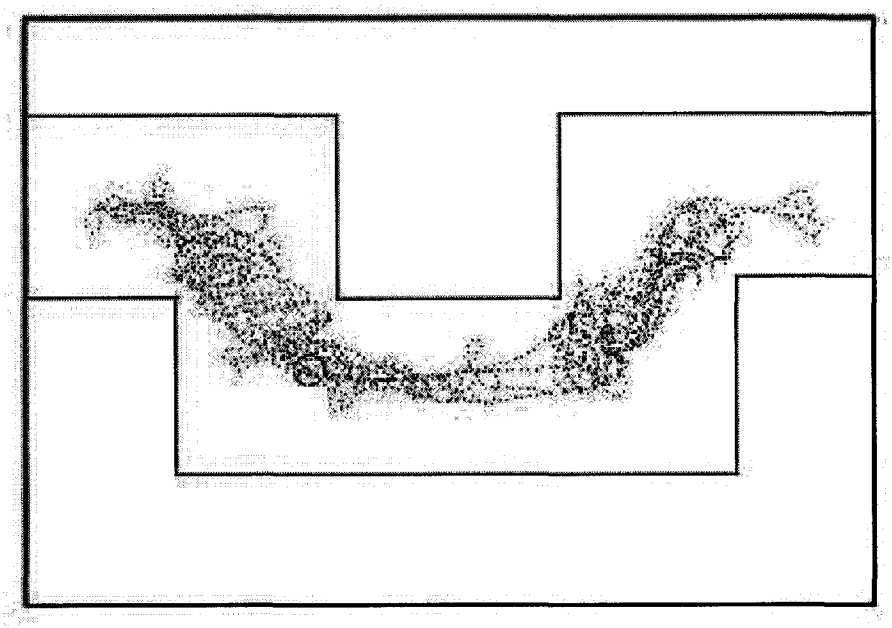

Figure A.5: Simulated trajectory with obstacle avoidance behaviour in novel environment, with 12 clusters

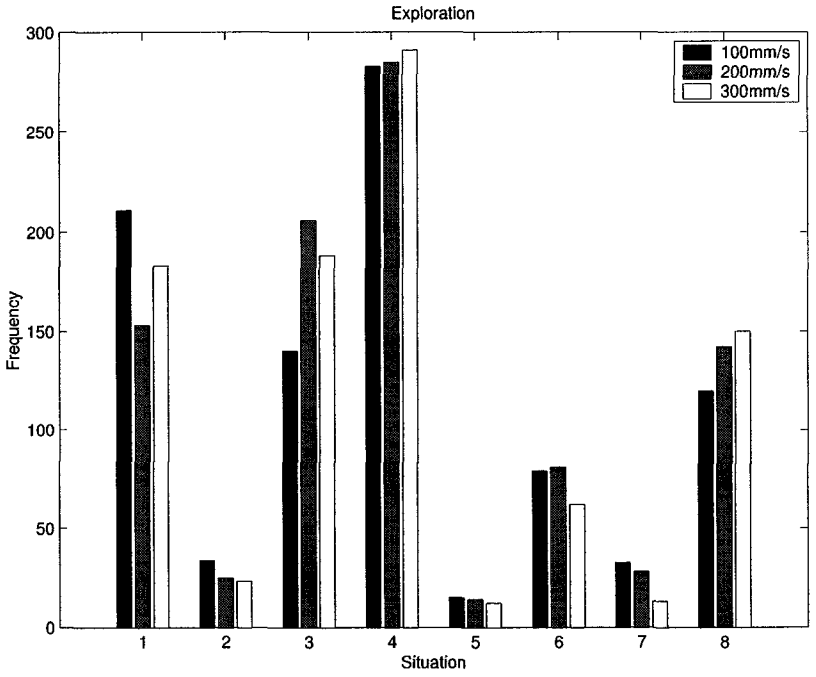

Figure A.6: Exploration distribution for 8 clusters in learning optimal velocity in obstacle avoidance experiment

Symbol	Description	Value
Fuzzy c-means Clustering		
c	Number of Clusters	8, 12
Fuzzy Logic Controller		
vel_h	Velocity for learning the heading control action	150 mm/s
vel_v	Velocity for learning the velocity control action	100 mm/s - 300 mm/s
$NumActions_h$	Number of output singleton functions for heading	7
$NumActions_v$	Number of output singleton functions for velocity	3
Q-Learning		
γ	Discount Factor	0.9
α	Learning Rate	0.3
Reinforcement Learning		
b	Number of collision free iterations between rule base changes	5
l	Maximum number of iterations	5000
p	Number of constant rule base changes for convergence	50

Table A.1: Design parameters for the situation-based FCS to learn the obstacle avoidance behaviour

Cluster	*Label*	*Action*
1	Front Wall	90^o, -90^o
2	Right Corner	90^o,60^o
3	Right Wall	60^o,90^o,30^o,0^o
4	Open Space	0^o,30^o,-30^o
5	Corridor	0^o
6	Left Wall	-60^o,-90^o,-30^o
7	Left Corner	-90^o,-60^o
8	Wide Corridor	0^o,30^o,-30^o

Table A.2: Observed alternate rules for obstacle avoidance among several trials in order of frequency (8 clusters)

Cluster	*Label*	*Action*
1	Left Corner 1	-90^o,-60^o
2	Open Space 1	0^o,30^o,-30^o
3	Front Wall	90^o, -90^o,60^o
4	Corridor 1	0^o
5	Right Corner 1	90^o,60^o
6	Right Corner 2	90^o,60^o,30^o
7	Dead End	90^o,-90^o
8	Corridor 2	0^o, 30^o
9	Right Wall	60^o,90^o,30^o,0^o
10	Left Corner	-90^o,-60^o
11	Left Wall	-60^o,-30^o,-90^o,
12	Open Space 2	0^o,-30^o,30^o

Table A.3: Observed alternate rules for obstacle avoidance among several trials in order of frequency (12 clusters)

Trial	*Iterations*
1	2561
2	1904
3	1679
4	1512
5	1716
6	1129
7	1334
8	1227
9	2472
10	2132
11	1441
Avg	1737

Table A.4: Number of iterations until convergence for 8 clusters in the optimal velocities in obstacle avoidance experiments.

B

Wall Following & Behaviour Fusion

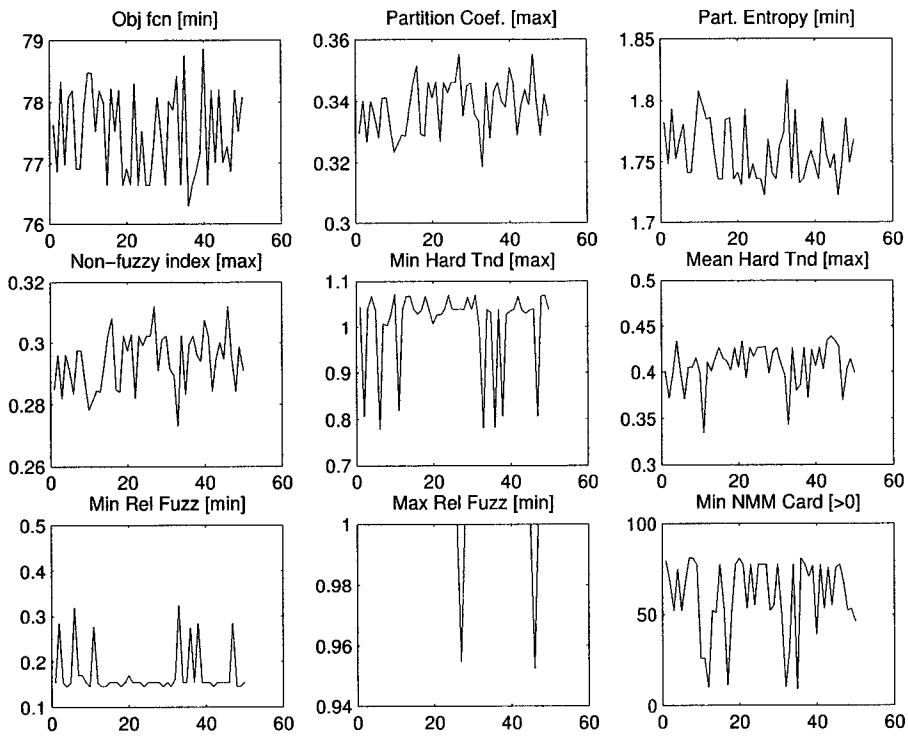

Figure B.1: Functionals with respect to trial number in fuzzy c-means clustering for 16 clusters

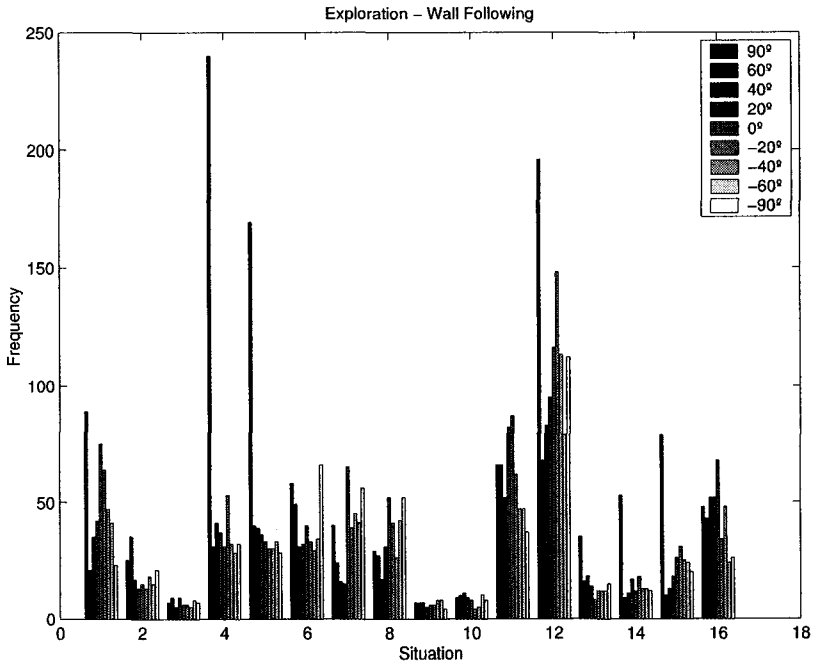

Figure B.2: Exploration distribution for 16 clusters in wall following experiments

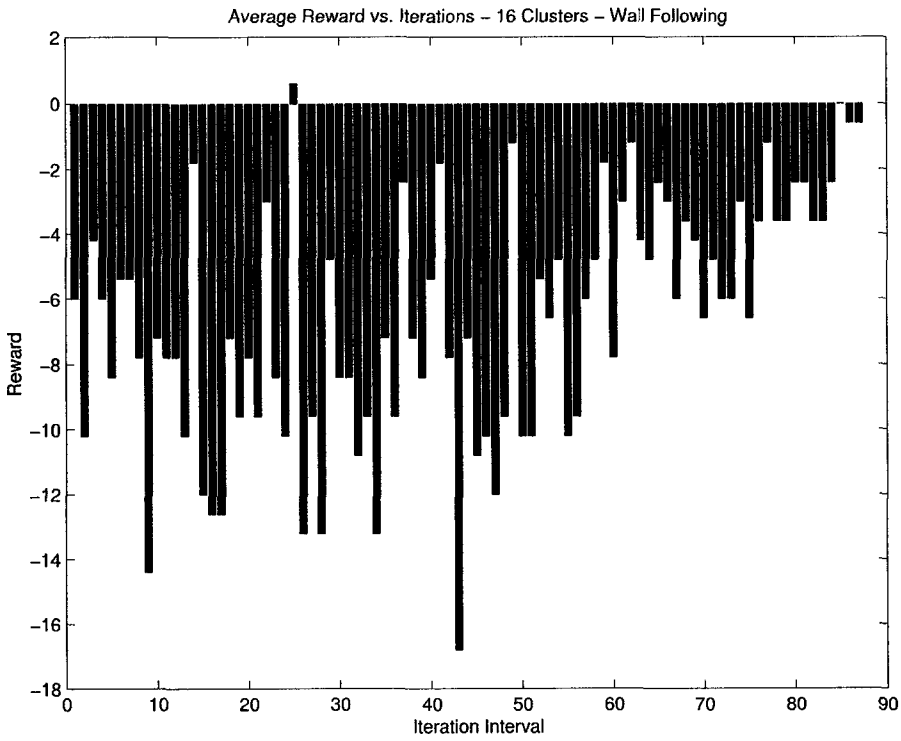

Figure B.3: Reward values with respect to 50 iteration intervals for 16 clusters, learning wall following

Symbol	Description	Value
Fuzzy c-means Clustering		
c	Number of Clusters	16
Fuzzy Logic Controller for Wall Following		
vel_h	Velocity for learning wall following	150 mm/s
min_d	Minimum allowable distance to a wall	15cm
max_d	Maximum allowable distance to a wall	35 cm
$NumActions_w$	Number of output singleton functions	9
Q-Learning		
γ	Discount Factor	0.9
α	Learning Rate	0.2
Reinforcement Learning		
b	Number of collision free iterations between rule base changes	5
l	Maximum number of iterations	6000
p	Number of constant rule base changes for convergence	50

Table B.1: Design parameters for the situation-based FCS to learn the wall following behaviour

Trial	*Iterations*
1	5115
2	5878
3	4379
4	5532
5	5674
6	5912
Avg	5415

Table B.2: Number of iterations until convergence for 16 clusters in the wall following experiments.

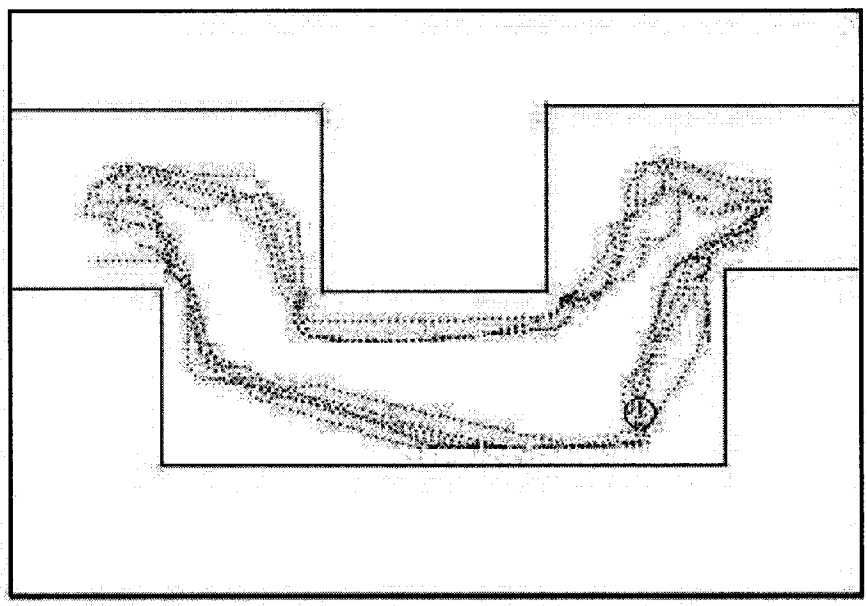

Figure B.4: Simulated trajectory with wall following behaviour in novel environment

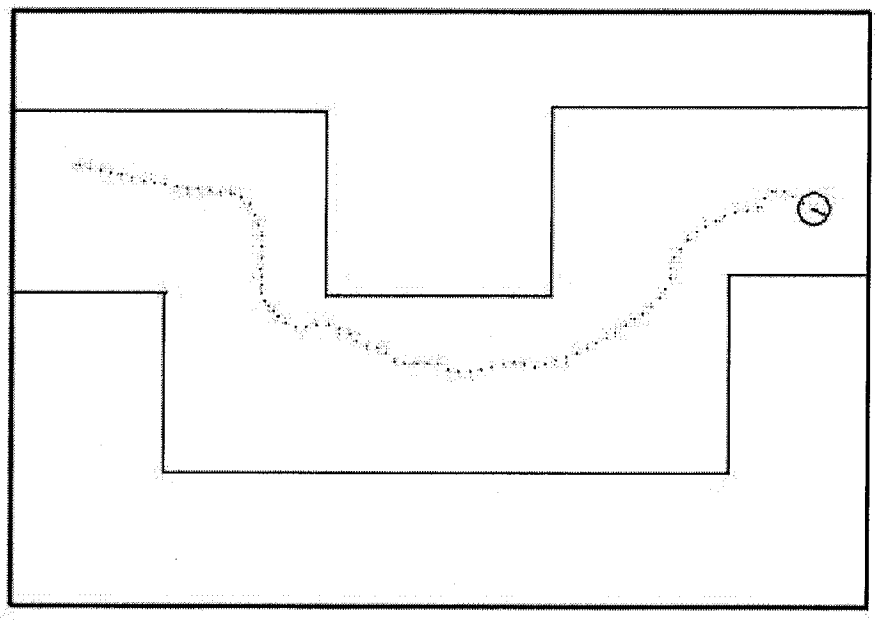

Figure B.5: Simulated trajectory with goal finding and obstacle avoidance in the same environment as in Figure B.4