Applying Reinforcement Learning in playing Robosoccer using the AIBO

Subhasis Mukherjee
Graduate School of Information Technology and Mathematical Science (GSITMS)
University of Ballarat
Ballarat, Victoria 3353, Australia

A thesis submitted for the degree of
Masters by Research in Computing
February 17, 2010

Thesis Supervisors:
Prof. John Yearwood and Dr. Peter Vamplew

Abstract

Robosoccer is a popular test bed for AI programs around the world in which AIBO entertainment robots take part in the middle sized soccer event. These robots need a variety of skills to perform in a semi-real environment like this. The three key challenges are manoeuvrability, image recognition and decision making skills. This research is focused on the decision making skills. This thesis considers one particular problem in decision making in robosoccer - The goal keeper problem. The work focuses on whether reinforcement learning as a form of semi supervised learning can effectively contribute to the goal keeper's decision making when a shot is taken.

The problem could be addressed in two ways: by using a hand-coded solution to the problem or by using a learning algorithm to learn the action to be taken. The hand coding technique is a set of input output pairs provided by the programmer. On the other hand, a learning process can start with zero knowledge and will gradually learn to accomplish a task without human interference. A specific Reinforcement Learning scheme (Q-learning) is used in this thesis to address the goalkeeping problem.

An agent decomposes a complex situation into basic parts and using Q-learning it tries to take a series of optimized actions to accomplish a task and finally learns how to reach the goal. In this work the goalkeeper was trained using multiple shots taken from different positions by an attacker. We applied the skill achieved against one attacker, in other situations where two attackers were used. There One attacker passes the ball to another and the second attacker shoots the moving ball towards the goal.

The Q-learning based results were compared with a base-line strategy using hand coded goalkeeping actions contained in the University of Pennsylvania 2003 Robosoccer code [1]. It was found that the Q-learning based technique was as good as the hand-coded technique

in both cases. In fact the goalkeeper basically develops skills to follow the ball irrespective of the attacker's position.

The results indicate that Q-learning was able to help the robot learn goalkeeping successfully without human interference. This suggests that a similar learning algorithm can be used to develop successful decision making strategies for performing other tasks in robosoccer. Q-learning uses a state x action table to record the training data and this forms a database of experience for the agent's use. The size of this table is determined by the number of states and actions required to accomplish the particular task. Future work would involve reducing the size of the state x action table using different methods and approximation techniques.

Statement of authorship

Except where reference is made in the text of the thesis, this thesis contains no material published elsewhere or extracted in whole or part from a thesis by which I have qualified for or been awarded another degree or diploma.No other persons work has been used without due acknowledgement in the main text of the thesis. This thesis has not been submitted for the award of any degree or diploma in any other tertiary institution. I understand that the work may be reproduced and/or communicated for the purposes of detecting plagiarism.

Signature _____

Date _____

Name: Subhasis Mukherjee

Signature _____

Date _____

Name: Prof. John Yearwood

(Principal supervisor)

Professor of Informatics

and Director, CIAO

Acknowledgement

This research project would not have been possible without the support of many people. I would like to express my gratitude to my principal supervisor Prof. Dr. John Yearwood for his constant invaluable support and guidance from the beginning. Deepest gratitude is also due to my associate supervisor Dr. Peter Vamplew without whose knowledge, experience and assistance this research would not have been successful. I am indebted to my sister and brother-in-law who helped me in learning the ways of research during my early days in Australia. Special thanks are conveyed to administration of the school of ITMS for providing few expensive pieces of hardware for due research. It is a pleasure to thank those my fellow colleagues who helped me time to time with their experience in several ways. I also would like to express my love and gratitude to my beloved family members for their understanding and endless love during the studies.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Robots and Artificial Intelligence

Humans have always dreamed of intelligent machines as a replacement for human being in different kinds of dangerous tasks and day to day activities. Science fiction writers have used their imaginations to paint robot characters in their stories for centuries. Mary Shelly came up with the idea of a biological machine in her famous novel, *Frankenstein* [4]. It was a partially mindless creature similar to a human. Later on Karel described a story of world domination by similar creatures in his play *Rossum's Universal Robots* [5]. Actually, he wrote about intelligent robot, those who were working for people. For the last few decades wide and extensive research has been conducted to create artificially intelligent machines. However, the level of AI [1] Karel described, is still out of reach. In contrast to the science fiction writers, scientists also have contributed to robotics in different centuries.

Once, Leonardo da Vinci, produced some designs of a mechanical knight, capable of waving hands, jaws and other body parts [6]. However, this mechanical knight had to be operated by a human. It was a fantasy for scientists to create machines for simple tasks requiring precision, before the invention of electronics. Even the basics of AI were also out of reach without the help of transistors and Integrated Circuits (ICs). Once these devices started to introduce complete new faces of technology, people explicitly started to work on the idea of analogue computers. However, these devices were not sufficient to process

[1] Artificial Intelligence

millions of bits of information in a short span of time, which is a key to the development of an intelligent machine.

Broadly, the process of creating a true artificially intelligent machine could be divided into two *main categories*. They can be addressed using hardware and software both techniques. The hardware solves physical functions such as movement, maneuverability, environmental interactions and data processing power. However, the software which controls hardware, starts working under the limitations of hardware. The extent of hardware is limited within a physical boundary. However, the scope of software development is indefinite in order to make a robot more intelligent.

By the middle of twentieth century, scientists first started to build stand alone robots of different shapes and sizes. Furthermore, the concept of multi-agent operations were taken into consideration. This idea needed both stand alone capabilities for a robot and team management skills to perform a task effectively. In order to standardize and compare the concerned research outcomes based on single and multi-agent strategies from all over the world, the Robocup competition was suggested as a tool in 1995 [7]. The ultimate goal of Robocup is to create a group of autonomous robots effective as stand alone player and a team worker too. An autonomous machine (namely robot) can be used as a replacement of human workers in the long run.

1.2 Robocup

Robocup is a globally accepted event where AI programs are being tested in the form of friendly events such as soccer, rescue league and so on. Robosoccer is a part of the Robocup competition. Robosoccer is a world wide event which was founded in 1997 and it is used as a test bed for research in robotics. At the beginning, Robosoccer was played by wheeled robots with cameras fitted on top. It was conducted as a part of Robocup. Later on, the legged robot was introduced there as well. Rescue League was introduced in the Robocup along with Robosoccer to focus a part of the robotics research on search and rescue purposes. Furthermore different kinds of simulation events were introduced as well. The ultimate goal of Robocup is set to build a group of humanoids by 2050 which can defeat the human world cup champion team [8].

Along with that, Robocup has several other impacts on society. First of all, it serves as a test bed for robotics and the AI society all over the world. It has different events to accommodate almost all kinds of existing robots (except very small sized robots). In this way different robots are being tested from different points of view. As an example, a stand alone humanoid goalkeeper in robosoccer is tested against a robot opponent or a non-professional human soccer player too. The idea behind choosing soccer as a major competition event of research is to incorporate both stand alone and multi-agent skills in a robot team while playing a friendly game. The point to be noted here is, if a program could serve as an efficient AI process in the soccer environments, then many real life issues with multi-agent environment, could be resolved using this program. As an example, inaccessible and dangerous areas could be managed by a group of robots with or without minimum human supervision. This application would definitely be a cost effective and risk-free solution from different perspectives. Apart from that, a lot of general community services could benefit from a team of skilled and intelligent nonhuman workers. A dedicated friend and teacher will be available for children. Aged and disabled persons will have a companion as well as a helper in their day to day activities. Industries can use the relevant technologies as a permanent solution for a cheap and infinite source of skilled labor. The multi agent strategy could be used in some other aspects as well. An efficient traffic system or an intelligent war strategy decision making program could be developed too.

Robocup not only focuses on soccer related issues as a primary form of competition, it also runs another event called the Rescue League parallel to soccer events. This activity involves the task of searching for and locating signs of life in an artificial disaster site. Some simulator events are also being conducted in Robocup for both soccer and Rescue League. Both 2D and 3D simulation are used for the purpose. Programmers can focus on the AI problems using a simulator leaving the maneuverabilities and pattern recognition problem apart. However, some 3 dimensional simulators are almost similar to a real life gaming environment and put the virtual agents to test.

1.3 Wheeled and Legged robots: Two different kinds of mobility mechanism used in Robocup

Although both wheeled and legged robots take part in Robocup, the latter is more suitable for the Rescue League. This is due to the uneven nature of surfaces in a disaster site and debris scattered all over the area. Again, in an unknown place, such as a Martian surface, one can least expect a smooth and even surface for wheeled robots. So, it might be said here that the future belongs to legged robots and not wheeled robots. Already, small and full sized humanoids are being developed by some companies. These robots are not fully capable of doing human like maneuvers yet, but the progress is still admirable enough. The QRIO [9] can run and jump. The ASIMO [10] can serve food, detect human faces, open up a combination lock, climb stair cases and so on [9]. Some of these abilities such as running, jumping and other slow but precise movements are highly recommended for robosoccer and rescue league. These qualities can transform a robot into a perfect opponent against a human soccer player. We have chosen a quadrupedal dog shaped robot, AIBO [11] by Sony, for use in this thesis. It is equipped with a small computer and different sensors. In terms of cost it is one of the most viable choice for researchers, too. The Figure 2.2 displays a complete picture of AIBO.

1.4 The AIBO and Robosoccer

Robosoccer is a part of robocup, found in 1997. Different types of legged robots that could take part in Robosoccer, are commercially available for research purposes now. These are QRIO [9], ASIMO [10], AIBO [11], ACTROID [12] and so on. The AIBO [11]is a quadruped robot made by Sony. The last released model, ERS-7 has a natural dog like structure. The touch sensors on the back, at the chin and on the head add more realistic features into it. Additionally its processing quality is impressive too. Other bi-pedal robots like QRIO and ASIMO are too expensive and only a few organizations have been able to afford those machines due to their high prices. In contrast with that the AIBO was initially meant for entertainment purposes. So, the relatively cheap price made it a

good choice for researchers throughout the world. In 1998 a new sub event was started in the Robosoccer section, namely four-legged robot(middle size) league. Only Sony AIBO robot teams compete in this section from then on. A comprehensive study on Robosoccer is available in Chapter 2.

1.5 Software aspects in Robosoccer and in the AIBO

Previously, people had limited choice over hardware to play robosoccer. However, a number of different models are available now for this purpose. Some of them are equipped with a high degree of data processing power. As a result a number of software application possibilities have been released over time. Sony released a Linux based Software Development Kit (SDK) for the AIBO, at the beginning. Programmers had no choice but using to use that SDK at that time. However, later on a few wrappers were designed to make other languages able to communicate with the core. It made the SDK, platform independent to some extent. The *Aperius* [13] real time Operating System (OS) is used to run the onboard computer in the AIBO. This OS basically boots up the robot and performs coordination between different hardware modules, so that parallel instructions can be processed at a time. The principle of a real time OS is to produce response within a reasonable amount of time after receiving an input. Using this, the program observes the environment using the onboard sensors and produce output accordingly. A few presumed states and corresponding action pairs were used to determine the right action in a particular state during the game. This *state x action pair* approach was acceptable as a starting effort for playing Robosoccer during the first few years. However, slowly and gradually the Robosoccer environment become more complex and close to real life. Different unexpected events made it impossible for programmers to use input-output action pair as a basic idea to program the robots further. A new approach was definitely needed to tackle this issue.

1.6 Machine learning

Researchers around the world looked for an alternative way to overcome the problem of having just a few input-output pairs only for programming a robot. Finally an idea was arrived at to create an automatic system which, in a new situation, would be able to decide an optimum move based on previous experiences. In that scenario, the concept of the learning paradigm was found to be more realistic than using a straight forward input-output pair. A learning algorithm could copy the way in which an animal gains experience throughout the whole life. Generally, an animal inherits some characteristics from its parents and develops other qualities over time. Accordingly, the learning algorithm could be divided into genetic and other kinds of learning algorithms. A machine learning algorithm may consists of either learning algorithm or genetic algorithm are both of them.

1.7 Reinforcement learning

The three existing learning paradigms are supervised, semi-supervised and unsupervised techniques. A fully supervised technique requires complete human interference. It learns a few input-output pairs for accomplishing a given task [14]. However, these input-output pairs follow a pattern strictly. It may not work properly in a situation which is not a part of the learned pattern. The unsupervised technique needs no human interference during the training period and so it considers all possible ways to accomplish a task. In addition to that the unsupervised learning also learns about the working environment. So, a fully unsupervised process takes a considerable amount of *time* to complete the training for a task. The semi-supervised technique tries to bridge these two extreme processes and optimizes the *training time* vs *output performance* trade off.

Human society itself uses the philosophy of using semi-supervised training exercises in different aspects. For example, a soccer player is usually trained in a similar way. A few basic maneuverability skills are provided to him and the player sharpens his skill through practice. A similar process could be followed in the case of Robosoccer as well. However, in case of a robot it needs to learn about the environment in addition to the soccer skill. The robot player could start its training with partial knowledge of the environment and

some basic actions to follow. However, no clue should be given about the right action in a given situation. A right action is strengthened by reward point and a wrong action looses its probability by receiving punishment in the form of negative factor. This kind of semi-supervised learning system such as Reinforcement Learning (RL) could be used to train a robot to take an optimum action at any situation. The same technique(RL) is used in this thesis to program the learning agent to perform a particular task in robosoccer.

1.8 Overview

Robosoccer encapsulates many dimensions in terms of robotics applications. Both basic maneuverabilities and team coordination skills are important in order to play Robosoccer efficiently. As discussed, reinforcement learning could be used in different aspects. It is already applied in optimizing some basic maneuvering skills in the AIBO for playing Robosoccer. However, decision making process currently use hand coding. In this thesis, we have applied one of the basic RL algorithms to develop an optimized decision making process for a given task. Robosoccer is an event where AI programs are being tested in the form of soccer, where goalkeeping is one of the important tasks. In Robosoccer an attacker may take a straight shot to score a goal or more than one attacker may attack as well. Based on these scenarios two major problem areas are listed in this thesis.

- The first and the basic problem is goalkeeping against one attacker, who shoot the ball from different positions.
- The second problem is an extension of the first one. The knowledge base achieved by the goalkeeper against one attacker was used to save the goal against two attackers. There, the first attacker passes the ball to the second while it takes a shot towards the goal using that flying pass.

The research question in this thesis is "Whether a basic reinforcement learning algorithm can perform as well as hand coding/input-output pairs to solve the goalkeeping problem"?

We have trained a robot as a goalkeeper against goal kicking using RL. In the experiment, the attacker took penalty shots towards the goal and the goalkeeper tried to save

it. The keeper gained experience using several training epochs using penalty shots. Afterwards, we tested the RL results against a base line of Upenn'03 goalkeeper's performance. The logic of Upenn'03 [1] goalkeeping code was used to create a benchmark to compare the goalkeeping efficiency achieved by RL. The outcome of both the experiments were more or less similar. The only difference exists between the benchmark code and our experimental process is the decision making technique.

The key is to introduce a dynamic decision making process using a learning technique over Upenn's hand coded static decision making process. The RL algorithm started with a zero efficiency, gained experience over time and finally proven as efficient as the bench mark without human supervision. In contrast with that the benchmark program used input-output pair which was made using human intelligence. We had further extended our experiment using two attackers with the existing knowledge base from the RL experiment. Another benchmark experiment was created with the upenn'03 code using two attackers as well. The result showed that the knowledge base from goalkeeping training using RL, performed similarly to that of the benchmark hand coded technique.

Chapter 2

Literature review

2.1 A brief history of robotics

The word *Robot* was first introduced by Czech writer Capek in his play, *Rossum's Universal Robots* [5], in 1920. The term 'Robot' was derived from the Czech noun *Robota* which means forced labor. Indeed, the aim behind the invention of a robot was to make a machine which could replace human workers. In fact, different kinds of automatic machines were created in different centuries for similar purposes. Reportedly, the first mechanical robot was made by Al-Jazari in 1206. It was a simple mechanical boat that consisted of four automatic musicians that would play music using the ups and downs of the waves in the water. In 1495, Leonardo da Vinci designed a mechanical knight which was able to stand up, wave hands and make a few movements of its jaws and other facial parts [6]. One more early automation was created by Japanese craftsman Hisashiga Tanaka in 1738. It was a group of mechanical toys capable of serving tea, firing arrows and even painting some Kenji characters [15]. In 1898, Nikola Tesla publicly demonstrated a radio controlled robot similar to modern remote operated vehicles [16].

Elsi was the first modern autonomous electronics robot [17] capable of sensing light and reacting to it. The age of digitally operated, programmable and teachable machines started with the *Unimate*, a robotic arm made by George Devol in 1954. This was the first robot used in a metal plant for collecting and placing red hot metal pieces into dice. One more contemporary invention was the wall mounted *Tentacle arm* by Marvin Minsky

in 1968 which was able to lift the weight of a mature person. The design of the *Stanford arm* by Victor Scheinman in 1969 still influences some of the related technologies currently as well. The first ever computerized mobile robot *Shakey* was equipped with wheels and a television camera. This type of robot was usually huge in size and expensive which researchers could not afford. The first Legged robot was made at MIT in 1989 [18]. This was one of the early successful steps in creating small and cheap robots for research purposes. A number of robots are available currently for entertainment as well as research purposes. They are relatively cheap, small and equipped with different sensors. As for an example *Lego mindstorm* robots are widely used at high school educational level. Other models such as Pioneer, Hemission, AIBO, Roomba, Khepera are used for different purposes. Khepera, Pioneer, Hemission are wheeled robots with IR sensors and cameras attached onboard. Roomba is a small domestic vacuum cleaner. AIBO is a quadruped robot and an example of a sound improvement in making small entertainment robots. Altogether it can be concluded that modern robotics has developed a lot with the help of electronics and digital technology. A time line view of the history of robotics is shown in in the Figure 2.1.

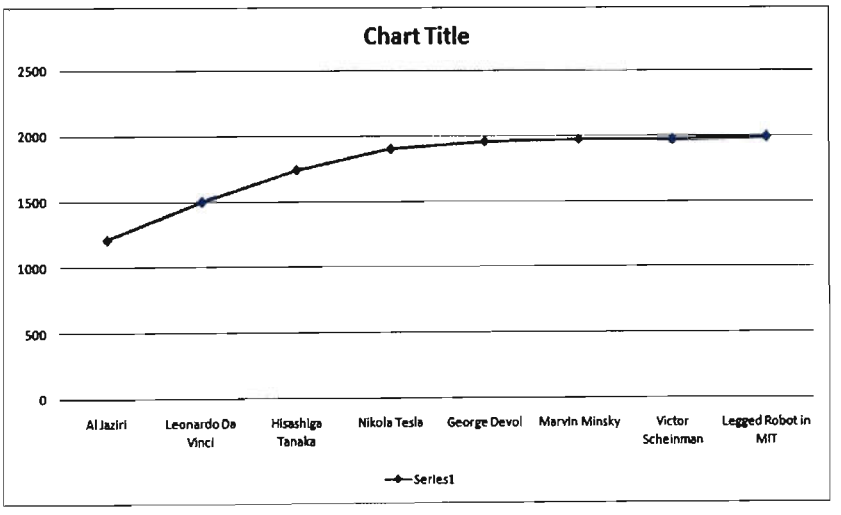

Figure 2.1: A brief history of Robotics

2.2 Overview of modern robotics

Modern robotics uses mechanical, electronics and software engineering combined to produce an intelligent piece of hardware, called a robot. The mechanical engineering deals with the design of body parts while electronics takes care of the sensory issues and the software relays between the sensors and the actuators. A legged robot has to perform different basic movements just to walk properly even on a plane. For the walking process, the center of gravity of the whole system starts shifting in such a manner that the torso must not fall down while moving. *Robotics Vision* is another prominent challenge. Pattern recognition, color detection and tracking more than one object are all part of it. These problems become more complex for a moving robot, especially for a legged robot due to its type of motion. Usually a legged robot leaned on left and right sides as well as back and forth at each and during each step for balancing issue. *Decision making* is another major problem in robotics. It is necessary for a robot to make decisions from its experience in a given situation. No human supervision should be used in this case after finishing the training exercises. This particular problem needs a lot of onboard computing resource and software expertise. As a whole, it can be concluded that a number of different technologies must be used at a time to build up a smart robot. Different researchers tried to solve these problems in different ways. The idea of 'robot playing soccer' was first mentioned as an international test bed for comparing those different approaches around the globe [7].

2.3 Introduction to Robosoccer

Robosoccer is an attempt to promote Artificial Intelligence (AI) and robotics research by providing a common task for the evaluation of various theories, algorithms and agent architectures [19]. "Robosoccer superseded chess as a challenging problem and benchmark for artificial intelligence research and robotics" [20]. The ultimate aim is to build a team of humanoid players by 2050 to defeat the human world cup champion team [8]. The robosoccer committee organizes its events based on two different goals. These are *Robosoccer* and *Robocup rescue League*. Robosoccer is a game of soccer between robot

teams in real life and in a simulated environment. Robocup rescue league consists of a search and rescue operation in an artificially created disaster area. The search is basically dedicated to look for signs of life.These two topics are divided into five main events to work out for robot teams around the world [21]. These events are discussed briefly in the next section.

2.4 Different types of Robosoccer

2.4.1 Robosoccer

- Simulation League

  Simulation league is arranged using a Unix server. Both 2D and 3D physical environments are provided for this virtual gaming event. Physical world sensory data such as vision, verbal communication and other position related data are supplied by the server to the software bots. In reply the software bot sends some signals to notify the server that it has made a physical movement such as run or kick or change in position and so on. The update is recorded on the server and used to define the world model. In such a match, up to 22 players can play in two teams against each other at a time. The soccer server is intended to provide a challenging environment for AI researchers, by allowing them to concentrate on designing only intelligence for the simulated bodies. Along with 2D and 3D league, a mixed reality or physical visualization league is also arranged.

- Small Robot League

  This small sized robosoccer event takes place between two teams of robots with five robots on either side. Each robot must fit within an 180mm diameter circle and must be no higher than $15cm$ unless they use on-board vision. The robots play soccer on a $4.9m$ by $3.4m$ green carpeted field with an orange golf ball. These robots come in two different models. One with local on-board vision sensors and other with global vision. Global vision robots use a centralized overhead camera and

a separate Personal Computer(PC) to identify and track the movements of other robots and the ball around the field. The overhead camera is attached to a camera bar located 4m above the playing surface. On the other hand, local vision robots have onboard sensors. The visual information is either processed within the robot or is transmitted back to the off-field PC for processing. The PC is used for different communication purposes such as sending referee commands, overhead vision, position information and so on. Typically, this PC also performs most, if not all, of the processing tasks required for coordination and control of the robots. Communication techniques are wireless and typically use dedicated commercial Frequency Modulator (FM) transceiver units [22].

- Four-Legged Robot League

  In 1998 Sony provided their first legged robot platform to three different research teams. These robots were a bit bigger than the small sized robots and consisted of an on-board computer. The most striking feature of this type of league is no off-field PC is used at all. Once the game starts, robots play on their own. Only built-in sensors are allowed in this case and robots are allowed to communicate to each other using verbal or wireless communication [23].

- Humanoid League

  The ultimate goal of robosoccer is to win against the football world cup champion team by 2050 [8]. Only humanoids are able to fulfil this goal. In year 2002, the Humanoid League started in robosoccer. The robots are divided into two size classes: kid-size ($30 - 60cm$ height) and teen-size ($80 - 130cm$ height). They are autonomous in nature and in addition to soccer games, penalty kick competitions and technical challenges take place as well. Walking, running, and kicking the ball while maintaining balance, visual perception of the ball, other players, and the field, self-localization, and team play are among the many research issues investigated in the Humanoid League [24].

2.4.2 Exhibitions

- RoboCup Commentator Exhibition

  RoboCup is not just for the teams who compete against each other in the leagues. In the year 1998 there was an exhibition of RoboCup-related technologies which are not directly related to competing teams. The *RoboCup Commentator Exhibition* demonstrates a number of systems which automatically generate soccer commentary for simulation league games. The commentator understands and analyzes the performance of each player in a game, creates hypotheses on interesting topics to provide comments on and generate fluent commentary in different languages. The applications of such technology are enormous and require a lot of attention [25].

2.4.3 Robocup Rescue League

- Rescue Simulation League

  The Rescue Simulation League is a relatively new concept in Robocup compared to that of soccer. It was first introduced in the year 2005 [26].The purpose is to provide emergency decision making support by the integration of disaster information, prediction, planning, and human interface. A generic urban disaster simulation environment is constructed on network computers. Heterogeneous agents such as fire fighters, commanders, victims, volunteers, etc. conduct search and rescue activities in that virtual disaster world. Real-world interfaces, such as a helicopter image, synchronize the virtuality and the reality by sensing data.

  This problem involves advanced and interdisciplinary research themes. The behavioral strategy consists of multi-agent planning, realtime/anytime planning, heterogeneity of agents, robust planning, mixed-initiative planning and so on. The RoboCup Rescue simulation league works as a standard platform to develop practical comprehensive simulators adding necessary disaster modules to disaster rescue researchers.

- Rescue Robot League This event is a real life version of the rescue simulation league. However, the scope of the problem here is much less due to the lack of

maneuverability skills and other technical problems in practical robotics. In this event, robots look for signs of life such as a waving hand in a disaster site. Team work and other skills are also being tested here.

2.4.4 Robosoccer@Home (since 2006)

The @Home league consists of an open challenge in the form of few tests to demonstrate the abilities of participant's robot. The robot should perform at least one test. The test should maintain the following cases:

- include human machine interaction
- be socially relevant
- be application directed/oriented
- be scientifically challenging
- be easy to set up and low in costs
- be simple and have self-explaining rules
- take a small amount of time [26]

2.4.5 Robosoccer Junior

This is an event organized for under graduates and school level students. The rules are more straight forward and relatively simple robots are taking part in this event. The challenges available in this case are as follows:

- Soccer Challenge
- Dance Challenge
- Rescue Challenge

2.4.6 A particular task of Robosoccer, used in this thesis

The research topic in this thesis is related to *Four-legged* robot league stated under Robosoccer events. A few real life situations of a soccer environment are imitated in there and only *AIBO* robots are allowed to take part. Both individual and team-work skills are required to play these soccer games. Basically, in this event, robots try to perform few tasks such as shooting a goal, defending their own territory and so on. The basic skills are maneuverability, shooting in a particular direction, blocking a ball, sharing information, visualizing the environment properly, team coordination, strategy acquisition and others. Some of these techniques could be used with little modification to make a robot team work in different situations efficiently. This research examines an approach to making a robot capable of acting appropriately in an unknown/complex situation. In this study the AIBO is programmed to respond to an unknown situation in the position of goalkeeper. It has been trained at the beginning and then saved the goal, both without human supervision. The description of the AIBO robot is available in next section.

2.5 The AIBO

AIBO means *companion* in the Japanese language. It is said that dogs are the best friend of human beings among other animals in our history. So, engineers at Sony Co. Ltd. decided to make a replica of dog as a robot. The initial structure of the AIBO was designed by the famous Japanese artist, Hajime Sorayama [11].

As described previously, different kinds of robots take part in Robosoccer. They are divided into two main categories depending upon their mobility mechanism, namely the wheeled robot and the legged robot. Wheeled robots are much faster and also have a relatively more stable vision while in motion to that of legged robots. This is due to the fact that a wheeled robot goes smoothly on a plain surface using wheels, whereas the moving technique of a legged robot is quite different. On the other hand, a legged robot can move through uneven surfaces which is completely impossible for a wheeled robot.

A quadrupedal robot like the AIBO, first adjusts its torso in order to put the center of gravity on any three legs. Then it lifts the free leg(which is not in use for balancing,

temporarily), puts it ahead and repeat this cycle for each of the other legs to move. The torso tilts on different angles during this whole process and the camera captures different frames of a particular object from different angles. So, the system receives multiple images of a single object. This brings a shaking effect (this is sometimes mentioned as "handshaking" effect as well) to the captured picture and makes it worse for further processing. Wheeled robots are largely free from this problem. However this problem in legged robots could be solved by different image processing techniques. But, wheels can not navigate through certain surfaces such as rocks, various gradients, staircases, places filled with scattered objects, forest, shallow water logged areas and so on. Only legged robots are free from those problems. Sony released a bi-pedal robot QRIO, which revealed the secret of jumping motion in bipedal movement. It can take both of its feet fully off the ground and regain balance after touching down. This facility allows it to run efficiently like a bipedal animal. Another bi-pedal robot, ASIMO, is capable of serving food, opening combination locks, equipped with voice and face detection technology and so on.

Sony started marketing the AIBO in the year 1999 for entertainment purpose. It was chosen for our experiment due to its portability, computing power, availability of different sensors and comparatively cheap price to other robots. Each and every model of AIBO is able to play soccer under Four-Legged robot league in Robosoccer. Three ERS-7 models were used for the experiments in this study. The ERS-7 was the latest model, released by Sony before they ceased production of their product.

This model resembles a dog and Sony programmed some spontaneous dog-like AI behavior to make it act like a real life animal. It has face detection, voice recognition, color detection (magenta by default) features and many more. It is capable of tuning its behavior depending upon the owner's mood. It detects harsh and polite tones of voice and acts accordingly. It has an in built wireless local area network card (IEEE 802.11b standard). One can set up an Simple Network Management Protocol (SNMP) server inside the AIBO and configure it to send the pictures taken by the nose camera over the internet using a workstation as a gateway. In this way it can be used as a moving in-house security device. The AIBO has a striking feature in that it can find its charging station while having a low battery signal. It uses complex pattern recognition technique for this purpose and locates a particular image printed in the tower of the charging station. This

feature makes it a perfect watch dog.

The different components of the AIBO are listed in Table 2.1.

Table 2.1: Specification

CPU	64-bit RISK Processor
CPU Clock Speed	576MHz
RAM	64 MB
Programming media	Memory $Stick^{TM}$
Operating Temperature	10^oC to 60^oC
Operating humidity	$10 - 80\%$
Built in sensors	Temperature Sensor, IR distance sensor, Acceleration sensor, Touch sensor, vibration sensor, Pressure sensor
Movable parts	Three parts in head module, Three parts in each of the four legs
Power Consumption	Approximately 9W (Standard operation in autonomous mode)
Operating time	1.5 Hour (In standard operation)
LCD Display on charger	Time date, volume, Battery condition
Operating system	Aperius
Weight	Approximately 1.6 Kg including battery and Memory stick
Dimension	180mm high and 18mm in diameter.

There are several different sensors available in the AIBO. One of those is a color camera situated at its nose with maximum $208/160$ pixels. It is capable of capturing 25 frames per second at the highest resolution. Four pressure sensors at the four paws are available to determine whether the robot has toppled or not. These four sensors are binary in nature and quite noisy too. These sensors sometimes do not fire even when the full body weight is resting upon them. Three touch sensor plates are available on the back, one at

the chin and one on the upper portion of the head. These plates are extremely sensitive and give the dog some realistic features. For example, the dog makes a happy sound if someone touches the chin sensor. The dog has three infrared distance sensors. Two of them are situated over the nose camera and one at the chest. The chest sensor is used to detect an edge or other close objects which can not be seen through the nose camera due to limited movements of head joints. Two distance sensors at the nose are regarded as *Near distance sensor* and *Rear distance sensor*. The working range of the near sensor is from 5.7cm to 50cm and the other one works between 20cm to 150cm. The last and most advanced sensor of the robot is the in built accelerometer. It is the only sensor in AIBO that is able to sense three dimensional readings. Its accelerometer can measure acceleration along the x axis, y axis and z axis. The velocity and displacement of the robot can be calculated using time difference and the data from this sensor. There are also two microphones fitted on top of the ears. These are capable of capturing stereophonic sound. They enable the AIBO to determine the direction of incoming sound. The AI program made by Sony allows users to register a name for a particular model, when someone calls the name AIBO can point its head towards that direction.

There are three stepper motors fitted at each of the four legs, two at the head joints and two at the tail. A stepper motor is a device which converts electrical pulses into discrete rotational motion; it creates a precision control over the motor. These motors are also fitted with safety devices in AIBO. This device shuts down the whole system whenever a motor is stuck at any point during operation. This feature is built to prevent any mechanical damage to the joints.

The AIBO has similar computational power to that of an early age Pentium III personal computer (PC) which was quite impressive at the time of release. However, the last upgraded model accepts at most 128MB memory stick made by Sony itself. This much memory is enough for programming the robot with existing languages such as C++, Java and Matlab, but it is not enough to record sensory data over a long period. Also, its infrared distance sensors are not good enough to measure the distance of any particular object. This is due to the reflection from adjacent materials around the target. So, a different method was developed to calculate the distance of the ball using the nose camera, which is described later on in this thesis.

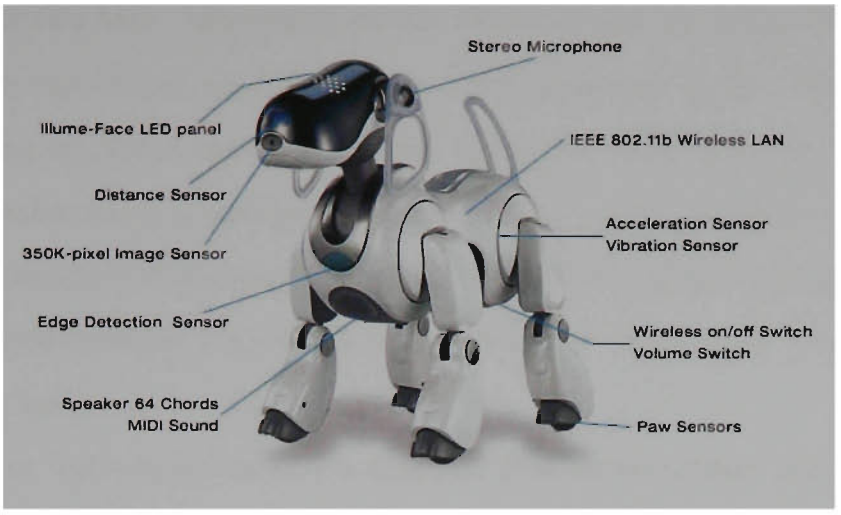

Figure 2.2: Sensors in ERS-7 [2]

In the year 2006 Sony announced it would stop producing and marketing AIBO. But, the last model ERS-7, which was used for the experiments, would be supported up to 2013 by the company. The next section describes reinforcement learning theory in general. A part of this RL theory is used as the programming logic in this thesis.

2.6 Reinforcement learning

2.6.1 Introduction

The theory of *reinforcement* initially started in the field of psychology a few decades ago. Perhaps the first signature of this idea was found in the words of Thorndike in expressing the trial-and-error learning. In his words "Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will, other things being equal, be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur; those which are accompanied or closely followed by discomfort to the animal will, other things being equal, have their connections with that situation weakened, so that, when it recurs, they will be less likely to occur. The greater the satisfaction or discomfort, the greater the strengthening or weakening of the bond" [27].

This theory is called the law-of-effect by Thorndike. There are two important things

associated with this idea: responding to the situation and repeating the similar actions associated with the reward when facing the same situations again. The law-of-effect is used in selecting an action over others while facing the same situation. Only the action that brings satisfaction in a particular situation is likely to occur again over others. This *satisfaction* is categorized as reward whilst the *discomfort* is denoted as punishment which weakens the bond between a particular situation and an action [27]. This idea is defined more generally below.

Reinforcement learning is basically a dynamic situation-to-action map which helps the actor to gain comfort or reward through the process. The word *dynamic* implies that the agent dynamically chooses the correct action for the current situation. An agent is a decision maker within a learning system and anything except the agent is denoted as the environment. The agent interacts with the environment, observing a situation which is called *state* in this regard. The agent responds with an action to the environment and receives a reward in terms of numerical value. It is the goal of the agent to maximize the reward over time. An agent carries out a particular task using reinforcement learning. Figure 2.3 below describes a hypothetical view of a system engaged in reinforcement learning.

2.7 A basic model of Reinforcement learning

In Figure 2.3 the agent finds itself in a state $s_t \epsilon \mathbf{S}$ at any discrete time t (t=1,2,3,4.....) in an environment where **S** is the set of all possible states. An action a_t is selected where $a_t \epsilon \mathbf{A}(s_t)$ and $\mathbf{A}(s_t)$ is the set of actions available in s_t. As a consequence of the action, the agent receives a reward $r_{t+1} \epsilon \mathbf{R}$ and finds itself in a new state s_{t+1}.

The following steps could be used to summarize reinforcement learning:

1. An input state is observed
2. An action is selected in response
3. The action is performed
4. An input state is observed

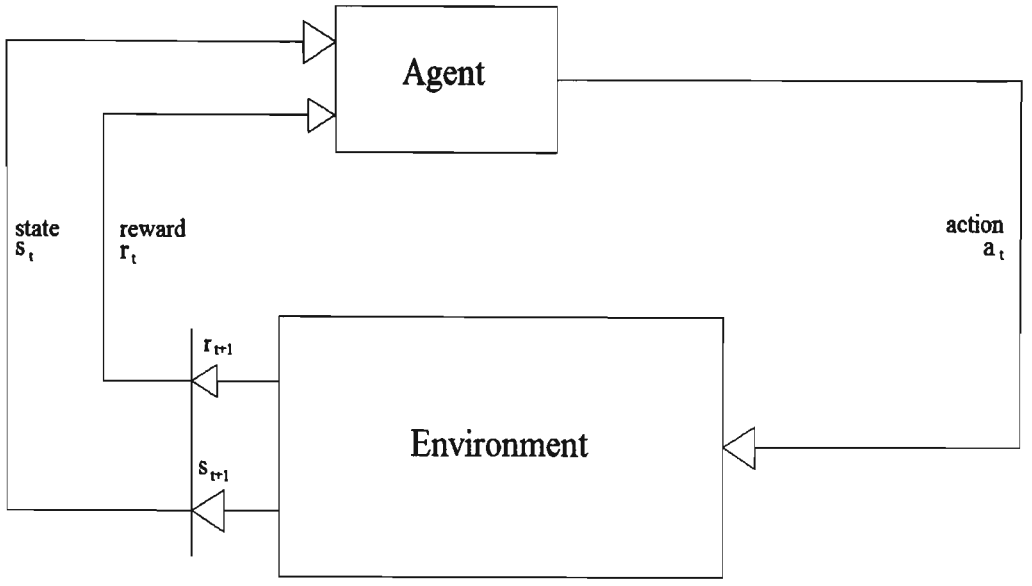

Figure 2.3: A general reinforcement learning system

5. A scalar reward or *reinforcement* is given

6. The reward for the state is recorded

The goal of the agent is to maximize the reward point through reinforcement learning. Finally, the agent follows the action associated with the highest reward point in each state and performs the given task at minimum effort. So, a *path* is ultimately defined to accomplish the task. A policy in reinforcement learning is how to define the actions to be chosen to meet the goal. The reinforcement learning techniques discussed in this paper are always following policies in one way or other. There are two basic techniques used in reinforcement learning along with other forms, namely Q-learning and SARSA. The following definitions explain the basic terminologies related to reinforcement learning, used in this thesis.

2.7.1 Exploration vs exploitation problem

One of the main trade offs in reinforcement learning is the exploration and exploitation dilemma. An agent has no choice other than taking random actions when it starts learning

with a zero value table. However, once it receives a reward, it starts acquiring an action sequence to reach the goal from the starting position. Now the question is whether the agent should follow this sequence repeatedly or whether it should try out some new actions. This dilemma between exploitation of the acquired knowledge and exploring some other actions is one of the main issues in reinforcement learning. Usually the agent needs to explore different actions to build up a policy whereas the policy has to be followed to finalize the table. One may try to keep the balance between exploration and exploitation and create a 50-50 ratio for reinforcement learning training. Some well known policies are described in the following part of this section.

2.7.2 On-policy Learning

It is said to be on-policy learning if an agent starts learning with a particular policy and finds out the state action values within the scope of that policy. SARSA [28] is an on-policy learning approach. Soft policies are used to ensure that the agent has enough room to explore. In other words, the policies are not so strict so as to simply only follow the successful moves. Three such policies are ε-greedy and softmax.

2.7.3 Off-policy Learning

In contrast with on policy learning, the agent starts updating the table with a strict greedy policy in the case of off-policy learning, but it is free to take any action without using any policies to make sure that enough opportunity is given for exploration. Finally, it learns a separate policy from the values updated in the table. So, this indicates that off-policy learning has the capability of working with a random action selection policy at the initial stage and finding out a new policy at the end. Because of this feature, off-policy learning is chosen to tackle our research question. Q-learning [29] does not use a policy to update the state x action table and so it is called off-policy learning

2.7.4 Different Action Selection Policies

A policy is a mapping from state to action. All RL methods learn and use policies during training and working period after completion. A soft-policy is a method which

gives a very small number of suggestions to the agent to choose between exploration and exploitation. The agent works throughout the training period with a random action selection process. Even after obtaining a reward it follows the successful moves with the same priority compared to the other less prioritized ways. On-policy learning only uses one of the available policies from the beginning. Some of these policies are described below.

2.7.4.1 ε-greedy

This policy is named *greedy* [29], which means the highest-reward action or the *greediest* is chosen for most of the time. Other actions are chosen only with a small probability ε and all the actions are given the same priority. This policy ends up with a converged state x action table, if enough trials are done. That is as the number of trials becomes very large the state x action Table 2.7.5 no longer changes.

2.7.4.2 Softmax

According to ε-greedy and ε-soft [29] methods all the actions are chosen uniformly except for the one associated with the highest-reward. A worse action is chosen with the same priority as the second-best action. The softmax remedies this by assigning a rank or weight to each of the actions according to their value-function estimate. These ranks are used as probabilities to choose an option. This approach is perfect where the worst possible action is not at all favorable.

2.7.5 Value function

A Value function is a state action pair function that estimates the return to any state after taking a particular action (A). This function is used to update the cell values mentioned in the previous definition. Once an action is taken, the agent finds itself in a new state, the value function calculates the cell value corresponding to that state and the action using the available reward.

$V^{\Pi}(S, A) \longrightarrow$ The value of a state S under policy Π. The expected return when starting in S and following Π thereafter.

2.7.6 State x action Table

State x action table is used as the data base of a reinforcement learning agent. The learning starts with a matrix with all the elements set to zero for most of the cases which is called the state x action table. However, occasionally the table may contain some numbers to make the learning faster. The Convergence of state x action table values indicates the end of the learning sequence. A state action table should resemble the example described in Table 2.2.

Table 2.2: State x Action table with no initial knowledge

	State 1	**State 2**	**State 3**
Action 1	0	0	0
Action 2	0	0	0
Action 3	0	0	0

The states and actions are different for different experiments. These kinds of tables are in use in this thesis. Zero values indicate that no experience is available so far. However, in some instances, the programmer may like to provide only a small amount of prior knowledge to the agent. In that case some of the cells of the state x action table will contain a non-zero number. Value function 2.7.5 formulae are used to update the cell values while learning.

Sometimes, the size of state x action tables is too large for the agent to try and reach the goal within a reasonable amount of time. In those cases the agent might look for a reward indefinitely and so the learning will not be effective. So, in those cases a few numerical values are supplied to make sure that the agent must has some prior knowledge to start with. In the long run, this technique yields a time efficient solution for a learning process with a large sized state x action table. This may look like the Table 2.3.

Only those state x action tables having all zero values are used in this thesis.

- Table 2.3: State x Action table with few existing knowledge

	State 1	State 2	State 3
Action 1	60	0	0
Action 2	50	0	36
Action 3	0	0	74

2.7.7 Temporal Difference

Temporal Difference (TD) Learning methods can be used to estimate these value functions. If the value functions were to be calculated without estimation, the agent would need to wait until the final reward was received before any state x action pair values could be updated. Once the final reward was received, the path taken to reach the final state would need to be traced back and each value updated accordingly [30].The formula stated below shows the mathematical form.

$$V(S_t) \longleftarrow V(S_t) + \alpha * [R_{final} - V(S_t)] \tag{2.1}$$

S_t = State visited at time t

R_{final} = Reward, received at the end

α = constant parameter

On the other hand, with TD methods, an estimate of the final reward is calculated at each state and the state x action value updated at every step of the way. Expressed formally:

$$V(S_t) \longleftarrow V(S_t) + \alpha * [R_{t+1} + \gamma * V(S_{t+1}) - V(S_t)] \tag{2.2}$$

R_{t+1} = Reward at time t+1

γ = Discount factor

2.7.7.1 SARSA

SARSA [28] is an on-policy learning technique and it updates the state x action table in reinforcement learning process. The algorithm is shown in Algorithm 1

Algorithm 1 SARSA

Initialize Q(S,A) Arbitrarily

repeat

Initialize S

choose A from $A(S_t)$ using selected policy

repeat

take action A, observe S and A'

choose A' from S' with selected policy

update table using V(S,A) $\longleftarrow$ V(S,A) + α[R + γV(S',A') - V(S,A)]

S $\longleftarrow$ S'

A $\longleftarrow$ A'

until terminal S reached

until Q-values have convergeds

RETURN V(S,A)

end

α is denoted as the learning rate. The value stays between 0 and 1. The learning does not take place while the value is zero; This is because of the fact that the table is never updated. Learning is the slowest at $\alpha = 0.1$ and quickest at $\alpha = 0.9$. It is preferable to keep the value of α at 1 in a noiseless environment.

γ is denoted as the discount factor. It also takes a value between 0 to 1.

The name SARSA is derived from the sequence S,A,R,S',A'. This sequence indicates that the agent starts with an action A at situation S, observes reward R, takes another action A' and finds itself in situation S'. Here the observed reward is based on the next action taken by the agent. This action is dependent on the policy which is chosen at the beginning of the experiment. Finally, the agent develops a policy based on the reward system and so restricts itself within the scope of the chosen policy.

The term V(S,A) is similar to Q(S,A) which is used in Q-learning. Actually, it is used to update the cell values in almost the same way except that the maximum value of the available actions from the next state is chosen. A detailed theory of Q-learning is as follows.

2.7.8 Q-Learning

In case of Q-learning [29] the state x action table converges with 100% probability to a close approximation of the value function provided enough training is given for any arbitrary target within a *Markov Decision process*. The step-wise procedural approach is displayed in Algorithm 2. Q learning differs from SARSA only at the point of selecting the

Algorithm 2 Q-Learning

Initialize Q(S,A) Arbitrarily

repeat

Initialize S

choose A from $A(S_t)$ using random/selected policy

for each step **do**

take action A, observe S and A'

choose A' from S' with random policy

update table using Q(S,A) $\longleftarrow$ Q(S,A) + α[R + γ max_{A^*} Q(S',A^*) - Q(S,A)]s

S $\longleftarrow$ S'

A $\longleftarrow$ A'

A^* ϵ All A

end for

until terminal S reached

RETURN V(S,A)

end

reward at a state-transition point. Although any action could be chosen using Q-learning, only the maximum reward is considered to update the state x action table; it guarantees the convergence of the table in most cases, whereas in SARSA, only the reward associated with the action taken is considered for calculation. Moreover, the policy tries to keep a balance between exploration and exploitation as well. It was found during a maze learning experiment [31] that SARSA sometimes ends up without convergence of the state x action table and the ultimate policy can not be revealed. This issue is addressed thoroughly in the methodology chapter with a maze-learning example.

2.8 Real life Examples

The following examples define a few real life problems where reinforcement learning can be used or is already in use [32].

2.8.1 Firing a machine gun at a moving target

Basically four factors simultaneously control the path of a moving bullet. These are gravity, air velocity, the viscosity of the air and the speed of the bullet itself. So, the resultant of these forces would determine the velocity of a bullet in a real life scenario. It becomes challenging when the target is moving fast such as a fighter plane, high speed vehicle and so on. In that case the resultant of the initial velocity of the bullet and the velocity of the moving body should intercept the target within a suitable range. So, all of these factors are involved in the successful firing from an anti-aircraft machine gun. A human operator takes many years to develop this skill. The reinforcement learning, in this case, starts with the trial and error method and gradually a policy is developed. It can be concluded here that this is a tune up process of different parameters to accomplish the task. Reinforcement learning could be used here better than any deterministic formula to do the same. In this case the learning will be based on two particular points. The flight pattern of the enemy aeroplane and the average velocity of air. The word *average* is used here as the velocity always changes with time. The agent finally should be able to develop a policy to fire a bullet with such velocity which will maximize the chance of intercepting a moving object in a 3-Dimensioned environment like an aeroplane.

2.8.2 Quadrupedal locomotion technique acquisition of a new borne gazelle calf

As Sutton said in his book [29], quadrupedal locomotion is one of the most complex locomotion techniques among animals. The parameters involved here are the center of gravity (CG) of the torso and the speed of the moving body. Whenever a quadrupedal animal walks, the CG transfers on top of three legs while the other leg is lifted and placed in a different position. While it runs, the CG rests upon two legs only and the torso works

in such a way that it goes forward instead of falling down. Both of these processes needs complex synchronization between movement of body and different postures of leg. A calf learns this critical process within half an hour after its birth, using the intuition and trial and error method. It should be mentioned here that the locomotion of the AIBO was also tuned successfully using reinforcement learning [33]. The hand tuned parameters were successfully discovering out using reinforcement learning [34].

2.8.3 Acquiring cycling technique for both human and machine

Preben and Jatte [35]applied reinforcement learning in a bicycle riding problem. The main problem in riding a bicycle is to properly maintain balance while riding. SARSA [28] algorithm was used to accomplish the learning process. This problem was extended to find the goal after the agent learned how to drive. So, the paper described and solved two problems using SARSA. The first problem is that of learning how to ride a bicycle after extensive training and the other is finding an optimal path to the final destination. Both were solved successfully by reinforcement learning.

2.8.4 Operation of an autonomous mobile cleaner robot

According to Sutton [29], as mobile robot cleaner should decide whether it enters a room for collecting trash or goes back to its battery charger. The decision is influenced by two different parameters. These are the expected time to clean up the trash and the time to reach the charger and accordingly the robot receives a reward or punishment. So, finally the cleaner learns to clean a place and charge itself in a balanced way. The previous knowledge of the house map could be used for defining the way back to the charger and the amount of the area to be cleaned [29].

These problems could be addressed with reinforcement learning in a better way than with the supervised learning process or hand coding. Usually training data with input output pairs are used with supervised learning. So, specific information is needed for each and every situation in the case of supervised learning. However, it is impossible to provide such information due to the vastness of the problem dimensions discussed above. The anti-aircraft problem has basically four different parameters involved and each of those

could take several continuous values. So a large number of training data is required to cover all possible situations. The second problem consists of the different positions of the four legs and the position of the torso. Altogether these parameters create a significant problem space to solve. The third problem is the leaning angle and the speed of the bicycle as input parameters. Again, together there could be several combinations between them. The fourth example has the two different problems. The first is getting used to the interior of a particular house, the second is finding a balanced way to capture the garbage and find the charger in time.

A handful of training data is not sufficient for any of the problems discussed above. Either they need the data from each and every segment of the environment while performing or a large amount of data needed to be stored well in advance. This indicates that supervised learning is not the answer to these problems. A more generalized solution is required and the agent should acquire experience rather than relying on a database. Reinforcement learning suits these points and has already proved worthy in some cases as cited with the examples.

2.9 Robosoccer using the AIBO

The ultimate goal of the Robosoccer event is to create a fully autonomous robot team which can defeat the human world-cup champion team [8]. In order to play soccer as efficiently as humans, a robot has to perform some basic maneuverability skills. Physically, humans are still much more capable than a robot. Moreover, a robot has to think like a human to take quick decisions on the field when playing soccer. But, until now, no machine on earth has the capability of taking independent decisions like a human. Robocup is an event to promote research all over the world towards making a machine as efficient as humans.

The first Robocup world cup took place in Paris in 1997 [36].The research platform, the AIBO, was released in July, 1998, and it also took part in Robocup in the same year in an exhibition match [37]. It was played at RoboCup98 in Paris. Three teams from Osaka University Baby Tigers (Japan), CarnegieMellon University CMTrio98(USA) and Laboratoire de Robotique de PARIS (LRP) Les 3 Mousquetaires (France) took part there

and used AIBO robots for the first time. In 1999, six more teams competed in the four legged autonomous robot league in a three-on-three match. Since any change of hardware was not allowed, only the software program could be developed through it. However it was discovered that the prototype had some serious hardware limitations.

Five issues were pointed out in the 1999 Robocup [37] as the key problems to Robocup using quadruped medium sized legged robots. These are:

- Vision
- Navigation
- Playing skill
- Localization
- Team work

Different universities worked on these key areas and tested their skill against other teams in the Robosoccer championship. A brief description of the winning team in some of the Robosoccer competitions is given below. There some of the rapid development of robosoccer are outlined to give an appreciation of the breadth and complexity of issues involved.

2.9.1 Robosoccer 1999

It is obvious that a robot easily loses sight of the ball due to the limited visual angle of the Charged Couple device (CCD) camera attached to the nose. This problem was considered to be a major one in Robosoccer 1999. It was decided that the key to winning the tournament was to make an efficient object recognition program to minimize the ball finding time. The challenge was there to make an optimized object recognition program to bridge between improved vision and available hardware resources.

In the 1999 Robocup most of the teams used the walking programs provided by Sony due to the limited availability to develop their own walking program. The preference was given to the tactics. However, the LRP [37], which developed a stable and robust walking program, won the league. This stable walking indeed helped improve the vision of the ball

while player and ball were both moving [38]. The team of Osaka university [37] developed a trot walking to achieve faster speed. Another most important issue in Robocup is localization. Due to slippage of Sony's walking technique, it was impossible to localize the AIBO properly. So, different universities took a different approach to overcome this problem. The Carnegie Melon university used probabilistic sampling to minimize errors caused by movement and unexpected errors. Multi-fidelity behavior was also introduced to gracefully degrade or upgrade it with a different localization model. Osaka university team used a landmark system for the purpose. It should be mentioned here that in the year 1999 no color pole was placed by the field side as a landmark. So, LRP, the winning team, used the goal post for the task accomplishment and achieved a relatively fast localization process. Finally, the playing tactics were more important than the others stated above. Unlike other problems, the LRP has its uniqueness hidden in the decision making activities. In an other way, it uses all four key skills as the basic action, but these actions have to be processed against a situation in order to take right action at right moment. In the year 1999, most of the team assigned one robot as the goalkeeper and two robots as players except Osaka team. All three robots in their team performed both defensive and offensive roles [37].

2.9.2 Robosoccer 2000

Some of the rules of the game in the year 2000 were changed compared to the last time. In particular a few beacons were added to the field to support localization more precisely. The overall significant improvement in this year was in ball controlling technique [39]. University of Osaka introduced head kick, which was effective as a long range shot [37]. Almost all contestants adopted this technique in the next year league. UNSW introduced an effective ball controlling technique in this season. Using it AIBO captured the ball within two front legs, moved to the desired direction and pushed it. It was proven to be an accurate and high powered kick for the competition. The winning team of UNSW year in 2000 [39] focussed on color detection. They have used a polygon growing model for the problem. AIBO has an in-built color detection mechanism. However, the same color could be considered as different colors in different brightness conditions. So, a code was developed to perform an off-line learning from different images taken from different

parts of the field. The image is not at all stable while the robot is on the move. So, a modified locomotion technique was developed for this reason [38]. The paws moved in a rectangular fashion to minimize the vertical and horizontal instability [39]. There, the behavioural strategy of goalkeeper was described by three high level strategies as found in UNSW golie code. These are as follows:

- Finding the ball
- Tracking the ball and acquiring a defensive position
- Clearing the ball

2.9.3 Robosoccer 2001

In 2001 UNSW defended their previous year's title. In this year many teams made significant changes in basic functions based on ERS-210 prototype of AIBO. its leg motors were stronger than previous models and onboard processor MIPS-4000 was faster too. A bunch of new physical maneuvers and skills could be realized using this model. The new model had a different body geometry over previous ERS-111 prototypes. So, a number of skills were obsolete too. But the new model was much better compared to the previous one due to the upgraded hardware. Also, due to faster CPU speed, a code was written to process high resolution images during game play [40]. A new keeper charge rule was introduced into this year's competition which gave the goalkeeper an advantage. The rule said an attacker would be removed for 30 seconds if it touched the ball while the goalkeeper had its grip on it. The goalkeeper was programmed to hold the ball between its front legs and turn forward within five seconds. But in the actual game it did not work, because the goalkeeper was never charged while holding the ball. A locking up problem at goal post corner was also noticed during weekly practice match [41], too. This problem seems to be related to real life oriented issue. In any automated process, a robot may stick to a corner if propels towards it at high speed. A simple solution was provided for this problem to the robots. It defined that if the distance between the ball and robot was not changing over a particular threshold then possibly the ball was hooked up at a corner. A separate maneuver was written to control the situation.

2.9.4 Robosoccer 2002

Carnegie Melon University won the title of legged Robosoccer in the year 2002 [42].This time they focused on single and multi robot control and multi robot team work [43]. The introduction of new robots and the increment of field size made it difficult for the programmer to take care of the role assignment issue. However, the wireless communication was used to coordinate movements and assign roles between available robots [44]. These roles were a primary attacker, which approaches the ball and attempts to move it up field; an offensive supporter, which moves up the field from the primary attacker and positions itself to recover the ball if the primary attacker misses its shot on goal; defensive supporter, which takes care of the ball if the opponent team approaches the goal with the ball and the goalkeeper takes care of the goal area. Three players always negotiate between each other and switch their roles accordingly. Also, they always keep in touch with the goalkeeper to avoid blocking or approaching the ball while the goalkeeper tries to clear it from the defense zone [45].

2.9.5 Robosoccer 2003

rUNSWIFT, the team from UNSW won the title again after losing it in 2002 to Carnegie Melon university of USA. They already had a large code base due to the previous years' experience. So, the code was divided into some Aperius [13] modules. Distributed sensor fusion using wireless communication was improved. Each robot team consisted of four players instead of three. It was observed in the year 2002 that uses of the fourth robot introduced more complexity into the game. So, this year UNSW team focused on the robustness of the gaming environment. In other words, their approach was to enable a robot to consider the complete environment. The overall efficiency of this approach was better in terms of managing the complexity than the previous years. But, this approach lacks in taking the right decision at any moment. So, the players were not able to take the optimum action although it was able to define the environment [46]. Theoretically, a learning system may solve the problem [29].

2.9.6 Robosoccer 2004

The German national RoboCup team won the middle size official robocup title in the year 2004. Four universities across the country took part in making the winning team that year. They had used *Agent Behavior Specification Language(XABSL)* [47], [48] for behavior control issues. This is an XML based programming language to take care of behavior in autonomous agents. According to the mentioned hierarchy, the German team had four main stages in their software coding, namely perception, object modeling, behavior control and motion control. In the perception stage, it collects data from sensors which leads to calculating world model in the object modeling section. This world model actually helps the agent to take decisions in the behavior control stage and provides input to the motion control state. In practice, less work was being done to improve high level skills in comparison to ball handling skills such as dribbling, kick selection, and navigation. But, they had introduced Dynamic Team Statistics(DTT) using wireless channel which was an extension of the XABSL behavior by a meta-layer that helps to represent the dynamics of the game and environment. Practically, agents shared world model information using DTT and had chosen the most appropriate job for it leaving other options for teammates. This approach lead to the problem of evaluating the position and prospect of success for every robot and each task. A hand coded solution was proposed for the purpose [49].

2.9.7 Robosoccer 2005

In the year 2005 Carnegie Melon University regained their honor after 2002 in the middle size legged robot championship. This time the software they developed was fully designed over the knowledge of previous teams from the university. Its world model acquisition process was equipped with sensor fusion technology. Each and every robot on the field tried to get the information of a single object from different angles. This information is unequal due to difference in offset values introduced by different sensors. Moreover when a team of mobile robots are trying to define the environment, a large amount of noise interferes with the actual data and makes the model extremely complex if not impossible to define. A method for reasoning over a discontinuous hypothesis space is used to solve the problem where the sources of information was used with strict ordering.

A prioritized hierarchy of state estimate is made by segmentation of information sources into different classes. This hierarchy is used in deploying the decision process that governs each individual robot's actions. It can easily select the most informative state estimate to use as its input. An efficient reasoning about the expected utility of certain classes of estimates over others can help the robot to select the best estimate from the set to act upon. The updating process of ordered hierarchy of possible estimates could benefited from that reasoning. This system is a prioritized state estimation technique. It could be applied to a real-time adversarial multi-agent domain. However the selection of priority used in this system is assigned statically by programmer. As a result, any undefined change in an environment may result in a malfunction or system crash [50].

Finally Figure 2.4 reveals a graphical view of the winners stated in last few paragraphs.

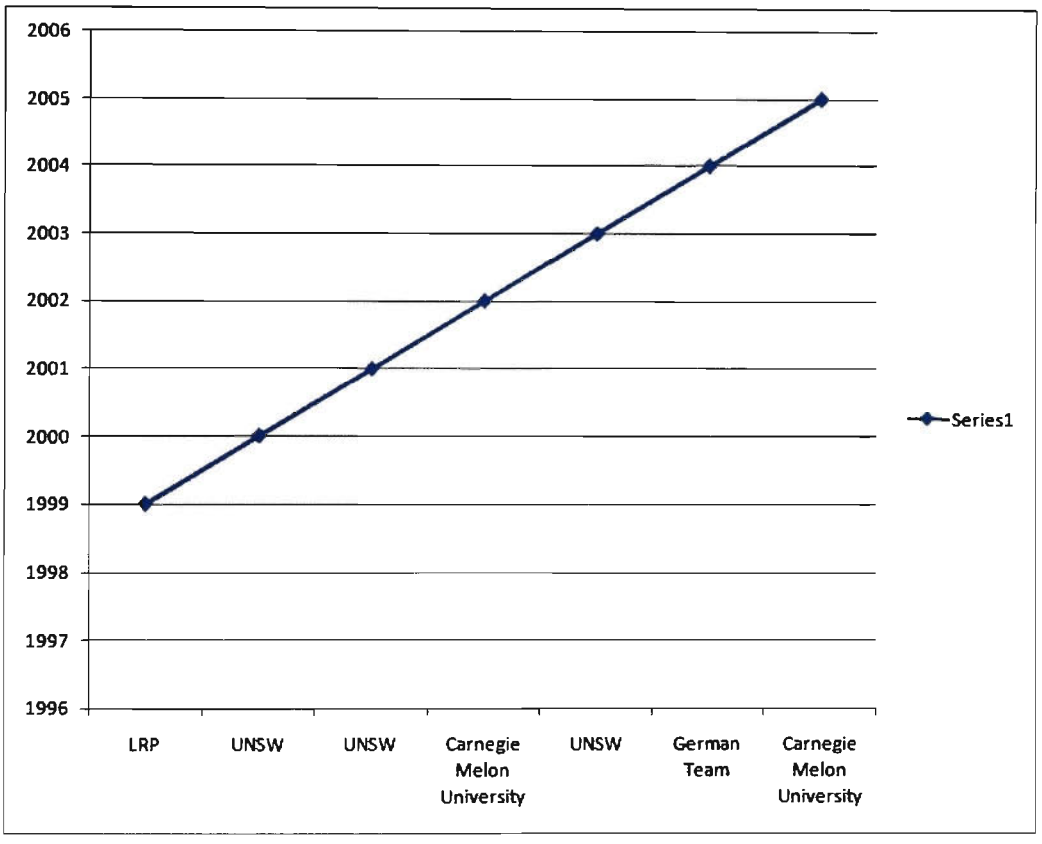

Figure 2.4: A time line displays the winning teacm

2.9.8 Summary

So from the above discussion, it could be concluded that various teams are designed from different points of view to play efficiently in Robosoccer using AIBOs. Physical maneuvers

could not be improved over a certain limit due to the limitation of hardware resources. Also, the robocup authority provides a partially dynamic environment for its participants. This is to make an agent encounter with different hidden states during game play. It is to make sure that the software may not only work for one particular purpose but also in other dynamic environments with similar ideology. World model prediction is the way of sensing the environment and already some efficient software techniques are devised by CMU'05 team based on prioritized hierarchy [50]. However, it is not defined how to take an optimized decision at any moment. In soccer, a number of different situations may come up. Hand coded techniques are not at all a permanent solution to encounter these large numbers of problems. It may be effective for an environment with limited dynamic nature, but it will definitely get out of control with the increment of complexity. So, a different technique was introduced to solve this problem. At the beginning the fuzzy logic [33] and later on the machine learning system was introduced to take care of the problems stated above.

2.9.9 Discussion about learning techniques introduced in Robosoccer using the AIBO

So far learning techniques are widely used to manage physical movements in the AIBO. However, some studies have also been conducted to apply the next level of action taking process in the AIBO. Studies for both basic and high level programming for the AIBO involving RL is described here.

In 1999 [51] and 2000 [52], two studies were being conducted to solve walking problem using reinforcement learning. The aim was to develop a walking strategy to minimize the trade off between speed and stability in the prototype. In the year 2004, a function approximation technique called policy gradient was applied to improve the walking gait [53].In 2001 some teams such as baby tigers [40] and Cerberus [40] used learning technique for their teams. In this year D.Gu and H.Hu applied fuzzy logic controller for AIBO [33]. Gradually people started to apply the learning methods for common problems of Robosoccer. For example, adaptive methods and real time decision making methods were applied to localization problems in 2002 [54]. Also, in the same year RL and fuzzy

logic both were applied [33] at the same time to manage the locomotion problem in the prototype. Other learning methods than reinforcement learning were applied for color detection at UNSW in 2003 [55] and in visual object recognition for legged robots [42]. A genetics based learning program was applied for ball chasing and position reaching [56]. Other physical actions like ball acquisition were also optimized using learning algorithm [57]. Again, a learning technique used to solve the walking problem, but this time to make it faster [14] yet stable. A slightly different approach was taken by applying learning algorithm in the camera pointing strategy [58]. That research was to find a bridge between pointing at a particular object and having an overall view of the field. The main idea of this study was to find a way to keep a watch on more than one object at a time. This paper described a high level action on the basis of a physical process using reinforcement learning. Few papers also described the similar walking gait problem using some new technique, such as the fitness function from genetic algorithms [59]. This discussion shows that researchers have a tendency to apply the learning technique to solve the basic maneuver skills.

2.9.10 Real world scenario vs simulated Robosoccer

Apart from real life experiments much effort has been made in soccer simulators using learning techniques. The Robosoccer simulation league was started in 2002. In this event, the 2D simulation league was first organized in a soccer server. The server consisted of a physical soccer simulation system. Matches were being displayed on screens using simulation monitors. This effort was made to present a semi physical environment to researchers to take care of particular AI related problems in Robosoccer. It relieves programmers from handling issues like object recognition, communication, maneuverability, ball handling and other hardware limitations. On the other hand it simulates the multi-agent environment not only with a high level of uncertainty, but also with real life demands such as a minimum time for thinking processes of agents, proper strategy between team mates and so on. As a result much time has already been spent to combat AI issues in simulation league. Reinforcement learning has already been implemented successfully in simulators for some critical sub tasks such as keep away soccer [60]. This paper shows a complex multi-agent learning in a noisy unpredictable environment. A team of robots try to keep the ball in

their possession within a rectangular area. At the same time they try to keep it away from the opponent team. Another multi agent learning was developed in *A Multi-agent Algorithm for Robosoccer games in Fira Simulation League* [61]. The paper concentrated on the placement of players in order to score a goal or to make an efficient pass which minimizes the risk of losing the ball to a nearby opponent player. The position of the ball, nearby goal posts and the position of enemy players were three main criteria for the purpose. The proposed algorithm was tested against the 2002 champion team and proved efficient over hand-coded technique. In another paper, Peter Stone, et al described the use of some simple reinforcement learning techniques for the multi-faceted learning process [53]. These examples solve that the implementation of the learning process in simulators is easier than in real life cases due to the absence of physical problems. Until 2003, only a 2D simulator was available in the Robosoccer simulation league.

However, in 2004 a new 3D simulation soccer was introduced. It was much more realistic than the previous 2D one. A large number of physical rules were used to build up the environment with the help of SPADES simulation system. A change to 3D from the 2D environment incorporated many differences in terms of complexity. First of all, the agent has to choose its task within a large number of states compared to the previous one. Secondly a number of physical rules make it realistic in terms of keeping the balance of the robot, even in a simulation environment. The model of a bipedal sumo Hoap-2 robot was used for this simulator. Its movements, maneuver and other environmental properties make it almost similar to a real life system. Moreover, the use of SPADES middle-ware system removes some of the drawbacks of the 2D system like fluctuation of the team performance due to the machine efficiency and the network load.

The 3D simulator might work as an alternate solution for bipedal robot used in some event of Robosoccer. However, it can not be a replacement for the middle size four legged official AIBO league due to the differences stated below.

- First of all the movement of quadrupedal locomotion is more difficult than that of bipedal motion. So, the maneuverabilities deployed in the simulator are no match for real life AIBO locomotion issues. This is because of the random nature of damping forces such as gravity, friction acting in real life. AIBO topples in real life if imbalanced a little from its equilibrium whereas it remains on its four legs for the

most of the time in simulators in similar situations.

- Secondly, visual object recognition is a significant issue in a real life event than in a simulator. Actually the object recognition technique is straight forward in the simulation environment. However, the presence of shadows may confuse the robot in a real life situation. The intensity and part of the simulated shadow can not resemble the real life situation.The real life object recognition techniques used for robots are crucial to determine each and every situation in Robosoccer.

In summary, the papers described in this subsection, used reinforcement learning in Robosoccer in both 2D and 3D simulator. So they describe the use of reinforcement learning in a much more synthetic and controlled environment and so are not directly comparable with the work in this thesis.

2.9.11 A general approach to the existing problems

Many difficulties exist for real life robot programming over a simulator. The robotics society classified a particular hierarchical approach for different types of problems. There are three basic approaches that exist for machine learning, namely black, white and grey box approach. In the black box approach a robot automatically and autonomously acquires all knowledge base and thus develops the desired skill. No human interference and no model is given in advance for this approach. On the other hand, the white box approach provides each and every piece of information to the agent to carry out a particular assignment. Everything is strictly determined by the programmer. A middle way in between these two extreme methods is the grey box approach. A partial environmental model is provided and the desired action sequence would be available for the agent as well [62]. A machine learning algorithm is used here to complete the world model and to tune the sequence parameter. Then, finally, the agent comes up with an optimal policy. This *semi-supervised* term explains that a part of the information about the environment and task is provided to the agent. Actually in this case the environment is quantized and supplied in advance to the agent with basic actions and basic situations. Only some of the basic situations are provided to the agent and the rest is available for exploration with some given actions.

However, there are a few practical problems that exist in real life machine learning which

make it challenging for programmers [63], [64], [65]. The main issues among them are briefly discussed below.

- **High level noise**

  Low resolution camera introduces noise in images taken during game play. Moreover we are working with quadruped robots. Quadruped motion is not smooth in terms of leveling. A bubble level measuring instrument reveals that the horizontal level position of the robot constantly varies during the walking gait of the AIBO. So loss of frame, overlapping images and so on make image processing a critical issue.

- **Stochastic actions**

  The stochastic process is a kind of non-deterministic process. The output of this process belongs to a probabilistic distribution [66]. Let us consider that a robot is working under a certain environment with few states and actions defined for it. It will receive some particular reward after performing any action according to a reward matrix. If it is in s_1 situation in time t and takes a_1 action, it could receive a particular amount of reward. Again at time t+n if it comes under s_1 situation and takes a_1 action again then the received reward would probably be a different one. This is the essence of the stochastic process in the machine learning algorithm.

- **Time and material constraint**

  Convergence of the reward matrix must be achieved by a small number of learning processes. It depends upon how fast the agent can react under a real world situation.

- **Real world real time requirements**

  So many real world applications, such as soccer, require quick decision making abilities. Using adaptive methods should enable a robot to process input information and act quickly like a living animal. Present hardware and associated software methods are not yet able to collaborate fast enough to yield such output.

- **Task complexity**

  Sometimes task complexity does not permit programmers to make a white box model. Quadruped locomotion is such a process. The locomotion process of the AIBOs quadruped, involves some basic steps. It takes one of its front legs up from

the floor and so the center of gravity of the torso will be balanced by three other legs. The lifted leg is placed in front then followed by the same action with the corner wise hind leg while the three other legs take care of the robot's weight. The level of the robot is affected a lot by this process and so are the images taken by nose camera. So, balance should be maintained between leg movement and horizontal movement of the torso by flap control. A learning process should work here better than a hand tuned model due to the wideness of available situations and corresponding solutions.

The complexity of real-world robotics tasks force researchers to use the complex white box model. Previously programmers used to learn about the particular task and environment and then tune the parameters accordingly to one or more autonomous agents. Only recently the robotics community is more openly suggesting to employ the grey box approach so that robot could be trained on selected aspects of the task and certain parameters set could be automatically tuned [62, 67, 68].

Already a lot of work has been done with reinforcement learning for managing basic maneuverability skills in AIBO. We have concentrated on developing the high level behavior (real time decision making) in the AIBO. One paper [69] already used which technique in high level action selection algorithm. Three challenges were pointed out for a robot to achieve the goals there:

- Exceedingly noisy action effects often with irregular noise distributions.
- Dynamically changing environments.
- Real time decision making despite limited processing power

In that paper the first point was considered and an instance-based action model was introduced. We have focused our effort on the third point with simple off line reinforcement learning algorithm Q-learning for the purpose.

2.9.12 Introduction to next chapter

Robosoccer involves many different challenges. We chose to focus on the goal keeping problem because it has a definite maneuverability aspect and its importance to the game

as well. We also have chosen the grey box approach to train our AIBO as a goal keeper to defend penalty shots. Reinforcement learning could be used to tune sequential actions in a grey box approach. In this thesis it was used to find out suitable actions in a given situation for goal keeping. The goalkeeper was chosen to train using penalty shots taken by another AIBO from different spots. The first experiment ended up with a single step reinforcement learning process. This single step calculation excluded some important features of reinforcement learning and so a second experiment was conducted with more simplified actions and using more precise quantization of the available environments. At this stage, it was shown that the learning system was working as efficiently as the hand coded technique. The UPenn2003 code base was used for producing a bench mark and to prove the efficiency of the learning system. Moreover this experiment was extended to the two attacker problem using knowledge acquired from one attacker experiment. In that case, one attacker passes the ball to another and the second agent takes a shot towards the goal using the running ball. The knowledge bank obtained from the second experiment was used for the two attacker problem as well. It was deemed useful for the purpose. The point to be noted here is the knowledge of the goalkeeper is completely determined by the programmer using hand coded technique. In contrast the reinforcement learning started with no knowledge and ended up with a similar output. So, the reinforcement learning is used to solve the goalkeeping problem in this thesis and the reason of using a particular RL technique will be addressed in the next chapter.

Chapter 3

Methodology

3.1 Introduction

Broadly, the research problem in this thesis is about how to apply reinforcement learning in Robosoccer. To be precise, the focus is on applying a specific RL technique in decision making for goalkeeping. The aim here is to use an RL technique in the AIBO for it to learn the best action in a given situation to perform goalkeeping. The reason for using a particular learning technique for this problem is addressed in this chapter. Moreover, it is shown here that the particular technique, Q-learning, is more appropriate than its counterpart SARSA in this regard.

The incorporation of a learning process into a physical robot is a significant issue for AI research groups [70]. In robotics, a learning process can be used in performing several tasks. The basic idea is to make an agent take decisions autonomously in a novel situation. In a controlled and limited environment, a robot can work efficiently with a white box model as described in Chapter 2. In such a model, a programmer is aware of each and every possible situation well in advance. So, a model with perfect input/output working pairs can be defined for the agent. The input signals are mostly free from noise in a controlled environment. As a result, the agent can detect any situation almost correctly and map the perfect output action in response. However, the problem considered here starts with a partially known or fully unknown environment which could better be addressed with the grey box or black box model respectively, described in Chapter 2. As an example,

the strategic actions become difficult due to the presence of team mates and opponent players in a partial noisy environment like Robosoccer. Moreover an agent often misjudges a situation due to a partly noisy environment. Secondly, an agent has to take an optimized or best action after correctly determining the situation. In this thesis, we will establish that a learning process proves as good as hand-coding in determining the right action for the right situation, without human supervision during the training exercise.

A learning process like RL has already been applied in learning low level actions [57] such as managing basic skills like locomotion, image processing in different versions of robots including AIBO and ball acquisition as discussed in Chapter 2. The RL is a semi supervised system and so it partially works without human supervision. The learning system used in this thesis, does not deal with physical behavior at all. The system only manages the decision making process for performing the goalkeeping task. More precisely, the RL is basically used here to develop the right maneuver at the right moment. Two basic experiments were conducted to demonstrate the efficiency of RL against Upenn'03 code. An environment set with Robosoccer field specifications was used for these experiments.

However, before proceeding with the main experiments it is necessary to decide on a particular RL method to use. Two basic RL methods are Q-learning and SARSA. A comparison between these two similar processes is described here based on a simulated maze-learning environment. Finally, the comparison led us to select one of them for the goalkeeping experiments.

In the first goalkeeping experiment, an attacker takes some shots from three fixed points towards the goal and the goalkeeper learns to block the ball using a 3x3 state x action table. However, after performing this experiment it was observed that the experiment was almost a trial and error method and was not actually using the temporal difference feature of RL as discussed in Section 2.6. The single step learning process was preventing it from doing so. As a result, it was decided to perform another experiment which used a three step, back-propagation learning process using the Q-learning formula.The theoretical overview behind these two experiments is discussed further in this chapter.

3.2 Choosing a preferred learning approach

3.2.1 A simulated maze learning example using Q-learning and SARSA

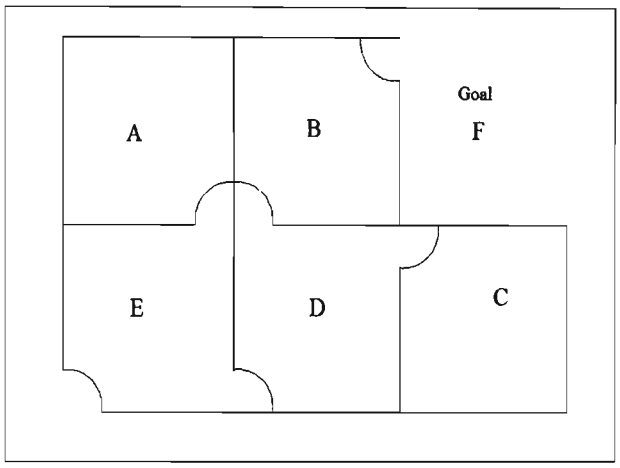

Figure 3.1: Maze learning environment

Let us consider an existing example with maze learning simulation systems to find the difference between two different learning techniques. This can be found in the existing example [31]. An agent is being placed in an environment consisting of six separated areas which are called as rooms in here. The task of the agent is to *navigate to a particular room using the shortest possible distance from any room*. The environment plan is given below.

According to Figure 3.1, the outside region F is the target room for the agent. The agent may start from any of the rooms including the target room 'F'. A quarter circle between two rooms represents a door. Every room name including F is denoted as a state. It will be considered as an action when the agent changes rooms, too. This letters (room names) are used to present the states in state x action Table 3.1 as well. However, the access between the rooms are restricted and described in the reward matrix shown in Table 3.1.

Both Q-learning and SARSA were applied to this problem. The value of the learning parameter (α) and discount factor (γ) used in Q-learning Equation 3.1 are 1.0 and 0.8

respectively. This experiment was designed in a computer simulated environment. As a result, it was a completely noiseless environment which allowed us to use the value of learning parameter as 1.

So, using $\alpha = 1$ and $\gamma = 0.8$ the simplified Q-learning shown in Algorithm 1 and SARSA shown in Algorithm 2 formula are described below

Q-learning

$$Q(state, action) \longleftarrow 1.0 * R(state, action) + 0.8 * Max(Q(nextstate, allaction)) \quad (3.1)$$

SARSA

$$V(state, action) \longleftarrow R(state, action) + 0.8 * V(nextstate, nextaction) \quad (3.2)$$

The value of R is considered as 100 in this case. This is used in terms of reward quantity in table 3.1 here. This matrix shows that the agent achieves a reward only when it moves to room 'F' from rooms 'B' and 'E'. If it starts from Room 'F' then it is rewarded straight away without making a move. The reward matrix also reveals the location of doors between rooms with a '0' at the intersections of two rooms mentioned in the corresponding row and column. The restriction of movement between rooms is shown using '−'. In other words, the action denoted by '−' can not be selected from that particular state. The agent can take an action at any state if the state x action pair is linked by either 0 or 100. The reward is denoted by the numerical value '100' at the proper state transition point. The full reward matrix with permissable actions is shown in at Table 3.1 with states along the rows and actions along the columns. The matrix shows that only rooms 'B','E' and 'F' have a door to the target room that is 'F'. If the agent starts at room 'F', then it stops right there and receives a reward. The agent starts with a blank memory which is a state x action matrix with all values set to zero, as displayed in Table 3.2.

3.2.2 Q-learning

The agent follows a few steps to complete the learning process using the Q-learning algorithm 2 in Chapter 2. The agent may choose an action randomly using this algorithm. Apart from that the agent could choose a different policy instead of taking actions randomly. The 85% greedy policy algorithm is described in this regard. This 85% greedy

Table 3.1: Reward table

	A	C	T	I	O	N	
S		A	B	C	D	E	F
T	A	-	-	-	-	0	-
A	B	-	-	-	0	-	100
T	C	-	-	-	0	-	-
E	D	-	0	0	-	0	-
	E	0	-	-	0	-	100
	F	-	0	-	-	0	100

action selection policy is nothing but the ϵ greedy policy with $\epsilon = 0.15$. Due to maintaining a balance between exploration and exploitation in SARSA we have chosen an 85% greedy policy with the maze-learning using SARSA. In order to present a comparison, the Q-learning with 85% greedy policy is described in Algorithm 3 in this chapter.

So, two different policies are used to update Table 3.2, using Q-learning in this case.

Policy 1 Use random action selection policy

Policy 2 Use 85% greedy action selection policy

The point to be noted here is that strictly the maximum available reward from the next state is chosen here to update the table irrespective of the action selection policy. The size of the full state x action matrix permits the agent to get the training done within a reasonable amount of time. The convergence of the matrix ensures that enough chances are given to try out all possible actions. In the next chapter, we have explicitly described the learning technique using a random training epoch.

Consider that the agent starts from room 'A', moving randomly. It will not receive any reward until it reaches room 'F'. Let us consider that at a point it reaches room 'F' through room 'E'. Using the Q-learning formula, the reward is derived from the state transition from 'E' to 'F' using the following steps.

Table 3.2: Initial State x Action Table for Q learning and SARSA

	ACTION					
S		A	B	C	D	E	F
T	A	0	0	0	0	0	0
A	B	0	0	0	0	0	0
T	C	0	0	0	0	0	0
E	D	0	0	0	0	0	0
	E	0	0	0	0	0	0
	F	0	0	0	0	0	0

- *Q(State, Action)* $= 100$ *(As R=* 100 *for the given state transition)*
- *It is the terminal state and so learning stops here.*
- *This is the terminal state and so the term* $0.8MaxQ(NextState, AllAction)$ *is invalid according to the Q-learning formula.*

The resultant state x action matrix would look like Table3.3.

Table 3.3: First intermediate State x Action Table by Q-learning

	ACTION					
S		A	B	C	D	E	F
T	A	0	0	0	0	0	0
A	B	0	0	0	0	0	0
T	C	0	0	0	0	0	0
E	D	0	0	0	0	0	0
	E	0	0	0	0	0	100
	F	0	0	0	0	0	0

Next time, consider that the agent starts somewhere except for room 'E' and 'F'. During

Algorithm 3 Q-Learning with 85% greedy policy

Initialize Q(S,A) Arbitrarily

Initialize S

choose A from $A(S_t)$ using 85% greedy policy (If no reward available use random policy)

repeat

take action A, observe S and A'

choose A' from S' with random policy

update table using Q(S,A) $\longleftarrow$ Q(S,A) + α[R + γ max_{A^*} Q(S',A^*) - Q(S,A)]

S $\longleftarrow$ S'

A $\longleftarrow$ A'

A ϵ A^* //A^* denotes all possible states

until terminal S reached

RETURN Q(S,A)

end

training it may go to room 'E' from room 'D' as there is a door that exists in between. The following mathematical steps are involved to upgrade the state x action table, in this case. So the summary we have from this training is as follows.

- Q (State, Action) = 0 (From reward matrix)
- Max (Q(next state, all action)) = 100

$$Q(state, action) \longleftarrow 0 + 0 + 0.8 * 100 \tag{3.3}$$

$$Q(state, action) \longleftarrow 80 \tag{3.4}$$

Now the state x action table would appear as in 3.4. In this way, the whole matrix would be updated and converge to Table 3.5 after completion of the training. This table contains the final output using both the 85% greedy and random action-selection policies. The result shows perfect convergence which indicates the end of training. Using this matrix, the agent can find the shortest path to the target room 'F' from any room. Let us consider that the agent starts at room 'C'. The shortest path to the target room would

Table 3.4: Second Intermediate State x Action Table by Q-learning

	A	C	T	I	O	N	
S		A	B	C	D	E	F
T	A	0	0	0	0	0	0
A	B	0	0	0	0	0	0
T	C	0	0	0	0	0	0
E	D	0	0	0	0	80	0
	E	0	0	0	0	0	100
	F	0	0	0	0	0	0

be C-D-E-F or it could be C-D-B-F as well. Accordingly if the agent starts at room 'E', it will go straight to room 'F' from there and if it starts from room 'F', it stays there. This table is a product, obtained by Q-learning formula, which uses the highest reward available on the next state to update the table and a random policy to pick up an action. This ensures a fare amount of exploration while exploiting the experience at the highest level. This is the reason why the Q-learning formula makes a perfect balance between the exploration and exploitation dilemma. This experiment takes more or less 4000 training cycles to converge while using an 85% greedy action selection policy and highest reward selection method. In contrast with that it completes the training using only 1800 to 2100 training cycle while using a completely random policy. A difference of less than 0.01 between the two consecutive output matrices is considered as convergence in this case. So, it is clear at this point that this maze-learning process has successfully completed using Q-learning [31]. Moreover, it takes less time while using the random action selection policy than using the 85% greedy policy.

3.2.3 SARSA

SARSA was applied to this problem in the same way. The same 85% greedy policy from Q-learning was used here. This level of exploration ensures that the agent uses strictly greedy policy for most of the training period. It also ensures that enough chance is given

Table 3.5: Complete State x Action table by Q-learning

	A	C	T	I	O	N	
S		A	B	C	D	E	F
T	A	0	0	0	0	80	0
A	B	0	0	0	64	0	100
T	C	0	0	0	64	0	0
E	D	0	80	48	0	80	0
	E	64	0	0	64	0	100
	F	0	80	0	0	80	100

to the agent to explore the environment while exploiting the available experience as well. The final outcome of using SARSA is shown in Table 3.6.

Table 3.6: Complete State x Action Table by SARSA

	A	C	T	I	O	N	
S		A	B	C	D	E	F
T	A	0	0	0	0	80	0
A	B	0	0	0	64	0	100
T	C	0	0	0	64	0	0
E	D	0	64	48	0	80	0
	E	64	0	0	64	0	100
	F	0	80	0	0	64	100

The optimal path to target room from 'D' is either D-B-F or D-E-F, whereas according to SARSA, the optimal path is only D-E-F and not D-B-F. The Q-learning revealed both the path by assigning an equal weight to the cells corresponding to columns 'B' and 'E' in row 'D' in Table 3.5. So, all the optimal paths were not fully revealed by SARSA. This happened due to choosing a particular action and the reward associated with it to

complete the table. But Q-learning always considers the maximum available reward out of all available actions from the next state. It was also noticed that the output state x action table by SARSA does not converged to that of Q-learning. Different output matrices were produced from multiple runs of the same program using SARSA. Each time 50000 training exercises did not end up with the same output matrix and each time one or two of the shortest paths were not discovered. So, it is clear that Q-learning with 85% greedy action choosing policy is better than SARSA for producing a full proof result for the simulated maze-learning problem. Again Q-learning with random action selection policy takes almost half the time to converge than that with 85% greedy action selection policy.

This discussion points out that the *high learning rate* influences the results of both the experiments using Q-learning and SARSA. Due to this high rate, SARSA is sensitive to the recent exploratory action. A high learning rate is also not recommended for a partially noisy environment like Robosoccer. The presence of different players and their activities are responsible for introducing noise in Robosoccer environment. On the other hand, Q-learning acts faster with random action selection policy. This policy is suitable for a partially noisy environment like goalkeeping. However, an overview of the concerned goalkeeping experiment should be discussed before choosing the right method for training.

3.3 A brief description of state x action tables used in experiments

There are two state x action tables used for goalkeeping experiments. These tables were developed according to the requirements of goalkeeping against penalty shots from different positions and also against two attackers. The first table was designed in such a way that an attacker takes shots from the penalty box three different positions. It is a $3X3$ state x action shown in Table 3.7.

Due to the absence of temporal difference learning feature of this experiment it was decided to extend it further with a table appeared as a $4X7$ matrix described in Table 3.8.

Table 3.7: Initial State x Action Table

	A	C	T	I	O	N
S			a_1	a_2	a_3	
T		s_1	0	0	0	
A		s_2	0	0	0	
T		s_3	0	0	0	
E						

One of the challenges in this research was to find the right learning method one that is suitable for suits two state x action Tables 3.7 and Tables 3.8.

3.4 Choosing the correct method for the goalkeeping experiments

So far, a simulation experiment has been discussed here which consists of a 6x6 table and uses a completely noiseless environment. The nature of a state x action table for the upcoming goalkeeping experiments is described as well. Accordingly, the goalkeeper experiments have two key features which influence the choice of a particular method to work with.

The size of the state x action table There are two state x action tables in use for the goalkeeping experiments. The biggest table consists of four columns and seven rows which is less in size than that used in the maze learning experiment.

Practical hazards AIBO robots are made to shut down when a software exception occurs in the operating system (OS) code or any of the joints experiences an obstruction beyond a threshold. This feature made the robots to go off line several times while experiment was on. Each time the boot up process was draining a large amount of energy and it resulted in draining the battery life quickly. The environment we used was far less noisy than a full scale Robosoccer environment. However, the jerks

Table 3.8: Initial State x action Table

	A	C	T	I	O	N
S			a_1	a_2	a_3	a_4
T		s_1	0	0	0	0
A		s_2	0	0	0	0
T		s_3	0	0	0	0
E		s_4	0	0	0	0
		s_5	0	0	0	0
		s_6	0	0	0	0
		s_7	0	0	0	0

due to movement of the goalkeeper introduced a little noise in calculating the ball distance using the nose camera.

These two points forced us to use a technique which could yield a suitable result with the 4x7 state x action table and at the same time enable it to converge as soon as possible. It is obvious from the previous discussion that Q-learning with random action selection policy is better than SARSA in case of noiseless maze learning for two main reasons:

- High speed converging rate
- Full consideration of each and every possible solution

So, finally it was decided to use Q-learning with random action selection policy but with a high value of alpha. It was chosen due to the fact that the Robosoccer environment involves negligible amount of noise signal only which makes it similar to maze-learning scenario. The experimental setup used in this thesis will be discussed in the next chapter. The goalkeeping training experiments will be performed using this setup and the Q-learning formula with random action selection policy.

3.5 Choice of the software environment for programming

Officially, Sony released a Standard Development Kit (SDK) for programmers around the globe [71]. It is a fully Linux based development environment. The script was written for Vine Linux distribution initially, but it has also been supported by other Linux distributions having some particular dependencies. Unlike windows, Linux softwares are available in the form of code. This is due to the *Copy Left* idea from Open Source Society. The user has to compile the code using some special commands and generate the particular machine code for the Linux distribution being used. This compilation process needs a different program compiler to be installed already in the system. In addition to that, those compilers should have some particular features which vary from one version to another, these are called dependencies for compiling a software code in Linux. Moreover more than one version of a similar compiler could be available in a single Linux system. But the system will recognize a particular version as the default option declared in the user profile. The Sony SDK was developed using the C++ programming language. First of all we started testing some of the programs which are already provided in the SDK. But, compilation of a new program in the local machine needed few more software dependencies. We started with Cygwin[1] and installed PERL to run some existing programs, but further packages, needed to compile a new program were not available with Cygwin for Windows. So, we have focussed on choosing an open source OS and installed Fedora Core 6. It was too advanced for the SDK compiler. As the SDK was released long ago and so some of the dependencies were not available for the the full installation in Fedora 6. SDK was made for low level programming. All the primitives were available in it and one could write a code from scratch for AIBO using it. A high level frame work Tekkotsu [72] was made on top of that to provide one step ahead solution for programmers around the world, but one still had to install Sony SDK with full features to play with Tekkotsu. So, the idea of using Sony SDK in the thesis was abandoned. In the recent past, some interpreters developed to write program for AIBO through Linux, Windows and MAC OS

[1]A command prompt utility work under windows environment to simulate a group of Unix commands

as well. In addition to that these wrappers were equipped with the facility to make some common programming language compatible with AIBO from the windows environment; these languages are namely Python, Matlab, Java and C++.

First of all, the focus was to concentrate on one of the simulators made for AIBO embedded inside the Knoppix Linux distribution. This distribution does not need to be installed in the hard drive and can run directly from CD ROM. Unfortunately, this feature does not allow the user to save any data in hard drive formatted with New Technology File System (NTFS). Moreover, we found that some of its features lead to frequent breakdown of the software and make it almost impossible to use. So, the Universal Realtime Behavior interface (URBI) system [73] was chosen at last. This is a software wrapper over Sony SDK and allows other programming languages to communicate with AIBO with an in-built program interpreter. A simulator for AIBO namely Webots was released to assist programmers to test their skills virtually using URBI scripts and other languages. Both the URBI and Webots were made for Windows and Linux. A user has the privilege to change the environmental variables in this simulator, which makes it more realistic. In the simulator there is a separate window to show the nose camera view of AIBO. But no option exists to program the AIBO camera using it. So, we have tested our maneuver programs only into the simulator before testing it practically. Those programs worked properly in real life situation with little calibration.

The essence of URBI control lies in its simple yet powerful nature. It has two different architectures for programmers. The first feature is available with the Windows installer. It allows a programmer to run AIBO using a scripting language only provided by URBI. The language consists of commands to control the primitives and some new features created, combining the existing techniques. The in built ball object helps the programmer to make the robot look at the ball and track its center with a single command, whereas the primitives allow for movement of a single leg joint at a desired angle. These codes could be copied to the memory stick or could be sent instantly via wireless link to the temporary memory of the robot. We used this feature and sent our raw codes through wireless link until it worked fully. Once it started working fully, it was copied into the stick. The "URBI" programmable memory stick (PMS) software contains an easy architecture to follow. The IP edition process is the same for both Sony SDK and URBI. The "WLANCONF.TXT" file

available at "OPEN-R SYSTEM CONF" location contains the IP and related information. One can toggle the dynamic host configuration protocol(DHCP) on and off by changing a command line in this file too. This is the feature of a local area network (LAN) card for acquiring a dynamic IP against the media access control address(MAC). But we experienced many problems with having the DHCP server on and so decided to use static IP using class C private IP pool. The AP was addressed as 192.168.10.1 and three robots with 192.168.10.2, 192.168.10.3 and 192.168.10.4 with 255.255.255.0/default subnetmask. Writing a new script and modifying an existing one is one more simple nature of the URBI. Being open source software, all the scripts are available inside the PMS. In fact we have started by altering some available scripts and worked on the existing walking procedure to make it efficient. However, the lay-to-stand function was not up to the mark. It caused the robot change its direction while standing up from a complete lying position. So, a different approach was introduced which is similar to the stand up technique of a camel. We put that script inside the PMS with the others and made it one of the default features. The URBI.inf file takes care of the list which is being used to load programs at the start up. We developed a new script for sidewise movement. This action is one of the most important among others during Robosoccer. The next stage program creation using URBI involves different object creation rather than writing simple functions. This feature would be helpful for the future work.

The following chapter describes about the experiments in detail, that have been performed to find an answer to the research question stated in Chapter 1.

Chapter 4

Experiments

4.1 introduction

This chapter describes seven experiments, stated below. The experiments stated below, will provide a practical overview of using Q-learning in playing Robosoccer using the AIBO.

1. Ball distance measurement experiment using nose camera
2. Ball distance measurement experiment using IR sensors
3. The experiment for goalkeeping training with single attacker and a 3x3 state x action table using Q-learning
4. The experiment for goalkeeping training with an extended 4x7 state x action table over the first Q-learning experiment
5. The two attacker experiment using the available knowledge base from the goalkeeping experiment with 4x7 state x action table
6. The one attacker benchmark experiment
7. The two attacker benchmark experiment

The next section specifies a common environment which was made using official Robosoccer field specifications. These will be used for all the experiments from now on.

4.2 Common setup for all experiments

The experimental setup for goalkeeping training and ball distance measurement technique were processed using the official field specification for Robosoccer using AIBO as mentioned in Figure 4.1. We used one half of the total field area without colored poles situated by the side. These poles are for ease of the localization process and the localization techniques were not coded or used in these experiments as we have discussed in chapter 1. This thesis is dedicated to study the application of the reinforcement learning process on AIBO and train it as a goalkeeper. The URBI system was used as the programming environment. Unfortunately, the URBI had no particular function to calculate the distance of the ball from AIBO. So, a separate method is described for that and an experiment was devised accordingly.

The entire thesis consists of seven experiments:

4.3 Ball distance measurement experiment

4.3.1 Background of the experiment

The AIBO is one of the few low cost robots, consisting of several different sensors in a single prototype. Engineers at Sony tried to make it similar to a real life dog and so tried to put all sensors on its different parts and made it as close as possible to a real-life dog. Some of these sensors are used in playing Robosoccer too. One of the mandatory actions in Robosoccer is to calculate the distance of the ball and other players from the robot. AIBO has two different Infra Red (IR) sensors at its nose. One is to sense the distance of nearer objects and the other is for the far objects. Infrared ray is one of the members of the electromagnetic spectrum family. It falls within the wave length range between 1 nanometer(nm) and 750nm and is invisible to human eyes. The working principle of those two IR sensors are stated below.

One sensor emits the infrared ray at a time at the direction of head pan angle. This reflected ray is received by two different sensors at a time, situated by the side of the transmitting device. The difference between the transmitted and reflected times are used

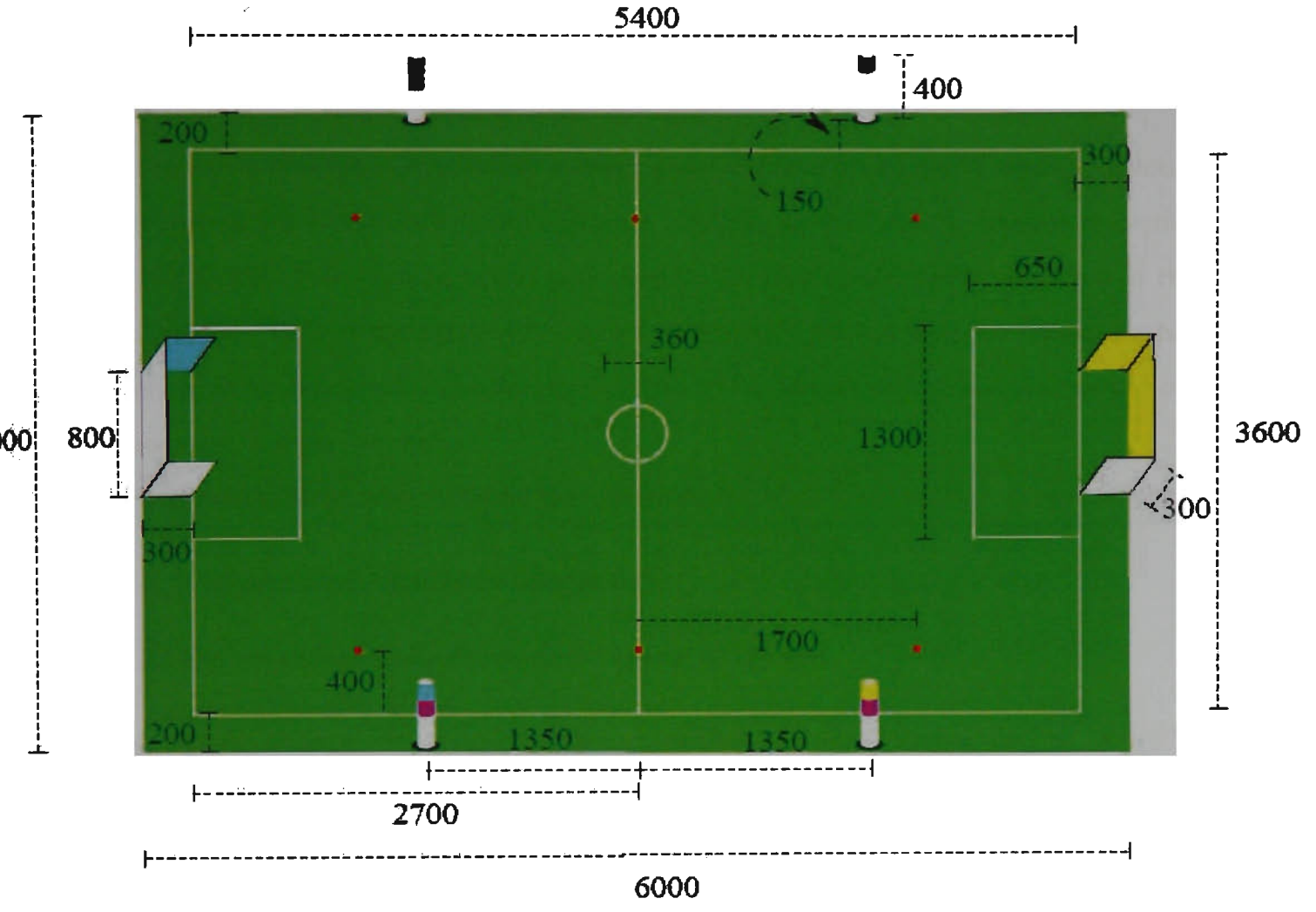

Figure 4.1: Arrangements for first experiment (Units are measured in millimeter) [3]

to determine the distance. It is wise to use the data from the near sensor when an object is situated closer than 20cm in front of the robot. On the other hand, the rear distance sensor is appropriate for the objects situated between 20cm and 150cm distance from the robot. These two IR sensors are supposed to return a rough idea of the nearest object in front of the robot, but in Robosoccer we need more accurate measurements of distances from few particular objects and their recognition is also important in this context. There are two basic problems that exist with using the IR sensors to obtain ball distance in the field. First of all, it has to move its head towards the floor when looking at the ball. At this moment, the IR receiver receives part of the reflected ray from the ball and some from the adjacent floor. As a result the data become fuzzy and fluctuating. Furthermore, IR sensors can not recognize the difference in color between the red ball and other objects.

So, if another player comes in front of the ball then the agent will consider the distance of that player as the distance of the ball. So, we have taken a different approach for ball detection to calculate its distance from the agent.

The AIBO also has a color video camera situated below the nose IR sensors. It has a Complementary Metal Oxide Semiconductor (CMOS) sensor with the highest resolution of 108 X 260. Few existing codes were used for to develop the main code used in this experiment. Two of the existing codes are used from URBI repository. These are ball tracking code (modified) and ball.ratio function. The way we used those functions in our experiment is described below.

The ball tracking code performs two functions.

1. It detects the pink colored circular ball

2. The nose camera is pointed at the center of the ball

The viewing angle of the video camera is such that it accommodates the ball within its view at a distance greater than 3cm from the nose. So, we can make sure whenever the ball is visible beyond that range, the agent is looking at its center. In that case, a part of the full available scenery to the nose camera is then occupied by the ball. At a particular distance, the ball will occupy a fixed amount of image area. Ball.ratio function yields the percentage of the area occupied by the ball in a picture. The major goal of this experiment is to calculate the distance of the ball using the value of ball.ratio. The robot also needs to know the ball direction, but the value of head tilt angle can be used directly to determine the direction of the ball. So this experiment emphasize on calculating the ball distance.

4.3.2 Experiment for measuring the distance between the ball and robot using the nose camera

4.3.2.1 Aim

This experiment was designed to find a method to determine the distance between the pink ball and the nose camera and to establish the accuracy of this method.

4.3.2.2 Equipment

We used one AIBO, the ball and points at fixed distances on the floor for this experiment.

4.3.2.3 Setup

The robot was placed in a fixed position with the ball tracking code enabled on it. It looked at center point of the ball with a fixed neck value and two variable parameters, namely *head pan* and *head tilt*. The ball.ratio function calculated the ratio of area occupied by the ball to the full visible area in picture and returned the value. We put the ball at different points in front of the robot. Five sets of values were taken at each point due to the minute fluctuation of placement by the natural forces and due to the variation in electromagnetic signal obtained from the IR receiver.

The radius of the ball is nearly equal to $3cm$. So we put points $3cm$ apart from each other for the experiment. Ball.ratio values were recorded after placing the ball at each point. A total of 5 samples were taken at every point and an average value was determined using them. We considered a total of 31 points starting from $3cm$ distance to $99cm$. The distance was measured approximately from the projection of the joint of neck and torso, on the base. So, the look up table we created using this experiment, permits the goalkeeper to detect the distance of the ball from $3cm$ to $99cm$ only. A picture shown at Figure 4.2 reveals the real life experimental scenario.

4.3.2.4 Method

We built up a table with the actual ball distance and corresponding ball.ratio values in order to find out a relation between them. But, no simple statistical method was able to generate an equation which could be used to reproduce the distance using ball.distance values as input. So we used the entire table for the distance measurement purpose and a two point linear equation method was used to calculate the distance in between each of the two extreme points situated at $3cm$ spacing.

Theoretically, if either the x or y coordinate of a point is given, the other could be calculated using two point equation form, provided the point is situated on a straight line

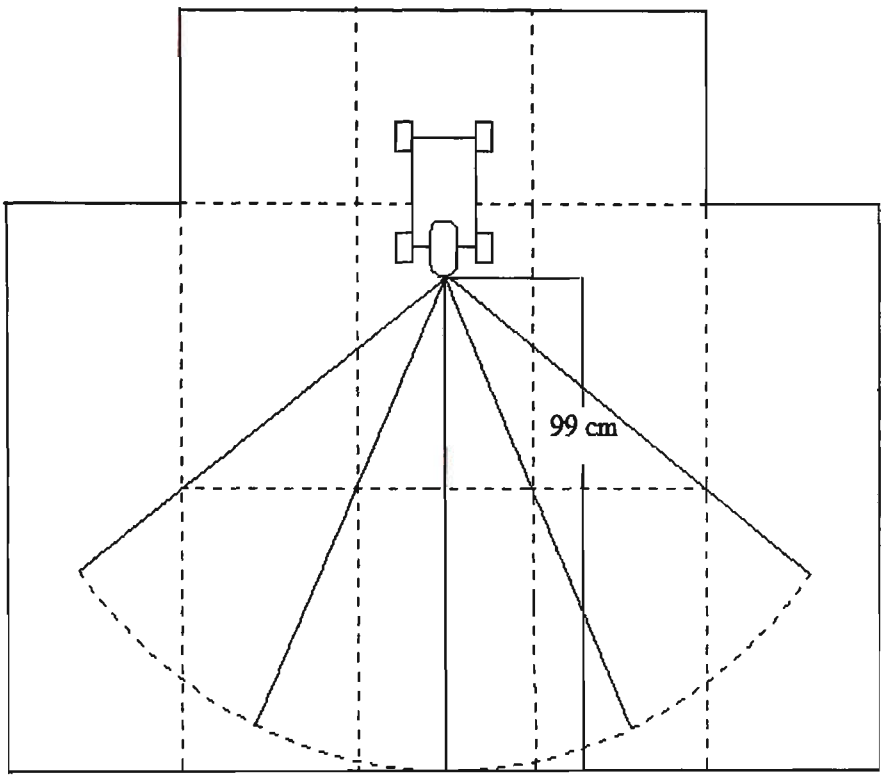

Figure 4.2: The ball distance measurement scenario from the point of view of a goalkeeper

going through two fixed points. The formula is as follows

$$(y - y_1) \div (x - x_1) = (y_2 - y_1) \div (x_2 - x_1) \tag{4.1}$$

(x, y) = any point on the straight line

$((x_1, y_1), (x_2, y_2))$ = Two fixed points on the line

Consider that the ball.ratio value at any instant is 0.0050.. It indicates that the ball is between $96cm$ to $99cm$ from the look up table. The values of the parameters would be as follows

$x_1 = 96,$

$x_2 = 99,$

$y_1 = 0.0048,\ y_2 = 0.0053,$

$y = 0.0050$

The value of x corresponds to the value of y that is 97.8, calculated using two point equation mentioned above. So the ball is at $97.8cm$ distance if ball.ratio value is 0.0050. Applying two-point equation of a straight line, a more accurate distance could be found with the assumption that change of distance is linear within each pair of readings. Table

5.1 shows the recorded ball.ratio values against the actual distance.

4.3.3 Experiment for Measuring the ball distance with IR sensors

4.3.3.1 Aim

This experiment was designed to measure the distance of the pink ball from the robot using IR sensors. Those sensors are situated just below the nose camera.

4.3.3.2 Equipments

An AIBO robot and a pink ball were used only for this experiment.

4.3.4 Setup

There are three IR sensors available just below the nose camera in a AIBO ERS-7 model of which two are receiving sensors. The *near* sensor is to determine the distance of an object within 20cm in front of it. The other one is denoted as the *far* sensor and is made to measure a distance between 20cm and 150cm. The pink ball is placed at each and every points 3cm apart.

4.3.4.1 Experimental Methods

The *near* and *far* IR sensor values were recorded here at each of every 31 points. Five readings were taken at each point due to the minute fluctuation of placement by the natural forces and the variation in electric signal obtained from sensor. These values are available in Chapter 5 in Table 5.2. The average of five readings was placed under the *average value* column and the two extreme readings as highest and lowest values. The results obtained using the IR sensors and those obtained from the camera were compared to know the relative accuracy of this two methods, discussed in Chapter 5

4.4 The experiment for goalkeeping training with single attacker and a 3x3 state x action table using Q-learning

4.4.1 Aim

The aim of this experiment was to train an AIBO as a goalkeeper using Q-learning method. At this stage, A 3x3 state action table was used as a state x action table.

4.4.2 Setup

Two robots, a pink ball and a penalty area with dimension set by official Robocup authorities were in this experiment shown in Figure 4.1. There the attacker shoots the ball straight away at the goal.

4.4.3 Experimental Method

One AIBO took penalty shots from a point situated at 63 centimeter distance from the goal line, at three different points as shown in Figure 4.3. The goalkeeping agent was waiting in the middle part to block the shot. The agent was allowed to take three different actions. These were, stay in the middle, go left and go right. The striker was also allowed to take shots at left, right and middle from the penalty point. At the end of the training the goalkeeper should learn to block these shots while start the training with zero knowledge.

According to Q-learning theory an agent starts learning from a zero Q-value matrix and updates its value after completion of each training cycle. The state x action table used here consists of basic states, and actions in the headings of rows and columns successively. It was decided to use three basic types of shots as three basic states and three corresponding movements as three basic actions. The attacker stayed at three different points on the penalty line and was allowed to take shots towards the right, middle or left side of the goal from each point. These shots are considered as states. The goalkeeping agent can watch over the incoming ball and decide the corresponding state according to the value

of headpan angle. The relevant Also the goalkeeper could take three different counter measures to block these shots, which are regarded as actions in this case. According to the defined actions, it could move towards the direction of incoming shots and block it. The initial state x action table is described in Table 4.1

Table 4.1: Initial State x Action table

	A	C	T	I	O	N
S				a_1	a_2	a_3
T			s_1	0	0	0
A			s_2	0	0	0
T			s_3	0	0	0
E						

States

s_1 = Ball is moving towards the left side of the goal.

s_2 = Ball is coming straight towards the goal through the middle area.

s_3 = Take shot at right side of the goal.

Actions

a_1 = Move towards left side of the goal and block/stop.

a_2 = Stay in middle position (and block). The agent will not take any action physically in this case because it always starts from this position of the goal.

a_3 = Move towards right side of the goal and block/stop.

Figure 4.3, displays the view of the field from the point of view of the goalkeeping agent. Figure 4.4 and Figure 4.5 state the actual situation with space quantized according to state x action table.

The three states, described above, are few particular situations sensed by the agent while playing. Here, the AIBO sensed the direction of the incoming ball from its head pan angel. According to the code, the AIBO always looks at the center of the ball. As

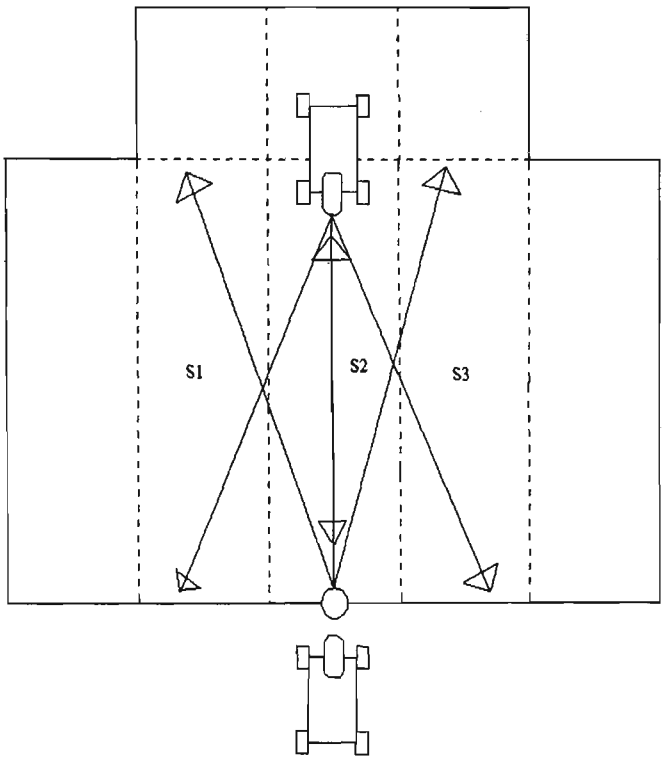

Figure 4.3: A goalkeeper's view of the experiment with 3x3 state x action table

a result, if the value of the head pan angel is equal to or more than $+7$ degrees, the ball is considered to be on the right side. If it is equal to or less than -7 degrees, the ball is assumed to be on the left side. So, when the ball is on the move, the AIBO calculates the distance after every 1.5 seconds and acts accordingly. The amount of this interval was chosen depending upon the processing power of the robots and the tasks involved. As the millisecond factor is allowed to be used with URBI, it was easy to define a fraction of seconds properly. Each of three actions devised here, are combination of two basic maneuvers. The first one is a sidewise movement and the second one is block (Stop in front of the ball). a_1 and a_3 are two actions that consist of the left and right side movements consecutively followed by block for each case. For a_2 describes the robot to stay in the middle and then block the ball when it is in close vicinity.

The values of the state x action table 4.1 indicate that the agent started with zero knowledge. It gains experience as the experiment goes on.The working formula to update the table is as follows.

$$V(S_t) \longleftarrow V(S_t) + \alpha * [R_{t+1} + \gamma * V(S_{t+1}) - V(S_t)] \tag{4.2}$$

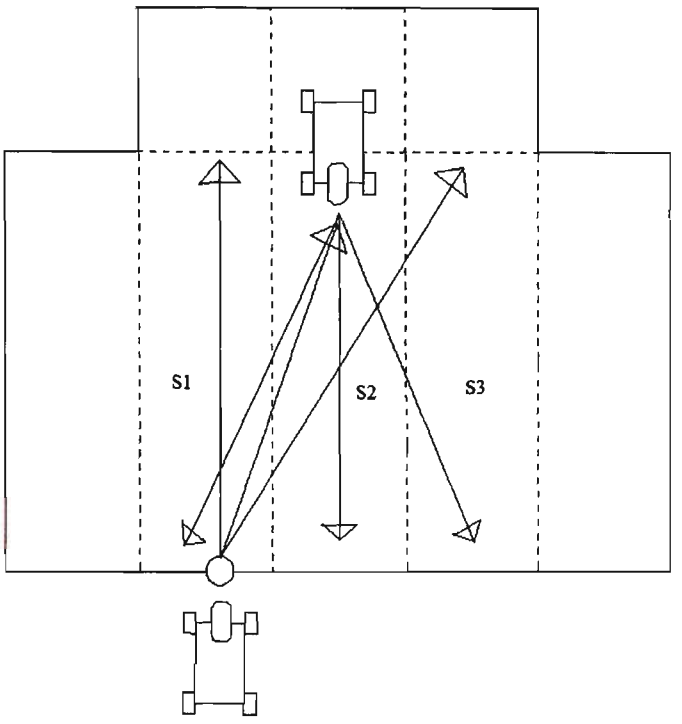

Figure 4.4: A goalkeeper's view of the experiment with 3x3 state x action table

R_{t+1} = Reward observed at time t+1

S_t = State visited at time t

R_t = Reward after time t

α = step size (a constant parameter throughout the whole experiment)

γ = Another constant parameter

We have used a simplified form of this formula. The value of α will be 1 due to low noise environment. So the formula would be as follows:

$$V(S_t) \longleftarrow R_{t+1} + \gamma * V(S_{t+1}) \tag{4.3}$$

Furthermore, single step learning is automatically used here due to the size of state x action table. So, the second parameter on the right side of the equation could be omitted as well and the final form of the operating equation would look like the following equation 4.4.

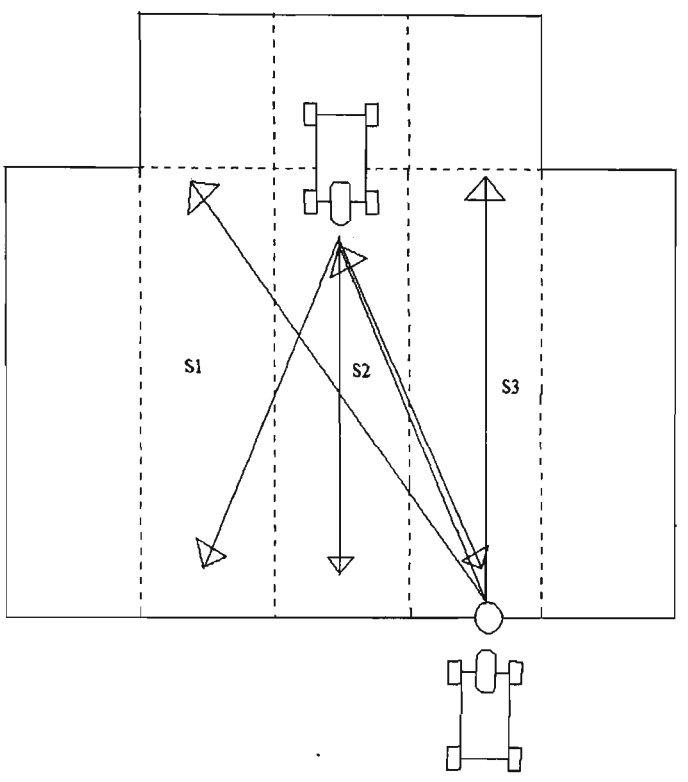

Figure 4.5: A goalkeeper's view of the experiment with 3x3 state x action table

$$V(S_t) \longleftarrow R_{t+1} \tag{4.4}$$

In this case the reward $R_{t+1} = 100$ as a reward of finishing the given task. During the experiment, the attacker took a total of 30 successful shots in three halves. Each of the 10 shots were directed at each of the three directions to cover all states. First of all, the agent (the goalkeeper here) tried to find out the particular state, in which the ball belongs. Practically, it pointed the nose camera at the ball using the ball tracking code and found the position (distance and direction) of the ball. Accordingly, it decided whether the ball was on the left, middle or on the right part of the penalty area. According to RL theory, the goalkeeper starts with no knowledge which is indicated by a state x action table filled with zeros. So, at the beginning it randomly chosen different actions to block the shots. However, whenever it finds some experience, it started to exploit it and acted accordingly.

4.5 The experiment for goalkeeping training with an extended 4x7 state x action table over the first Q-learning experiment

4.5.1 Aim

The aim of this experiment is to train the goalkeeper using 4x7 state x action table. It is an extension of the first experiment using Q-learning.

4.5.2 Setup

Two robots, a pink ball and a penalty area with dimensions set by the official Robocup authorities shown in Figure 4.1 were used in this experiment. The attacker AIBO took few penalty shots from different spots towards the goal.

4.5.3 Experimental Method

This experiment was conducted using a similar setup and characteristics as the previous one. But the initial state x action table we used here is more descriptive than the one used in previous experiment. The initial state x action table is given listed in Table 4.2

s_1 = Ball is at far left

s_2 = Ball is at far right

s_3 = Ball is at far and middle

s_4 = Ball is right in front and heading straight towards goalkeeper

s_5 = Ball is located at close left corner

s_6 = Ball is located at close right corner

s_7 = Ball is located and close in front

a_1 = Go right

a_2 = Stay where you are

a_3 = Go left

Table 4.2: Initial State x Action Table

	A	C	T	I	O	N
			a_1	a_2	a_3	a_4
S		s_1	0	0	0	0
T		s_2	0	0	0	0
A		s_3	0	0	0	0
T		s_4	0	0	0	0
E		s_5	0	0	0	0
		s_6	0	0	0	0
		s_7	0	0	0	0

According to the code, the ball will be considered at far side when it is at least 0.52 meter away from the goalkeeping agent. Figure 4.4 states the actual situation with space quantized according to the state x action table. The working formula to update the table is displayed in Equation 4.5.

$$V(S_t) \longleftarrow V(S_t) + \alpha * [R_{t+1} + \gamma * V(S_{t+1}) - V(S_t)] \tag{4.5}$$

R_{t+1} = Reward observed at time t+1

S_t = State visited at time t

R_t = Reward after time t

α = Step size

γ = Constant parameter

This formula is evaluated using the value of $\alpha = 0.9$ and $\gamma = 0.8$ and stated in Equation 4.6.

$$V(S_t) \longleftarrow 0.9 * R_{t+1} + \gamma V(S_{t+1}) - 0.1 * V(S_t) \tag{4.6}$$

So, the following values of the parameters are used for the system where applied.

R = 100 (The amount of reward when the task is accomplished)

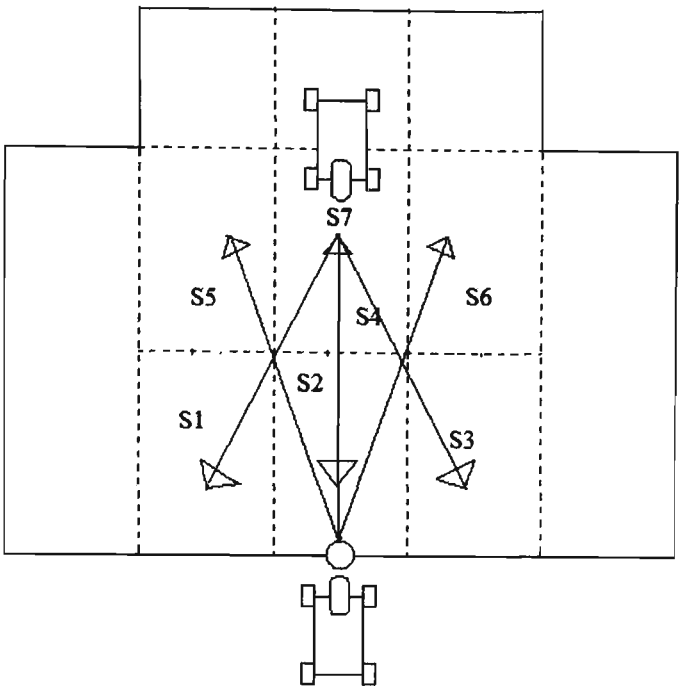

Figure 4.6: A goalkeeper's view for the experiment with 4x7 state x action table

R = 0 (Rather than 100) $\gamma = 0.8$

$\alpha = 0.9$

And the formula becomes as stated in Figure 4.7

$$V(S_t) \longleftarrow 0.9 * R_{t+1} + \gamma * V(S_{t+1}) - 0.1 * V(S_t) \tag{4.7}$$

The data obtained from this experiment is displayed in Chapter 5.

4.6 The two attacker experiment using available knowledge base from the goalkeeping experiment with 4x7 state x action table

4.6.1 Aim

The aim of this experiment was to test the goalkeeping skills obtained in the previous experiment against two attackers. The point to be noted here is that the training was

completed against one attacker. However, in this experiment, two attackers were introduced to check whether that experience works against those attackers or not.

4.6.2 Setup

We used three robots, a pink ball and a penalty area with dimensions set by official Robocup authorities for the purpose.

4.6.3 Experimental Method

This experiment was conducted inside the penalty area and with two attackers. We conducted four different experiments with the different positions of those two attackers. The attackers were static in all four cases. One of them was located at penalty line (Far region) and the other is in the near region. The attacker at far region pass the ball towards its team mate located at near region and the second player took a shot towards goal using the running pass from his team mate. The first two test cases are explained with Figure 4.7 and Figure 4.8. There the second attacker directed the ball towards the opposite direction it was destined for.

The other two setups are displayed in Figure 4.9 and Figure 4.10. Here the attacker could take two different shots. However, most of the time it was taking shots towards the far end and only a small number of shots were directed towards the middle position.

4.7 An experiment to find out the efficiency of Upenn'03 code to create a benchmark for one attacker goalkeeping experiment using Q-learning

4.7.1 Aim

A bench mark result was created through this experiment using the logic from UPenn'03 code base.

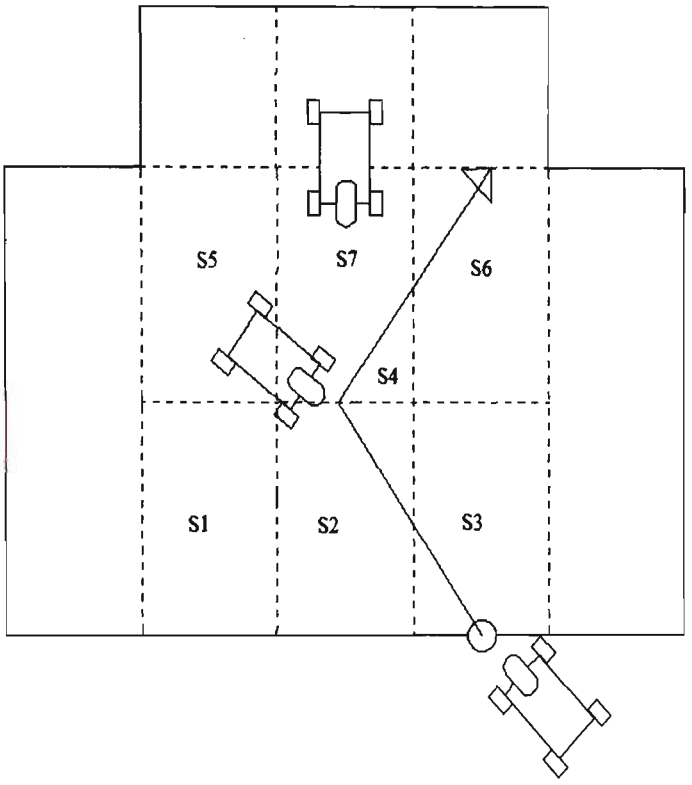

Figure 4.7: First setup for two attackers experiment

4.7.2 Setup

There were two AIBOs, a pink ball and a penalty area with dimensions set by official Robocup authorities shown in Figure 4.1 used in this experiment.

4.7.3 Experimental Method

This experiment was conducted using similar setup inside a penalty area created with the field specification stated in Figure 4.1. Altogether, three major high level decisions were found in Upenn'03 code base for the goalkeeping task:

- Find the ball if it is not within $0.8m$
- Move to intercept the ball on the left or right side if it is within $0.8m$
- Block the ball if it is found in front (Less than or equal to $0.03m$)

The data obtained from this experiment are displayed in the next chapter.

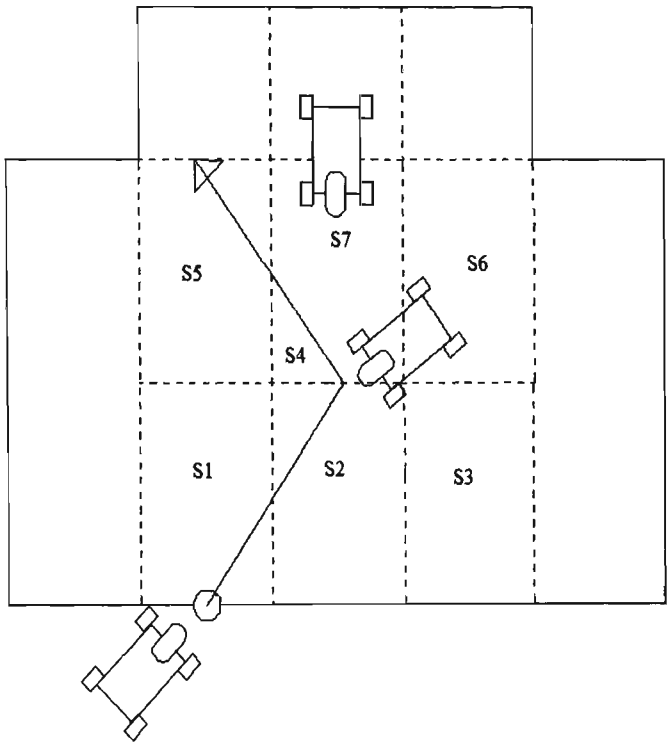

Figure 4.8: Second setup for two attackers experiment

4.8 An experiment to create benchmark for two attackers experiment using Upenn'03 code

4.8.1 Aim

The aim of this experiment was to create a bench mark result for the two attackers experiment by using a hand coded goal keeping set of actions.

4.8.2 Setup

Three AIBOs, a pink ball and a penalty area with dimensions set by official Robocup authorities from Figure 4.1 were used here.

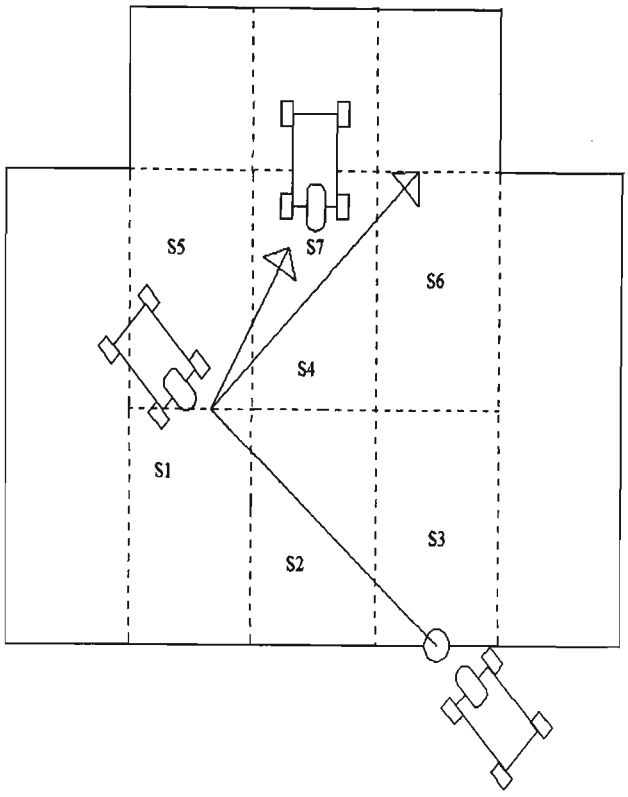

Figure 4.9: Third setup for two attackers experiment

4.8.3 Experimental Method

This experiment was conducted using a similar setup to that described in Figure 4.3 and Figure 4.4 for one attacker experiment and using Figures 4.7, 4.8, 4.9, 4.10 for two attackers experiments. Three major high level decisions were used in those experiments to block the shots using the Upenn'03 code logic described above. The data obtained from this experiment is displayed in the next chapter.

4.9 Summary

So far the working principle and environmental setup of all experiments has been described in detail in the previous sections. The next chapter will describe the results obtained from these experiments.

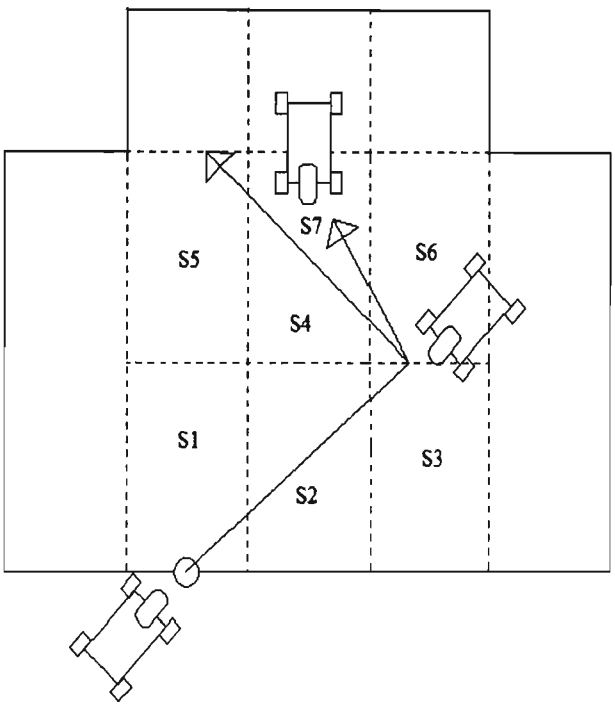

Figure 4.10: Fourth setup for two attackers experiment

Chapter 5

Experimental results

5.1 Introduction

The following four sections describe, the results in detail, achieved for the experiments described in Chapter 4.

5.2 Results for the ball distance measurement experiment using nose camera

5.2.1 The ball.ratio values at several distances using nose camera

The following table consists of ball.ratio (discussed in Chapter 4) values at different distances.

Table 5.1: Experimental readings from nose camera

Actual ball distance	**Maximum ball.ratio value**	**Average ball.ratio value**	**Minimum ball.ratio value**
3	0.6088	0.6050	0.6013

Continued on next page

Table 5.1 – continued from previous page

Actual ball distance	Maximum ball.ratio value	Average ball.ratio value	Minimum ball.ratio value
6	0.5549	0.5476	0.5338
9	0.3699	0.3610	0.3533
12	0.2696	0.2673	0.265
15	0.2	0.1957	0.1902
18	0.1431	0.1371	0.1324
21	0.1058	0.1022	0.0994
24	0.0831	0.0809	0.0792
27	0.0672	0.0652	0.0636
30	0.0548	0.0541	0.053
33	0.0466	0.0443	0.0433
36	0.0402	0.0401	0.0401
39	0.04	0.0391	0.0387
42	0.0323	0.0321	0.0315
45	0.029	0.0275	0.0253
48	0.0246	0.0245	0.0245
51	0.0215	0.0211	0.0209
54	0.0191	0.0185	0.0182
57	0.017	0.0168	0.0165
60	0.0151	0.0149	0.0149
63	0.0135	0.0134	0.0134
66	0.0125	0.0123	0.0122
69	0.0111	0.0111	0.011
72	0.0105	0.0104	0.0103
75	0.0087	0.0085	0.0084
78	0.0081	0.0080	0.0079

Continued on next page

• Table 5.1 – continued from previous page

Actual ball distance	Maximum ball.ratio value	Average ball.ratio value	Minimum ball.ratio value
81	0.0077	0.0075	0.0074
84	0.0071	0.0070	0.007
87	0.0065	0.0065	0.0064
90	0.0061	0.0061	0.006
93	0.0058	0.0057	0.0056
96	0.0052	0.0053	0.0054
99	0.005	0.0048	0.0048

5.2.2 Conclusion

This distance measurement method was evaluated against the actual distance table. The nose camera yielded an average accuracy of 97% over a ball distance range of 3cm to 100cm. The average readings from the table stated above were tested with the ball placed at 33 fixed points with given distance. The range could be increased with readings taken above 99cm. However, due to the speed of the ball and the limited maneuverability of the AIBO, it would not be necessary to measure the distance of the ball in Robosoccer over 1meter distance. The next section reveals the readings taken using IR sensors.

5.3 Results for the ball distance measurement experiment using IR sensors

5.3.1 The ball distance measured by near and far IR sensors

Table 5.2: Experimental readings from IR sensors

Actual ball distance (cm)	Near IR Sensor value (cm)	Far IR Sensor value (cm)
3	10.44	20.00
6	12.26	20.00
9	14.53	20.00
12	15.49	21.34
15	16.32	27.06
18	18.31	30.94
21	21.25	31.59
24	19.31	33.51
27	21.47	37.93
30	22.86	37.00
33	33.84	48.59
36	30.75	51.93
39	38.80	55.76
42	36.15	61.59
45	34.39	66.85
48	32.79	68.36
51	31.33	70.76
54	47.52	88.97
57	47.67	93.79
60	38.80	87.05
63	40.28	85.81
66	50.00	87.05
69	50.00	93.79
72	50.00	115.23
75	50.00	119.80

Continued on next page

Table 5.2 – continued from previous page

Actual ball distance (cm)	Near IR Sensor value (cm)	Far IR Sensor value (cm)
78	50.00	120.45
81	50.00	150.00
84	50.00	150.00
87	50.00	150.00
90	50.00	150.00
93	50.00	150.00
96	50.00	150.00
99	50.00	150.00

5.3.2 Conclusion

The result displayed in Table 5.2 was tested using 15 given distance points while the ball was in a moving condition. The IR method reproduces the actual *ball distance* with 34.7% overall accuracy only. It is suspected that the IR reflections from adjacent objects of the ball are creating an interference with the IR signal reflected from the ball itself. This problem makes the IR measurement system invalid in any multi-agent environment, due to reflection from other team mates. There the reflection of IR from other players would make it worse. Furthermore there are two different IR sensors to choose for sensing near and far distance. The right sensor should be chosen at the correct moment to measure the accurate distance of the ball. However, the readings of two sensors are fuzzy at the range transition region. A decision making process is needed here to choose the right sensor at the transition period. So, for these reasons, we have decided to use the distance measurement system using nose camera for our experiments.

Sony released an SDK with the AIBO and there they had introduced a similar function to *ball.ratio*. However, no mapping between the ratio value and the mapping existed to find out the ball distance. Moreover, due to downward compatibility issues of the available

Linux OS, we have used URBI in stead of the Sony SDK. It had no in-built function for measurement the distance of an object from the robot. So the ball distance measurement experiment had to be performed using nose camera. At the end, we suggest that our technique could be used in calculating the speed and direction of a moving ball while the agent is in a static position. However, the readings changes tremendously within a very small amount of time if both the agent and the ball are moving simultaneously. A *less system hungry image stabilizing technique* can be used in order to resolve this problem.

5.4 The goalkeeping experiment with a 3x3 state x action table using Q-learning

5.4.1 Results

The state x action table converges perfectly in this experiment. The final form is described in Table 5.3.

Table 5.3: Goal keeping experiment with 3x3 state x action table

	A	C	T	I	O	N
				a_1	a_2	a_3
S			s_1	100	0	0
T			s_2	0	100	0
A			s_3	0	0	100
T						
E						

5.4.2 Conclusion

This table indicates that if the ball goes left (state s_1), the goalkeeper goes left (action a_1), stays in the middle (action a_2) if the ball moves towards middle (state s_2) and moves

right (action a_3) if the shot is directed to the right side (state s_3). It is a single stage learning, and so the agent learns with a single successful movement in each case. The efficiency of the goalkeeper would be 100 percent using the knowledge database using the final state x action table. In practice, we observed that our robot was unable to save 8 shots only during evaluation due to the unpredictable trajectory of the incoming ball caused by natural damping forces. Actually the ball took an unexpected turn at the eleventh hour and so the robot was unable to intercept it due to lack of maneuverabilities issues, although it chose the right action in the right situation.

The above results establish the truth that Q-learning technique worked as well as hand coded technique in defending penalty shots. Although, one hundred percent success should be achieved, we have almost 83.3% of average success in this experiment. However, this experiment was turned out to be a trial and error method due to the size of the state x action table. So, we have split one single state into two simpler states in the next experiment. First of all, this approach adds the lost essence of temporal difference technique of the Q-learning method. Secondly, it made the learning more effective and enables what to perform more accurate actions in an unknown situation.

5.5 The Goalkeeping experiment with one attacker using a 4x7 state x action table using Q-learning

5.5.1 Results

The second experiment was devised due to the absence of a back propagation feature in this experiment. This experiment consists of a 4x7 state x action table and the quantization of the spaces is more accurate in this case. The final score is displayed and listed in Table 5.4.

5.5.2 Conclusion

The experimental results showed that the agent has learned the knowledge of goalkeeping using Q-learning from zero experience and without human interference. The agent overall

Table 5.4: Expected final State x action Table

	A	C	T	I	O	N
				a_1	a_2	a_3
S			s_1	0	64	0
T			s_2	0	64	0
A			s_3	0	64	0
T			s_4	0	100	0
E			s_5	0	0	80
			s_6	80	0	0
			s_7	0	80	0

achieved 80.7 percent success in goalkeeping after completion of the training.

5.5.3 Experimental results using Upenn'03 code

The one-attacker Q-learning experiment with 4x7 state x action table was an extension of the previous Q-learning experiment which was designed with 3x3 state x action table. We have used Upenn'03 code base as a base line code to compare the efficiency of the goalkeeping training using Q-learning. A benchmark was produced that shows an overall success of 79.9% achieved by Upenn'03.

It was noticed that the Q-learning experiment with 4x7 state x action table yielded a success rate of 80.7 percent after completing the training. This success rate is similar to the result obtained from the experiment conducted with Upenn'03 code. However, unlike the hand-coding, the agent started from zero knowledge and ended up with virtually 100 percent efficiency.

5.6 Results from the third experiment with two attackers using the knowledge base obtained in second Q-learning experiment and Upenn'code

The results of the four cases are described in Table 5.5 using both Q-learning and Upenn'03 code base. Altogether 20 shots were used to evaluate each and every separate attacker formation.

Table 5.5: Score boards of two attacker experiment

	Success rate achieved by UPen'03 code base	Success rate achieved by Q-Learning
Attacker Arrangement 1	15	15
Attacker Arrangement 2	16	14
Attacker Arrangement 3	14	15
Attacker Arrangement 4	17	14

5.6.1 Conclusion

The experimental result shows that the achievement of Q-learning and Upenn'03 is more or less similar for each formation of the attackers.

5.7 Summary

The experimental results of the ball distance measurement experiment, Q-learning experiments and benchmark experiments led us to compare the effect of the approach stated in the methodology chapter. The next chapter focuses on the comparison and a detailed discussion of the comparison between the Q-learning approach and benchmark experiments.

Chapter 6

Discussion

6.1 Research aim

The research question in this thesis was whether a particular reinforcement learning (RL) technique can be used to achieve comparable results with standard hand coding for a decision making task in Robosoccer or not. In particular, the goalkeeping problem was chosen and Q-learning was used as the chosen RL algorithm, described in Chapter 3.

Hand coding specifies input-output state x action pairs to carry out a specific task or a group of tasks; there a programmer has to define the environment specifically to the agent as well. On the other hand, a programmer has to define only the environment and an empty state x action table, using reinforcement learning. No specific input-output pair is defined in advance to perform the action in a given situation. The agent learns the right action for a novel situation using training exercises and a reward system. After completion, the robot uses the obtained experience to complete tasks for acting in the future. Q-learning is such a process and it also enables an agent to learn while performing. The state x action table provides the necessary solution to a given situation. The table contains some basic situations which could be combined to represent more complex ones. Accordingly, the agent produces a series of actions to suit the anticipated situation optimally.

Altogether, three RL experiments were performed to find out an answer to the research question. The experimental results are described in Chapter 5. However, before commencing with the RL experiments, another experiment was performed to find out a better

method for measuring the distance between the ball and the AIBO. The outcome of the experiment was an improvement over the existing method of URBI that uses IR sensors for sensing the ball distance [73].

6.1.1 The outcomes of the experiment designed to measure the distance between the ball and robot

The nose camera method was evaluated using Table 5.2 obtained from the experiment and the two point equation described in Equation 4.2. This test revealed that an overall 97% accuracy was achieved in regenerating the distance using that method. This evaluation test was performed while only the ball was moving an the robot was at rest. The same condition produced 35% accuracy in regeneration of the distance using IR sensors. The problems that prevent the IR sensors from producing a clear distance measurement are described below.

- There are two different IR sensors available for *near* and *far* distance measurement. The near distance sensor measures effectively from 3cm to 20cm and the far sensor takes care of distances from 20cm to 150cm. The border between the two distance sensors is often mixed up during real life operation. So, at that point, it usually becomes critical to decide which sensor data should be used. The experiments were designed in such a way that a point at 20cm distance from the goalkeeper lies in the *near* region. The goalkeeper starts moving whenever the ball comes within the *near* region. As a result it is highly recommended to calculate the distance properly at this point to anticipate the state correctly in order to save the goal. So, a fuzzy output at this point by IR sensors is not acceptable for the experiments.

- IR rays reflect from all the other sources around the ball. These could be the carpet around the ball, other players roaming around or the boundary wall of the fields. The rays reflected from these object creates interference and produce confused results out of the IR sensors.

Considering these two major problems, the IR sensors were not used to measure the distance of the ball in both hand coding and Q-learning experiments. Separate approach is

devised here to solve the problem using the nose camera and pink ball detection algorithm. The following points describe the logic behind that proposed technique.

- The pink ball detection algorithm enables the AIBO to find the pink colored ball and look at its center.
- The AIBO has a 2D vision system which represents the ball as a circle within the field. So if it goes away the circle will appear smaller and vice versa. This was the logic applied to find out the actual distance between the nose camera and the pink ball. After performing ball distance measurement experiment a relationship table was established. The table indicates the amount of area the ball occupied in the the robot's 2D vision and the corresponding ball distance.

It was concluded that the AIBO measured the distance of the moving ball with an acceptable degree of accuracy as described at the beginning of this section. The Q-learning therefore extensively used the ball measurement technique. The next section describes the experiments, conducted using Q-learning.

6.1.2 The outcomes from goalkeeping experiments using Q-learning

6.1.2.1 The achievement from the one attacker experiment with 3x3 state x action table

This experiment was designed using a 3x3 state x action table. The Q-learning learning was accomplished using a single action form the starting position. The point to be noted here is that two basic actions were combined to form a single action for the goalkeeper as displayed in Table 3.7. These two actions were moving and blocking the ball. So, the combined action was called *move and block/stop*. Finally, the state x action table ended up with a converged matrix. The goalkeeper followed the table after completing the training against the three different shots, it was trained with. All three different shots were tested with the final state x action table and an average of 83.33% success was achieved by the goalkeeper. The limitations of maneuverability stopped the AIBO from achieving complete success. However, the single step learning prevented the learning agent from

using the temporal difference feature of Q-learning. As a result, the second experiment was devised with a more generalized state x action table.

6.1.2.2 The achievement from the other one attacker experiment with 4x7 state x action table

This experiment was carried out with a state x action table displayed in Figure 3.8 and it showed convergence after completion of the training sessions. The point to be noted here is that the agent started without any previous knowledge taken from previous 3x3 state x action based experiments. The state x action table permitted more than single step learning to block the ball. It means that the reward was propagated through few steps backward from the final state. Thus the temporal difference technique involved in this experiment. Finally, it produced a stable output matrix in the form of the defined state x action table.

The research question we started with was whether a basic Q-learning technique is able to take decisions as well as standard hand coded technique in the Robosoccer environment or not. In this case, the goalkeeping problem was solved using Q-learning. The goal of the thesis was to enable the robot to take an optimum decision using the acquired knowledge base. The final outcome should resembles the real life goalkeeping scenario hypothetically. Usually, in real life soccer the human goalkeeper observes the ball from a safe distance and at close proximity he may choose to move and intercept the ball. He must stop the ball if it comes in front.

The goalkeeper in our experiment learned to use similar logic to defend the goal. Primarily three different positions were defined according to the space quantization for the experiment designed with 3x3 state x action table. These are denoted as different states, namely, *far region*, *near region* and *right in front*. *Far region* and *near region* were further divided into six more regions for more precision training in the experiment consist of 4x7 state x action table. So, altogether 7 states (Table 3.8) were defined. The final state x action table reveals that the agent *stays* at its starting position (middle of the goal line) when the ball is located at the *far region*. The goalkeeper starts moving when the ball comes inside the *near region*. Whenever the agent detects the ball is detected *in front*, the agent stops there to block it. It moves to the left or right side according to the movement

of the pink ball when it is found in *near region* only. These actions hypothetically resemble to that of a human goalkeeper in a real life soccer game, too.

It was noticed that a few shots took an unexpected turn due to friction and spin of the ball at the eleventh hour during the evaluation of the acquired knowledge. The goalkeeper tracked that change efficiently using the training data and tried to move in the direction of the ball. However, sometimes it was a failure due to the physical limitations of the AIBO. Altogether, it saved 81.7% of all the shots out of the 50 shots upon completing the training. It clearly portrays that the learning system had worked properly in the decision making process irrespective of the few failures due to physical limitations.

6.1.2.3 The achievement from the two attacker experiment

The final state x action table acquired from the one attacker experiment with 4x7 state x action table, was used in this experiment. It was designed to check the efficiency of goalkeeper using two attackers. The design of different kinds of this experiment are shown in Chapter 4. Two attackers took shots from different points and distances. The first attacker took shot into the second attacker and the second attacker pushed the flying ball towards the goal. The first setup of the two attacker experiment (Figure 4.5 and 4.6) reveals that the first attacker is situated at far middle region while the second attacker is in near left or in near right. This near attacker took a direct shot using the pass from the first attacker. In the other two setup the first attacker was at far right or far left, whereas the second attacker was at the near middle position. Even at this time, the goalkeeper started from the middle position and followed as per the real time decision making process using Q-learning to save the game.

These complex situations introduced with the two attacker problem added a little hassle to the goalkeeper. Throughout the time it followed the *pink ball* only and not the attackers due to the distance measurement code shown in Table 3.8. However, the ball was rapidly changing direction from the point of second attacker. So, the goalkeeper considered sweep changes in its movement during the operation. The second attacker caused a swift change in the ball direction and also added additional speed to it. It was found that the Q-learning system took the right action at this time. So, the goalkeeper needed to act faster here than the previous experiment. However, the physical limitation of AIBO did

not allow it to do so and a few more shots went into the goal. The scoreboard of the four separate attacker arrangements for the two attacker experiment is given in Table 5.5.

The average achievement of these four experiments is 15.5 saves out of 20 shots for hand coded technique by Upenn'03 and 14.5 saves by Q-learning technique out of 50 shots. It can be concluded from this point that both the hand coding and learning method produced a similar achievement in two attacker experiment.

It was found that the movement of the ball was completely controlling the decision making process of AIBO. According to real life soccer, the goalkeeper could start up from either left or right side according to the state of incoming ball. However, our table does not permit the goalkeeper to do so. Moreover, in Robosoccer it is better to stay in the middle of the goal to get a complete 180^o wide view in both sides using minimum amount of head pan. So, in all the cases, the goalkeeper started from the middle of the goal line and then acted according to the training data thereafter. Finally, the overall achievements of the RL experiments are stated below.

6.1.3 The results achieved with AIBO using Upenn code

The Upenn code describes the field information using a world model to the goalkeeper. It uses a polar coordinate system to find the distance and angle of the ball. Actually the hand-coded technique in this case asks the goal keeper to follow different actions at different ball distances. According to the decision making file written in MATLAB, four different activities are described for the goalkeeper:

- Search for the ball if it is not in the visual range
- Move forward if the ball is coming straight and not in close proximity
- Move side ways to the ball if the ball is not too far and coming in from either side
- Stop it using the torso if the ball is detected just in front

This logic was applied in a similar environment for three different RL experiments stated in the experiment section. The success rate for one attacker experiment using Upenn code is 79.9% on average and that of Q-learning is 80.7%. The outcome of the multi-attacker experiment is described in Table 5.5.

6.1.4 The performance of Q-learning over the hand coding

- In the first two, one attacker experiments, the goalkeeper (AIBO) updated the state x action table through the training session and produced a stable matrix without human supervision. It properly used this data afterwards to block the goal accordingly.

- The knowledge database from the experiment designed with 4x7 size state x action table was used in the two attacker experiment. It was found that the agent was saving the goal perfectly after completing the training. Using it, AIBO performed efficiently against two attackers with the same knowledge base learned against one attacker and a 4x7 state x action table.

The physical limitations of AIBO put a limit to its actions in some situations during goalkeeping. However, the outputs using hand coding and Q-learning were similar in all the experimental trials. It proves that a particular RL technique (Q-learning) can perform as well as the hand coding method.

Chapter 7

Conclusion

7.1 The research question and its origin

The ultimate goal of Robosoccer is to prepare a team of humanoids to defeat human world soccer champions [24]. It demands a robot (humanoid) to be equipped with human like maneuverability, image and pattern recognition systems and thinking and decision making capacity. The research aim in this thesis was to test a specific artificial intelligence method, namely Reinforcement learning (RL), against a fixed hand coding on a particular robocup based problem. We have applied a particular RL technique, Q-learning, in the AIBO to address the research question. The research question we started with was "Whether a basic RL algorithm can perform as well as hand coding/input-output pairs to solve the goalkeeping problem?" The performance of Q-learning was compared with a baseline created by Upenn'03 hand-coding technique using the goalkeeping problem. The hand-coding technique specifies an input versus output table for the agent to carry out a task. In contrast with that, the Q-learning helps the agent to learn how to perform a task using a state x action table. The comparison of these two approaches is used to determine the superiority of a learning process over the fixed hand-coding technique. However, before proceeding with the experiments, we developed the reason for choosing Q-learning as the proposed method.

7.2 Methodology

The baseline experiment was completed using the Upenn'03 Robosoccer code base given in Chapter 3. As discussed, a limited number of input versus output pairs were used there. The goalkeeper (AIBO) block the goal against one and two attackers, using this code. In contrast with that, this thesis focused on a learning method which learns to block the goal in similar situations using a learning method. Reinforcement Learning was used for this purpose and a suitable RL method (Q-learning) was ultimately chosen to perform the goalkeeping experiments. We selected Q-learning for its efficiency in a simulation based maze learning experiment as discussed in Chapter 3. The similarity between the maze learning experiment and the goalkeeping experiment exists in the form of the size of the state x action tables and the information about the environment used in both cases. The maze learning used a 6x6 size table whereas the goalkeeping experiment used a 4x7 size table and both the experiments was carried out in a noiseless environment. Q-learning and SARSA are two similar algorithms of RL with a few difference in action selection policy. So we tested both of them using the maze learning environment. The output showed that Q-learning produced a perfectly converged output using approximately 2500 training exercises whereas SARSA produced a non-converged output using all 50000 training exercises each time.

An AIBO runs on battery and its maximum duration is 1.5hr. So, we chose a method which can provide a relatively faster learning rate. It was noticed that Q-learning considered all the parallel routes in maze learning with equal probabilities. SARSA failed in this regard and assigned different weights to parallel routes. This indicates that Q-learning reveals all the similar possibilities using less number of training epoch that SARSA. However SARSA failed to produce a converged matrix even after using $50,000$ training epochs. As a result, finally we used Q-learning to perform the goalkeeping experiment.

7.3 The experiments

The hand-coding experiments were performed at the beginning to set up a base line result for the Q-learning method. The experimental setup was made using an official

Robosoccer field layout displayed in Figure 4.1. One attacker took three different shots from three different points situated on the penalty line. Two attackers were used in another experiment against the goalkeeper. In this experiment, one attacker took a direct shot towards the other player and the other player pushed the ball into the goal. The knowledge obtained from the one attacker experiment was used in the two attacker scenario.

Subsequently, the Q-learning technique was also tested using similar experiments. We started with the one attacker problem in this case. There the attacker took shots from three positions as well. A 3 x 3 sized state x action table was used to record the learning experience. All available space in front of the goalkeeper was quantized in three situations and three different basic actions were chosen for it. The agent learned to take the right decision at the right moment to save the goal in the training exercises. However, the essence of back propagation of the reward was lost in this small size state x action table due to its single step learning process. So it was decided to break both the situations and actions into simpler forms and proceed with another experiment with one attacker only. A state x action table with 7x4 size was defined for it. All the available spaces were quantized into seven different areas based on different possible positions of the pink ball. One more action was added to the table which instructs to stop the robot immediately in order to block the ball if it is detected in front. This experiment also showed that the given agent had learned successfully to block the shots using the given training exercises. In the second attacker experiment, we put two attackers against the goalkeeper while the goalkeeper used the knowledge data base obtained from the experiment contains 4x7 state x action table. The first attacker took a shot towards the second and it pushed the ball towards the goal. The goalkeeper showed a satisfactory result in this experiment in performing the task.

7.4 A comparison between the hand-coding and Q-learning experiment results

The success rate of Q-learning in the one attacker experiment with 4x7 state x action table was similar to that of Upenn'03 code base. Moreover, the same database was applied

to the two attacker problem, both Q-learning and hand coding methods produced similar efficiency using four differently designed experiments as indicated in Chapter 5.

The Upenn'03 code base supplies a code for a hand coding technique. The term *Hand Coding* indicates that all the input versus output pairs are supplied by the programmer. However, this approach does not provide any instruction for an unknown situation that an agent may encounter during action. So, an unknown situation may result in an undesirable action.

The Q-learning method allows the programmer to define the environment and some basic actions only. However, the agent has to learn the correct action in a given situation. The agent, in this thesis, focused on the position and distance of the ball. Using the one attacker experiment the agent successfully learned to block the ball using two different state x action tables. To prove the extent of the acquired skill, we decided to continue the third experiment with the knowledge database obtained from the previous experiments. It was found that the goalkeeper took the right decisions in order to block the incoming shots, even against two attackers. Moreover, it was pointed out that Q-learning allowed the agent to work satisfactorily in a novel situation.

To support this idea, it should be mentioned here that AIBO tracked and followed all the shots properly even if the ball took an unexpected turn, described in Chapter 6. Due to the spinning action of the ball and friction of the carpet, some of the shots took such turns. These sudden state transitions represented new situations for the AIBO which were not mentioned in the state x action table. The goalkeeper first developed the database using Q-learning and managed these new situations using the learned experience. So, it can be concluded that the agent has learned to take the right decision at the right time, independently.

So, Q-learning proved better than hand coding due to the fact that the goalkeeper was able to accomplish the task from the beginning and worked out a way to deal with a novel situation.

7.5 The contributions made in this thesis

So far, humans have created a number of different robotic models. These are humanoids that also include replicas of other animals such as dog, cat, fish and so on. However, even a living dog still supersedes its existing robot replica in two major ways. These are maneuverability and decision making skills. In some cases, robotic arms and other robotic machine parts can deliver heavier duty output and more precision service than living bodies. Those can be proven worthy in hazardous areas such as a nuclear reactor, under water structures, space stations and so on. However, in non-hazardous and day to day jobs, a living animal is still preferred over robots. The examples of police dogs and/or guide dogs can be considered in this case. This is due to the fact that an artificial robot is not yet smart enough to replace a living animal.

In this thesis we used Q-learning to make a robot solve a novel situation without providing specific knowledge about it. The Q-learning technique was tested using Robosoccer as a test bed. The goal keeping act was chosen for the purpose and finally it was found that the agent had successfully performed the task. We trained it against one attacker to save the goal. Using that one attacker knowledge database, it performed as well as hand coding technique against two attackers. It was found that the goalkeeper could not only learn to block the ball using the given situation, but also it successfully applied the knowledge database to block a few unexpected state transitions. This shows that Q-learning taught the AIBO to act independently, like living animals do. Further development possibilities using complex learning features and ball distance measurement techniques are discussed in the next section.

7.6 Future work

A brief description is given below for the possible improvements on the ball distance measurement technique. It is also suggested that the outcome of the Q-learning algorithm could be improved by adding an approximation technique on top of it.

Possible improvement for ball distance measurement experiment

We had set up a separate experiment for measuring the distance of the pink ball from

the robot before performing Q-learning. The method was designed for a scenario where the AIBO was standing still and the ball was either static or moving. The final technique was evaluated while both the objects were moving. The result was not at all acceptable for activities for playing soccer. A huge amount of fluctuation of the captured images made the result impossible to use in ball distance measurement technique. After a detailed investigation, it was concluded that jerks were introduced in the picture due to the movement of the nose camera while the AIBO was in motion. This problem could be addressed using some low overhead digital image stabilization methods or an approximation function involving the different dynamic parameters responsible for the jerks in the image.

Recommended upgrade for Q-learning experiment

The proposed learning technique, Q-learning, involves updating a predefined table and managing it during the experiment. It means the uses of a large size table can slow down the complete process in time. A relatively small table is used with 4 columns and 7 rows in this thesis. However, a multi player game of Robosoccer involves a lot more basic states and actions which in turn increase the size of the table and so the complexity of the learning process. An approximation technique could be useful here to solve this issue. Thus a policy could be setup after completion of a particular learning process which would be useful in ignoring the useless values in the table. The use of computational resources would be far less in that case. Those resources could be allocated to other accompanying functions such as ball distance measurement to increase the efficiency of the complete process. Some of the existing work was already focused on a policy gradient [74] function approximation technique to maximize the efficiency of the existing walking gait [75]. A similar approach can be useful for the approximation of the decision making process as well.

Appendix-A

This chapter is divided into two halves. First half describes the method of using wireless access point to communicate with AIBO remotely. The next part consists of the distance measurement code that has been developed during the research.

A-1 Communication technique with AIBO using wireless

The model of AIBO comes with an in-built wireless Local Area Network (LAN) card with 802.11*b* protocol enabled. It enables the AIBO to connect with another similar device at 2.4GHz radio frequency spectrum, typically with 4.5Mbps speed. It allows simultaneous transfer of video and other sensory data back and forth between the robot and another wireless device, mostly a wireless gateway. There are two differen ways to connect an AIBO to one or more PCs at a time.

- Setup a connection using a wireless gateway to connect AIBO and other PCs at a time.
- Setup a single peer to peer connection between a wireless LAN enabled PC and AIBO

Figure A.1 shows the schematic representation of the points stated above.

There are two different files that exist to configure the onboard LAN. These are WLANDFLT.TXT and WLANCONF.TXT. The first one is the default configuration and should not be altered for safety reasons. The second file contains the following fields to

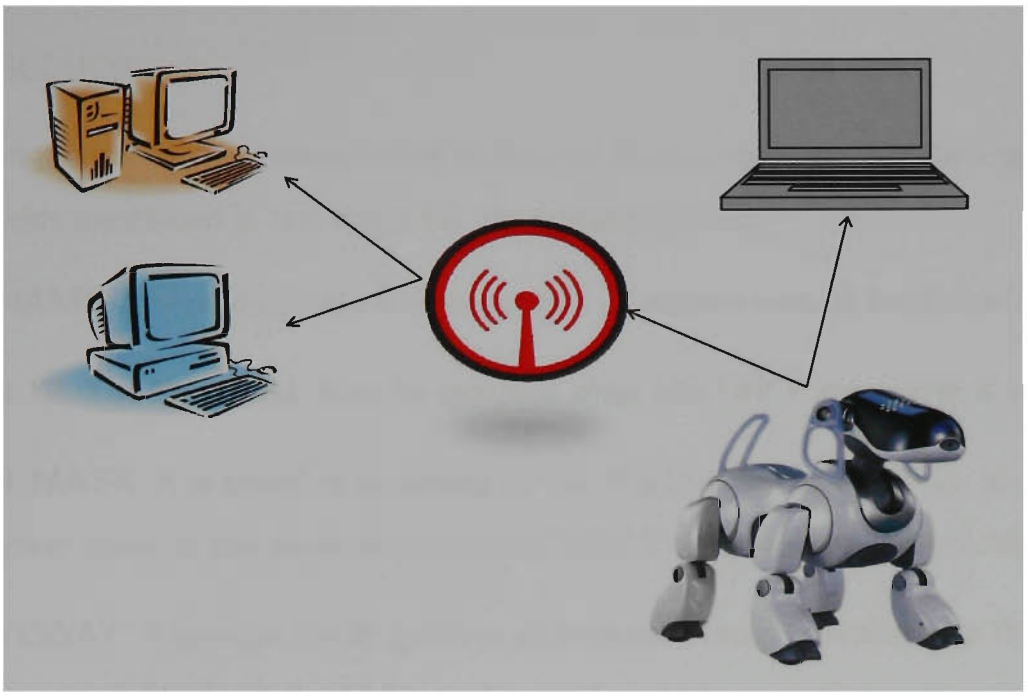

Figure A.1: Different methods of communication with AIBO

edit the IP address and other options for establishing a connection over wireless LAN with AIBO. The parameters of WLANCONF.TXT are listed below.

- HOSTNAME = AIBO
- ETHER_IP = 192.168.1.6
- ETHER_MASK = 255.255.255.0
- IP_GATEWAY = 192.168.1.1
- ESSID = $AIBO1$
- #WEPKEY = $ABCD1234$
- #WEPENABLE = 1
- APMODE = 2
- CHANNEL = 3
- #DNS_SERVER_1 = 192.168.1.1

- #DNS_DEFDNAME = ouraibo.net
- USEDHCP = 0

The configuration written above refers to the one of our robots used in the experiments. Those fields mentioned in the above list are explained below.

HOSTNAME It should contain 8 alphanumeric characters with at least one letter.

ETHER_IP The IP address must be specified when USEDHCP parameter is set to 0.

ETHER_MASK It is specified according to the IP address class. One can use different subnet mask at this point in contrast to that of other device used in LAN.

IP_GATEWAY It specifies the IP gateway address and if none is available on the network then the *ETHER_IP* should be used instead.

ESSID It is the name of the wireless network. Up to 32 alphanumeric character could be used for the purpose.

WEPKEY Wep key is the set of characters used as a passkey to the wireless network. It may be called as encryption characters to the network as well. Only WEP64(40) bit or WEP128(104) bit wep key could be defined in this case. An alphanumeric character is 8 bit long and the length of a hexadecimal character is 4 bit. Accordingly, 5 alphanumeric characters or 10hexadecimal characters must be used to define a WEP64 key and 13 alphanumeric characters or 26 hexadecimal characters for WEP128 key. The hex characters defines must precedes a 0x and only $0-9$,A-F and a-f characters could be used after that with appropriate length set for two different version of WEP key.

WEPENABLE The value of these parameters enables or disables the use of WEP key on the wireless network.

APMODE This parameter sets the mode of LAN into an AIBO. 0 defines Ad Hoc demo mode, 1 defines infrastructure mode and 2 defines automatic detection mode.

CHANNEL It defines a channel at Ad Hoc Demo mode only. Any value from 1 to 11 could be used here.

DNS_SERVER_1 It defines "Domain Name Server"(DNS) IP. The IP address of the wireless gateway is used in this place.

DNS_DEFDNAME It is used to specify the domain name and could be used instead of the ETHER_IP.

USEDHCP It defines whether "Dynamic Host Configuration Protocol"(DHCP) has to be used to automatically detect IP address or not.

The wireless setup used for this thesis is given below.

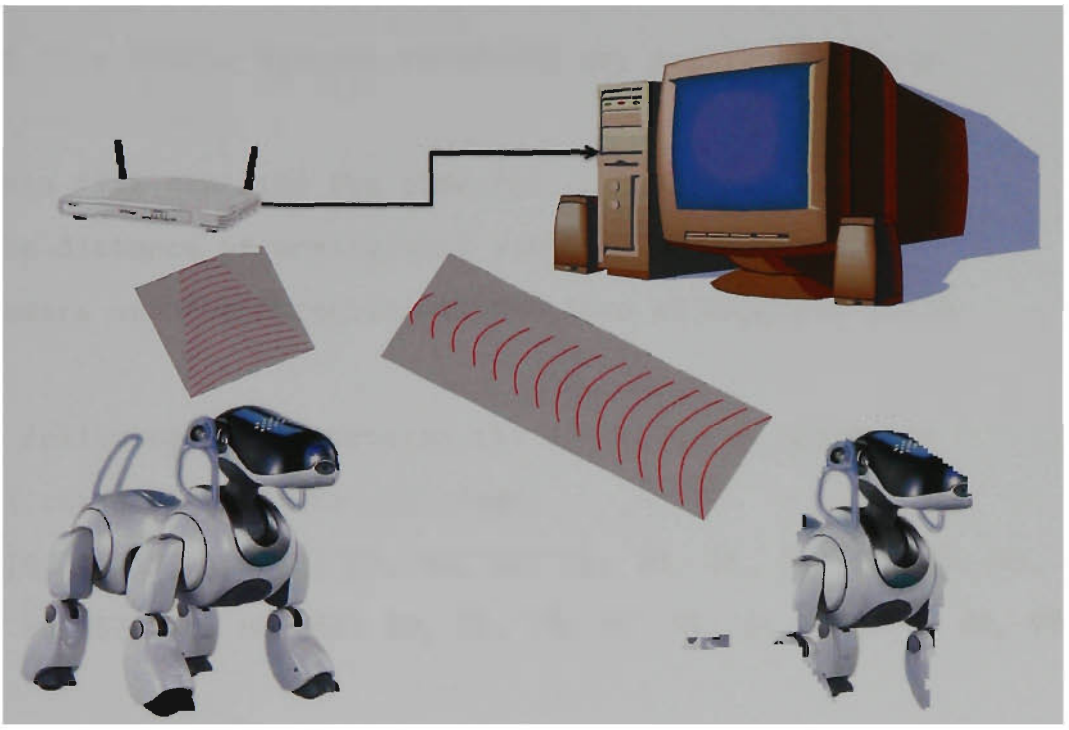

Figure A.2: Setting up multiple communication channel to different AIBO's using single PC

A-2 The code developed for the experiment

There is a major code used in the experiments for distance measurement, developed using URBI script. The code is

- Distance measurement and direction calculating process

A-2.1 Distance measurement code

The distance measurement code is made to work according to the supplied table. It consists of ball.ratio values at 33 points separated by 3cm distance from each other. The AIBO acquires ball.ratio values and the cod returns the corresponding distance from the table. The spline technique is used to find out the distance between the two adjacent points. The direction calculator returns the head pan angle only. The code and its description is as follows.

```
#This code is solely developed for the RL experiments
#of this thesis without referring any previous sources.

#This file contains the code for  calculating
#the distance of moving ball with respect to nose
#camera and the direction in the form of head pan angle.

#The following array contains the distances at which the
#ball.ratio values were collected.
x = [6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51,
     54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99];

#The next array contains the ball.ratio values collected
#at the distances mentioned in the previous array.
y = [0.6094,0.5476,0.361,0.2673,0.1957,0.1371,0.1022,0.0809,0.0652,0.0541,0.0443,
     0.0401,0.0391,0.0321,0.0274,0.0245,0.0211,0.0185,0.0168,0.0149,0.0134,0.0123
     0.0111,0.0104,0.0085,0.0080,0.0075,0.007,0.0065,0.0061,0.0057,0.0053,0.0048]

#Variable declaration
distn = 0;
a = b = c = d = e = u = m = t = 0;

#It is important to assign zero value to each and
```

```
#every variable in order to avoid any possible garbage value


if (isdef(robot.dist))
  {
  delete robot.dist;
  delete robot.pointcal;
  delete robot.stopdist;
  };



if (isdef(robot.pointcal))
  {
  delete robot.pointcal;
  };



if (isdef(robot.stopdist))
  {
  delete robot.stopdist;
  delete robot.dist;
  delete robot.pointcal;
  };



if (isdef(robot.showdist))
  {
  delete robot.showdist
  };



if (isdef(robot.showdir))
```

```
{
delete robot.showdir
};

#The next function introduces the two point function
#scheme for finding distance between two consecutive points



function pointcal(a,b,p)
  {
      pointcal:
      {
          m = (y(b)-y(a))/(x(b)-x(a));
          t = (p-y(a));
          u = t/m;
          d = u + x(a);
          return(d);
      }
  };

function robot.dist()
  {
      dist:
      {
      whenever(ball.visible)
          {
          a = ball.ratio;
          if(a $\leq0.6094$ \&\& a $\geq0.5476$) distn $=$ pointcal(1,2,a);
          if(a $< 0.5476$ \&\& a $\geq0.3610$) distn $=$ pointcal(2,3,a);
          if(a $< 0.2673$ \&\& a $\geq0.1957$) distn $=$ pointcal(3,4,a);
```

```
if(a $< 0.1957$ \&\& a $\geq0.1371$) distn $=$ pointcal(4,5,a);
if(a $< 0.1371$ \&\& a $\geq0.1022$) distn $=$ pointcal(5,6,a);
if(a $< 0.0809$ \&\& a $\geq0.0652$) distn $=$ pointcal(6,7,a);
if(a $< 0.0652$ \&\& a $\geq0.0541$) distn $=$ pointcal(7,8,a);
if(a $< 0.0541$ \&\& a $\geq0.0443$) distn $=$ pointcal(8,9,a);
if(a $< 0.0443$ \&\& a $\geq0.0401$) distn $=$ pointcal(9,10,a);
if(a $< 0.0401$ \&\& a $\geq0.0391$) distn $=$ pointcal(10,11,a);
if(a $< 0.0391$ \&\& a $\geq0.0321$) distn $=$ pointcal(11,12,a);
if(a $< 0.0321$ \&\& a $\geq0.0274$) distn $=$ pointcal(12,13,a);
if(a $< 0.0274$ \&\& a $\geq0.0245$) distn $=$ pointcal(13,14,a);
if(a $< 0.0245$ \&\& a $\geq0.0211$) distn $=$ pointcal(14,15,a);
if(a $< 0.0211$ \&\& a $\geq0.0185$) distn $=$ pointcal(15,16,a);
if(a $< 0.0185$ \&\& a $\geq0.0168$) distn $=$ pointcal(16,17,a);
if(a $< 0.0168$ \&\& a $\geq0.0149$) distn $=$ pointcal(17,18,a);
if(a $< 0.0149$ \&\& a $\geq0.0134$) distn $=$ pointcal(18,19,a);
if(a $< 0.0134$ \&\& a $\geq0.0123$) distn $=$ pointcal(19,20,a);
if(a $< 0.0123$ \&\& a $\geq0.0111$) distn $=$ pointcal(20,21,a);
if(a $< 0.0111$ \&\& a $\geq0.0104$) distn $=$ pointcal(21,22,a);
if(a $< 0.0104$ \&\& a $\geq0.0085$) distn $=$ pointcal(22,23,a);
if(a $< 0.0085$ \&\& a $\geq0.0080$) distn $=$ pointcal(23,24,a);
if(a $< 0.0080$ \&\& a $\geq0.0075$) distn $=$ pointcal(24,25,a);
if(a $< 0.0075$ \&\& a $\geq0.0070$) distn $=$ pointcal(25,26,a);
if(a $< 0.0070$ \&\& a $\geq0.0065$) distn $=$ pointcal(26,27,a);
if(a $< 0.0065$ \&\& a $\geq0.0061$) distn $=$ pointcal(27,28,a);
if(a $< 0.0061$ \&\& a $\geq0.0057$) distn $=$ pointcal(28,29,a);
if(a $< 0.0057$ \&\& a $\geq0.0063$) distn $=$ pointcal(29,30,a);
if(a $< 0.0063$ \&\& a $\geq0.0059$) distn $=$ pointcal(30,31,a);
if(a $< 0.0059$ \&\& a $\geq0.0053$) distn $=$ pointcal(31,32,a);
if(a $< 0.0053$ \&\& a $\geq0.0048$) distn $=$ pointcal(32,33,a);
if(a $< 0.0048$) distn = 99;
```

```
        b = headPan;
        };
    echo distn;
    } #End of "dist" table.
};


function robot.stopdist()
  {
  stopdist:
      {
      stop dist
      }
  };

function robot.showdist()
  {
  showdist:
      {
      c = robot.dist();
      }
  };


function robot.showdir()
  {
  echo head.Pan;
  };

  #The end of distance measurement code
```

Bibliography

[1] D. Cohen, H. Ooi, P. Vernaza, and D. Lee, "The university of pennsylvania robocup 2003 legged soccer team." [Online]. Available: http://fling.seas.upenn.edu/ robocup/wiki/index.php

[2] http://www.fredhouse.net/images/aibo.jpg.

[3] P. Stone, K. Dresner, P. Fidelman, N. Kohl, G. Kuhlmann, M. Sridharan, and D. Stronger, "The ut austin villa 2005 robocup four-legged team," The University of Texas at Austin, Department of Computer Sciences, AI Laboratory, Tech. Rep., 2005.

[4] M. W. Shelley, *Frankenstein.* New York : C. N. Potter : distributed by Crown Publishers, 1977.

[5] K. Capek, "Rossum's universal robots," 1921.

[6] M. E. Moran, "The da vinci robot," *Journal of Endourology*, pp. 986–990, December 2006,.

[7] A. Mackworth, *Computer Vision: System, Theory, and Applications.* World Scientific Press, Singapore, 1993.

[8] *Robocup 1997: Robo Soccer World Cup Initiative*, vol. 1395, 1997.

[9] Sony, "Qrio," http://www.sonyaibo.net/aboutqrio.htm, 2004.

[10] Honda, "Asimo," http://asimo.honda.com/, 2008.

[11] Sony, "Aibo," http://support.sony-europe.com/aibo, 2006.

[12] Kokoro, "Actroidder," http://www.kokoro-dreams.co.jp/english/robot/act/index.html, 2008.

[13] "Operating system documentation project," http://www.operating-system.org, 2003.

[14] M. Saggar, T. D. Silva, N. Kohl, and P. Stone, "Autonomous learning of stable quadruped locomotion," *Lecture notes in computer science*, no. 4434, pp. 98–109, 2007.

[15] T. Hornyak, *Loving the machine: The art and science of Japanese robots.* New York : Kodansha International, 2006.

[16] L. I. Anderson, *Nikola Tesla - Guided Weapon and Computer Technology*, L. I. Anderson, Ed. Twenty First Century Books, 1998.

[17] A. Currie, "The history of robotics."

[18] R. Brooks, , Flynn, and M. Anita, "Fast cheap and out of control: A robot invasion of the solar system," pp. 478–485, 1989.

[19] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara, "Robocup: A challange problem for ai and robotics," *Lecture Notes In Computer Science*, vol. 1395, pp. 1–19, 1998.

[20] S. Behnke, "See, walk and kick: Humanoid robots start to play soccer," *See, Walk and Kick: Humanoid Robots Start to Play Soccer*, vol. 5, pp. 497–503, December 2006.

[21] RoboCup, "What is robocup," http://www.robocup.org/overview/21.html, 2007.

[22] ——, "Small size robot league," http://small-size.informatik.uni-bremen.de/, 2007.

[23] ——, "Robocup," http://www.robocup.org/games/01Seattle/cfa2001legged.html, 2000.

[24] Robocup, "Humanoid league 2006," http://www.humanoidsoccer.org/.

[25] ——, "Robot world cup," http://www.fira.net/media/newsletters/read.html, 1999.

[26] A. Kleiner, "Rescue simulation project," http://kaspar.informatik.uni-freiburg.de/rcr2005/, 14th December 2008.

[27] E. Thorndike, *Animal Intelligence: Experimental Studies.* CT: Wesleyan University Press, 1911.

[28] M. N. GA Rummery, "On-line q-learning using connectionist systems," Ph.D. dissertation, Cambridge University Engineering Department, 1994.

[29] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* The MIT Press Cambridge, Massachusetts London, England, 1998.

[30] R. S. Sutton, "Learning to predict by the methods of temporal differences," in *Machine Learning*, 1988, pp. 9–44.

[31] K. Teknomo, "Q learning numerical example," http://people.revoledu.com/kardi/tutorial/ReinforcementLearning/Q-Learning-Example.htm, 2006.

[32] "Reinforcement learning," http://www.cse.unsw.edu.au/ cs9417ml/RL1/tdlearning.html.

[33] D. Gu and H. Hu, "Reinforcement learning of fuzzy logic controllers for quadruped walking robots," 15th Triennial World Congress, Barcelona, Spain, 2002.

[34] "Redhat cygwin," http://www.cygwin.com/.

[35] J. Randlov and P. Alstrom, "Learning to drive a bicycle using reinforcement learning and shaping," ser. Fifteenth International Conference on Machine Learning, 1998.

[36] H. Kand and M. Veloso, "Physical model based multi-objects tracking and prediction in robosoccer," in *In Working Note of the AAAI 1997 Fall Symposium. AAAI*, 1997.

[37] S. Coradeschi, L. Karlsson, P. Stone, T. Balch, G. Kraetzschmar, M. Asad, and M. Veloso, "Overview of robocup 99," Springer-Verlag, 2000.

[38] J. R. Austin and A. Zelinsky, *Robust monte carlo localization for mobile robots*, ser. The Tenth International Symposium 3 on ScienceDirect Search. Springer, 2003.

[39] P. Stone, B. Tucker, and K. Kraetzschmar, *Robocup 2000: Robot Soccer World Cup IV*, P. Stone, Ed. Springer, 2001.

[40] A. Birk, S. Coradeschi, and S. Tadokoro, *Robocup 2001: Robot Soccer World Cup V*, Springer, Ed. Springer, 2002.

[41] S. Chen, M. Sue, T. Y. amd T Hengst, S. Pham, C. Sammut, and T. Vogelg, "The unsw robocup 2001 sony legged league team," *Lecture notes in computer science*, no. 2377, pp. 39–48, 2002.

[42] R. Gal A. Kaminka, Pedro U. Lima, *Robocup 2002: Robot Soccer World Cup VI*. Springer, 2003.

[43] B. Browning, *RoboCup 2002: Robot Soccer World Cup VI*, ser. Lecture Notes in Computer Science. Springer, 2003, vol. 2752/2003, ch. RoboCup 2002 Small-Size League Review, pp. 453–459.

[44] M. Velosoa, S. Lenser, D. Vail, M. Roth, A. Stroupe, and S. Chernova, "Cmpack-02: Cmu's legged robot soccer team," in *RoboCup 2002: Robot Soccer World Cup VI*. Springer-Verlag, 2003.

[45] D. Vail and M. Veloso, Eds., *Multi-robot dynamic role assignment and coordination through shared potential fields*, ser. ICRA, the 2003 IEEE International Conference on Robotics and Automation, Taiwan, May 2003.

[46] C. Sammut, W. Uther, and B. Hengst, "runswift 2003," school of Computer Science and Engineering University of New South Wales and National ICT Australia.

[47] M. otzsch, J. Bach, H. Burkhard, and M. Ungel, *Designing agent behavior with the extensible agent behavior specification language XABSL*. Springer, 2004, 7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences).

[48] M. L. otzsch, "Xabsl web site," http://www.xabsl.de/, 2003.

[49] R. Thomas, B. Ronnie, D. Ingo, H. Matthias, and H. Jan, "Grmanteam 2004: The grman national robocup teram," in *Germanteam 2004: Robot Soccer Worlaperiusaperiusd Cup VIII*. Springer, 2005.

[50] M. Veloso, P. E. Rybski, S. Chernova, C. McMillen, J. Fasola, F. Hundelshausen, D. Vail, A. Trevor, S. Hauert, and R. Espinoza, "Cmdash'05: Team report," School of Computer Science, Carnegie Mellon University, Tech. Rep., 2005.

[51] G. Hornby, M. Fujita, S. Takamura, T. Yamamoto, and O. Hanagata, Eds., *Autonomous evolution of gaits with the Sony quadruped robots*, ser. Genetic and and Evolutionary Computation conference(GECCO), vol. 2, 1999.

[52] G. Hornby, J. Yokono, S. Takamura, T. Yamamoto, and O. Hanagata, "Evolving robust gaits with aibo," *IEEE international conference on robotics and automation*, vol. 3, pp. 3040–3045, 2000.

[53] G. Kuhlmann, P. Stone, and J. Lallinger, "The ut austin villa 2003 champion simulator coach: A machine learning approach," in *RoboCup-2004: Robot Soccer World Cup VIII*, D. Nardi, M. Riedmiller, and C. Sammut, Eds. Berlin: Springer Verlag, 2005, vol. 3276, pp. 636–644.

[54] I. Dahm and J. Ziegler, *Robocup 2002: Robot Soccer World Cup VI*. Springer, 2002, ch. Adaptive methods to improve self localization, pp. 393–406.

[55] J. Chen, E. Chung, R. Edwards, N. Wong, B. Hengst, C. Sammut, and W. Uther, "A description of the runswift 2003 legged robot soccer team," University of New South Wales, Tech. Rep., 2003.

[56] D. Gu, J. Hu, and E. Tsang, "Genetics-based machine learning and behavior-based robotics: a new synthesis," in *Computational intelligence in robotics and automation*, 2003.

[57] P. Fidelman and P. Stone, "Learning ball acquisition on a physical robot," *International Symposium on Robotics and Automation*, 2004.

[58] C. Kwok and D. Fox, "Reinforcement learning for sensing strategies," in *Intelligent Robots and Systems*, vol. 4, 2nd October 2004, pp. 3158–3163.

[59] N. Kohl and P. Stone, "Machine learning for fast quadrupedal locomotion," *Proceedings of The National Conference of Artificial Intelligence*, pp. 611–616, 2004.

[60] S. Kalyanakrishnan, Y. Liu, and P. Stone, "Half field offense in robocup soccer: A multi agent reinforcement learning case study," in *RoboCup-2006: Robot Soccer World Cup X.* Springer Verlag, 2007, vol. 4434, pp. 72–85.

[61] S. L. Wuy, Y. R. Liou, W. H. Lin, and M. H. Wu, "A multi-agent algorithm for robot soccer games in fira simulation league," department of Computer Science and Information Engineering Chang Gung University, Kwei-Shan Tao-Yuan, Taiwan, R.O.C.

[62] M. Fujita and H. Kitano, *Autonomous Agents.* Springer, 1998, ch. Development of autonomous robot quadruped robot for for robot entertainment, pp. 7–18.

[63] J. Connell and S. Mahadevan, *Robot Learning.* Springer, 1993.

[64] S. Mahadevan, "Machine learning for robots: A comparison of different paradigms."

[65] M. Mataric and D. Cliff, "Challenges in evolving controllers for physical robots," *Robotics and autonomous systems*, vol. 19, no. 1, pp. 67–84, 1996.

[66] I. I. Gikhman, A. V. Skorokhod, and S. Kotz, *The theory of stochastic processes.* Springer, 2004.

[67] A. Kleiner, M. Dietl, and B. Nebel, *Robocup 2002: Robot Soccer World Cup VI.* Springer, 2003, ch. Towards a Life-Long Learning Soccer Agent, pp. 126–135.

[68] M. Riedmiller and A. Merke, "Using machine learning techniques in complex multi-agent domains," in *Adaptivity and Learning.* Springer, 2002.

[69] M. Ahmadi and P. Stone, "Instance-based action models for fast action planning," in *RoboCup-2007: Robot Soccer World Cup XI*, U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, Eds. Springer Verlag, Berlin, 2008.

[70] S. K. Chalup and C. L. Murch, "Machine learning in the four-legged league," *3rd IFAC Symposium on Mechatronic Systems*, 2004.

[71] Sony, "Openr," http://www.aibo.com/openr, 2006.

[72] E. T. D. Touretzky, "Tekkotsu: A framework for aibo cognitive robotics," *Proceedings of the national conference on artificial intelegence*, vol. 20, pp. 1741–1742, 2005.

[73] "Urbi forge," http://www.urbiforge.com, 2005.

[74] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *In Advances in Neural Information Processing Systems 12*. MIT Press, 199, pp. 1057–1063.

[75] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," ser. IEEE International Conference on Robotics and Automation (ICRA 2004), May 2004, pp. 2619–2624.