Learning obstacle avoidance by a mobile robot

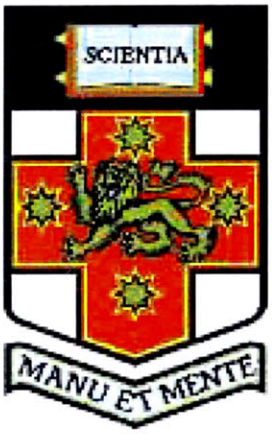
Author:
Esmaili, Nasser

Publication Date:
1999

DOI:
https://doi.org/10.26190/unsworks/8502

THE UNIVERSITY OF NEW SOUTH WALES

Learning Obstacle Avoidance by a Mobile Robot



Nasser Esmaili

B.Sc. Eng. – Bahonar University of Kerman

Artificial Intelligence Research Group
School of Computer Science and Engineering
University of New South Wales
Sydney 2052, Australia

A thesis submitted as partial requirement for the degree of Master of Engineering (Computer Science and Engineering)

August, 1999

I dedicate this thesis to the one who inspired me most;

"THE SPIRIT OF GOD"

Acknowledgment

I wish to thank my supervisor, Professor Claude Sammut for his patience and guidance through the course of my thesis.

I specially express my gratitude towards my family who have patiently carried me along during the time I was needed most.

My special thank to my wife who has stood by me and who has always been there with her love, support, and understanding that I could never do without.

Thank you so much.

Table of Content

1. Introduction

The rapid development of computer technology has brought intelligent systems within the realm of possibility. The methods of monitoring, control and operation of all types of modern systems have changed dramatically. Intelligent machines now accomplish many of the tasks formerly performed by human operators. Many of today's control systems are benefiting from the complex and very fast decisions that augment traditional automation using machine intelligence, making them capable of handling such functions as problem solving, perception and learning.

1-1. Aim

Mobile robots are gradually entering the real world. There have been demands on autonomous robot systems to accomplish given tasks in unknown and/or dynamic environments. They are employed in automated factories, used for plant supervision, and more recently are increasingly becoming successful in service tasks such as health care and rescue missions. However, in order to have a wider use in the real world, mobile robots must be able to learn from their past experiences. To achieve this goal, one way is to dramatically increase the complexity of the robot's control software. This research is aiming at providing an alternative.

One of the basic tasks of a mobile robot is to move from one point to another (preferably the target) quickly and collision-free. To do so, it needs to be equipped with one or more sensory systems to perceive information from the world (the environment in which it is wandering). Recent mobile robots have been using distance-measuring sensors based on infrared, sonar and /or laser technologies widely. Due to the fact that the information provided by these sensors can be processed quickly (referred to as on-line), they are ideal for real-time monitoring of the environment.

Mobile robots also require an interface or a control link between the sensory systems and the actuators. Using this link the robots can alter their action with respect to their perception and also possibly with respect to the given task. The control link arises from situation-action rules provided to the robot.

If we can foresee all the situations likely to be encountered by the robot, it would be possible (but very costly) to program the robot to handle those situations appropriately. However, in the real world, with all its noise and variability, this approach becomes rather impossible. Therefore, robots should be able to learn from the experiences gained during the operations, or even more preferably be able to learn by observation. Applying machine learning techniques can help mobile robots meet the need for increased safety and adaptivity that real-world operation demands.

This thesis has adopted a machine learning method, ***Behavioural Cloning*** [MBM90], to attempt to solve the problem of Mobile Robot Navigation in an

unknown terrain. At the time this research started, it was one of the first efforts ever to use a physical platform rather than a simulation program. Furthermore it is one of the first attempts to use behavioural cloning technique on a physical system.

The aim is to learn how to navigate a mobile robot using behavioural cloning, a term introduced by Michie describing systems and methods to clone a human-operated system [MBM90]. This job is performed by learning specific rules to describe the operation of the machine, which can be used to emulate the operation of that machine. Behavioural cloning is the imitation of human's sub-cognitive skills.

Developing a mobile robot gives us the opportunity to investigate issues in the design of intelligent systems because the robot's mobility forces us to deal with many unpredictable environmental situations. One dictionary definition of intelligence is the ability to deal with new or changing situations. Thus, a mobile robot that reliably navigates in unknown environments gives the appearance of intelligent behaviour. The main idea of an autonomous vehicle is quite simple: given a task to perform, it must have the ability to perceive the environment and act appropriately without human intervention. This ability requires a feedback control system to link the vehicle's sensing and control. Unfortunately, autonomous robots have characteristics not yet satisfactorily addressed by the classical control community:

- Solving the problems encountered by the mobile robot generally requires the integration of several methodologies.
- The robot's decision space is discrete and composed of distinct elements as opposed to continuous functions. The system must react to the environment in an appropriate time period.
- Due to the limitations of the sensors and sensory processing, most of the knowledge the robot acquires is either incomplete or uncertain.

1-2. Task in Hand

Interest in the robot learning field has been growing fast in the last few years. Providing learning ability to robots offers certain benefits:

- Increasing the ability of the machine in a dynamic environment, where the propagated knowledge will eventually become obsolete or even is not available at all.
- Reducing the cost of programming robots to perform specific tasks.
- Furthermore, increasing the robots' ability to overcome changes in their own physical specifications, such as sensor drift or power failure.

Navigation is one of the standard tasks in the mobile robots' domain. In order for a mobile robot to accomplish a non-trivial task, the task should be described in terms of primitive actions of the robot's actuators. The complete navigation problem can be broken down in to related sub-tasks, which will be referred to as behaviours here after in this thesis.

Autonomous agents, like mobile robots, typically operate in dynamic and uncertain environments. Such environments can only be sensed imperfectly, so the effects of those are not always predictable. Also, the robots may not usually control the changes due to the environment. Prominent among the approaches to design agents operating in these kind of environments are the so-called behaviour based, situated, and animat methods [Bro86b][KR90][Mae89][Wil, 1987].

As useful as we imagine the robots to be, they must be capable of obtaining (learning) new behaviours, either primitive or complex. There have been several methods used to obtain more complex behaviours [MB90][Mit89][WDEP89]. These techniques mostly depend on hand-coded programs or built-in circuitry. In our experiments the acquisition of behaviours has been carried out by imitating human trainers.

To control a dynamic system one requires the sort of skill that can not be completely described, but can be behaviourally demonstrated. Behavioural cloning [MBM90], is the process of reconstructing skills from operators' behavioural traces by means of machine learning techniques.

To build a controller for a physical process, different methods can be used. Some examples of these methods are optimal control [MKK86], fuzzy control [Wil87], neural network [Np90], expert systems [Dev87], and machine learning [UB94], each of which has its own capabilities and limitations [WK90]. Although, each method may be suitable for a task, it will not suit

another task. Also, there is no clear criterion to decide on which method is most suitable for a specific task. But, we know that a competent human operator can control most dynamic systems, even those difficult to control by classical methods. As a result, there is growing interest in mimicking the skills of the human operator [MBM90][SUKM92][UB93][UB94].

Also, learning through interacting with an operator is an efficient method to increase the knowledge of an intelligent robot. Through its experiences, the robot can become more and more autonomous, improving its reactions to events in the environment. There have been several attempts to develop a system based on explanation-based learning [Dej86][MMS85][MKK86] or external guidance [GRL87][LNR87][RL86] methods.

Inductive learning [MC94] is concerned with the extraction of general principles from examples. In the other words, in this method "*inductive inferences are obtained from facts provided by a teacher or the environment*" [Mic83]. It is one of the major goals of machine learning.

In the experiments related to this thesis a mobile robot is demonstrated to acquire a primitive motor skill by being guided through a set of trials by a human operator. The robot has 2 main wheels and a leading front wheel, 6 infra-red sensors for obstacle detection, one rotary sonar sensor on top, and a 386 IBM-PC compatible computer as its controller [ECM94][ECS95].

A log file of the robot's sensory and motor data is recorded as a trainer steers the robot through a set of obstacle avoidance scenarios. The final logged file

consists of data gathered during more than 300 sets of training runs in several different scenarios by different trainers. Later, the file is pre-processed to remove unwanted lines of data or noise. If the consecutive lines show a little change in sonar data, or there is no change in action control, then the last line of data is kept and the early history is removed. The reason is to keep the latest occurrence which will eventually produce the action.

The results then are processed by two induction algorithms, each of which extracts a decision tree that represents appropriate skirting movements for a range of obstacle patterns. The skill is then evaluated by observing the robot executing the codes generated from the obtained decision trees in response to objects encountered during autonomous movements.

We have used Ripple Down Rules (RDR) [CH89][Gc92] as the knowledge acquisition tool and Induct & C4.5 as machine learning mechanisms to automatically create rules from the logged data. We compare the decision trees induced by different algorithms. The 'skill' is then evaluated by observing the robot as it executes code generated from the decision trees in response to objects encountered during autonomous movement.

1-3. Organization

This thesis is organized in five chapters. Chapter 2 is a survey of a variety of robot navigation systems, mostly similar to the employed method in this thesis. It covers a detailed discussion on Behaviour-based or reactive control

as well as representational world modeling. It also covers a survey on available literatures on learning, concentrating on machine learning techniques.

Chapter 3 will explain in details the design and implementation of a mobile robot which specifically was designed as an experimental platform for this thesis, as well as covering detailed technical information on the second robot used for the second parts of the experiments.

Chapter 4 covers the experiments performed as partial requirements for this thesis. And finally, chapter 5 will be the discussion and conclusion of this thesis.

1-4. Contribution

So far, the majority of mobile navigation systems in general and object avoidance behaviour in particular have been implemented either by hard coded programming or by simulation.

However, just as medical diagnosis can be learnt by observing a physician at work, or riding bicycle can be learnt from experience, we should be able to learn how to control a dynamic system just by watching a human operator in action.

Learning to control by observation is only possible by means of having a vast history of the previous situations and actions taken based on those situations. In the case of the experiments related to this thesis, the data collected during the training period was logged to a file. The logging only applied when an action was taken by the operator in response to changes in the robot and/or environment's states.

Very few attempts prior to the start of this research have used a physical platform (a mobile robot in this case) as opposed to a simulation environment for the purpose of navigation learning. Also, during the course of these experiments, for the first time a physical mobile robot has been used for behavioural cloning purpose. For this thesis a specific robot was designed, built, and trained by a human operator through sets of different obstacle avoidance scenarios. The logged data from the training sessions were then used to build sets of control rules using an induction program.

We have shown that by using machine learning technique to control a dynamic physical system in a dynamic environment, the control tasks could be learnt. Therefore, learning control rules by observation is another way of building more complex but yet cheaper control systems quickly and easily.

"Clean-up" effect was an important issue of the results from these experiments, confirming the previous findings [SUKM92], which shows that the physical systems can over-perform the human trainer. While our experiments have been primarily concentrated on mobile robot navigation in

general and obstacle avoidance in particular, inductive methods and learning by observation can be applied to a variety of similar problems, such as educational purposes, training simulators and plant control.

This thesis, based on the current research, was aimed at producing a reliable and reproducible method for building controllers for a mobile robot. The following papers have been published relating to the topic of this thesis:

Esmaili, N., C.A. Sammut, and G.A. Mann (1994). *"Navigation Learning by a Mobile Robot"*, Proceedings of ICEE-94, Tarbiat Modarres University, Iran, pp. 142-150.

Esmaili, N., C.A. Sammut and G.M. Shiraz (1995). *"Behavioural Cloning in Control of Dynamic Systems"*, Proceedings of 1995 IEEE International Conference on Systems, Man & Cybernetics SMC'9, Vancouver - BC, Canada, Oct. 22-25, pp. 2904-2909.

Shiraz, G.M., C.A. Sammut and N. Esmaili (1995). *"Man, Machine Cooperation for Learning to Control Dynamic Systems"*, Proceedings of1995 IEEE International Conference on Systems, Man & Cybernetics SMC'95, Vancouver - BC, Canada, Oct. 22-25, pp. 1108-1112.

2. Background

2.1 Navigation

This section provides a general description of existing Robot Navigational Strategies. This investigation of available algorithms will provide useful information on how program based control works.

It is perhaps advantageous to make a distinction between high-level global navigation (i.e., planning an optimal path to some desired goal location in the world coordinates), and local navigation (i.e., piloting the robot around unexpected obstructions). This chapter shall address only the latter category, from the two perspectives of:

- *Required sensors.*
- *Interpretation of data collected by sensors.*

2.1.1 Behaviour-based (Reactive Control)

Sometimes, in the control strategy of a mobile robot there is no usage of the intervening symbolic representations attempting to model, in absolute sense, a part of the robot's operating environment. Then, the behaviour-based strategy directly coupling real-time sensory information to motor actions is referred to

as *"reactive control"*. In this method the control strategy is clearly based on the state of the robot and its environment.

Arkin lists the following general characteristics of reactive control [Ar92a]:

- It is typically manifested by decomposition into primitive behaviours.
- Global representations are avoided.
- Sensor de-coupling is preferred over sensor fusion.
- It is well suited for dynamically changing environments.

The most simple reactive collision avoidance capability for an autonomous mobile robot is perhaps illustrated by the basic wander routine implemented by several research-groups [Eve82][Bro86a][Ark87]. The term, *wander,* is used here to describe a behavioural primitive that involves travelling more or less in a straight line until an obstacle is encountered, altering course to avoid collision, then resuming straight-line motion. Such a capability can be simply hard-coded, rule-based, or inherent in a more sophisticated layered subsumption architecture [Bro86a]. Brooks' layered subsumption architecture will be discussed later in greater detail.

2.1.1.1 Subsumption Architecture

A mobile robot's control system should be able to process complex information real time due to the fact that it performs in an environment with rapid condition changes. An ordinary control system would be one big program loop, while a better method is to break this complex system into many simple and basic

modules, each in charge of a simple task. This method will not be vulnerable to unexpected errors during operation and also would allow a simple layering modulation for applications of higher level.

Figure 2-1 shows a decomposed control system into a series of functional units by a series of vertical boxes. The control feedback loop is made from the environment information collected from the sensors and fed through the modules and finally returned back to the environment by the actuators as

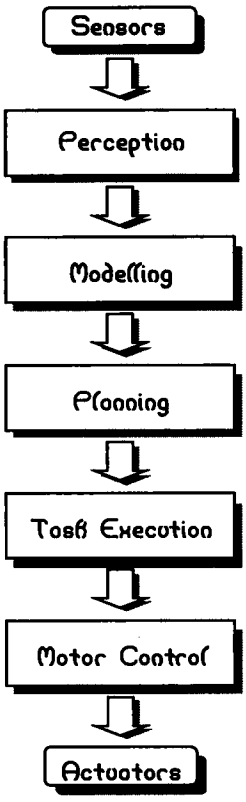


Figure 2-1. **Traditional decomposition of a mobile robot control system into functional modules**

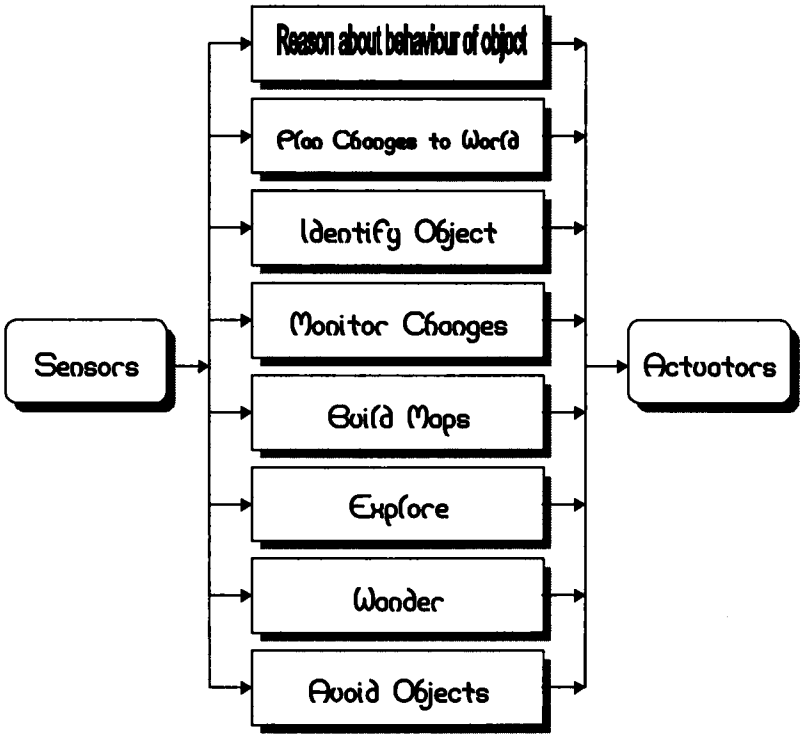


Figure 2-2. **Decomposition of a mobile robot control system based on task achieving behaviours (Brook's layered control system)**

action control decisions. Instances of each layer must be built in order to run the robot. Later changes to a particular layer must either be done in such a way

that the interfaces to neighboring layer do not change, or the effects of the change must be also incorporated to neighboring layer, changing their functionality. [Bro86a].

Alternatively, Brooks used task-achieving behaviours as his primary decomposition of the problem, which is illustrated in Figure 2-2. In his powerful and versatile subsumption architecture, layers are implemented as additional finite-state machines to support progressively intelligent control. This decomposition method has several advantages over the other type of control system in terms of:

- ***Multiple Goals:*** The control system is responsive to higher priority goals, while still servicing necessary "low-level" goals.
- ***Multiple Sensors:*** As all sensors' readings include an error component and there is no direct analytic mapping from sensor values to desired physical quantities, the readings are often consistent. So, the robot must make decisions under these conditions.
- ***Robustness:*** When some sensors fail, the robot should be able to adapt and cope by relying on the functional ones.

Control is layered with higher level layers subsuming the roles of lower level layers when they decide to take control. The system can be broken at any level, and the remaining layers can form as a complete operational control system. Brooks also defined a number of levels of competence for an autonomous mobile robot. A level of competence is an informal specification of a desired class of behaviours, which are:

1. Object avoidance.
2. Aimless and collision-free wander.
3. Heading for observed accessible targets in the distance.
4. Notice changes in the "static" environment.
5. Reason about the world in terms of identifiable objects and perform tasks related to certain objects.
6. Formulate and execute plans that involve changing the world's state in some desirable way.
7. Reason about the behaviour of world's objects and modify plans accordingly.

According to Brooks: *"the main idea of the competence levels is that corresponding to each level, layers of control system can be built and every new layer simply is added to the existing set to move to the higher level of overall competence. In addition, the lower layer will continue to run unaware of the layer above it, which sometimes interferes with its data path. The same process is repeated to achieve higher levels of competence, which is described as* ***Subsumption Architecture"*** [Bro86a].

2.1.1.2 Wander

By way of illustration, the wander routine employed on ROBART I [Eve82] was based on a six-level scheme of proximity and impact detection using the following sensor inputs:

- ❊ A positional near-IR proximity scanner.
- ❊ Forward-looking sonar sensor.

- ❊ Ten near-IR proximity sensors.
- ❊ Projecting "cat-whisker" tactile sensor.
- ❊ Contact bumpers.
- ❊ Drive motor current sensors.

The first two categories, looking out ahead of the robot for planning purposes, were loosely classified as non-contact ranging sensors, while the next three were considered close-in proximity and tactile sensors requiring immediate action. Drive motor overload was used as the last resort in the event that no other sensors did not detect collision.

In some ways the software implementation was similar to Brooks' subsumption architecture approach [Bro86a]. There were two distinctly separate hierarchical layers in a bottom-up design:

- low-level interrupt-driven layer,
- and, intermediate-level polling layer in the main program loop.

This layering was basically an algorithmic differentiation of software categories running on a single processor, however limited in actual embodiment to only two layers, although a future higher-level expansion was suggested [Eve82].

The sensors responsible for close-in environment monitoring of ROBART (i.e., proximity detectors, feeler probes, and bumpers drive current overload) were highly priotorized. Therefore, a maskable interrupt request (IRQ) routine was used to read them. This routine monitoring the state of sensors' output would

redirect the action of the robot according to the hard-coded reactions specifically designed for individual sensors, unless stopped by the main loop of the program. A hard-coded response for a right-front bumper impact would consist of the following steps:

- Stop all forward direction travel.
- Turn steering full right.
- Back up for x number of seconds while monitoring rear bumper.
- Stop and center steering.
- Resume forward travel.

Both the main loop and the IRQ routine contained multiple behaviours according to the assigned priority in case of conflict. For example, in order to specify which device has requested the service (by triggering the interrupt handler), all the potential inputs would be checked by the collision avoidance interrupt service. Those inputs representing actual impact with an obstacle had higher ranks of being checked, followed by inputs associated with "cat-whisker" probes, near-IR proximity detectors, and so forth. Based on the ranking associated with the sensors, the interrupt service would initiate the appropriate action for the first active condition, ensuring that higher concern situations received priority attention. The issued avoidance response in turn would also check the other inputs to ensure appropriate reaction in chain of the event (i.e., monitoring rear bumper while reversing).

On the occasion of triggering of an interrupt by the close-in collision avoidance sensor, the intermediate level software's execution was temporarily suspended as the control switched to the interrupt service routine. At this stage, the low-level avoidance maneuvers would be in charge until the robot was clear from the obstacle (unlike the subsumption architecture approach). On the other hand, the intermediate level software was able to disable IRQ interrupts associated with collision avoidance sensors, or otherwise suppress or inhibit the lower level behaviours.

The intermediate-level software also would continuously check the sonar and head-mounted near-IR scanner on each pass of the main loop, to avoid any poor randomly "bump-and-recover" situation. These sensors were responsible for monitoring of the distance up to 1.5 meters ahead of the robot and in turn producing a suitable representation of detected targets, relatively. If the forward path found blocked, the wander algorithm would choose the least obstructed direction to continue its movement. Since all zones were equally weighted in a binary fashion (i.e., either blocked or clear), the least obstructed direction would be the one with the largest number of adjacent clear zones. The simplicity of this model enabled real-time on-the-fly response, without the robot having to "stop and think" before continuing its path.

A major shortcoming of this world representation is the problem with continuous update of the polar model to produce more accurate probability zone occupancy.

For example, a detected obstacle located at point P1 in Figure 2.3 would transition due to robot motion from Zone 05 to Zone 00, crossing through all zones in between. Repetitive sightings would likely not be associated with the same zone number. As a result, reactions are always made to "snapshot" sensor information subject to numerous sources of potential error, usually resulting in jerky or erratic vehicle movement. Borenstein and Koren [BK90a][BK90b] solve this problem by deriving the polar model in real time from a certainty grid representation.

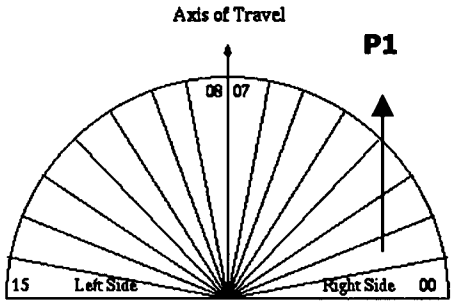


Figure 2-3. **The world model employed on ROBART I consisted of sixteen wedges shaped zones relative to the direction of travel [Eve82].**

2.1.1.3 Circumnavigation

The term *circumnavigation* describes a collision avoidance behaviour in which the robot moves around an obstacle, while still attempting to move in the general direction of the goal. When the on-board sensors show the object is no longer a threat, the desired path is recovered. In a sense, circumnavigation can be regarded as a wander behaviour that turns to a goal-seeking behaviour when clear, instead of simply resuming straight-line motion.

A good example of circumnavigation collision avoidance behaviour is the one implemented by Cyber-motion, Inc. on their K2A Navmaster autonomous vehicle. In normal operation, the K2A controller calculates a motion vector from its current position to a down loaded X-Y goal location. To reset the robot's heading according to the vector orientation, this vector is re-calculated on-the-fly as the robot moves. If a threatening obstacle is detected in front of the robot, advancing speed is reduced and a fixed bias is added to the heading command. The sign of the bias chosen in a way to deviate the robot away in the direction of free space. Once the obstruction is cleared, the steering bias is removed, and the robot moves towards the goal location.

Another example is the work of Lumeslsky and Stepanov [LS87]. In their work, they present the mobile automaton as a point and presume any shape of obstacles with continuos boundaries and finite sizes with no restriction on the size of the scene. While the only information available to the automaton is only its own coordinates and the target's positions', the obstacles are detected by the automaton's sensors only when they are hit. Lumeslsky and Stepanov have presented three basic path planning algorithms, as well as analysing their performances and deriving the upper bounds on the length of their generated paths.

Under *'Bug1'* basic algorithm, the automaton never meets the same obstacle twice between the origin and target point, while it only can meet finite number of obstacles in that path. *'Bug2'* sharing the second characteristic of *'Bug1'*, also defines that the automaton will pass any point of the i^{th} obstacle boundary

at most $n_i/2$ times in which n_i is the number of intersections between the straight line (Origin, Target) and the i^{th} obstacle. While Bug1 never creates any local cycles, it is over-cautious and never covers less than the full perimeter of the obstacle. On the other hand, Bug2 takes advantage of the simple situations, but seems to be quite inefficient in more complicated cases. To overcome the shortcomings of these two procedures, they combined the better features of the two in a new algorithm and called it *'BugM1'*. This new procedure combined the efficiency of Bug2 in simpler scenes with the more conservative strategy of Bug1. In this new algorithm, the number of local cycles for a given point on the path, containing this point, never exceeds two which means that the automaton never passes the same point of the obstacle boundary more than three times.

The very obvious advantages of the circumnavigation approach are the simplicity and speed of execution without any need for high complexity of processing power. However, The technique is limited to the occurrence of minor advances into the intended path. Any obstacle significantly blocking the desired route can push the robot too far from its intended target causing any normal path resumption to be close to impossible. Furthermore, circumnavigating robot must always travel forward in the general direction of the goal without backtracking. No further arrangement can be made for choosing alternate routes if the original path is completely blocked.

In addition, as specific behaviours, both wander and circumnavigation can also be considered stand-alone collision avoidance control strategies.

The following sections deal with additional examples of control strategies for collision avoidance (and other purposes) that are capable of implementing not only wander and circumnavigation but various other behaviours as well.

2.1.1.4 Unmanned Vehicle System AFV-I

Montgomery et al., designing a flying vehicle, illustrated the advantages of hierarchical, behaviour-based control, with low-level behaviours ensuring robot's survival and high-level behaviours performing tasks such as navigation and object location [MFB95]. They have used a radio-controlled model helicopter, powered by a two-stroke engine as their robotic platform. This craft has 5 degrees of control including roll, pitch, yaw, collective, and throttle (both controlling thrust). Various sensors including a compass (measuring heading), three ultra-sonic sensors (down-facing, measuring distance from the ground), RPM (measuring the speed of the main rotor), and a CCD camera (providing visual information) were mounted on the craft. In addition the chopper used three solid-state gyros. The AFV-I can not use its sensors to find the targets or move around, depending heavily on the vision system.

Montgomery et al. claim that creating a flying robot with the capabilities to locate and manipulate objects and transport them from one location to another presents many challenges. To achieve its goals under hazardous conditions, without human guidance, and within a fixed time limit, the robot must make control decisions based on imperfect sensory data, while adapting to unexpected situations such as gusts of wind or sensor failure. It must make

these decisions in real time to maintain the craft's safety and ensure system survival. In order to do so, the AFV-I uses a behaviour-based control architecture, which partitions the control problem into a set of loosely coupled computing modules. Each module, or behaviour, is responsible for achieving a specific task. These behaviours interact to achieve the robot's overall goals. The modules are organized hierarchically, with low-level, reflexive behaviours responsible for craft survival and high-level behaviours responsible for tasks such as navigation and object location.

They emphasised that a behaviour-based approach has many advantages over traditional methods of controlling autonomous mobile robots, they claim that traditional methods attack the control problem sequentially. In another word, a robot first senses, perceives, and models its environment; then it plans and acts in that environment. Since the world is full of information, traditional methods are bound to be overloaded by information, which makes the robot incapable of functioning in real time. In addition, these methods assume that the robot can construct accurate, global world models based on the incoming sensory information. A number of factors such as rapidly changing world, limited computer processing power and inaccurate and incomplete sensor models make this matter very difficult if not impossible. In contrast, a behaviour-based approach solves the problem in a parallel fashion. Each behaviour, acting concurrently with other behaviours, in order to prevent information overload, only absorbs the information required to complete a given task at a given time (from the environment). This reduction of work also reduces the robot's

computational load by eliminating the need for construction and maintenance of a global world model.

Montgomery and et al. also believe that another advantage of the behaviour-based approach is that it provides the ability to create layers of increasingly complex behaviours. If necessary, higher-level behaviours can inhibit or modulate lower-level behaviours. Thus, one can incrementally build and test a robot control system with increasing capabilities, without losing low-level capabilities already created. This control approach has been explored by others, including in the subsumption architecture [Bro86a] and in an architecture for reactive navigation, AuRa [Ark90].

However, the behaviour-based approach has its own limitations. Interactions and possible combining of behaviours, which may critically affect the robot's stability, are unknown beforehand. It may be necessary to do experiments in order to find such combinations. However, since no models are available, this operation can be time consuming and potentially hazardous to the craft. And they have observed that any attempt to expand the complexity of the system can worsen the coupling problem by increasing the number of behaviours and layers. To overcome this problem, they have proposed to develop behaviour-based models from performance data, including methods by which the robot can obtain relevant parameters by learning.

2.1.1.5 Potential Fields

Krogh [Kro84] introduced the concept of potential fields for simulation of localised mobile robot control. The classical approach mathematically involves an artificial force acting on the robot, which is the vector summation of an attractive force representing the goal and number of repulsive forces associated with the individual known obstacles [Til90]:

$$\vec{F}_t(X) = \vec{F}_o(X) + \vec{F}_g(X)$$

Where:

$\vec{F}_t(X)$ = *resultant artificial force vector*

$\vec{F}_o(X)$ = *resultant of repulsive obstacle forces*

$\vec{F}_g(X)$ = *attractive goal force*

The attractive goal forces, which are the priority and the weights of the goal(s) can be classically represented as the following equation [Til90]:

$$\vec{F}_g(X) = Q_{goal} \frac{\vec{X} - \vec{X}_g}{|\vec{X} - \vec{X}_g|}$$

Where:

Q_{goal} = *a positive constant (ie, the "charge" of the goal).*

The classical potential field is the summation of the attractive goal force and the repulsive force contributions from those directions defined by the various fields of view of the obstacle detection sensors. The individual repulsive forces are aligned away from their respective obstacles and towards the robot, falling

off with the k^{th} power of separation distance [Til90]. For example, an early MIT implementation on the robot treated each detected sonar target as the origin of a repulsive force decaying as the square of the indicated range [Brooks, 1986]. The desired vehicle heading was represented as an attractive force. The resultant of all such virtual forces acting on the robot, if greater than a predetermined threshold, was used to compute the instantaneous drive motor commands for steering and velocity, effectively moving the platform away from obstacles and in the general direction of the goal.

Alternatively, Arkin [Arc92b] uses a repulsive force magnitude that is a linear function of obstacle range:

$$O_m = G\frac{S-d}{S-R} \text{ for } R < d \leq S$$

Where:

O_m = magnitude of repulsive force associated with obstacle

G = gain constant

S = sphere of influence from centre of obstacle (ie, $O_m = 0$ for $d > S$)

d = distance from centre of obstacle to robot

R = radius of obstacle.

In both of the preceding examples, since the resulting field depends only upon the relative positions of nearby obstacles, it is possible for repulsive forces to be generated by object that in fact do not lie along the intended path of travel. Such a situation is illustrated in Figure 2-4.

In recognition of the above concerns, Krogh [Kro84] had introduced the concept of generalised potential fields, wherein the potential field intensity is a function of not only relative position with respect to obstacles but also the robot's instantaneous velocity vector at that position. The generalised potential is the inverse of what Krogh calls the reverse avoidance time.

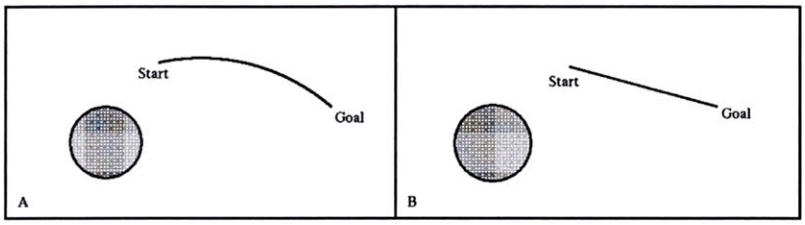


Figure 2-4. **(a) The classical potential field method considers only separation distance, causing the robot to deviate from a straight-line path segment even though moving away from the circular obstacle;**

(b) The generalised potential field method considers relative velocity in addition to separation distance [Til90].

Consider a robot approaching a stationary object at some constant velocity $\mathbf{V}_0$ as illustrated in Figure 2-5. There is some maximum allowable deceleration rate $\mathbf{a}_{max}$ that will bring the robot to a halt in the shortest possible length of time $\mathbf{t}_1$. Similarly there is some minimum deceleration rate $\mathbf{a}_{min}$ that will cause the robot to stop just before impact over some longer time interval $\mathbf{t}_2$. Reserve avoidance time is simply the distance in time required to stop for the two cases of maximum-allowed versus minimum-required decelerations, (i.e., $\mathbf{t}_2 - \mathbf{t}_1$). The generalised potential field is thus sensitive to time to impact as opposed to separation distance (Figure 2-4, part B), and approaches infinity as the reserve avoidance time approaches zero [Til90].

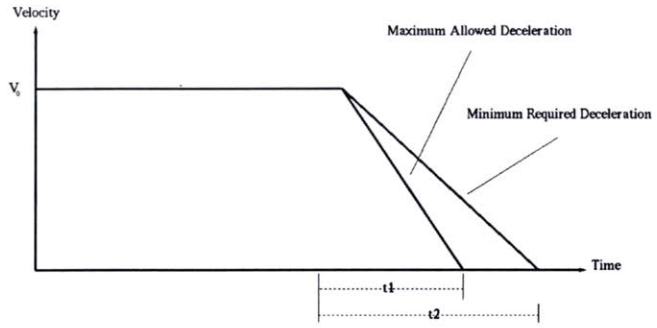


Figure 2-5. **Krogh [Kro84] defines the generalised potential as the inverse of reverse avoidance time, which is the difference in stopping times associated with maximum-allowed and minimum required decelerations.**

The principle limitation of the potential field approach is its vulnerability to becoming boxed in or "trapped" by intervening obstacles as illustrated in Figure 2-6. Culbertson [Cul63] predicted this problem for more general cases of "memory-less robots" that react to current stimuli in a deterministic fashion without taking into consideration the results of previous behaviour under similar conditions.

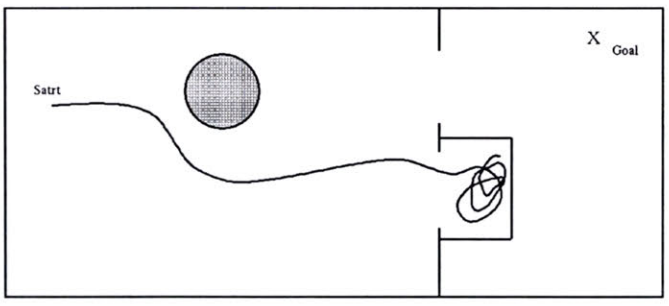


Figure 2-6. **The robot successfully negotiated first obstruction but has become trapped by the U-shaped structure of the closest and is unable to reach the goal in the next room.**

The likely occurrence of cyclic behaviour as well as local maxima and Minima make any system that relies sorely on the potential-field navigation approach somewhat unreliable [Arc92a]. To get around this problem, Thorpe [Thorpe, 1984a][Thorpe, 1984b] employed a grid-based search to find a good low-cost path towards the goal, adjusted the path off grid to further minimise costs, then executed the path with a variant of potential fields to keep the vehicle on the path. Krogh & Thorpe [Kro86] discuss the integration of a generalised potential field collision avoidance scheme, with a global path planner based on certainty grids (to be discussed in a later section) for optimal route planing and trap recovery.

In a different approach, Nam and et al. have proposed a unified method for avoiding obstacles while moving [NL95]. Their method combines the artificial potential field (APF) concept and view-time based motion planning, where the driving force is generated at every interval of the view-time [KL93]. Prior to their attempt, APF has already been used on stationary obstacle avoidance [Kro84][Kor91]. The whole idea is to split the moving obstacle avoidance into two separate problems, path planning for stationary obstacles and velocity planning for moving obstacles.

A mathematically formulated approach has been achieved to "P*lan the motion of a robot from the initial to the final location avoiding a moving obstacle in two dimensional space subject to the various constraints of robot and obstacle motions*". The time set from one sampling time instant to the next is defined as view-time. They also assume that velocity and acceleration of the moving

obstacle are monitored or known at each sampling time. At each sampling instance, an accessible region that will be swept by the obstacle in the next view-time is predicted based on the velocity, acceleration, and dynamic limitations of the obstacle. Then, an artificial potential field is constructed around the accessible region, which exerts repulsive force on the robot. During the view-time, the force induced by the artificial potential field drives the robot away from the accessible obstacle trajectories in real-time. The dynamic physical limitations of the robot are also considered. Application of the described procedure at each successive sampling time from the initial to final location suggests the collision-free trajectory for moving obstacle avoidance.

2.2 Learning

This Chapter is the result of a survey of some machine learning methods related to this research. It includes a survey of different methods of machine learning and a survey of previous works performed in learning to control dynamic systems. The conclusion comes with some discussions about previous works in this area and the way that it has inspired, to some extent, the methodology used in this thesis.

2.2.1. Machine Learning

The beginning of artificial intelligence marks the start of research on machine learning techniques. Samuel's checker system [Sam59] and Winston's programs for learning structural descriptions [Win75] can be named as examples of early machine learning researches. The last twenty years have produced a remarkable expansion of research in machine learning with a rapid growth in this field, despite the slow growth in 50s and early 60s.

There are several reasons for this growth. First, the ability to learn and to modify behaviour is part of intelligence and it is hard to consider an intelligent system without a capability for learning and self-improvement. Also, being able to learn makes it easier to build high performance systems.

Early success of symbolic learning systems in various areas such as medicine [DeDo72], agriculture [Mic83], robotics [AB75], reinforcement learning

[Wat92], process control of optimizing a Nuclear Fuel plant [Lea86], and chemistry [Buc78] helped this growth and turned more attention to this field.

To determine the rules or decisions mimicking the human trainer's actions, machine learning programs normally take a set of examples called ***the training set.*** This includes a sufficiently large and diverse set of cases and the action taken or decisions made accordingly by the trainer. It should be pointed out that the training set does not cover all possible situations, so the learning system should be able to generalize the possible decisions when coming across unseen (untrained) situations.

Machine learning methods could be classified in several different ways depending on: the underlying strategy that they used, the presence or absence of a teacher, the type of knowledge representation, and domain application. [Mic84]. Before we describe each of these categories, it is necessary to explain what machine learning means. Machine learning is the "Modification or construction by program of stored information structures, so that the machine-deliverable information becomes more accurate, larger in amount, or cheaper to obtain " [Mic82]. Machine learning is a sub-field of AI, whose ultimate goal is to replace explicit programming by teaching. By teaching in this thesis we mean any form of instruction, ranging from examples of the desired behaviour, domain knowledge of the task, or even weak performance feedback. Teaching is usually less difficult than explicit programming.

2.2.1.1. Machine learning methods' classification

Generally speaking, there are two types of learning, supervised and unsupervised. In the former, a teacher carefully selects examples for the learner, whereas in the latter the learner is given little or no feedback on the learning task.

Supervised Learning: In this method, a teacher assists the learning system, by providing a set of examples (a training set) presenting the appropriate action or output for a given situation to the learning system. The aim of the learning system is to minimize the difference between the reaction of the system and the reaction provided by the teacher in the training set for the same situation. In practice, the training set does not cover all the possible situations and the examples can be noisy. Hence, the learning system can be considered successful if it produces the correct output with a high degree of probability when exposed to unseen situations.

"Learning with a critic", too, can be classified as supervised learning. The critic, like a teacher, assists the learning system by providing useful information. However, the role of the critic is different to the role of the teacher. In this case, the critic provides an evaluation of the learning system's reaction to a given input instead of providing the correct reaction that is provided by the teacher in the supervised learning. The learning system itself is responsible for finding the correct reaction to a given input. The evaluation (credit score) and previous experience will be used to find the correct action.

As examples of which method, we can name two paradigms, inductive concept learning and explanation-based learning. The first paradigm assumes that the teacher presents examples of the reactions for the learner. In this paradigm, since the teacher is essentially, in some particular situations, is telling the learner what action to perform, the temporal credit assignment problem does not exist. However, the task credit assignment problem has to be solved, since the examples provided by the teacher pertain to some particular tasks. In the second paradigm, the teacher not only supplies the examples of the reactions, but also provides a domain theory for determination of which reaction is useful in certain situation.

Unsupervised Learning: There are different views on the definition of unsupervised learning, such as labeled and unlabeled. According to some [RN95], "Learning when there is no hint at all about the correct outputs is called unsupervised learning. An unsupervised learner can always learn relationships among its percepts using supervised learning methods – that is, it can learn to predict its future percepts given its previous percepts. It can not learn what to do unless it already has a utility function."

However, the hereunder text is based on Sutton's definition. In this method, the learner is not provided with which actions to take, and must discover which actions yield the best output by trying them (trial and error). As described by Sutton [Sut92] reinforcement learning is "the learning of a mapping from situation to action so as to minimize a scalar reward or reinforcement signal".

The underlying strategy is one of the most important parts of any learning system, the ability of which it can influence. The machine learning categories can also be classified based on the strategy used by a machine learning system.

An important aspect of learning systems affecting their ability to learn different concepts is the representation language. How to represent the training information (objects and their relations) and how the learning system expresses its acquired knowledge about the system, are some of the related problems. Many different representational systems and techniques have been used so far. Just as examples we can name some, including semantic networks (Winston's work [Win75]), predicate calculus (Sprouter [HaRo77], Marvin [Sam81], CIGOL [Mug88]), attribute/value vectors [Qui79], FOIL [Qui89], neural networks [WL90], and GRAIL [BS97].

Other sources of problem in many systems are the existence of noise in training set and missing attribute values. Misclassification of the examples and increase in size of the resulting classifier can be caused by these problems. Some learning systems assuming all the training examples are provided by an expert, so being complete and without noise, ignore these problems. Other systems use techniques such as tree pruning to deal with the noise [Qui93][Kon84][Bre84].

2.2.1.1.1. Inductive Learning

We begin with the most classical paradigm in machine learning, namely inductive concept learning. The learning system, knowing that the concept to be learned comes from some space of hypotheses, proceeds from individual cases to general principals. The system is also provided with a set of training examples of the target concept. Normally, these examples are drawn from some space of instances according to some unknown but fixed probability distribution. In the other words, the main task is to build a concept description being able to cover as many of the positive instances, and as few of the negative instances as possible, especially in the presence of noise in the system.

Many methods have been developed for solving the inductive learning problem; two of the most well known methods are decision trees [Qui86] and neural networks [MR86]. A classic example of a robot learning system based on the inductive learning paradigm is the ALVINN system [Pom90]. There are many other examples of inductive concept learning applied to robot learning systems [CM93][Dor96][Fra96][Mah94].

Inductive learning systems are suitable for cases where there exists a large number of training examples and the objective learning task is classification [Mic83][Mic84]. Well, of course it is also true to say that this problem somehow exists with all types of learnings. Another problem with this paradigm is that function approximators, such as neural nets, often take huge number of

exemplars to converge. One critical issue is how to speed up learning by incorporating some form of search strategy.

2.2.1.1.2. Deductive Learning

Humans appear to be capable of generalizing from a few examples. Clearly, humans bring to bear considerable amount of background knowledge to the learning task. This method is based on learning from a few examples but with a large amount of prior knowledge. The goal is to analyze and transform existing knowledge (past problem solving experience) into a more effective, efficient and operational form, using deduction as the primary inference mechanism. The result of deductive learning is as valid as the background knowledge and the input information. In this method, the emphasis is on improving the efficiency of the system, rather than extending the set of cases that the system can handle.

Explanation-based learning captured the major focus in this field during the last decade [Dej86][MKK86][Sil86][Min90][MC90]. Explanation-base learning does not assume any particular representation for the domain theory. It can be a logical theory, a neural network, or even an approximate qualitative physics based theory. An explanation-based learner tries to explain a new instance in terms of its background, and then creates as many new descriptions as possible by using this explanation. They produce a description of the concept that enables instances of the concept to be recognized efficiently and in an effective manner.

The primary advantage of the explanation-based learning formulation of the robot learning problem is that it provides a way of incorporating previous domain knowledge or bias to speed up learning. The savings in number of examples needed to learn the policy could be quite significant.

In spite of many advantages of this learning method, there are also some disadvantages. The requirement of complete, consistent, and traceable background knowledge not being available in most real systems is one of these disadvantages [Mic90]. In addition, if the domain theory is very approximate, the learning system requires to exercise some care in using the theory to guide its generalization. Much work remains to be done in addressing these issues, especially in robot learning problems.

2.2.1.1.3. Reinforcement Learning

Reinforcement Learning (RL) studies the problem of learning by trial and error, a policy that maximizes a fixed performance measure, or reward [BS83][Kae93][Sut90]. It is a trial and error paradigm, so the examples are not carefully chosen by the teacher. Instead, since the states and rewards experienced by the system depend on the actions it takes, the distribution of examples is influenced by its actions.

In other words, RL is a method for learning how to map situations in to actions, so the learner does not learn which actions to take but rather must discover the action based on the reward taken from it. In some cases not only the immediate

rewards are affected by the action taken, but also the future rewards could be affected. This makes trail-and-error and delayed reward characteristics the most important features of reinforcement learning.

Besides the learner and the environment, four other elements of the RL are the *policy, reward function, value function,* and *model* of the environment (optional). A policy defines the learner's behaviour pattern at a given time, i.e. a set of stimulus-response rules or associations. A reward function specifies the immediate goal, whereas the value function specifies the long-term goal. And finally the model of the environment is in charge of mimicking the environment and is used for planning or deciding on the course of action based on considered future situations.

Reinforcement learning does not require a domain theory (as required by explanation-based learners), which might be a substantial undertaking in any real world robotics task. Also, RL can be used for online learning in contrast to some types of inductive learning methods, so the robot continually improves its performance. In addition, for many tasks, it is relatively straightforward to supply the robot with appropriate reward functions.

On the other hand, reinforcement learning suffers from a number of limitations. Sometimes, requiring many steps to converge, it can be very slow. This is due to the fact that to obtain a better reward, the RL agent should try all actions in order to be able to compare the rewards yielded by them to the previously tried producing better reward actions. Moreover, in corporation of domain

knowledge to speed up the learning is difficult. Also, RL assumes that the current state and the action determines a probability distribution on future states, and in other words the environment can be modeled as Markov decision process [Put94]. The robots can rarely sense the true state of their environment, due to the lack of high-tech cheap sensory devices, hence this "memory-less" property is most likely to be false.

2.2.1.1.5. Evolutionary Learning

Evolutionary learning includes Genetic Algorithms [Gol89] and Genetic Programming [Koz93]. A population of policies is chosen as the starting point, and combined to achieve better policies until an optimal policy is produced. Genetic operators such as crossover and mutation are responsible for this combination. In order to measure the quality of the achieved policy, the paradigm uses a fitness function.

A number of studies involving robots' training has been performed using evolutionary learning method. The training includes obstacle avoidance, navigation around a close area, and learning goal-seeking behaviours [GS94][Dor96].

Important advantages of the evolutionary learning paradigm include the ability to start in a pre-set state – allowing speed learning – and being able to learn arbitrary policies – against being constrained to stationary policies.

The key disadvantage of this paradigm is the limitation on off-line learning. Before testing the learnt policies on a real robotic platform, all training should be performed on a simulator.

2.2.2. Learning to Control Dynamic Systems

Controlling dynamic systems, specially the complex ones, require such skills that can not be practically explained using words, but can be demonstrated. This section presents some basic background on this subject. Several approaches have been so far used to build controllers for dynamic systems. Machine learning [SUKM92][UJ92][Ben96], Optimal control [SS82], fuzzy control [Lee90][YB94], neural networks [MS90], and expert systems [Dev87] can be named as some of these methods. Several papers [WL90] have discussed the capabilities and limitations of these approaches.

In order to construct a controller for a physical process by classical methods, constructing a model of the system is necessary. Unfortunately, it is often very difficult to construct an accurate model of such systems. This is due to the complexity and lack of information about the environment of many physical systems.

An alternative approach to deal with this problem is using qualitative reasoning [Mak91][Bra91][Bra93][BU95][Bra97]. Another approach is learning from experience [MC68][SS85][SUKM92][ESM94][ESS95][UB94][Shi94]. In this method, the system can learn by emulating the behaviour of a skilled operator

(behavioural Cloning), that is "Learning by observing a human operator", or it may perform trials repeatedly until a given success criterion is met. This is known as "Learning by trial and error". These two methods are explained in more detail in the following sections.

In contrast to the success of the behavioural cloning in some applications , qualitative reasoning also has been used in some experiments such as pole balancing [Mak91] and discovering dynamics [Dze93], and is the one more widely studied and used.

2.2.2.1. Learning by trial-and-error

In this method, the learning agent repeatedly performs trials in order to gain knowledge about the effects of different control decisions. Often, a trial starts by positioning the system in a random state, and ends when a failure occurs or successful control is performed for a specified period of time. Repeated trials stop when a certain success criterion is met.

The BOXES algorithm is one of the earliest trial-and-error learning methods [MC68]. The algorithm has been used as a benchmark for many studies of subsequent machine learning methods. It also demonstrated an effective and flexible method for learning to control physical dynamic systems.

In many situations the learning system is notified about an incorrect action after some delay. Therefore, it is difficult for the learning system to find out which of its actions has caused the failure. It is known as "The Credit Assignment

Problem" which is one of the major problems that a trial-and-error learning system must deal with. In other word, credit assignment is how to assign the credit or blame for a final output among several outputs that lead to the final output.

A further problem with trial-and-error learning is finding the best trade-off between "exploitation" and "exploration". "Exploitation" means using what has already been learned trying to obtain the best performance. On the other hand, "exploitation" means possibly sacrificing performance versus gaining more information about the system by trying something new. The BOXES algorithm and Watkin's [Wat92] Q-learning provide some solutions to this problem.

One of the important advantages of trial-and-error systems is that they do not require a teacher. However, the slow rate of learning and the necessity to solve additional problems such as credit assignment and balancing between exploration versus exploitation, make it difficult to scale these methods up to problems with big dimensionality. This method is effective when the domain is simple like the pole and cart. However, when the domain is complicated (e.g. navigation or flight control) dealing with a large search space makes this method non-applicable. Furthermore, larger search spaces will cause slower rates of learning, and sometimes the big search space can kill the learning process.

2.2.2.2. Learning by observing a human operator

Although learning by trial-and-error can be used for controlling physical systems, it can be very time consuming. The alternative method is to emulate the behaviour of a skilled operator. So far only a few attempts have been made to build a controller using this method [MBM90][SUKM92][UB93] [ESS95][SSM95].

One approach is to extract the skill from the operator in a dialogue manner; i.e. question and answer, and then using the description to formalize and build an appropriate automatic controller. However, the problem is that the skills can be mostly described approximately and not completely; in other words, the skills are sub-cognitive. Thus, these descriptions can only be used as the guidelines for construction of the controllers.

Considering this problem, an alternative approach in transferring the skills would be to trace the operator's actions. The idea is to use them as examples, extracting the operational descriptions of the skill by machine learning techniques. This is known as ***"Behavioural Cloning"*** [Mic93], and contains three stages. First, a skilled operator is asked to control the system, while the states of the system along with the operator's action are logged in to a file. In the next stage, a learning algorithm is used to construct control the rules for the system from the logged information. And finally during the third stage, the rules are made operational on the system.

Behavioural cloning has been studied and used in various dynamic domains [BU95]. The summarized conclusion from these experiments and studies shows that successful clones have been produced in several domains. In some, the clone surpasses the skill of the operator (so-called ***Clean-Up effect***).

Bratko believes that the current approaches suffer from the lack of robustness in providing the perfect clone, clones do not show enough robustness following the changes in control task, and finally they generally lack conceptual structure and representation. This is a clear sign of slow progress in the process of creating clones representing the sub-cognitive skills of operators. It is shown that often such representations do not match those operators work based upon [Bra97]. However, we have seen otherwise, and clear examples are learning to fly [SUKM92] and the experiment related to this thesis.

In addition, we can also name the neural network based system called ALVINN that learns to drive a real robot truck by Pomerleau [CM93]. Although he has used the training by observation method, but since the work classifies under supervised learning and the work is based on neural networks, a more detailed study of his work will be included later in this chapter.

2.2.2.2.1. Learning to Fly

Learning to fly an aircraft is one of the earliest successful learning systems that used behavioural cloning strategy [SUKM92]. The aim of this experiment was

to automatically build a controller to pilot an aircraft by observing the operation of a skilled human operator during the task.

For the experiments a Cessna aircraft flight simulator on a Silicon Graphic computer was used, limiting the flight to a predefined route. The core of the simulator is a loop, which receives control input and updates the state of the simulation according to a set of motion equations.

Human experts were asked to fly the aircraft thirty times. The aim of this project was to build a learning system to emulate the behaviour of a particular pilot (behavioural cloning). Three pilots were used to demonstrate repeatability of the experiment. Different pilots have their own strategies for controlling an aircraft. This strategy is different for each pilot even in the same flight plan. Thus, induction was restricted to one set of pilot data at a time and a separate set of controllers was constructed for each pilot.

During a flight, different stages may require different strategies. In this experiment the flight path was broken into seven different maneuvers. These maneuvers are take off, level out and fly to a specific distance, turn right, turn left, lining up on the runway, descend to the runway and land. This technique also reduced the complexity of the system.

During each flight, information about the flight status was logged in to a data file only when an action was taken. This information contains the state of the system at that instant and the response of the pilot to that situation (log action

and state). During each flight, up to 1000 events can be recorded, while an event refers to the performance of a control action.

The accumulated file of thirty flights for each pilot was segmented into seven files corresponding to the seven stages. In each stage, four separate decision trees were constructed, one for each of the elevators, ailerons, thrust, and flaps (classes) using C4.5 [Qui93] as the induction program. For the induction task, it is necessary to have a set of training examples. Each example in this set must be described as a collection of attributes. The class of each example also must be known in advance. A program filters the flight logs of each stage and generates four input files for the induction program. The logged flight parameters in the data file are the attributes of the training examples. The class value or dependent variable is the attribute describing a control action. For example, in generating a decision tree for the elevators, the elevators column is considered to be the class value and the other columns in the file are considered to be ordinary attributes. It is the same when generating decision trees for flaps, ailerons, and elevators.

The decision trees were then converted into if-statements in "C" by using a post-processor. The auto-pilot code of the flight simulator was replaced by the induced rules of all the decision trees to test the correctness of these rules.

During this experiment, Sammut and his colleagues encountered a number of problems. It is essential to record the state of the system along with the response of the pilot to that state when an event occurs. However, there is

always a delay in human response to a stimulus. This delay is known as the ***response time***. In order to log accurate data, the state of the system should be logged sometimes before the action. To be able to do this, the response time of the pilot must be estimated. Clearly, pilot response time will vary according to person and state. The response time of a pilot depending on the position in the flight and the type of reaction required can vary considerably. Moreover, during the flight, the pilot usually anticipates where the aircraft will be in near future and prepares the response before the stimulus happen [SUKM92].

To overcome this problem, Sammut and his colleagues used a circular buffer to store the current state of the simulation each time the simulator passes its main control loop. When a control action is performed, the action is logged along with the previous state of the simulation from the circular buffer. Although by these techniques, they solved the problems in some sense and the learning process was successful, they stated that the problem of "what is the actual delay between the stimulus and the action" is still unsolved.

The C4.5 program [Qui87b][Qui93] was used as the induction program. Similar results have since been obtained using a regression tree algorithm [Bre84][Kar92].

Another problem reported by Sammut and colleagues [SUKM92] is that the induced rules in some stages are very complex and difficult to understand by an expert. This problem exists because the learning system constructs rules based on the primitive attributes. Introducing some high-level parameters

containing more information about the system may help producing smaller and more understandable decision tree [SUKM92].

Michie and Camacho also reported a similar experiment in learning to pilot an aircraft using a flight simulator for an F-16 combat plane [MC94]. They used the same technique as Sammut et al. in previous experiment [SUKM92]. In an attempt to improve the robustness of clones in the "Learning to Fly" experiment, Arentz introduced disturbance into the system. He successfully created a set of clones, which he claimed to be robust to disturbance.

2.2.2.2.2. Learning to Control Container Cranes

Traditional control theory sometimes can not be used to build a controller for some physical systems. Controlling a container crane is an example of such a system. Sakawa [SS82] has shown that due to some unpredictable factors such as wind, it is not possible to build an accurate controller by means of the traditional control theory.

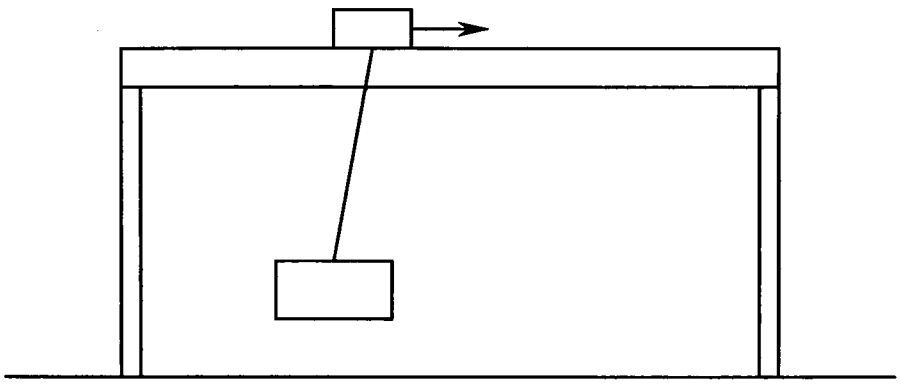


Figure 2-10. **A container crane**

An attempt at building a controller for a container crane was done using machine learning techniques [UB93]. In that experiment the aim was to automatically construct control rules from the recorded performance of a skilled operator following the strategy used [MBM90][SUKM92][MST94].

The crane is made of a trolley at the end of a rope, which are moved together. In this experiment, the task was to transport a container from one place to a target position. An operator, who picks up a load from some point and transports it to a goal point, performs the task. The speed of the trolley and the length of the rope can be altered in order to lift, transfer, or drop the load.

The basic performance requirements for this system as described by Urbančič and et al. [UB93] are:

- **Basic Safety:** Keeping the system within the defined limits and avoid any obstacle,
- **Stop-gap accuracy:** Keeping the gap between target and load positions within the defined limit, and
- **High capacity:** Minimizing the time of transportation.

To minimize the transportation time, two simultaneous operations should be performed. These operations bring the trolley above the target position and the container to the specified height. In this experiment, a real time simulator of a real crane is used. Six unskilled human operators were asked to learn to operate the crane using the simulation. All subjects succeeded in learning the task, working from 1 to 10 hours on the simulator. During the operation, state

variables were logged to the computer showing the state of the system. These variables were:

- *trolley's position,*
- *trolley's velocity,*
- *the rope inclination's angle,*
- *the rope inclination's angular velocity,*
- *the rope's length,*
- *the rope's length velocity.*

The RETIS program [Kar92] for regression tree construction was used as the induction program and every 0.1 seconds samples were recorded. Different levels of delay were considered in that experiment. However, at the end they decided to use zero delay for the response to any event. In the construction of the controller rules, they found it hard to find a common strategy among the recorded information from different operators. Thus, they decided to work on collected trials performed by the same operator instead of working on all of the information.

Using the recorded information they built a controller that they claim "... is conservative and minimizes the swinging but at the cost of time" [UB93].

As explained earlier, Urbančič and Bratko introduced the human operator's instruction into the "cloning cycle" by using six volunteers to learn to control a crane simulator. After the operators had mastered the task, they were asked to write down instructions for how to perform the task and were also encouraged

to discuss their experience and their written instructions with each other to improve their performance. A clone was also induced from a set of successful traces of each operator by using machine learning techniques. The clone was able to control the task in a manner similar to the human operator.

The main goal in this experiment was to establish a bridge between the induced clones and the operators' instructions to improve the transparency, robustness, and generality of the clones. Also, improving the operators' instructions using the information gained from the clones set to be the other goal. In the paper titled "Reconstructing Human Skill with Machine Learning" [UB94], they made it clear how these two sets of rules can be used to complement each other. In this paper, they explained how the induced clone could uncover some of the operator's subconscious control skill. However, they did not make it clear how the human operator's instruction can be used as background knowledge by the machine learning techniques, especially induction. Moreover, although human experts are able to control dynamic tasks quite easily, it is often hard or impossible for them to explain how they perform the task. Compton has argued that even when the operators provide some explanation about their strategy, it is often a justification for their action rather than the way they have reached their conclusion [Com92]. Furthermore, this justification depends on the context in which it is provided. Therefore, it is important to use the knowledge that the expert has provided in the context within which it has been acquired [CJ89]. Hence, the approach of Urbančič and

Bratko could be strengthened to provide a more reliable and accurate way of capturing the expert's explanations.

Later, Bratko in a further study has investigated reasons for the slow progress in generating clones which is directly in relation to the lack of conceptual structure and representation that would clearly capture structure and style of the operator's control strategy [Bra97]. He has claimed that one of the reasons is that since the usual representation used in reconstruction of the skill is inherited from traditional control theory, it is entirely numerical. He shows that more appropriate representations are largely qualitative and involve history and not just the current state of the system.

2.2.2.2.3 Learning to fly, Parvaz

Following Sammut and colleagues' work, Shiraz in his Ph.D. thesis used an interactive method [Shi97], in which the expert cooperates with the learning program to create the controller. Shiraz broke down the system in to two parts. He used Dynamic Ripple Down Rules (DRDR) for the parts of the flight simulator where the pilot had the ability of verbalizing the control strategy with the system. DRDR is a modification of the Ripple Down Rules (RDR) which has been developed for dynamic system control. For other parts, where it was difficult or even impossible for the pilot to formulate the strategy, he also developed Learning Dynamic Ripple Down Rules (LDRDR). LDRDR automatically produces rules from the logged data from the pilots actions instead of C4.5 which was used in "Learning to Fly" experiments by Sammut

and colleagues [SSE95]. A schematic diagram presenting the structure of Parvaz is presented in Figure 2-11. He claims that the use of a pilot's advice resulted in the creation of more transparent and robust rules [SS95]. However as he also cites that the major limitation of his work was the separation of the knowledge acquisition and learning part.

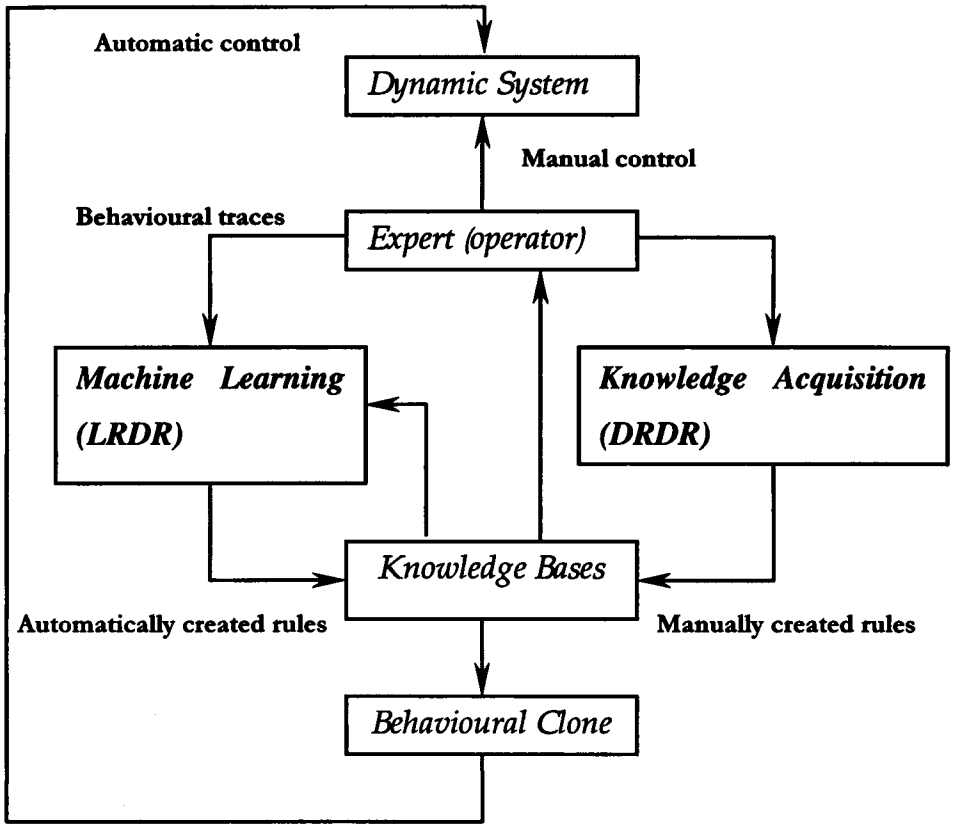


Figure 2-11. **The basic structure of Parvaz**

He presented a new machine learning program using sequential data and called it LDRDR, short for Learning Dynamic Ripple Down Rules [Shi97]. The real reason behind the introduction of this program was the need for a learning program capable of dealing with sequential data and also its compatibility towards DRDR that is an incremental learning tool. DRDR is also a

modification of RDR, a simple and powerful method for knowledge acquisition and representation. The basic form of a ripple-down rule is and if statement:

If *condition* **then** *conclusion* **because** *case* **except**

If *condition* **then** *conclusion* **because** *case* **except**

If ...

else if ...

And in the absence of any other information, the RDR recommends taking the default action, that is:

If *true* **then** default *conclusion* **because** *default case*

In his experiments, Shiraz also divided the flight procedure in to seven stages. For each stage he created four separate RDRs, each for every control action (elevators, flaps, ailerons and throttle). Each RDR (28 altogether) was constructed either using DRDR interactive knowledge acquisition method or by applying Induct/RDR (LDRDR) to the logged data. He applied both machine learning and knowledge acquisition procedures independently.

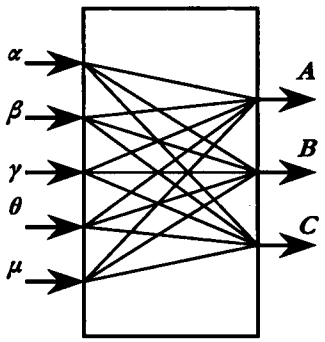
2.2.2.2.4. Flying with CHURPS

In another experiment, Stirling for his Ph.D. experiments employed CHURPs (Compressed Heuristic Universal Reaction Planners) to achieve a new technique for uncovering and synthesizing control skills evolved by human

pilots. He developed CHURPS as a method to capture human control knowledge.

He observes that behavioural cloning seeks to learn control skills by observing and learning the interactions between the machine/process and human operator/agent. However, he claims that "there are fundamental problems about the scope of training exemplars one must provide to obtain robust and reasonably generalized skills" [Sti95a]. Thus, using CHURPs he tends to overcome the problems of brittleness and generality, by separating the planning and execution phases. He has actually emphasized on robust controllers built to stand failure in actuators.

To do so, his approach tends to acquire a starting point from the expert from which the controller can be automatically generated.



Effector set allocations			
SE	A	B	C
UE	—	μ	—
ME	γ	β	θ
SE	α,β	α,θ	β,γ

Controls	Goals		
	A	B	C
α	0.6	0.2	0.0
β	0.1	0.7	0.3
γ	0.8	0.0	0.6
θ	0.0	0.4	0.8
μ	0.0	0.2	0.0

Figure 2-12. **(a) An example of a plant control system**
(b) Agent's effector view of the system
(c) Influence matrix for control inputs over output goals

"Influence Factors", numbers in the range of 0 to 1 specifying the level of effect of input on an output goal, are asked from the expert for a certain system. As shown in Figure 2-12, in the plant control system with 5 inputs and 3 output

goals, input or control action α has no effect on goal variable C, while it has greater effects on A and B. The control action α is called the main effector for goal variable A, meaning that it has the most influence on the goal variable A.

From the influence matrix, Figure 2-12 (b), for each goal variable there can be three sets of control actions:

- **UE**, a *Unique Effector*, the only effector influencing the goal variable
- **ME**, a *Maximal Effector*, the one with highest level of influence over the goal variable
- **SE**, *Secondary Effectors*, all effectors over a goal variable except the maximal effector

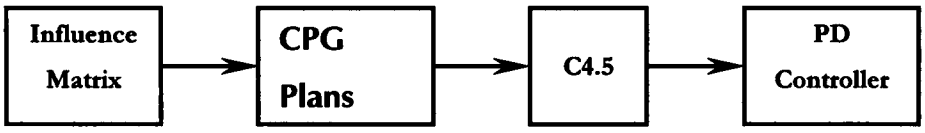


Figure 2-13. **CHURPS Architecture**

Stirling has used these three sets for his CPG (Control Plan Generator) algorithm to generate plans for the operation control. According to this algorithm, an UE control action can not be chosen as ME or SE for other goal variables and also an SE control action has the least effects on the other goal variables.

Stirling proposes that in the CHURPs paradigm, only the planning skill is learnt using machine learning technique and the skills can be modeled by common

feedback control loops. He also assumes that any complex task can be broken down in to a series of sub-tasks. Each sub-task has a clearly defined objective and target, and once the current sub-task's objective has been achieved the next sub-task is entered. Arguing that since both CHURPs and the behavioural cloning auto-pilots were constructed using the same learning method, he suggests that the difference in their performance is due to the source of examples and the addition of multi-controller feed-back. Apart from the flight domain, he also has demonstrated CHURPs on other multi-input/multi-output domains [Sti95b].

2.2.2.3. Learning using neural networks

In the last two decades, we have witnessed a rapid increase in the use of neural network in different areas [WL90]. They have been applied in several different domains such as pattern recognition [CG83][WW88], Image processing [NP90], hand writing recognition [Bar90], natural language processing, and economics [BF91]. The ability of a neural network to deal with non-linear tasks make them a suitable choice for learning to control dynamic systems [HA90][JS90]. Auto-lander [JS90], Bio-reactor [Un90], and pole cart [WW88] are some examples of this.

The following section provides a brief review of the auto-lander project, which is related to some aspects of the work reported in this thesis.

2.2.2.3.1. ALVINN

ALVINN, Pomerleau's trainable road tracker is one of the world's most famous neural network applications. The specialty of ALVINN is not its driving capability, but it can also learn how to drive just by watching a human driver for a while [Pom93a]. ALVINN uses a training system designed in four layers. In the first layer the system uses a low resolution video camera to input three 30x32 pixel images, one for each of the video color bands (red, green and blue) to the neural network. These images are taken from the road ahead and the current position of the steering wheel. There is a connection between every single pixel in each of these three images and the corresponding pixel in the 30x32 unit array of hidden units, which is the second layer. The system was designed in such a way that there were only three distinct weights, one for each distinct color, between the color bands and the array of hidden units. On top of this layer four hidden units were used to connect the hidden unit array to the forth layer. Finally, the forth layer was made from 30 output units corresponding to the direction control units, ranging from sharp-left to straight-ahead and to sharp-right (figure 2-14).

ALVINN was able to drive on single lane, multi-lane and unpaved roads. To drive at night, it was equipped with laser reflectance imaging system. Using a laser range-finder unit, ALVINN was able to perform object avoidance task maintaining a fixed distance from the parked cars.

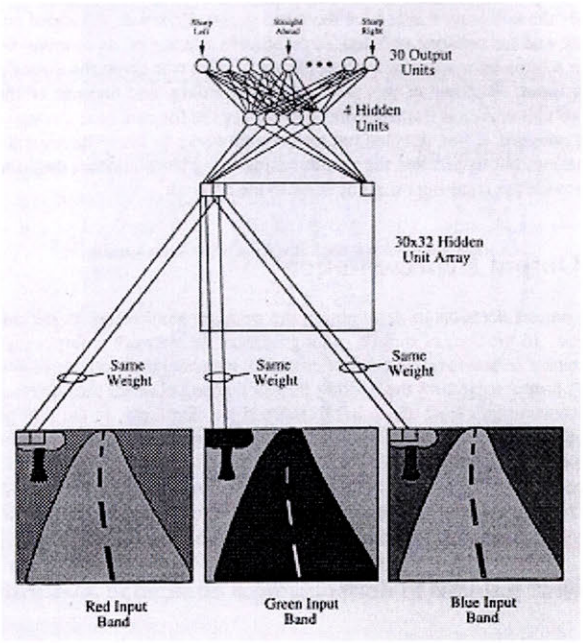


Figure 2-14. **The four-layered architecture used to train ALVINN**

Pomerleau used a new methodology for "on-the-fly" training in neural networks. In this methodology, the training set is kept small in order to meet the limitations of real-time processing. However, the examples were removed only when they were completely studied and efficiently learnt. In addition, his new methodology requires a training set representing a balance variety of cases.

To train on-the-fly (Figure 2-15), Pomerleau first considered generating artificial images of situations likely to be encountered by the robot, to ensure enough diversity in the training set. To do this, he developed a simulated road generator program. He then used this program to create 1200 examples used to train the network by randomly changing its parameters.

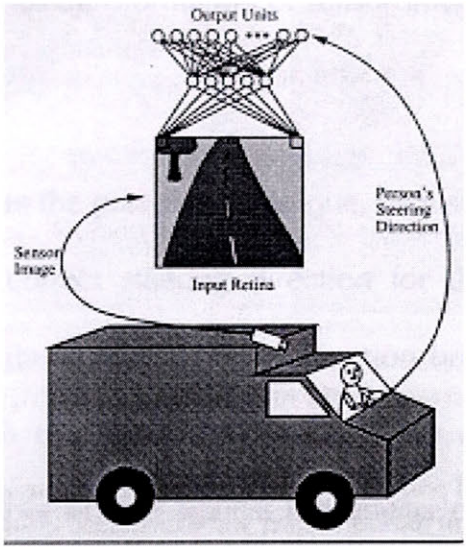


Figure 2-15. **Schematic representation of training "on-the-fly"**

There were serious drawbacks in this approach, ranging from the log time needed for artificial road generation, poor performance of the network due to the differences between the artificial roads and the real ones, and finally the need for prohibitively complex training data generators for multi-lane and off-road driving trainings. As a result, Pomerleau developed another training scheme in which the network imitates a human driver under actual driving conditions. He has called this technique, "*training on-the-fly*". He believes that training on real images would need lesser human effort to develop networks for new situations [Pom93b].

There were also two potential problems associated with training on live sensor images, recovering from misalignment errors due to road center concentrated training and over-learning recent inputs due to training with only the current image and steering direction. To prevent those problems Pomerleau came up

with another technique for transformation of sensor images to create additional training exemplars.

Furthermore, to complete the previous technique, he also used the pure pursuit steering to adopt the correct steering direction for the transformed sensor images. In this model, the correct steering direction brings the vehicle to the desired location (which usually is the center of the road) a fixed distance ahead. In order to further ensure against the effects of repeated exemplars, Pomerleau increased the diversity of the training set by maintaining a buffer of previously encountered training patterns. In this method, the newly digitized and transformed sensor images are added to the buffer while the older patterns matching these new ones are removed from it. This will ensure having an updated history of the recently encountered driving situations in the training pattern buffer.

Pomerleau claims that training on-the-fly scheme gives ALVINN a better flexibility that is new among other navigational control systems.

2.2.2.3.2. Auto-lander

An auto-lander is usually available in most commercial aircraft. However, they are not designed to handle large winds and turbulence [AM90]. To explore an alternative for the current auto-landers, Jorgensen and Schley suggested using a neural network [JS90].

They believed, because of the capability of the neural network to generate a map from large set of variables (e.g. sensors) to another set of variables (e.g. control actions), they may be able to capture some critical behaviour of the human pilot. A pilot's skills, usually developed through years of experience, help the pilot to make appropriate responses in a new state. They are called sub-cognitive skills and are very difficult to explore.

Jorgensen and Schley believed also that the use of neural networks might make it possible to capture some of variable interrelationships in an aircraft, which are not normally considered by a design engineer.

Jorgensen and Schley in their experiments found that it is impossible to capture the performance capabilities of a linear auto-lander controller by a single network. Therefore, they decided instead to use a set of neural networks. They used a simple and linearized mathematical model of a controller to train these networks. Their model considers just longitudinal and vertical movement of the aircraft. To simulate some of environmental disturbances, they modeled head and tail wind. This model exhibits wind shear at different altitudes.

The auto-lander system has four major components:

a) ***Auto throttle:*** To maintain constant airspeed

b) ***Pitch auto-pilot:*** To provide adaptive damping and a speed response to desired pitch attitude

c) ***Wind disturbance calculation:*** To generate random gusts of head or tail winds.

d) ***Glide slope and flare controller:*** To provide pitch commands in response to desired altitude and altitude rate of change.

The input to the system consists of current altitude, altitude rate of change, desired altitude, and desired altitude rate of change. The goal of the system is to generate elevator angles at any time to land the aircraft on the runway.

In this experiment, Jorgensen and Schley successfully generated a set of networks that could land the aircraft in the presence of headwinds and tailwind.

This work was limited in that it used a very simple flight model and only considered the elevator actions during landing, not an entire flight. Moreover, it did not explore the areas of interest in this thesis, namely combining human and machine knowledge.

2.2.2.4. Learning using Fuzzy Logic

2.2.2.4.1. Mobile Robot Navigation

Recently there have been many attempts at mobile robot navigation using *Fuzzy Logic* techniques. Aycard and colleagues have shown a new method to design, in two levels, a fuzzy controller for reactive navigation of a mobile robot in a structured unknown environment [ACH97]. Using a Nomad200 robot for their behaviour based control of reactive navigation, they have conducted two experiments with different local behaviours and different

mechanism of integration. At the first level, the adjacent sensors of the robot were grouped in areas and were used to define the local behaviours, which were later in the second level gathered in order to define a global behaviour.

Adaptive behavioural capabilities are necessary for robust mobile robot navigation in non-engineered environments. Robust behaviour requires that uncertainty be accommodated in the robot control system, especially when autonomy is desired. Claiming that fuzzy logic control technology enables development of controllers which can provide the necessary computational intelligence in real-time, Tunstel et al. have presented the incorporation of fuzzy logic, into the framework of behaviour-based control [TDLJ97]. They have implemented an architecture for hierarchical behaviour control in which control decisions result from a consensus of behavioural recommendations. Applying multiple fuzzy-behaviour coordination to autonomous navigation without explicit maps, they have declared that performance and robustness is demonstrated by implementation on a mobile robot with significant mechanical imperfections.

2.3. Discussion and Conclusion

In this chapter we presented the background work on both navigation and learning approaches. The first section covered different approaches for localised navigation, while the second attempted to cover as much as possible a complete related survey of various methods of machine learning.

In the navigation section, it was clearly pointed out that the two widely used navigational methods (i.e. localised and global) are clearly separable and distinguishable by means of implemented strategies and data processing. Consequently, in this thesis, we have proposed to use reactive control of a locally developed mobile robot. We gradually develop the navigation system based on the previously attempted strategies, however the difference is that the strategies are not hard coded and will be learnt eventually by the robot during trial sessions.

In the second section, it was shown that machine learning techniques could be successful in constructing a controller for complex dynamic systems such as flying an aircraft and robot navigation, in simulation or using physical platforms. The main focus of this section was on describing behavioural cloning (behaviour based control) which seems to be one of the promising methods for building controllers.

It has been observed that behavioural cloning, seemingly, has been successful in building control systems. However, many problems remain to be solved, and much more research is to be done before this method can be practically applicable.

In our experiments, we have concentrated on applicability of a combination of behavioural cloning and other conventional navigational methods to a physical system. At first, we have tried simple and primitive obstacle avoidance behaviour by observation and changing the training strategy.

3. Robotic Platforms

Experiments related to this thesis have been performed in two different phases. Two different robots have been used, one for each phase. There are similarities between both robots, and the major difference lies in the way the experiments have been performed. Robots have been used to acquire the primitive behaviour of object avoidance. Also, two different algorithms have been applied for comparison purposes.

3.1. Purpose Built Robot

The robot was designed and built by the author, based on the previous physical frame by Graham Mann (Graham built the base and frame for a different purpose, no reference is available on his work though). The structure of the robot was based on an IBM-Compatible 386 computer. It adopted a simple peripheral interface to the main board via Xeltek MCP-550 data acquisition and MCP-330 digital interface cards. The physical structure of the robot is shown in Figure 4-1. The robot is made in three individual separable sections, top, middle and the base. Located atop the robot is an ultra-sound rotating scanner, the computer and motor controllers are in the middle section, and all the power related parts and motors are also residing on the base section.

The overall system architecture of the robot is shown in Fig. 4-2. The left-hand side illustrates the sensors, while the lower right section shows the computer and communication boards. The picture also shows the interconnection between different parts of the robot. The different parts and sections of the robot are described in details in the following sections.

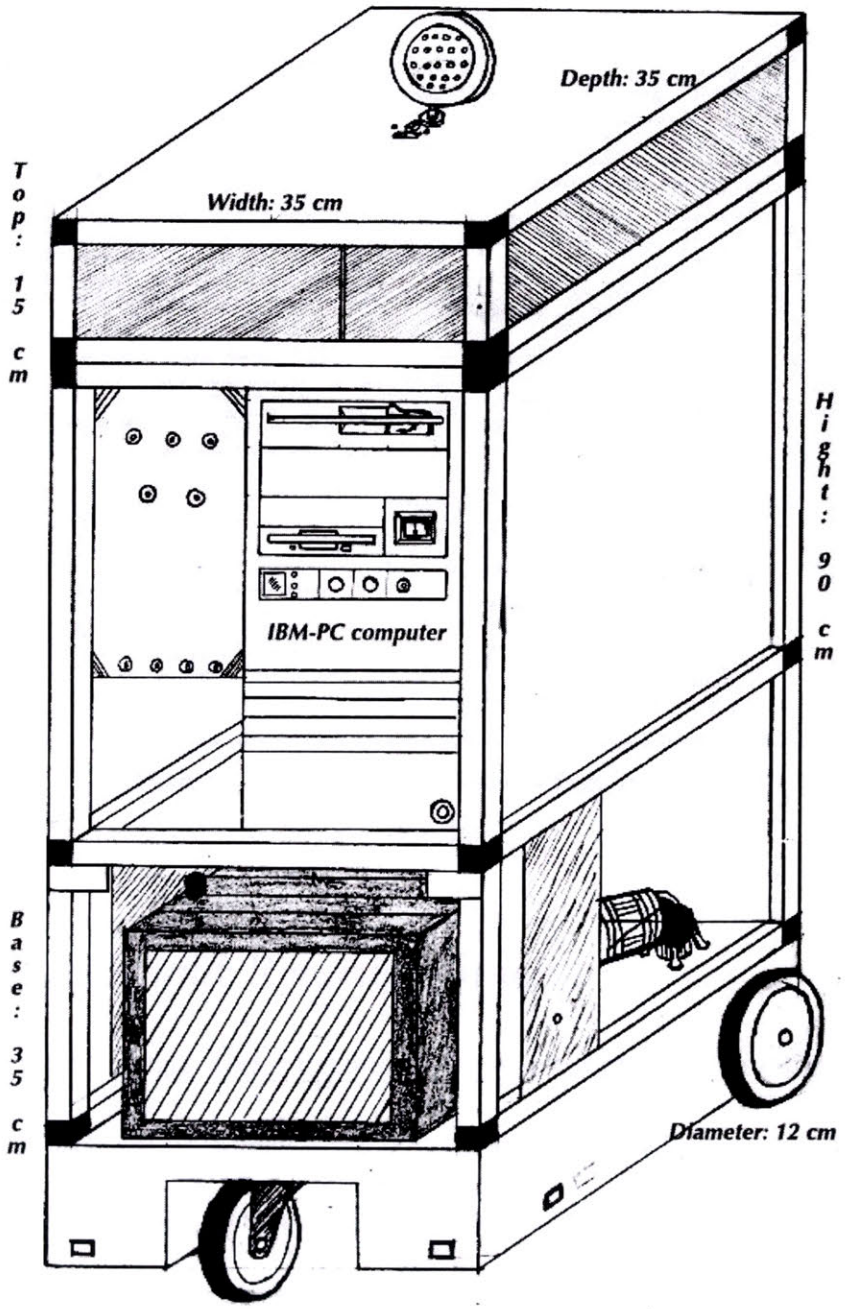


Figure 4-1. **The Robot Sketch**

3.1.1 Sensors

The purpose built robot is equipped with a rotary ultra-sound sensor, as well as five infrared proximity sensors. These sensors are used for long distance object detection and close proximity obstacle avoidance.

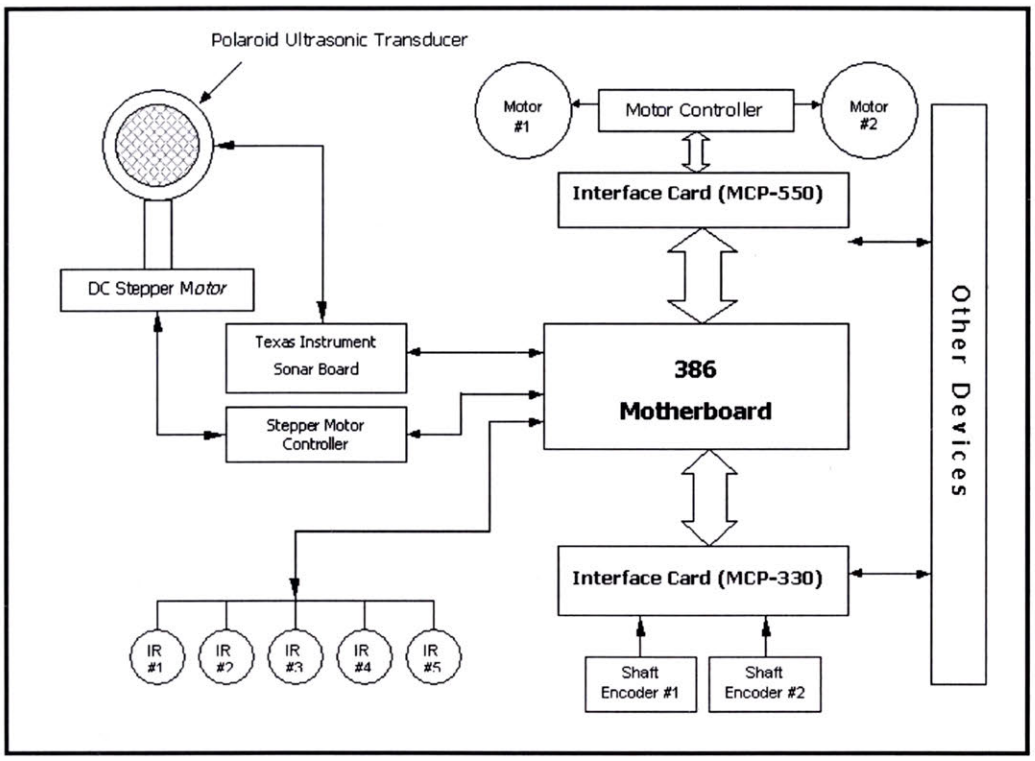


Figure 4-2. **Overall system architecture of Purpose-Built Robot**

3.1.1.1. Ultra-Sound Rotary Scanner

A rotary ultrasonic range finder mounted atop the robot scans the surroundings continuously, generating a 48-value polar proximity map from a 360-degree scan every 15 seconds. The stepper motor in charge of rotation has the resolution of 7.5 degrees.

The sonar ranging module is made by Texas Instrument, which matches the Polaroid Ultrasonic transducer. For each reading, there should be an initialization from the controller board to the transducer, followed by emition of a series of vibrations which sounds as a single click noise to human ears. Then the ranging system goes to receiving mode, waiting for an echo to receive from the object(s) ahead in case there is any. In the meantime, an external counting system which is also initialized at the beginning will continue counting and will only stop when either an echo is received or its buffer is full. If the reading shows an overflow in the counter value, it simply means that there is not sensible object in th evicinity of the next 10 meters.

The ranging module is capable of operation in both single-echo and multiple-echo mode. We have only used the module in single-echo mode, so in any sigle reading only the closest object in that direction is sensed.

3.1.1.2. Infrared Sensors

The robot is also equipped with five infra-red proximity sensors (modulated IR detector, coupled transmitter and receiver) arranged around the base half-way up the robot; each sets one bit of a digital input port (on MCP-330 board) if an object falls within range of approximately 10 to 15cm.

The IS471F is an infrared detector with integrated modulation system. It is a perfect device to be used in high atmosphere light conditions by elimination of detected unwanted light emissions.

3.1.2. Motors, Controller and Shaft Encoders

The robot has employed two motors on its base. There is also a gearbox located under the motors and connected to the side wheels, facilitating the engagement of the motors with the wheels during the start-up moment and stop moment of the operations. The gearbox acts as a damper during the stop procedure, stopping the robot without any residual movement.

The robot is also equipped with a motor controller system, in order to command the motors. We have chosen the current control method, so it is possible to drive the motors using *Pulse Width Modulation* (PWM) in order to save power and avoid over-consumption of the battery module. Generally there are two types of motor controllers in respect to the design, *Linear Servo-Amplifier* based and *PWM Servo-Amplifier* based. The PWM based controller will keep the driving transistor in switching mode, causing less power consumed as well as lesser time needed for switching.

In many applications it is preferred that the motor's driving current to be directly controlled by the input command (signal). This is simply because the developed torque in the motor that is the most important factor for creation of the rotational motion in speed/position controls is directly proportional to the motor's driving current:

$$m = K_T i_m$$

where:

m = instantaneous torque

i_m = motor's driving current

K_T = torque constant of the motor

So, the servo-amplifier is designed in such a way that the input signal controls the motor's driving current, so by changing the level of current we can change the speed of the motors.

The current control signal is generated by a Pulse Width Modulation (PWM) technique. To do this a triangular wave generator is used in conjunction to the speed command, which is created by the built-in communication board (using software) and fed to an operational amplifier (as a level comparator). We define the analogue command signal to be modulated by the source (triangular -- modulator) signal (Figure 4-3).

The chosen motors can be directly driven forward or backward, simply by reversing the applied voltage direction. This has been made possible using simple relay modules. Figure 4-4. Shows a block diagram of the motor controller designed specifically for this robot.

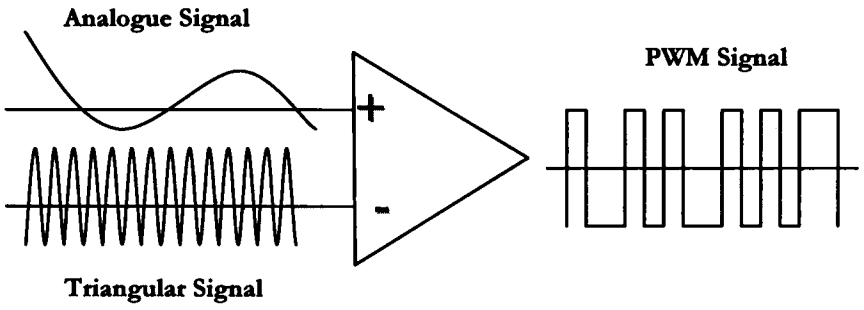


Figure 4-3. **PWM generation**

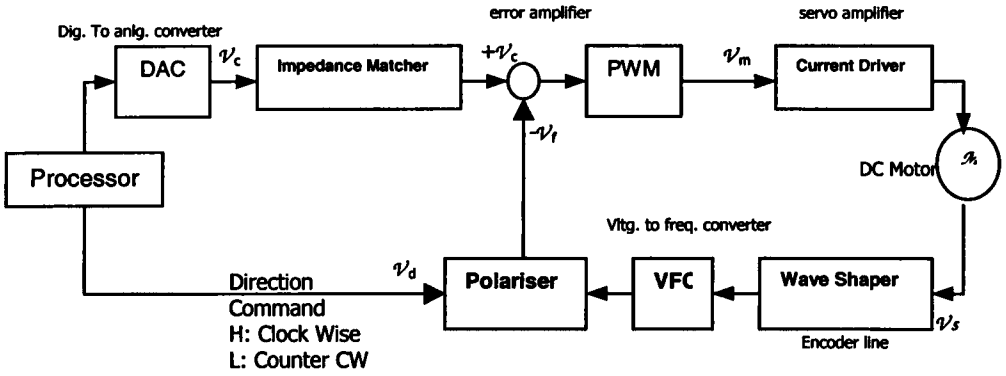


Figure 4-4. **Motor controller block diagram**

It has been observed that while both motors are working simultaneously, whether forward or backward, there will be race condition in which one motor rotates faster than the other. To prevent this, we have designed a couple of shaft encoders, located on the shafts of the motors. The encoders are in a clear circular disk with 36 black stripes lying from the center towards the edge of the circle. The main purpose of the shaft encoders is to measure the wheel rotation. The encoders rotate within standard infrared optical units (encapsulated transmitter and receiver), connected to a pair of 8 bits counters through signal conditioning circuits creating square wave signals. Each transition from clear zone to a black zone is counted as one tick, 36 zones in total, 10-degree resolution.

To detect any racing between the motors, it is imperative to check the shaft encoders continuously only when both motors are driven at the same time in one direction. Figure 4-5 illustrates a flow chart of the process responsible for taking care of the race condition. Each motor can be set on two different speeds in both forward and backward directions.

3.1.3. Power Supplies

Power is provided by a 12-Volt/40 Amps high-duty car battery, which is then

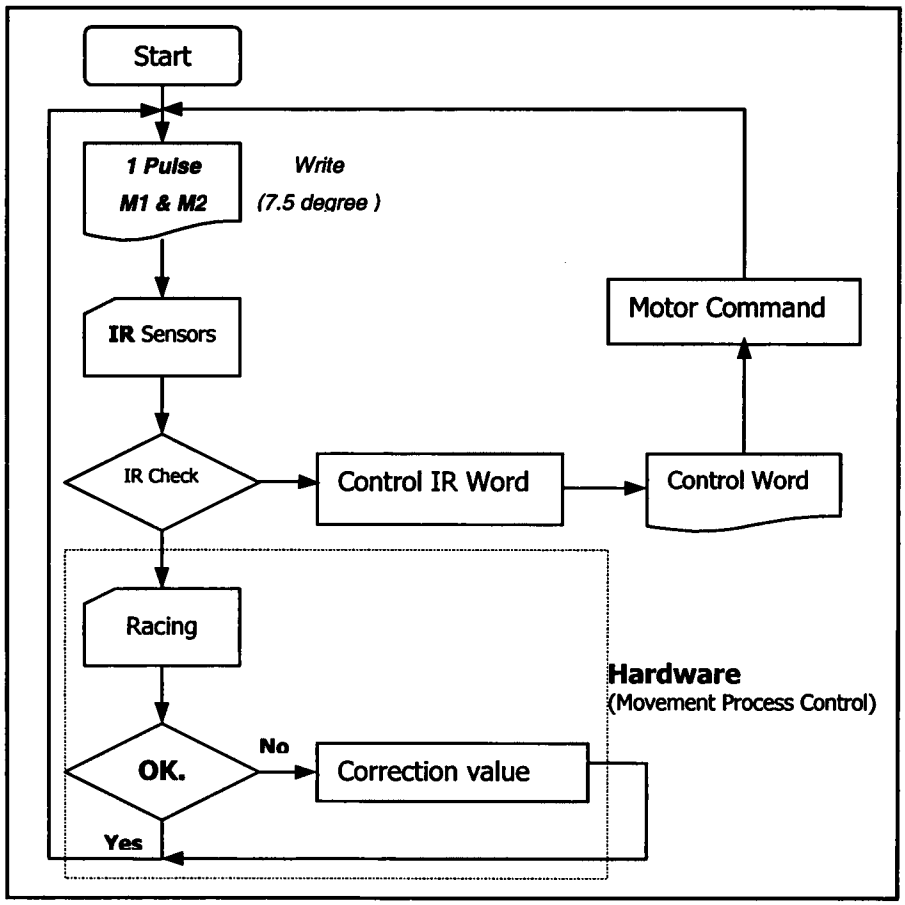


Figure 4-5. **Flow chart of motor control**

fed through a combination of converters to different parts of the robot,

providing positive and negative 5-Volt and negative 12-Volt. The 5-Volt converter also provides the fine +5-Volt necessary for the motherboard.

Also, the robot is provided with a power supply supervisor chip, which is responsible for lower power warnings. The power supply supervisor unit RS3543 contains full monitoring and shutdown control systems. It is suitable for both linear and switch mode power supplies, and is programmable for over-voltage and under-voltage protection.

3.1.4. Computer and Communication Boards

The robot is equipped with two communication boards, which sit in two slots in the IBM compatible computer. MCP-550 data acquisition and MCP-330 digital input/output cards are products of Xeltek.

3.1.4.1. MCP-550 data Acquisition Card

The data acquisition system converts the raw analogue/digital outputs from transducer readings into equivalent digital signals/data usable for further processing. MCP-550 card provides an analogue Multiplexer, a sample and hold circuit, two Digital to Analogue Converters (DAC), two Analogue to Digital Converters (ADC) with reference, programmable clock, and buffers. Fast speed and multi-function data acquisitions are the main features of this card.

DAC ports are used for motor control task, while the timer/counters are used for shaft encoders reading. The output of each digital to analogue converter is

connected to the reference analogue voltage of the comparator described in previous section, and then modulated with a triangular waveform generating the pulse width modulation driving power for the robot's motors. The level of the analogue voltage determines the speed of the robot.

The ADC (analogue port) was used for joystick control reading. The joystick was made of 4 linear potentiometers, one for each direction of movement. The analogue reading from joystick is changed to digital and then based on the position of the stick, the control command will be issued through the two DAC ports.

3.1.4.2. MCP-330 Digital Input/Output Card

The general interface method among personal computers is programmable digital input and output registers. MCP-330 card provides 32 digital input channels, 32 digital output channels and 2 interrupt functions. The digital input/output channels are TTL compatible, but the board provides higher driving capacity for digital output channels and lower loading current consumption for digital input channels than normal TTL circuits.

The ultra-sound scanning system's counter is connected to 16 digital input channels, while 8 input channels have been used for infrared proximity sensors.

3.1.5. The Main Control Program Loop

The robot is programmed by two simple nested control loops. The program is responsible for sensors checking, movement commands, and data logging.

At the beginning of the program, the sonar transducer, MCP-330 and MCP-550 boards are initialized. The inner loop is repeated four times, before the outer loop can be completed once. The outer loop is only for sonar readings. It takes 4 seconds to scan the full 360º by the sonar transducer, forcing us to do it in four equal intervals. After every quarter of scanning, the inner loop starts, and monitors the infrared sensors, battery level, and joystick movement respectively.

Also, the loop contains file initialization and manipulation processes for data logging purposes. The logging can be turned on and off using a physical switch connected to one of digital input channels of MCP-330 board, which is also monitored within the inner loop. The main purpose of this switch is to prevent the data logging while in transition and movement with out a specific target; i.e. when the robot is moved from one location to another outside of the training periods.

3.2. Upgrade and Changes

In the continuation of our experiments, the robot was upgraded to use an IBM-compatible 486 processor based computer and also the interfaced boards were replaced with a Motorola M68HC11 Micro-controller based mini-board (Mini-

computer). This change was followed by the change to a newer sonar transducer, which is basically the same as the original one, but the only difference is the controller that is matched to the mini-board controller and is designed for this specific transducer, which makes it faster. Also, the analogue joystick is replaced with a digital one specifically designed to be connected to the parallel port of the IBM compatible computer.

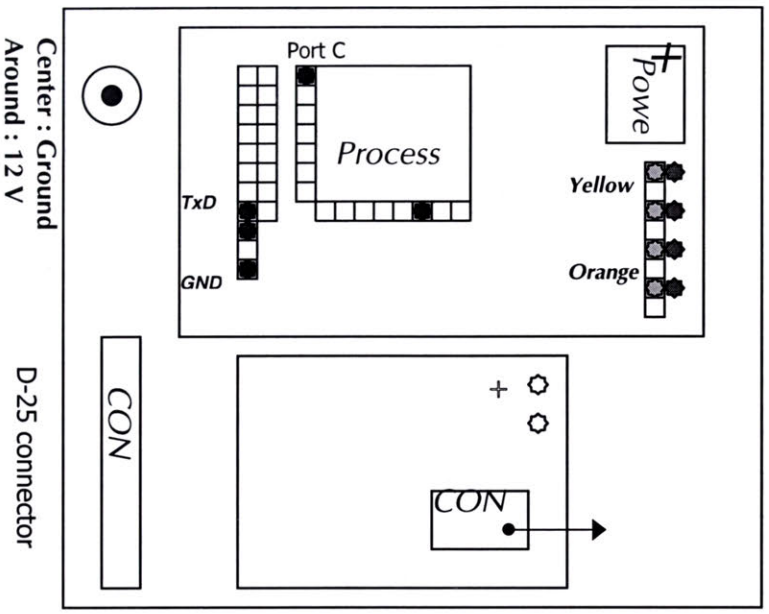
3.2.1. The 68HC11 Mini-Controller board

The mini-board 2.0, designed by Fred G. Martin at Media Laboratory of MIT, is a single board micro controller/computer designed to control small DC motors and receive data from a variety of electronic sensors. It is smaller than a business card in size, low power, and programmable. These features make the board ideal for mobile robot control. It is capable of communication with a desktop PC via a standard RS-232 serial port, which provides a good opportunity for desktop PC based control as well.

The mini-board is capable of:

- Control 4 DC motors using software based PWM in 16 different levels of speed
- Eight analogue inputs for analogue sensors and devices
- Eight digital inputs
- Three/four programmable counters
- RS-232 compatible serial port for communication with a PC

- Includes 256 bytes on internal memory on the 68HC11 chip, and takes 2K bytes EPROM for onboard programming purposes.



Connector CON2 is connected to the Mini-Board using a D9:
1- 5 V input (from regulator – Red)
2- Echo (orange)
3- Ground (Black)
4- Trigger (Blue)

Port A 3 : Echo (Orange)

Port C0 : Trigger , Sonar INIT (Blue D-9)
C1 : Stepper HOME (White D25-P21)
C2 : Left Shaft Encoder (Orange D25-P6)
C3 : Right Shaft Encoder (Red D25-P7)
C4 : Left Motor driver (Green D25-P8)
C5 : Left Motor Drctn (Yellow D25-P9)
C6 : Right Motor Drive (Cyan D25-P10)
C7 : Right Motor Drctn (Brown D25-P11)

D25 Connector :

P2	: RS232 RXD	P6	: Left Shaft Encoder, RED	P18	: Transducer, Supply
P3	: RS232 TXD	P7	: Right Shaft Encoder, GREEN	P19	: Transducer, GND
P5	: RS232 SG	P8	: Left Motor Drive, YELLOW	P21	: Transducer, HOME
P4,6,8	: Ground (D9 - Computer Side)	P9	: Left Motor Direction, CYAN	P22	: Stepper, YELLOW
P14	: Sonar – (NEG)	P10	: Right Motor Drive, BROWN	P23	: Stepper, ORANGE
P15	: Sonar + (POS)	P11	: Right Motor Direction, BLACK	P24	: Stepper, BROWN
		P12	: GND P13 : +5V	P25	: Stepper, RED

Figure 4-6. **Mini-board's wiring diagram**

Figure 3-6 is an illustration of the mini-board and its connections as used in our locally built robot. The board is connected to the stepper motor of the sonar scanning system, sonar transducer, motor control driving circuits, motors direction control, and RS-232 standard serial connector of the onboard PC.

One of the reasons for choosing 68HC11 based-controller/computer is the vast existence of the C cross compilers. One can simply write a control program in C and then use the cross compilers to compile it to the 68HC11 assembly

language. Furthermore, a small uploading program is used to send the translated program in to the onboard EPROM.

The motors are also connected directly to the Mini-Board, and are to be driven using pulse width modulation provided by micro-programmed pieces. Wheel's IR detectors (rotation counter from the shaft encoders) are also connected to internal counters on the Mini-Board.

3.2.2. The Parallel Port Control Box

The other change involved was the replacement of the joy-stick with an electronic control box, also providing switches and LEDs for better control and navigation. The box was connected to the PCs parallel port.

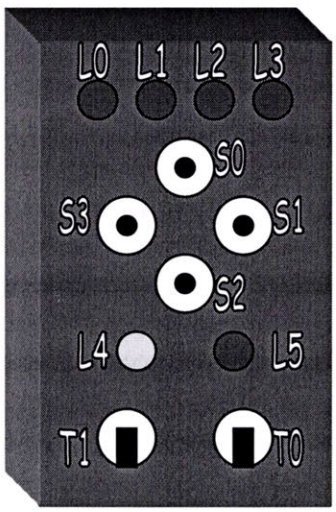


Figure 4-7. **Parallel port control box**

There are 4 push-buttons, 2 switches and 6 LEDs on the box. The following explains the specification and use of each part:

- S0 to S3 are push-buttons connecting to ground when pressed, and are used for direction control of the robot.

- L0 to L5 are LEDs and can be programmed for any purpose. For the moment, L4 and L5 are related to the situation of toggle switches T1 and T0 respectively.

- T0 and T1 are programmable toggle switches. The training program uses them just for recognition of automatic / manual control and data log on / off status of the system.

3.2.3. The Control Program

The control program explained in the previous sections has also been changed accordingly. The main loop has been broken in to two different parts. The sensors reading part has been changed and transferred on board of the 68HC11 based micro-controller board. The onboard program scans all the sensors and input devices, including the shaft encoder digital square wave signals, ultra-sound transducer echo signal, infrared sensors' output and the parallel port digital controller box. The program is also responsible for sending control signals to the sonar controller, and motor controllers.

The other part of the program is executed on the IBM compatible PC, and communicating with the mini-board processes the incoming data. It also logs the environment and robot's status data into a file in case the related switch is on (the switch is located on the parallel port control box). In addition, this program can be executed in two different modes, manual control and

automatic control. Within the automatic control mode, extra steps would be executed, which are explained in the next chapter under automatic control section.

3.3. Fander-I Robot

This robot is very similar in structure to what we have used before. The robot's architecture is illustrated using Figure 4-8, and the only difference is that there also exists a LCD display and a keypad, which are not shown in the picture. The use of this keypad is for direct control of the system as specified in the provided program with the robot. The LCD display is used for information display.

Figure 4-8. **Fander-I Robot**

3.3.1. Sensory system

Fander-I robot is equipped with many built-in sensors that can be used and adapted to any robot navigational approaches. The sensors include, rotary ultrasonic range-finder transducer mounted on top of the robot with 48 steps (7.5º /step angular resolution), three sets of infrared proximity sensors, two shaft encoders located on its main wheels, bumper detectors, and a line sensor for following designated lines on the ground. Of all these sensors we have decided to use only the sonar transducer, the infrared proximity sensors, and the shaft encoders.

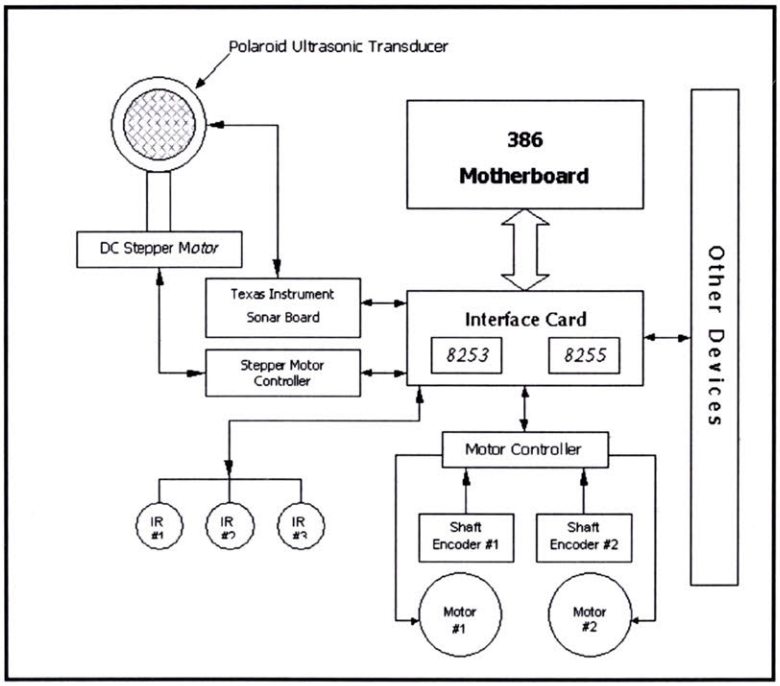


Figure 4-9. **System architecture for Fander-I**

The sonar transducer, made by Polaroid and coupled with the Texas Instrument controller board, was used for range finding and object detection in longer distances up to 10 meters. The process is exactly the same as the

one described in an earlier section for the soar system used in purpose built robot. The transducer is sitting on top of a stepper motor with 7.5° per step resolution. The step control signals are provided by a special purpose designed control board, which also include ADC, DAC, digital input/output channels and timer/counters.

Three sets of IR proximity sensors, as well as two wheel shaft encoders enable the motors to detect the obstacles unrecognized by other means. The other use of shaft encoders can be to measure up the rotations of each wheel, especially useful for racing detection as has been described earlier in this chapter.

The infrared transmitter and receiver pairs are attached to the front of the robot's body, just below the lower bumper strip. The transmitters and the receivers are separate modules, unlike the purpose built robot's, and are also connected to square wave signal generators.

3.3.2. Motors and Shaft Encoders

Each wheel has a shaft encoder measuring the wheel's rotation. The shaft encoder is made of two parts quite similar to the previous ones, and include the rotating striped disc and a coupled infra-red optical detector. The disc has 16 stripes so the total of 16 pulses are generated on a full rotation (22.5° resolution). The signal conditioning circuit is also responsible for generating the square wave signal on each transition from clear to black stripe. The generated square wave signal is TTL compatible.

Unfortunately, the resolution of the shaft encoders is not enough for most of the robot navigation approaches like grid-base strategy, however other strategies that do not need high level odometer information can still be applied.

The main motors are located on the sides of the robot's body towards the back. The control system drives the two motors separately, providing straight ahead, right, left and backward motions.

3.3.3. Programs

The robot comes with pre-compiled monitoring software, and also provided in disks are the library modules that can be used for control programming purposes. The functions are mostly useful in primitive action-response processes. For further information, we refer the reader to the programming manual provided by the manufacturer.

The monitoring program is designed to be used by a serial connection between a monitoring computer and the robot. Since we had decided to use wire-less communication, it was necessary to implement the monitoring program from the scratch based on our needs. The new monitoring program included features such as, graphical display updating the representation of the environment in 2 seconds intervals. The data would be provided by the robot through the wire-less modems, processed and screened in shape of colored dots. The control should have been performed using arrow keys on

the keyboard. During training periods, the data logging would be turned on by the trainer.

4. Learning Navigation

To be as useful as we imagine they could be, robots must be capable of learning new behaviours. Truly adaptive behaviour means not only the ability to put together sequences of behavioural primitives in new ways, but even to acquire new behavioural primitives. A number of methods of acquiring complex skills from primitives have been developed including plan learning [Mit89] and coordination of behaviours on a subsumption network using positive and negative feedback [MB90]. These techniques depend on primitives implemented as hand-written code or purpose-built circuits. The focus of this thesis is the acquisition of behaviours by imitating a human trainer.

In these experiments a mobile robot is demonstrated to acquire a primitive motor skill by being guided through a set of trials by a human operator. A log file of the robot's sensory and motor data is recorded as a trainer steers the robot through a set of obstacle avoidance scenarios. The logged data are then processed by an induction algorithm, which extracts a set of decision trees representing appropriate skirting movements for a range of obstacle patterns. The skill is then evaluated by observing the robot as it executes code generated from the decision trees in response to objects encountered during autonomous movement.

Also, for comparison purposes, another set of rules is extracted using Induct/RDR from exactly the same set of data. This is to show that there is a

little difference between the extracted rules, and the only difference lies on the strategy applied by the trainer (demonstrator).

4.1. The Problem

In these experiments, we have demonstrated that a simple but important behavioural primitive, obstacle avoidance, can be acquired by induction from steering motions performed by a human operator. When the robot encounters an object large enough to cause obstruction, it must make movements to skirt the object and continue on its way. These movements are not always a simple function of the size and shape of the object, but also depend on the movement capabilities, size and shape of the robot, and on the local spatial situation. Human operators, controlling the robot by a joystick, can perform this skill. If sensory and motor data are logged by the robot during a number of avoidance trials, an inductive learning algorithm can be used to extract a number of decision trees. Encoded rules derived from the decision trees can then be used to control the robot during autonomous movements. The quality of the behaviour is then assessed by observation.

This learning method, called *"behavioural cloning"* by Michie [MBM90], has been successfully applied to the control of pole-balancing machines [MC68], aircraft [SUKM92][SSE95], cranes [SS82][UB93], all of which have been done by simulation models. This is apparently the first use of the technique for

navigation in a mobile robot. It is likely that the technique will find other uses for robot systems.

4.2. Manual Control

The central control mechanism of the trainer program is a loop that interrogates the robot controls and updates the state of the robot according to a set of sensors readings. Before repeating the loop, the requested data is logged in the data file.

The robot can be manually steered using a joystick. Continuous motion of the joystick are converted into numbers in the range of 0 to 12 (Figure 4-1). Each number represents the robot's heading and it's speed. In manual training mode, these signals are converted into motor commands, which move the machine in a direction roughly corresponding to the angle of the stick. The speed of movement is related to the amount by which the stick is displaced from the origin. With practice, a human can drive the robot as if it were a toy car. While the trainer controls the motors, the sensors just continue their regular job that is sensing the environment's changes. However, the sensors readings are not reported to the trainer.

As shown in Figure 4-1, there are three speed regions in the joystick control. The central region is stop, the mid-region is low speed, and the outer region for high speed. The directions are forward, turn_right_forward, turn_right_rear, reverse, turn_left_rear, and turn_left_forward.

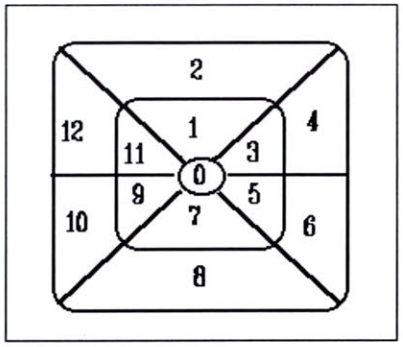


Figure 4-1. **The control region of *joystick***

300 different obstacle avoidance trials, performed by 5 trainers (demonstrators), in which 3 objects of various sizes and shapes were placed in 7 different positions (Figures 4-2 through 4-5) were set up. The pictures show a general sketch of the position of the object and the robot in different situations, as well as the direction of the movement.

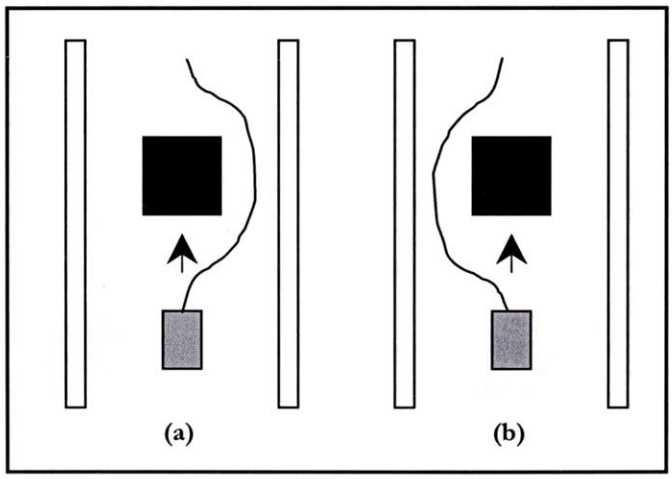


Figure 4-2. **Center Obstacle**

In Figure 4-2, The object has been placed on the center of the road in between two parallel walls. The robot has been steered by the trainer in either direction around the obstacle. The process was that the steering continues towards the object until the front infrared sensor has sensed the obstacle, after the data

logging switch is turned on. Then the robot is steered backward (backing off procedure in order to make enough space to steer around the object without bumping to it) until the sensor(s) were cleared. Finally the trainer would decide to continue the path on the right or left side of the robot, clearing the obstacle. Finally, the original direction and path would be picked up and the trainer would continue the steering for some more time. After that the data logging would be turned off, and reproduce the same scene again or try another scene.

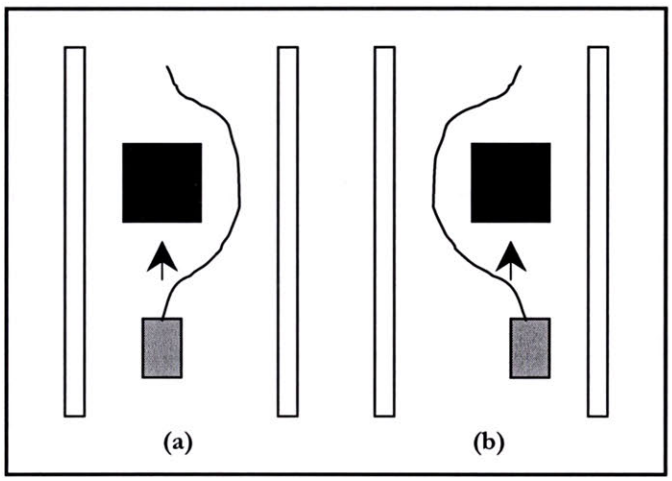


Figure 4-3. **Side Obstacle**

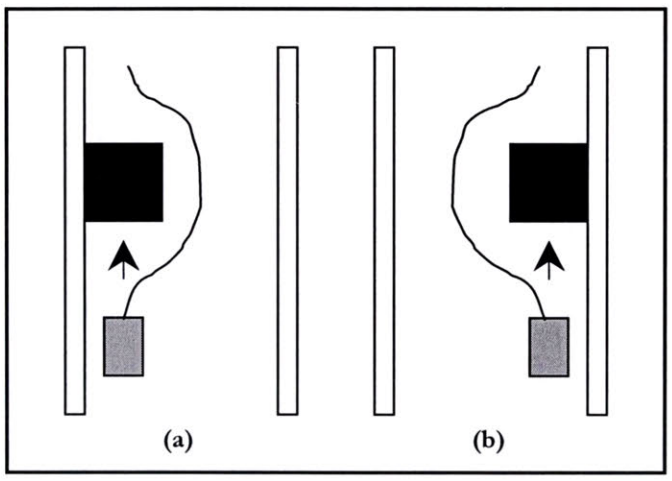


Figure 4-4. **Side Obstacle**

In the situation illustrated by Figure 4-3 and 4-4 the obstacle was located closer to wall or just beside the wall, making it only possible to travel in one direction. The difference between the two situation is that the robot can sense a

free space in Figure 4-3 between the obstacle and the wall, but observes that can not pass through, while in Figure 4-4 the wall and the obstacle would make the situation like a corner.

In scenarios illustrated by Figure 4-5, the robot was steered towards an object located on a corner, distanced differently from the wall in different scenarios. The steering similarly continued until the object was sensed, then the robot was cleared from the obstacle turning right or left respectively in scenario (a) and (b), finally continuing the travel along the wall further away from the object.

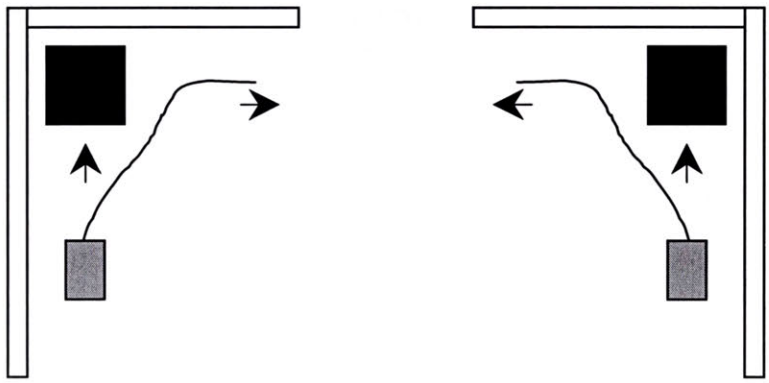


Figure 4-5. **Corner Obstacle**

The strategy for the obstacle avoidance is chosen by the trainer, and naturally is different for each individual. The following picture (Figure 4-6.) shows typical strategy's sketch for covering and avoiding an object. The steps are numbered for easier recognition of the procedure. The solid shape in the center is the obstacle, and the square shapes around it represent the robot in different positions during the skirting operation.

In steps 1 and 2 the robot is steered towards the object. Step 3 is the back-off step, and during the steps 4 to 7 the trainer is trying to cover the obstacle. Steps

8 and 9 represent the original path recovery and in step 10 the data logging is turned off.

The dotted lines represent another possible path for obstacle avoidance in this particular case. Of course, in the other scenarios, only one possible path may exist.

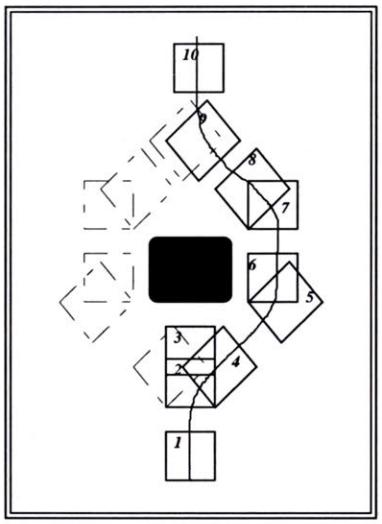


Figure 4-6. **The strategy of obstacle avoidance**

4.3. Logging Control Information

Every single trainer/operator took the robot through 5 training sessions for each scenario, with all sensors operational during the training. All the sessions involved three phases:

1. Sense obstruction and approach object to within 9 to 15cm, until one of the IR sensors detects the object's presence.

2. Skirt object, minimizing complications with respect to other objects, that is if the object has more than one corner in one side, the robot should have recognized it and start the procedure for object avoidance from the beginning.

3. Recover original path with minimum disturbance.

During each trial, a set of vectors for the following sensory and motor data was recorded:

polar_map	Vector of 8 bytes each representing the distance from the environment around at increments of 7.5° clockwise from origin, collected by a complete rotation of the transducer (sonar-sensor). The value of each byte can be one of ***Cl_rng*** (close range), ***Md_cls*** (medium-close range), ***Md_rng*** (medium range), or ***Fr_rng*** (far range) values corresponding to 0-0.5m, 0.5-1m, 1-2m, and more than 2 meters respectively ***(sense_1 to sense_8)***.
ir_sensors	A vector of 5 Boolean values, indicating the close proximity of the objects around the lower part (base) of the robot. Each vector can show either ***Sns*** (sensed) or ***Nsns*** (not-sensed) values ***(IR_1 to IR_5)***.
wheel_count_l	One byte from the left shaft encoder, representing the distance covered since the last reset ***(wl_ctl)***.
wheel_count_r	One byte from the right shaft encoder, representing the distance covered since the last reset ***(wl_ctr)***.
collision	Boolean, derived from a disparity between motor speed variables and wheel counts ***(Bang)***.

obstacle_flag	Boolean, showing the presence of the obstacle ***(Warning)***.
2_previous_control	One byte, representing the second-previous action's direction and speed of the motors ***(stat_2)***.
previous_control	One byte, corresponding to the previous action's direction and speed of the motors ***(stat_1)***.
Current_control	One byte, indicating the direction and speed of the motors' control ***(class)***.

It is essential to point out that originally the *polar_map* data were in numeric form made up of 48 different numbers between 0 and 255. Those numbers were the content of counter connected to the sonar system. However, the *Polar_map* was changed later to be in the range of specified discrete values, later which is explained in the following sections.

Each control attribute can take a value from the set of Stop, ***Bkwrd*** (backward), ***Frwrd*** (forward), ***Rght*** (right), Left, ***Frwd_Rght*** (right-forward), ***Frwd_Left*** (left-forward), ***Bk_Rght*** (right-backward), or ***Bk_Left*** (left-backward).

The number of recorded line (each line representing one event, including sonar, infrared sensors, shaft encoder counters, traveling direction for the last two events and the current control c0mmand) during each trial varies and depends on the trial's duration and chosen strategy. The program has been written so that when the trainer turns a specified switch on, the state of the whole system is written to the log file. And also, when the switch is turned off, logging would be stopped.

4.4. Data Analysis

Even with a well-specified navigating plan such as the one we are using here, there is a large degree of variation in the navigation strategies. Because of this variation, the number of trials we have is not sufficient to allow an induction program to distinguish useful actions from noise, using the raw data. However, it would not be very practical if it were necessary to have hundreds of trials before anything useful could be obtained. So, before applying the induction program to the data, we perform some pre-processing to assist it.

We define the similar cases to be the ones that are different only in few sonar vector values. This means that other environment variables are the same. These kinds of cases can be created by having the robot stationary for a short time and the trainer has moved a little, or there has been a slight bump to the robot, and etc.

The data file(s) were scanned for any exactly repeated or similar cases of events. The repeated events were omitted from the data file(s) and also within the similar events range, the one seeming most different to others were chosen and kept, resulting in deletion of the rest. To discard the similar or repeated cases, we used two different approaches. First, when the *polar_map* was in numeric form, we used a weight factor on the full map using an array of vectors kept for the last 10 readings and the summation of the whole vector as another comparison factor.

Furthermore, the continuos data collected from the sonar scanner were also changed in to discrete values of Close, near, near far and far distance as mentioned in the previous section. The original data collected from the sonar range-finder system were in numeric format, causing a lot of branching in decision-makings. In this case, to establish the similarity or duplication of the events, the summation comparison factor was dropped and we only used the buffer method for only last 5 readings.

We have used Quinlan's C4.5 [Qui93] as the induction program in these experiments. Learning reactive strategies is a task for which C4.5 was never intended. However, having used this algorithm before and also the availability of an auto-generator, which translates decision trees to nested "if-else-statements" in the C language, we decided to use C4.5 in this experiment. The transition to C was necessary so that the decision tree code can be inserted into the main automatic control program.

Like the learning to fly experiments [SUKM92], we also observed that at the early stages, data logging during each control cycle caused a vast amount of data to be recorded, producing inaccurate and huge decision trees. So, we decided to log the data only when the state of the robot was changed or an action has occurred. Each control cycle includes reading the position of the joystick, control switches and IR-sensors, and finally one sonar reading cycle. The motor speeds and heading are then recorded.

```
138,    ...    , 255,255,    ...    , 255,  0,  0,255,0,0,1
138,    ...    , 126, 58,...    , 70, 50, 50,223,1,1,0
143,    ...    , 126, 58,...    , 70, 20, 20,239,1,0,4
:
:
203,    ...    , 116, 164,    ...    , 252, 55, 55, 223, 0, 1, 0
203,    ...    , 116, 164,    ...    , 252, 155, 155, 239, 1, 0, 4
146,    ...    , 119, 231,    ...    , 179, 250, 250, 127, 1, 2, 2
:
:
146,    ...    , 119, 231,    ...    , 179, 255, 255, 255, 0, 1, 1
146,    ...    , 119, 231,    ...    , 179, 0, 0, 255, 0, 0, 1
146,    ...    , 119, 231,    ...    , 179, 75,75, 159, 1, 1, 0
146,    ...    , 119, 231,    ...    , 179, 55, 55, 223, 1, 0, 4
```

Figure 4-7. A portion *of typical file 'robot.data'*

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12.

Snes_1 : Cl_rng, Md_cls, Md_rng, Fr_rng.
Snes_2 : Cl_rng, Md_cls, Md_rng, Fr_rng.
 :
Snes_8 : Cl_rng, Md_cls, Md_rng, Fr_rng.
IR_1   : Sns, NSns.
:
IR_5   : Sns, NSns.
Wl_ctl: continuous.
Wl_ctr: continuous.
Bang: 0, 1.
Warning: 0, 1.
Stat_2:0,1,2,3,4,5,6,7,8,9,10,11,12.
Stat_1:0,1,2,3,4,5,6,7,8,9,10,11,12.
```

Figure 4-8. A typical names file

C4.5 requires two input files. The first file is the data file gathered during the experiments, which is called '***robot.data***' (Figure 4-7) and the second file is named '***robot.names***' (Figure 4-8) and contains the class values, attribute names and legal attribute values of the data in the data file.

Figure 4-7 is a partial data collected during the experiments with the *polar_map* vectors were set to be a continuous value between 0 and 255. While the ***Snes_1*** through ***Snes_8*** attributes illustrated in Figure 4-8, represent the latter case, when the *polar_map* was changed to a discrete value. The original names file contained 48 ***Snes*** attribute names with the continuous values.

During the first stages, given the two files, C4.5 was used to build a preliminary decision tree from the raw data, i.e. without filtering or pre-processing of data.

4.5. Generating the Control Rules

After processing the data as described above, we can finally apply C4.5 and summarize them as rules that can be executed in an automatic controller.

C4.5 has two parameters that can be changed by the user to adjust tree pruning. We have experimented with them trying to obtain the simplest workable rules. One parameter controls C4.5's confidence level. That is, the algorithm will prune the decision tree so that it maintains a minimum classification accuracy with respect to test data. The second parameter controls the minimum number of instances required for a split. For example, if this

parameter is set to 10, then no branch in tree will be created unless at least 10 examples descend down that branch.

Proceeding with the default parameter settings for C4.5, the decision tree is generated and the rules are tested using the automatic control program. The automatic control program is a part of the main program implemented for the robot training. The main loop was implemented in a way that reading a specific switch position, it would either be executed in manual control mode or in automatic control mode. Most of the codes were common, and only certain portions of each section would have been executed based on the value of a designated switch. The code were put among if statements, and satisfying the situation (for example, if switch is on), the automatic part would have been executed. Otherwise, the manual control would be performed. The sensor reading related codes were part of the common sections. Also, other parts involved the movement command (writing to motor controllers) and sensor system activity.

Furthermore, the simpler rules are obtained by adjusting the C4.5 parameters, gradually. The procedure is continued until the rule breaks, i.e. the system is no longer able to control the robot.

Table 1 shows two sample decision trees and the error rate differences. The left part of the table is the result of the first execution of C4.5 on a small logged file. This file was logged during two short training scenarios. However, the right hand side part was the result of the execution of C4.5 on one of the last logged files that was a combination of over 250 training trials. It can be observed that

the more logged events, the more complex the decision tree would be. Having more rules in the decision tree is not necessarily a good thing. However, table 1 shows the opposite. The reason is that since the smaller tree is based on a shorter log file, it does not necessarily represents the complete decision making process. But, the longer version on the right is based on a longer logged file, which is also covering more obstacle avoidance training sessions' data.

An event is recorded only when there is a change in the robot's status; that is when any control setting is changed. A buffer is designated for each of previous actions (***stat_2*** and ***stat_1***), and a change is always determined by observing the current robot's state (***stat***) and in comparison with the content of the buffer. If the logging speed is too high, then the side effect of recording all the intermediate values between the two changes can occur. That is the majority of the event would be the same as the previously recorded one, causing unnecessary recorded events as well as confusion in decision-making process. This undesirable side effect can be discarded with the choice of appropriate scanning and recording speed.

Read 46 cases (16 attributes) from robot1.data

Decision Tree:

```
IR_1 = Sns: Bkwrd (7.0)
IR_1 = NSns:
|  stat_1 = Stop: Frwd (5.0)
|  stat_1 = Frwd_Rght: Frwd (0.0)
|  stat_1 = Rght: Frwd (8.0)
|  stat_1 = Bck_Rght: Frwd
|  stat_1 = Bck_left: Frwd
|  stat_1 = Left: Frwd
|  stat_1 = Frw_left: Frwd
|  stat_1 = Frwd:
|  |  sens_7 = Cl_rng: Left
|  |  sens_7 = Md_cls: Left
|  |  sens_7 = Md_rng: Left
|  |  sens_7 = Fr_rng: Rght
|  stat_1 = Bkwrd:
|  |  sens_4 = Cl_rng: Rght
|  |  sens_4 = Md_cls: Left
|  |  sens_4 = Md_rng: Left
|  |  sens_4 = Fr_rng: Rght
```

Evaluation on training data (46 items):

Before Pruning			After Pruning			
Size	Errors		Size	Errors		Estimate
20	1	(2.2%)	20	1	(2.2%)	(26.8%)

(a) Decision tree from first run

Read 966 cases (8 attributes) from robot2.data

Decision Tree:

```
IR_1 = Sns: Bkwrd (147.0)
IR_1 = NSns:
|  stat_1 = Stop: Frwd (105.0)
|  stat_1 = Frwd_Rght: Frwd (0.0)
|  stat_1 = Rght: Frwd (168.0)
|  stat_1 = Bck_Rght: Frwd
|  stat_1 = Bkwrd: Rght
|  stat_1 = Bck_left: Frwd
|  stat_1 = Left: Frwd
|  stat_1 = Frw_left: Frwd
|  stat_1 = Frwd:
|  |  IR_4 = Sns:
|  |  |  stat_2 = Stop: Rght
|  |  |  stat_2 = Frwd: Rght
|  |  |  stat_2 = Frwd_Rght: Rght
|  |  |  stat_2 = Rght: Left
|  |  |  stat_2 = Bck_Rght: Rght
|  |  |  stat_2 = Bkwrd: Rght
|  |  |  stat_2 = Bck_left: Rght
|  |  |  stat_2 = Left: Rght
|  |  |  stat_2 = Frw_left: Rght
|  |  IR_4 = NSns:
|  |  |  IR_5 = Sns: Left
|  |  |  IR_5 = NSns:
|  |  |  |  stat_2 = Stop: Left
|  |  |  |  stat_2 = Frwd: Left
|  |  |  |  stat_2 = Frwd_Rght: Left
|  |  |  |  stat_2 = Rght: Rght
|  |  |  |  stat_2 = Bck_Rght: Left
|  |  |  |  stat_2 = Bkwrd: Left
|  |  |  |  stat_2 = Bck_left: Left
|  |  |  |  stat_2 = Left: Left
|  |  |  |  stat_2 = Frw_left: Left
```

Evaluation on training data (966 items):

Before Pruning			After Pruning			
Size	Errors		Size	Errors		Estimate
34	63	(6.5%)	34	63	(6.5%)	(8.3%)

(b) Decision tree from final run

Table 1. **Two sample decision tree and the differences**

By trial and error, we found out that the best rate of scan and recording is equivalent to a full sonar scan. However, since there are four sensor readings (except the sonar) during a complete cycle of sonar scanning, it is necessary to keep a buffer of previous events based on other sensor readings. In case of difference(s), the previous immediate data were recorded.

4.6. Cause and Time of Action

C4.5 only constructs purely reactive rules, which make decisions base on the values in a single state of the control program. Time and causality of the data is not part of the induction program's concepts. That is the rules are not situation based -- the time and the cause of actions are not considered.

```
(line 1)    IR_1 = Sns :
(line 2)    IR_2 = Sns:
(rule 1)    |  Stat = 1 : 0
(rule 2)    |  Stat = 2 : 2
            IR_ ... :
            :
            :
            :
```

Figure 4-9. **Part of a typical decision tree**

As a result some strange rules can turn up. For example, the above rule for IR sensors in the approaching stage (towards the obstacle) was derived from data that was not filtered as described in the previous section. There were about 300 examples in the training set, and since there were 30 trials combined, the minimum split size was set to 30, and finally the confidence parameter was set to 10%. The outcome is shown in Figure 4-9.

The first two lines (line 1 and line 2) state that if the two front Infrared sensors have detected an object, the previous direction of motion should be tested. If it was forward, the robot should stop, and if it was making a

turn_right_movement, it should continue. Thus C4.5 has detected a correct relationship between previous movement and IR sensors. Unfortunately, the second rule is not correct. By applying the filtering described in section 4.2.5, problems caused because of the absence of the cause of decision can be overcome to some extent, but rules like this sometimes still may occur. For the above case, C4.5 was executed again with minimum split size set to 50 resulting in the following rule:

```
IR_1 = Sns :
IR_2 = Sns:
|  Stat = 1 : 0
:
```

Figure 4-9. **Corrected rule**

This is quite sensible, if the robot has been moving forward sensing an object it should stop. As Sammut and colleagues also mentioned in learning to fly experiments [SUKM92]: *"We believe that learning could be improved by including some knowledge of causality in the system so that it is able to correctly identify dependencies among variables"*.

4.7. Automatic Control

In applying the above decision tree (Table 1, right hand side) to the robot and it's control program, some level of success have been achieved. The main part of the automatic control section of the program is made of if-then statements. The if-then statements are the final result of C-translated rules induced by C4.5

from the logged data. During the automatic travel, the appropriate rules for each control action are selected according to the sensors' readings [SUKM92].Using this control program the robot can approach an object, sense it, avoid it, and choose a good direction for the next movement. But, still there has not been a success in making the robot to recover the original travelling path after passing the object.

One way to overcome this problem can be providing more history in the data file. This approach would need to add more buffers into the program, incorporating more history of the state of the robot, environment and control actions. However, this can backfire, simply because of the size of the rules and branches of decisions created based on this complex data file.

The other way, using force field approach, is to calculate the angle of movement before every log and recording it along with the rest of the vectors. Thus, later on, while the robot is trying to recover the original travelling path (and direction), it should be able to reproduce the previous angle with negative sign. However, this method can cause other problems including entrapment of the robot around the object. Nevertheless, it may be necessary to change the approach partially in this regard. This matter is discussed in further details in the last chapter.

It is very interesting to mention that if we apply the data file taken from only one trainer to C4.5, then the robot will follow the personal strategy of the trainer in movements control and decision makings provided by the individual.

And that is exactly why it is considered a clone behaviour based control strategy.

4.8. Induct / XRDR approach

The same procedure has been carried out using Induct/XRDR, but in this case

Induct on 966 cases

```
1 RULE: 1: IR_1 = NSns & stat_2 = Stop -> stat_1 = Stop
2 RULE: 2: IR_1 = Sns -> stat_1 = Frwd
3 RULE: 3: stat_2 = Left -> stat_1 = Frwd
4 RULE: 4: stat_2 = Rght -> stat_1 = Frwd
5 RULE: 5: IR_3 = Sns & stat_2 = Frwd -> stat_1 = Rght
6 RULE: 6: IR_3 = NSns & stat_2 = Bkwrd -> stat_1 = Rght
7 RULE: 7: IR_5 = NSns & IR_6 = NSns & stat_2 = Frwd -> stat_1 = Rght
8 RULE: 8: IR_5 = Sns & stat_2 = Frwd -> stat_1 = Bkwrd
9 RULE: 9: IR_3 = NSns & IR_5 = NSns & IR_6 = NSns
          & stat_2 = Frwd -> stat_1 = Bkwrd
10 RULE: 10: IR_6 = Sns & stat_2 = Frwd -> stat_1 = Left
11 RULE: 11: IR_3 = Sns & stat_2 = Bkwrd -> stat_1 = Left
12 RULE: 12: IR_5 = Sns & stat_2 = Frwd -> stat_1 = Left
13 RULE: 13: IR_3 = NSns & IR_5 = NSns
          & stat_2 = Frwd -> stat_1 = Left

Rules 13  Clauses 0 (Mean 0.0)
Evaluating Induced Rules :

 Total Correct 714/966  Errors  +:252  -:0  Total: 252
26.09% when testing on itself
```

Figure 4-10. **Corresponding Induct rule sample**

there is only one file involved starting with the names of classes and attributes followed by their values. Induct produces a binary tree, and a primary rule set. The induced rule set by Induct and XRDR format of the same rule set is shown in Fig. 4-10 and 4-11.

While there is a very straightforward way of interpreting the rules produced by C4.5, the XRDR rules are a bit complicated. Rule 11 of the C4.5 rule set (Table 1, right side) implies that if the robot has been going to the right it should now move forward (unless one of the sensors is sensing an object on the way forward). The second rule of XRDR (ꕥ #2, in Figure 4-11) specifies that if infrared sensor number 3 has sensed the object, and the second previous action (***Stat_2***) has been move-forward, then turn right. Also, it says the left & right branches of the tree are rules #1 (ꕥ #1, in Figure 4-11) and rule #3 (ꕥ #3, in Figure 4-11) respectively, and the upper node for this node is node #0.

Naturally, the default class value for both is moving forward, which is what we would expect.

When translated from Induct format to XRDR format it looks like this:
Rule num, conditions, parent true fail branch, concl, corner
ꕥ #1 true = true ≪0 ❷ 0≫ stat_1 = Frwd
ꕥ #2 IR_3 = Sns;stat_2 = Frwd ≪1 ❶ 3≫ stat_1 = Rght
ꕥ #3 IR_3 = Sns ≪2 ❶ 4≫ stat_1 = Left
ꕥ #4 IR_6 = NSns;stat_2 = Frwd ≪3 ❶ 5≫ stat_1 = Bkwrd
ꕥ #5 IR_1 = NSns;stat_2 = Stop ≪4 ❶ 6≫ stat_1 = Stop
ꕥ #6 stat_2 = Bkwrd ≪5 ❶ 7≫ stat_1 = Rght
ꕥ #7 IR_6 = Sns ≪6 ❶ 0≫ stat_1 = Left
ꕥ #8 IR_5 = Sns ≪4 ❶ 0≫ stat_1 = Left

Figure 4-11. **XRDR Rule format**

The reason for this short comparison is to show that the both c4.5 and RDR methods produce very similar results. So, in order to avoid the complexity we have decided to continue using the C4.5 technique.

4.9. Experiments after Upgrade

After making changes for upgrade, as discussed in the last chapter, the experiments were reproduced and the results were almost the same. However, better sensory devices, specially the ultra-sound transducer and the micro-controller made a lot of difference in the speed and accuracy. This means that besides the improvement of speed and accuracy, the experiments were device-independent. However, it is also expected that changing the robotic platform all together will need a little bit of change in approach, especially in program parts that are sensor dependent.

The next step was to use wire-less modems in place of the parallel control box (electronic joystick). Unfortunately, the modems did not match the robotic platform and we were forced to employ another robot, Fander-I.

4.10. Experiments on Fander-I

After the unfortunate problem(s) with the wire-less modems on the purpose built robot used in the previous experiments, we decided to also change the approach of the experiments to *"grid-based navigational strategies"* and combining it with the previously done trial scenarios.

Also, using the wire-less modems instead of the joystick would provide observation of the environment from the eyes of the robot, or as the robot sees the world around it. In the previous set of experiments, since the trainer is using

his/her natural senses which are much more complex than the robot's primitive sensors, the training to some extent would have been based on the operator's senses and not the information mostly gathered by the robot. But, if the trainer steers the robot from a distance without being able to use human senses directly, and only observe the environment based on the robot's sensor, we believe that the training would be more successful. The reason is the involvement of the electronic device errors and delays in the observation as well, and so the decision making process of the trainer is also based on the robot's observation.

Furthermore, this strategy would have let us to make a degree of map of the area visited by the robot during the wandering process.

But, yet again, there were set backs due to technical problems with the wireless modems. In addition the Fander-I shaft encoders do not give high enough resolutions to be used in many navigational strategies including grid-based. However, it should be mentioned that the resolution is enough to achieve any navigational task not relying on too much of odometer information. For example, the previously described experiments could be repeated using Fander-I. But, as we mentioned before, since the Fander's sensory system is somehow different in design, the manual and automatic control programs should be somehow different as well.

4.11. Performance

The performance observed from the automatic control systems, specially the one based on the rules generated by C4.5 from the filtered and pre-processed original logged data file have been successful to some extent. As explained before, the obstacle avoidance behaviour was learnt successfully. However, the robot could not successfully recover the original travelling path, after skirting and clearing the obstacle.

To have a better performance, or even further in order to make an accurate clone, we still need more training trials and more scenarios. Also, better history buffers and more of them are needed, in case the physical platform stays the same.

Nevertheless, using improved sensory system, specially purpose designed gyro-systems, the process can be improved dramatically. For example using gyros, approaching an obstacle the coordinates can be saved and after clearing the obstacle and according to the situation newer coordination can be calculated on-the-fly. The calculation can also be based on the saved coordination.

5. Discussion and Conclusion

5.1. Discussion

One of the interesting things we have learnt in this study is that even good trainers could not control the robot very smoothly. But, it is essential to note that the control quality is actually based on the hardware and the control software, and due to these shortcomings we can not blame all the problems on trainers. As discussed in previous chapter by changing the hardware, the software and as a result the training approach should be also altered. Nevertheless, the quality of the training to some extent is based on the approach of the trainer.

For example, if the trainer(s) make a mistake and in some control responses due to special situations choose the wrong action, and the number of the wrong actions are more that the right ones for the same situation, then the induction algorithm can come up with the wrong rule as well. However, this problem can be overcome by choosing more training scenarios and more trainers/trials.

We expected the behavioural cloning simply to be similar to copying a trainers behaviour. However, similar to the same results observed in Learning to Fly

experiment, it was also observed that the robot's behaviour somehow would be different to that of the trainer. This has been called "clean-up" effect as noted by Michie, Bain and Hayes-Michie [MBM90] and Sammut [SUKM92]. The navigation log of any trainer will contain many different and opposite actions due to human inconsistency and corrections required as a result of inattention. However, it seems due to the fact that the inducting algorithm built in C4.5 makes the rules based on the majority of matched action instances, it would actually prune away the wrong affects of the bad logged data. This would make the control rules, navigating the robot, much more smoother.

This effect was particularly noticeable in the stage of making a turn around and skirting an obstacle. Primarily, it has been observed that the automatic controller does a much better and smoother skirting job during this stage.

5.2. Conclusion

So far, most applications of inductive learning have been in classification tasks such as medical diagnosis. Just as diagnostic rules can be learned by observing a physician at work, we should be able to learn how to control a system by watching a human operator at work.

In the case our experiments, the data provided to the induction program (C4.5) are the data logged to a file when an action taken by the operator/trainer in response to changes in the system's and/or environment's state.

We have used a procedure to inductively build sets of control rules. An induced rule-set makes up a "strategy" for the given sub-task - a kind of classifier that maps state records into action names, just like mapping/assigning patient records into disease names. In our preliminary study we were able to partly demonstrate the feasibility of learning a specific control task.

Machine learning of control systems may lead to a better understanding of sub cognitive skills, which are inaccessible to introspection.

For example, if you have been asked about the method that you use to ride a bicycle, you will not be able to provide an adequate answer because the skill has been learnt and is executed at a subconscious level. However, mathematical operations can be explained step-by-step.

As another example, when someone is driving home, no matter how engaged that person is while driving, he/she would never go the wrong way, of course unless they have recently moved to the new place. This example provides a different method of learning, and surely it is not memorization. It can also be regarded as sort of association in memory of the person, i.e. somehow people specify some remarkable spots on the way home and then by comparing those spots would find their way home. However, we believe that is not completely true.

Furthermore, we believe that it's possible to construct a functional description of a sub cognitive skill in the form of symbolic rules, just by monitoring the performance of that skill. This will not only reveal the nature of the skill, but

also will aid the training in different aspects of automation. So, as in first example, by riding over and over again, the person will automatically acquire the skill in his/her sub-conscious. While in the latter example, the map has been automatically saved into the sub-conscious of the driver, forcing him automatically drive home.

Learning control rules by induction/observation provides a new way of building complex control systems quickly and easily. Where these involve safety critical tasks, the "clean-up" effect mentioned in the previous section holds particular interest. While our experiments have been primarily concerned with the navigation of a mobile robot, inductive methods can be applied to a wide range of related problems.

Though, our experiments have been preliminary concentrated on obstacle avoidance related problem(s), one can apply the discussed method to a wide range of related problems such as manufacturing plant control, educational purposes, or even pattern recognition. This research was aimed at producing a reliable and reproducible method for building (learning or imitating) controllers for a mobile robot.

5.3. Future Work

The robot designed for the purpose of these experiments was built during the time that no affordable robotic platforms were available. And since we started the experiments with the aim of working on a real physical platform rather than

a simulated environment, we were bounded to use the available equipment and affordable devices.

To be able to better understand the nature of the problems arising from experiments in the real world, we will have to use a robot benefiting from today's technology. For example, gyro-sensors, better sonar equipment, wire-less LAN-based network cards (instead of wire-less modems) and so can provide better opportunity in tackling the problem discussed in this thesis.

Future experiments might attempt to learn other behaviours such as following another moving object, or finding a spar along a wall. Teaching more primitive behaviours to the robot can be combined with incremental learning boosting the ability of the system towards producing a subsumption like structure of higher-level behaviours.

Also, it can be noteworthy if the wire-less modem based training is combined with the grid-base occupancy approach. Or even further, one can combine the learning/cloning procedure with other methods of providing information. For example, the obstacle avoidance behaviour can be broken into two parts. The hard-coded obstacle skirting and clearance can be provided to the robot and the trainer can choose the specific hard-coded behaviour on the arising of the condition. This would make the learning procedure faster and easier. It is exactly similar to the situations where kids are trained to do something specific associated with certain condition. For example, one should put down the fire

only when in hazardous situation. No one in the right mind would kill the heating fire in a winter night.

Future work can also be directed to finalize the work started, and also to combine these control behaviours together into a subsumption architecture to control the navigation of a mobile robot [Bro86a]. The same method of training can be used to obtain other simple and primitive behaviours like the ones mentioned above. Combining those primitive behaviours by teaching/training the robot when to use them can be implemented in to a behavioural cloning subsumption architecture.

Bibliography

[AB75] **Ambler, A.P. and H.G. Barrow (1975).** *"A Versatile System for Computer Controlled Assembly"*, Artificial Intelligence, Vol.6: pp.129-156.

[ACH97] **Aycard, O., F. Charpillet and J.P. Haton (1997).** *"A New Approach to Design Fuzzy Controllers for Mobile Robots Navigation"*, 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '97),

[AM90] **Anderson, C.W. and W.T. Miller (1990).** *"A Challenging Set of Control Problems"*, Neural Network for Control, MIT Press, pp.475-511.

[Ama97] **Amant, R.S. (1997).** *"Navigation and Planning in a Mixed-Initiative User Interface"*, Proceedings of 14th International Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference.

[Arc87] **Arkin, R.C. (1987).** *"Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior"*, Proceedings of the IEEE International Conference on Robotics and Automation.

[Arc89] **Arkin, R.C. (1989).** "*Motor Schema Based Mobile Robot Navigation*", International Journal of Robotics Research, Vol.8 (4): pp.91-112.

[Arc90] **Arkin, R.C. (1990).** "*Integrating Behavioral, Perceptual, and World Knowledge in Reactive Navigation*", Robotics and Autonomous Systems, Vol.6, pp.17-34.

[Arc92a] **Arkin, R.C. (1992a).** "*Cooperation Without Communication: Multi-Agent Scema-Based Robot Navigation*", Journal of Robotic Systems.

[Arc92b] **Arkin, R.C. (1992b).** "*Behavior-based Robot Navigation in Extended Domains*", Journal of Adaptive Behavior, Vol.1 (2): pp.201-225.

[AS97] **Atkeson, C.G. and S. Schall (1997).** "*Robot Learning from Demonstration*", Proceedings of the 14th International Conference on Machine Learning (ICML '97).

[AT90] **Amidi, O. and C. Thorpe (1990).** "*Integrated Mobile Robot Control*", SPIE, Vol.1388 : pp.504-523.

[Bar90] **Barto, A.G. (1990).** "*Connectionist Learning for Control*", Neural Network for Control, MIT Press, pp.5-59.

[BB83] **Buchanan, G., D. Barstaw, et al. (1983).** "*Constructing an Expert System*", Building Expert Systems, Massachusetts, Addison-Wesley,

Vol.36: pp.127-168.

[BB94] **Bell, D.A. and J. Borenstein (1994).** "*An Assistive System for Wheelchairs Based Upon Mobile Robot Obstacle Avoidance*", IEEE International Conference on Robotics and Navigation.

[BD91] **Basye, K., T. Dean, et al. (1991).** "*Coping with Uncertainty in Map Learning*", Autonomous Mobile Robots, Vol.1: Perception, Mapping, and Navigation, IEEE Computer Society, pp.347-352.

[Ben96] **Benson, S.S. (1996).** "*Learning Action Model for Reactive Autonomous Agents*", Stanford University.

[BF91] **Bradshaw, J.M., K.M. Ford, et al. (1991).** "*Knowledge Representation of Knowledge Acquisition: A Three Schemata Approach*", Proceeding of 6th AAAI Sponsored Banff Knowledge Acquisition for Knowledge Based Systems Workshop, Banff, Canada.

[BK90a] **Borenstein, J. and Y. Koren (1990a).** "*Real-Time Map Building for Fast Mobile Robot Obstacle Avoidance*", SPIE, Vol.1388: Mobile Robots V.

[BK90b] **Borenstein, J. and Y. Koren (1990b).** "*Real-Time Obstacle Avoidance for Fast Mobile Robots in Cluttered Environments*", IEEE International Conference on Robotics and Automation, Cincinnati, OH.

[BK91] **Borenstein, J. and Y. Koren (1991).** *"The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots"*, IEEE Journal of Robotics and Automation, Vol.7 (3): pp.278-288.

[BK91] **Bozma, O. and R. Kuc (1991).** *"Building a Sonar Map in a Specula Environment Using a Single Mobile Sensor"*, IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.13 (12): pp.1260-1269.

[BM78] **Buchanan, G. and T.M. Mitchell (1978).** *"Model Directed Learning Production Rules"*, Pattern-Directed Inference Systems, D.A. Waterman and F. Hayes-Roth (Eds.), New York, Academic Press, pp.297-312.

[BN92] **Banta, L., R.S. Nutter, et al. (1992).** *"Mode-Based Navigation for Autonomous Mine Vehicles"*, IEEE Transaction on Industry Applications, Vol.28 (1): pp.181-184.

[BO90] **Beckerman, M. and E.M. Oblow (1990).** *"Treatment of Systematic Errors in the Processing of Wide-Angle Sonar Sensor Data for Robotic Navigation"*, IEEE International Conference on Robotics and Automation, Vol.6 (2).

[BR97] **Blank, D.S. and J.O. Ross (1997).** *"Learning in a Fuzzy Logic Robot Controller"*, Proceedings of 14th International Conference on Artificial

Intelligence and 9th Innovative Applications of Artificial Intelligence Conference.

[Bra91] **Bratko, I. (1991).** "*Qualitative Modeling: Learning and Control*", Proceedings of the 6th Czechoslovak Conference on Artificial Intelligence, Prague.

[Bra93] **Bratko, I. (1993).** "*Qualitative Reasoning about Control*", Proceedings of ETFA'93 Conference, Cairns, Austria.

[Bra96] **Bratko, I. (1996).** "*Deriving Qualitative Control for Dynamic Systems*", Machine Intelligence 14, K. Furukawa, D. Michie and S. Muggleton (Eds.), Oxford: Clarendon Press.

[Bra97] **Bratko, I. (1997).** "*Qualitative Reconstruction of Control Skills*", 11th International Workshop on Qualitative Reasoning, Cortona, Italy.

[Bre84] **Breiman, L. (1984).** Classification and Regression Trees, Belmont, Wadsworth.

[Bro83] **Brooks, R.A. (1983).** "*Find-Path for a PUMA-Class Robot*", Proceeding of AAAI '83, pp. 40-44.

[Bro86a] **Brooks, R.A. (1986a).** "*A Robust Layered Control System for a Mobile Robot*", IEEE Journal of Robotics and Automation, Vol.RA-2

(1): pp.14-23.

[Bro86b] **Brooks, R.A. (1986b).** *"Achieving Artificial Intelligence Through Building Robots"*, MIT AI Memo 899.

[Bro89] **Brooks, R.A. (1989).** *"A Robot that Walks: Emergent Behavior from a Carefully Evolved Network"*, Neural Computation, Vol.1 (2).

[Bro90] **Brooks, R.A. (1990).** *"The Behavior Language"*, MIT AI Memo 1227.

[BS83] **Barto, A., R. Sutton, et al. (1983).** *"Neuronlike Adaptive Elements that can Solve Difficult Learning Control Problems"*, IEEE Transaction on Systems, Man, and Cybernetics, Vol.13 (5): pp.834-846.

[BS97] **Bain, M. and C.A. Sammut (1997).** *"Structuring Learning for Behavioral Cloning Tasks"*.

[BU95] **Bratko, I., T. Urbančič, et al. (1995).** *"Behavioral Cloning: Phenomena, Results and Problems"*, Automated Systems Based on Human Skills, IFAC Symposium, Berlin.

[Buc78] **Buchanan, G. (1978).** *"Dendral And Meta-Dendral: Their Application Dimension"*, Artificial Intelligence, Vol.11, pp.5-24.

[Buc86] **Buchanan, B. (1986).** *"Expert Systems: Working Systems and the*

Research Literature", Expert Systems, Vol.3: pp.32-51.

[Bun88] **Buntine, W. (1988).** "*Generalized Subsumption and its Applications to Induction and Redundancy*", Artificial Intelligence, Vol.36: pp.149-176.

[Car83a] **Carbonell, J.G. and et al. (1983a).** "*An Overview of Machine Learning*", Machine Learning and Artificial Intelligence Approach, R.S. Michalski, J.G. Carbonell and T. M. Mitchell (Eds.), Palo Alto, California, Morgan-Kaufmann, pp.3-23.

[Car83b] **Carbonell, J.G. and et al. (1983b).** "*Learning by Analogy: Formulating and Generalizing Plans from Past Experience*", Machine Learning and Artificial Intelligence Approach, R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), Palo Alto, California, Morgan-Kaufmann, pp.137-161.

[Car89] **Carbonell, J.G. (1989).** "*Introduction: Paradigms for Machine Learning*", Artificial Intelligence, Vol.40 (1-3): pp.1-9.

[CG83] **Carpenter, G.A. and S. Grossberg (1983).** "*A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine*", Computer Vision, Graphics, and Image Processing, pp.15-54.

[CH89] **Compton, P.J. and R. Horn (1989).** "*Maintaining an Expert System*", Applications of Expert Systems, Compton, P.J. and R. Horn (Eds.), London, Addison-Wesley, pp.366-385.

[CJ89] **Compton, P.J. and R. Jansen (1989).** "*A Philosophical Basis for Knowledge Acquisition*", Proceedings of 3rd European Knowledge Acquisition for Knowledge Systems Workshop.

[CM93] **Connell, J. and S. Mahadevan (1993).** Robot Learning, Kluwer Academic Publishers.

[CN90] **Constant, P., S. Natwin, et al. (1990).** "*LEW: Learning by Watching*", IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.12 (3): pp.294-308.

[Com92] **Compton, P.J. (1992).** "*Insight and Knowledge*", Proceedings of AAAI, Spring Symposium: Cognitive Aspects of Knowledge Acquisition, Stanford University.

[Cro85] **Crowley, J.L. (1985).** "*Navigation for an Intelligent Mobile Robot*", IEEE journal of Robotics and Automation, Vol.RA-1 (1).

[CU87] **Connell, M.E. and P.E. Utgoff (1987).** "*Learning to Control a Dynamic Physical System*", Proceeding of the 6th AAAI International Conference.

[Dedo72] **DeDombal, F.T. (1972).** *"Computer Aided Diagnosis f Acute Abdominal Pain"*, British Medicine Journal, Vol.2: pp.9-13.

[Dej86] **Dejong, G.F. (1986).** *"Explanation Based Learning: An Alternative View"*, Machine Learning, Vol.1: pp.145-176.

[DeLa92] **DeLafontaine, J. (1992).** *"Autonomous Spacecraft Navigation and Control for Comet Landing"*, Journal f Guidance, Control, and Dynamics, Vol.15 (3): pp.567-576.

[Dev87] **Devorak, D.L. (1987).** *"Expert Systems for Monitoring and Control"*, The University of Texas at Austin.

[DK90] **Davies, H.C., A.E. Kayaalp, et al. (1990).** *"Autonomous Navigation in a Dynamic Environment"*, SPIE, Vol.1388 (Mobile Robots V): pp.165-175.

[Dod90] **Dodds, D.R. (1990).** *"Coping With Complexity in the Navigation of an Autonomous Mobile Robot"*, SPIE, Vol.1388 (Mobile Robots V): pp.448-452.

[Dor96] **Dorigo, M. (1996).** *"Introduction to the Special Issue on Learning Autonomous Robots"*, IEEE Transactions on Systems, Man, and Cybernetics, Vol.26 (June): pp.361-364.

[DS97] **DeRougement, M. and C. Schlieder (1997).** "*Spatial Navigation with Uncertain Deviations*", Proceedings of 14th International Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference.

[Dze93] **Dzeroski, S. (1993).** "*Discovering Dynamics*", Prceeding of the 10th International Cnference on Machine Learning, Amhert, Massachusetts.

[EG94a] **Everett, H.R., G.A. Gilbreath, et al. (1994a).** "*Controlling Multiple Security Robots in a Warehouse Environment*", Proceedings of the Conference n Intelligent Robotics in Field, Factory, Service, and Space (CIRFFSS '04), NASA, Linthicum Heights, MD, USA.

[EG94b] **Everett, H.R., G.A. Gilbreath, et al. (1994b).** "*Coordinated Control of Multiple Security Robots*", SPIE, Vol.2058 : pp.292-305.

[Elf86] **Elfes, A. (1986).** "*A Distributed Control Architecture for an Autonomous Mobile Robot*", Artificial Intelligence, Vol.1 (2): pp.99-108.

[Elf89a] **Elfes, A. (1989a).** "*Using Occupancy Grids for Mobile Robot Perception and Navigation*", IEEE Computer Magazine, Vol.22 (6): pp.46-57.

[Elf89b] **Elfes, A. (1989b).** *"Occupance Grids: A Probabilistic Framework for Mobile Robot Perception and Navigation"*, Electrical and Computer Engineering Department / Robotics Institute, Carnegie Mellon University,.

[ESM94] **Esmaili, N., C.A. Sammut, and G.A. Mann (1994).** "*Navigation Learning by a Mobile Robot*", Proceedings of ICEE'94, Tarbial-Modaress University, Tehran, Iran.

[ESS95] **Esmaili, N., C.A. Sammut, and G.H.M. Shiraz (1995).** "*Behavioral Cloning in Control of Dynamic Systems*", Proceedings of 1995 IEEE International Conference on Systems, Man, and Cybernetics SMC '95, Vancouver, Canada.

[Eve82] **Everett, H.R. (1982).** *"A Microprocessor Controlled Autonomous Sentry Robot"*, Monterey, CA, Naval Postgraduate School.

[Fau84] **Faugeras, O.D. (1984).** "*Object Representation, Identification, and Positioning from Range Data*", 1st International Symposium on Robotics Research, Cambridge, MA.

[FH90] **Fennema, C., A. Hanson, et al. (1990).** *"Model-Directed Mobile Robot Navigation"*, IEEE Transaction on Systems, Man, and Cybernetics, Vol.20 (6): pp.1352-1369.

[Fly88] **Flynn, A.M. (1988).** *"Combining Sonar and Infra-Red Sensors for Mobile Robot Navigation"*, The International Journal of Robotics Research, Vol.7 (6).

[FR86] **Forsyth, R. and R. Rada (1986).** Machine Learning: Applications in Expert Systems and Information Retrieval, NewYork, Printed by Horwood Halsted.

[Fra96] **Franklin, J. (1996).** Robot Learning. Number 2-3, Kluwer Academic Press.

[Fry87] **Fryxell, R.C. (1987).** *"Navigation Planning Using Quadtrees"*, SPIE, Cambridge, MA, Vol. Mobile Robots II,: pp.256-261.

[GC92] **Gaines, B.R. and P.J. Compton (1992).** *"Induction of Ripple Down Rules"*, Proceedings of AI '92: 5th Australian Joint Conference on Artificial Intelligence, Hobart, Australia, World Scientific, Singapore,

[GE88] **Gilbreath, G.A. and H.R. Everett (1988).** *"Path Planning and Collision Avoidance for an Indoor Security Robot"*, SPIE, Cambridge, MA pp.19-27.

[Geh89] **Gehani, N.H. (1989).** *"Concurrent Programming and Robotics"*, The International Journal of Robotics Research, Vol.8 (2).

[Gol89] **Goldberg, T. (1989).** Genetic Algorithms, Addison-Wesley.

[GP86] **Grefenstette, J.J. and C. B. Pettey (1986).** "*Approaches to Machine Learning with Algorithms*", IEEE, pp.55-60.

[Gp97] **Gaudiano, P. and C.C. Phone (1997).** "*Adaptive Obstacle Avoidance with a Neural Network for Operant Conditioning: Experiments with Real Robots*", 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '97),

[GRL87] **Golding, A., P.S. Rosenbloom and J.E. Laird (1987).** "*Learning General Search Control from Outside Guidance*", Proceedings of IJCAI '87: International Joint Conference on Artificial Intelligence, Milano, Italy.

[GS94] **Grefenstette, J.J. and A. Schultz (1994).** "*An Evolutionary Approach to Learning in Robotics*", Proceedings of the 1994 Workshop on Robot Learning.

[HA90] **Herman, M., J.S. Albus, et al. (1990).** "*Intelligent Control for Multiple Autonomous Undersea Vehicles*", Neural Network for Control, MIT Press, pp.427-511.

[HaRo77] **Hayes-Roth, F. (1977).** "*Knowledge Acquisition from Structural Descriptions*", Proceedings of Fifth International Joint Conference on

Artificial Intelligence IJCAI '77, Cambridge, Massachusetts.

[HoRo83] **Hayes-Roth, F. (Ed.) (1983)**, Building Expert Systems, Teknowledge series in knowledge engineering, Reading, Massachusetts, Addison-Wesley.

[HP91] **Hartley, R. and F. Pipitone (1991).** "*Experiments with the Subsumption Architecture*", Proceedings f the 1991 IEEE International Conference on Robotics and Automation.

[IK91] **Ikegami, T., J.I. Kato, et al. (1991).** "*Sensor Data Integration Based on the Border Distance Model*", Autonomous Mobile Robots, Vol.1: Perception, Mapping, and Navigation (IEEE Computer Society): pp.339-346.

[JS90] **Jorgensen, C.C. and C. Schley (1990).** "*A Neural Network Baseline Problem for Control of Aircraft Flair and Touchdown*", Neural Network for Control, MIT Press, pp.403-425.

[KA90] **Krozel, J. and D. Andisani (1990).** "*Navigation Path Planning for Autonomous Aircraft: Voronoi Diagram Approach*", Journal of Guidance, Control, and Dynamics, Vol.13 (6): pp.1152-1154.

[Kae93] **Kaelbling, L.P. (1993).** Learning in Embedded Systems, Cambridge, Massachusetts, MIT Press.

[Kar92] **Karalic, A. (1992).** "*Employing Linear Regression in Regression Tree Leaves*", Proceeding of the 10th European Conference on Artificial Intelligence, Wein, Austria, John Wiley & Sons.

[KB89] **Kuc, R. and B. Barshan (1989).** "*Navigating Vehicles Through an Unstructured Environment with Sonar*", IEEE International Conference on Robotics and Automation.

[KB91] **Koren, Y. and J. Borenstein (1991).** "*Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigations*", Proceeding of 1991 IEEE International Conference on Robotics and Automation,

[KK90] **Kahn, P., L. Kitchen, et al. (1990).** "*A Fast Line Finder for Vision Guided Robot Navigation*", IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.12 (11): pp.1098-1105.

[KL90] **Kibler, D. and P. Langley (1990).** "*Machine Learning as an Experimental Science*", Readings in Machine Learning, J.W. Shavlik and T.G. Dietterich (Eds.), Morgan Kauffman Publishers Inc., pp.38-43.

[Kli71] **Kling, R.E. (1971).** "*A Paradigm for Reasoning by Analogy*", Artificial Intelligence, Vol.2: pp.147-187.

[KL93] **Ko, N.Y., B.H. Lee, et al. (1993).** "*An Approach t Robot Motion*

Planning for Time-Varying Obstacle Avoidance Using the View-Time Concept", Robotica, Vol.11 (4): pp.315-327.

[Kon84] **Kononenko (1984).** *"Experiments in Automatic Learning of Medical Diagnostic Rules"*, Ljubjana, Yogoslavia, Jozef Stefan Institute.

[Kor92] **Kortenkamp, D. (1992).** *"Integrating Obstacle Avoidance, Global Path Planning, Visual Cue Detection and Landmark Triangulation in a Mobile Robot*", SPIE, Vol.1831 (Mobile Robots VII): pp.515.

[Koz93] **Koza, J. (1993).** Genetic Programming, MIT Press.

[KR90] **Kaelbling, L.P. and S.J. Rosenschein (1990).** *"Action and Planning in Embedded Agents*", Robotics and Autonomous Systems, Vol.6 (1&2): pp.35-48.

[Kro84] **Krogh, B.H. (1984).** "*A Generalized Potential Field Approach to Obstacle Avoidance Control*", SME Conference Proceeding for Robotics Research: The Next Five Years and Beyond, Bethlehem, Pennsylvania USA.

[KS87] **Kuc, R. and M.W. Siegel (1987).** *"Physically Based Simulation Model for Acoustic Sensor Robot Navigation*", IEEE Transaction on Pattern analysis and Machine Intelligence, Vol.PAMI-9 (6).

[KT86] **Krogh, B.H. and C.E. Thorpe (1986).** "*Integrated Path Planning and Dynamic Steering for Autonomous Vehicles*", Proceedings of 1986 IEEE International Conference on Robotics and Automation, Washington DC, USA, IEEE Computer Society Press.

[Kuf97] **Kufrin, R. (1997).** "*Generating C4.5 Production Rules in Parallel*", Proceedings of 14th International Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference.

[LD92] **Leonard, J.J. and H.F. Durrant-Whyte (1992).** Directed Sonar Sensing for Mobile Robot Navigation, Kluwer Academic Publishers.

[Lea86] **Leach, W.J. (1986).** "*Proceedings of the International Conference and Exhibit*", Instrument Society f America, Research Triangle Park, NC.

[Lee90] **Lee, C.C. (1990).** IEEE Transaction on Systems, Man, and Cybernetics.

[Leo91] **Leonard, J.J. (1991).** "*Mobile Robot Localization by Tracking Geometric Beacons*", IEEE Transaction on Robotics and Automation, Vol.7 (3).

[LK96] **Lee, L.F. and A. Kean (1996).** "*An Architecture for Autonomous Flying Vehicles: A Preliminary Report*", Proceedings of PRICAI '96:

Topics in Artificial Intelligence, 4th Pacific Rim International Conference on Artificial Intelligence, Springer-Verlag.

[LL90] **Levitt, T.S. and D. T. Laeton (1990).** *"Qualitative Navigation for Mobile Robots"*, Artificial Intelligence, Vol.44: pp.305-360.

[LNR86] **Laird, J., A. Newell, and P.S. Rosenbloom (1986).** *"Chunking in Soar: The Anatomy of a General Learning Mechanism"*, Machine Learning, Vol.1 (1): pp.11-46.

[LNR87] **Laird, J., A. Newell, and P.S. Rosenbloom (1987).** *"Soar: An Architecture for General Intelligence"*, Artificial Intelligence, Vol.33 (3): pp.1-64.

[LoPe79] **Lozano-Perez, T. (1979).** *"An Algorithm for Planning Collision Free Paths among Polyhedral Objects"*, Communications of the ACM, Vol.22 (10): pp.560-570.

[LS90] **Lumelsky, V. and T. Skewis (1990).** *"Incorporating Range Sensing in the Robot Navigation Function"*, IEEE Transaction on Systems, Man, and Cybernetics, Vol.20 (5): pp.1058-1991.

[LS87] **Lumelsky, V.J. and A.A. Stepanov (1987).** *" Path-Planning Strategies for a Point Mobile Automaton Moving Admist Unknown Obstacles of Arbitrary Shape"*, Algorithmica, Vol.2

(4): pp.403-430.

[Mae89] **Maes, P. (1989).** "*How to Do the Right Thing*", Connection Science, Vol.1 (3): pp.291-323.

[Mah94] **Mahadevan, S. (1994).** Proceedings of the Workshop on Robot Learning, held in conjunction with the 11th International Conference on Machine Learning IMLC '94.

[Mak91] **Makarovic, A. (1991).** "*A Qualitative Way of Solving the Pole Balancing Problem*", Machine Intelligence, J. Hayes, D. Michie and E. Tyugu (Eds.), Oxford, pp.241-258.

[MB90] **Maes, P. and R.A. Brooks (1990).** "*Learning to Coordinate Behaviors*", Proceedings of AAAI-90, Boston, MA.

[MBM90] **Michie, D., M. Bain, J.E. Heyes-Michie (1990).** "*Cognitive Model from Sub-Cognitive Skills*", Knowledge Base Systems in Industrial Control, M. Grimble, S. McGhee and P. Mowforth (Eds.), Peter Peregrinus, pp.71-99.

[MC68] **Michie, D. and R.A. Chambers (Eds.) (1968)**, Boxes: An Experiment in Adaptive Control, Machine Intelligence 2, Edinburgh, Oliver and Boyd.

[Mc90] **Minton, S., J.G. Carbonell, et al. (1990).** "*Explanation-Based Learning: A Problem Solving Perspective*", Machine Learning: Paradigms and Methods, J.G. Carbonell (Ed.), MIT-Elseveir Press.

[MC91] **Mahadevan, S. and J. Connell (1991).** "*Scaling Reinforcement Learning to Robotics by Exploiting the Subsumption Architecture*", Proceedings of the 8th International Workshop on Machine Learning.

[MC94] **Michie, D. and R. Camacho (1994).** "*Building Symbolic Representation of Intuitive Real-Time Skills from Performance Data*", Machine Intelligence, K. Furukawa, D. Michie and S. Muggleton (Eds.), Vol.13: pp.1-30.

[McKe] **McKerrow, P.J.** "*A Data Fusion Architecture for Ultrasonic Mapping*".

[ME85] **Moravec, H.P. and A.E. Elfes (1985).** "*High Resolution Maps from Wide Angle Sonar*", proceedings of the 1985 IEEE International Conference on Robotics and Automation, St. Louis pp.116-121.

[MFB95] **Montgomery, J.F., A.H. Flagg, G.A. Bekey (1995).** "*The USC AFV-I: A Behavior Based Entry in the 1994 International Aerial Robotics Competition*", IEEE Expert / Intelligent Systems & Their Applications, Vol.10 (2): pp.16-22.

[Mic82] **Michie, D. (1982).** "*The State of the Art in Machine Learning*", Introductory Reading in Expert Systems, D. Michie, Gordon and Breach (Eds.), pp.208-229.

[Mic83] **Michalski, R.S. (1983).** "*A Theory and Methodology of Inductive Learning*", Artificial Intelligence, Vol.20 (2): pp.111-161.

[Mic84] **Michalski, R.S. (1984).** "*A Theory and Methodology of Inductive Learning*", Machine Learning: An Artificial Intelligence Approach, R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.) Morgan-Kaufmann Publishers, pp.83-134.

[Mic90] **Michalski, R.S. (1990).** "*Research in Machine Learning: Recent Progress, Classification of Methods, and Future Directions*", Machine Learning, and Artificial Intelligence Approach, Y. Kordratoff and R.S. Michalski (Eds.), San Mateo, California, Morgan Kaufmann Pub. Inc., Vol.III.

[Mic93] **Michie, D. (Ed.) (1993),** Knowledge, Learning and Machine Intelligence, Intelligent Systems, New York, Plenum Press.

[Min61] **Minsky, M. (1961).** "*Steps towards Artificial Intelligence*", The Institute of Radio Engineers Proceedings, Vol.49.

[Min90] **Minton, S. (1990).** Machine Learning: Paradigms and Methods, MIT-

Elseveir Press.

[Mit88] **Mitchell, J.S.B. (1988).** *"An Algorithm Approach to Some Problems in Terrain Navigation"*, Artificial Intelligence, Vol.37: pp.171-201.

[Mit89] **Mitchell, T.M. (1989).** *"Towards a Learning Robot"*, Carnegie-Mellon University.

[Mit97] **Mitchell, T.M. (1997).** Machine Learning, New York, McGraw Hill.

[MKCS91] **Mansouri, Y., J.G. Kim, P.J. Compton and C.A. Sammut (1991).** *"An Evaluation of Ripple Down Rules"*, Australia, KAW 1991.

[MKK86] **Mitchell, T., R.M. Keller and S.T. Kedar-Cabelli (1986).** *"Explanation Based Generalization: A Unifying View"*, Machine Learning, Vol.1: pp.47-80.

[MMS85] **Mitchell, T.M., S. Mahadevan and L.I. Steinberg (1985).** *"LEAP: A Learning Apprentice for VLSI Design"*, Proceedings of IJCAI '85, Los Angeles, CA pp.616-623.

[MP90] **Malik, R. and E.T. Polkowski (1990).** *"Robot Self-Location Based on Corner Detection"*, SPIE, Vol.1388 (Mobile Robots V): pp.306-316.

[MR86] **McClelland, J. and D. Rumelhrt (Eds.) (1986)**, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Bradfrd

Books.

[MS90] **Miller, W.T., R.S. Sutton, et al. (1990).** Neural Networks for Control,, MIT-Elseveir Press.

[MST94] **Michie, D., D.J. Spiegelhalter and C.C. Taylor (Eds.) (1994),** Machine Learning, Neural and Statistical Classification, England, Ellis Horwood.

[Mug88] **Muggleton, S. (1988).** "*Machine Invention of First-Order Predicates by Inverting Resolution*", Proceedings of the Fifth Machine Learning Workshop, Ann Arbor, MI.

[NL95] **Nam, Y.S., B.H. Lee, et al. (1995).** "*An Analytic Approach to Moving Obstacle Avoidance Using an Artificial Potential Field*", Proceedings of 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '95), Pittsburg, Pennsylvania USA, IEEE Computer Society Press.

[NP90] **Narendra, K.S. and K. Parthasarathy (1990).** "*Identification and Control of Dynamic Systems Using Neural Networks*", IEEE Transaction on Neural Networks,: pp.4-27.

[PH93] **Pearson, D.J., S.B. Huffman, et al. (1993).** "*A Symbolic Solution to Intelligent Real-Time Control*", Vol.11: pp.279-291.

[Pom90] **Pomerleau, D. (1990).** "*Neural Network Based Autonomous Navigation*", Vision and Navigation: The CMU Navlab, C. Thorpe (Ed.), Kluwer Academic Publishers.

[Pom93a] **Pomerleau, D.A. (1993a).** Neural Network Perception for Mobile Robot Guidance, Norwell, Massachusetts, Kluwer Academic Publishers.

[Pom93b] **Pomerleau, D.A. (1993b).** "*Knowledge-Based Training of Artificial Neural Networks for Autonomous Robot Driving*", Robot Learning,,Pomerleau, D.A. (Ed. Norwell, Massachusetts, Kluwer Academic Publishers.

[Put94] **Puterman, M. (1994).** Markov Decision Processes: Discrete Dynamic Stochastic Programming,, John Wiley.

[Qc93] **Quinlan, J.R. and R.M. Cameron-Jones (1993).** "*FOIL: A Midterm report*", Proceedings of European Conference n Machine Learning ECML '97, Springer-Verlag.

[Qui79] **Quinlan, J.R. (1979).** "*Discovering Rules by Induction from Large Collectin of Examples*", Expert Systems in The Micro-electronic Age, D. Michie (Ed.), Edinburgh, Edinburg University Press.

[Qui86] **Quinlan, J.R. (1986).** "*Induction of Decision Tress*", Machine Learning, Vol.1 (1): pp.81-106.

[Qui87a] **Quinlan, J.R. and e. al. (1987a).** "*Induction Knowledge Acquisition: A case Study, Application of Expert Systems*", Application of Expert Systems, J.R. Quinlan (Ed.), Turing Institute Press, pp.157-173.

[Qui87b] **Quinlan, J.R. (1987b).** "*Simplifying Decision Tress*", International Journal of Man-Machine Studies, Vol.27: pp.221-234.

[Qui89] **Quinlan, J.R. (1989).** "*Learning Relations: Comparison of Symbolic and Connectionist Approach*", Proceedings of ISSEK Workshop, Udine, Italy.

[Qui90] **Quinlan, J.R. (1990).** "*Learning Logical Definition from Relations*", Machine Learning, Vol.5 (239-266).

[Qui93] **Quinlan, J.R. (1993).** C4.5: Programs for Machine Learning,, Morgan Kaufmann.

[Rao89] **Rao, N.S.V. (1989).** "*Algorithmic Framework for Learned Robot Navigation in Unknown Terrains*", IEEE Computer Magazine, Vol.22 (6): pp.37-43.

[RB90] **Raschke, U. and J. Borenstein (1990).** "*A Comparison of Grid-Type Map-Building Techniques by Index of Performance*", Proceedings of 1990 IEEE International Conference on Robotics and Automation, Los Alamitos, CA, USA, IEEE Computer Society Press, pp.1828-1832.

[RB92] **Roberts, B. and B. Bhanu (1992).** *"Inertial Navigation Sensor Integrated Motion Analysis for Autonomous Vehicle Navigation"*, Journal of Robotic Systems, Vol.9 (6): pp.817-842.

[RL86] **Rosenbloom, P.S. and J.E. Laird (1986).** *"Mapping Explanation Based Generalization onto Soar"*, Proceedings of AAAI '86, Philadelphia, PA

[RN95] **Russel, S. and P. Norvig (1995).** Arificial Intellignece, A Modern Approach, Prentice Hall Inc.

[Rog97] **Rogers, S.O. (1997).** *"Symbolic Performance & Learning in Continuos-Valued Environment"*, Michigan, The University of Michigan.

[RR90] **Roning, J., J. Riekki, et al. (1990).** *"Simulator for Developing Mobile Robot Control Systems"*, SPIE, Vol.1388 (Mobile Robots V): pp.350-360.

[RS97] **Rybski, P., S. Stoeter, et al. (1997).** *"A Cooperative Multi-Robot Approach to the Mapping and Exploration of Mars"*, Proceedings of 14th International Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference.

[Sam59] **Samuel, A.L. (1959).** *"Some Studies in Machine Learning Using the Game of Checkers"*, IBM Journal of Research and Development,

pp.211-229.

[Sam81] **Sammut, C.A. (1981).** *"Learning Concepts by Performing Experiments"*, "Department of Computer Science", Sydney, Australia, University of New South Wales.

[SB86] **Sammut, C.A. and R.B. Banerji (1986).** *"Learning Concepts by Asking Questions"*, Machine Learning: An Artificial Intelligence Approach, R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), Los Altos, CA, Morgan Kaufmann Pub. Inc., Vol.II: pp.167-191.

[Sc88] **Sobek, R.P. and R.G. Chatila (1988).** *"Integrated Planning and Execution Control for an Autonomous Mobile Robot"*, Artificial Intelligence in Engineering, Vol.3 (2): pp.103-113.

[SC97] **Stormont, D., J. Canulette, et al. (1997).** *"Lbokhod: The University of New Mexico's Robotic Mars Rover"*, Proceedings of 14th International Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference.

[SD94] **Sestito, S. and T.S. Dillon (1994).** Automatic Knowledge Acquisition, New York, Printice Hall.

[She93] **Shen, W.M. (1993).** *"Discovery as Autonomous Learning from the Environment"*, Machine Learning, Vol.12: pp.143-165.

[Shi94] **Shiraz, G.M. (1994).** *"Learning to Control Dynamic System"*, Ofogh, The Journal of Computer Sc. & Eng., Vol.3 (2): pp.65-71.

[Shi97] **Shiraz, G.M. (1997).** *"Building Controller for Dynamic Systems Using Machine Learning and Knowledge Acquisition"*. "School of Computer Sc. & Eng.", Sydney, University of New South Wales, pp.260.

[Sil86] **Silver, B. (1986).** *"Precondition Analysis: Learning Control Information"*, Machine Learning: An Artificial Intelligence Approach, R.S. Michalski and e. al. (Eds.), Los Alts, CA, Morgan Kaufmann, Vol.II: pp.647-670.

[SM91] **Sammut, C.A. and D. Michie (1991).** *"Controlling a Black Box Simulation of a Spacecraft"*, AI Magazine, Vol.12 (1): pp.56-63.

[SS82] **Sakawa, Y. and Y. Shinido (1982).** *"Optimal Control of Container Crane"*, Autmatica, Vol.18 (3): pp.257-266.

[SS85] **Selfridge, O.G., R.S. Sutton, et al. (1985).** *"Training and Tracking in Robotics"*, Proceedings of the 9th International Conference on Artificial Intelligence, Los Altos, CA, Morgan Kaufmann.

[SS95] **Shiraz, G.M. and C.A. Sammut (1995).** *"Learning to Fly in Presence of an Expert"*, Control Proceedings of Third Iranian Conference in

Electrical Engineering ICEE '95, Tehran, Iran.

[SS95] **Stevens, A. and M. Stevens (1995).** *"OxNav: Reliable Autonomous Navigation"*, IEEE International Conference on Robotics and Automation, Vol.3: pp.2607-2612.

[SSE95] **Shiraz, G.M., C.A. Sammut, N. Esmaili (1995).** *"Man-Machine Cooperation for Learning to Control Dynamic Systems"*, Proceedings of 1995 IEEE International Conference on Systems, Man, and Cybernetics SMC '95, Vancouver, Canada.

[ST94] **Safra, S. and M. Tennenholtz (1994).** *"On Planning while Learning"*, Journal of Artificial Intelligence Research, Vol.2: pp.111-129.

[Sti95a] **Stirling, D. (1995a).** *"Learning to Fly with CHURPS"*, The 8th Australian Joint Conference on Artificial Intelligence AI'95, Canberra, Australia, World Scientific.

[Sti95b] **Stirling, D. (1995b).** *"CHURPS: Compressed Heuristic Universal Reaction Planners"*. Basser Department of Computer Science, Sydney, University of Sydney.

[SUKM92] **Sammut, C.A., S. Uurst, D. Kedzier, and D. Michie (1992).** *"Learning to Fly"*, Proceedings of the 9th international Workshop on Machine Learning, Morgan Kaufmann.

[Sut90] **Sutton, R. (1990).** "*Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming*", Proceedings of Seventh International Conference on Machine Learning, Morgan Kaufmann.

[Sut92] **Sutton, R.S. (1992).** "*The Challenge of Reinforcement Learning*", Machine Learning, Vol.8: pp.225-227.

[SW92] **Salin, E.D. and P.H. Winston (1992).** "*Machine Learning and Artificial Intelligence: An Introduction*", Analytical Chemistry, Vol.64 (1): pp.49-60.

[TDLJ97] **Tunstel, E., H. Danny, T. Lippincott and M. Jamshidi (1997).** "*Autonomous Navigation using an Adaptive Hierarchy of Multiple Fuzzy-Behaviours*", 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '97).

[Tho84a] **Thorpe, C. (1984a).** "*Path Relaxation: Path Planning for a Mobile Robot*", Robotics Institute, Carnegie Mellon University.

[Tho84b] **Thorpe, C. (1984b).** "*FIDO: Vision and Navigation for a Mobile Robot*", Robotics Institute, Carnegie Mellon University.

[Til90] **Tilove, R.B. (1990).** "*Local Obstacle Avoidance for Mobile Robots Based n The Method of Artificial Potentials*", Proceedings of 1990

IEEE International Conference on Robotics and Automation, IEEE Computer Society Press.

[TL97] **Tam, K., J. Lloyd, et al. (1997).** "*Controlling Autonomous Robots with GOLOG*", 10th International Conference on Artificial Intelligence AI' 97, Springer-Verlag.

[UB93] **Urbančič, T. and I. Bratko (1993).** "*Learning to Control Dynamic Systems*", Machine Learning, Neural and Statistical Classification, D. Michie, D. Spiegelhalter and C. Taylor (Eds.), Ellis-Harwood.

[UB94] **Urbančič, T. and I. Bratko (1994).** "*Reconstructing Human Skill With Machine Learning*", Proceedings of 11th European Conference on Artificial Intelligence.

[UJ92] **Urbančič, T., D. Jurii, et al. (1992).** "*Automated Synthetic of Control for Non-Linear Dynamic Systems*", Proceedings of IFAC/IFIP/IMACS International Symposium on Artificial Intelligence in Real-Time Control.

[Ung90] **Ungar, L.H. (1990).** "*A Bio-reactor Benchmark for Adaptive Network-Based Process Control*", Neural Network for Control, MIT Press, pp.387-403.

[Wat92] **Watkins, C. (1992).** "*Technical Notes on Q-Learning*", Machine

Learning, Vol.8: pp.278-292.

[WD95] **Weckesser, P., R. Dillmann, et al. (1995).** "*Multiple Sensor-processing for High- Precision Navigation and Environmental Modeling with a mobile Robot*", Proceedings of 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '95), Pittsburgh, Pennsylvania USA, IEEE Computer Society Press.

[WDEP89] **Weisbin, C.R., G. DeSaussure, J.R. Einestein and F.G. Pin (1989).** "*Autonomous Mobile Robot Navigation and Learning*", IEEE Computer Magazine, Vol.22 (6): pp.29-35.

[Wei76] **Weizenbaum, J. (1976).** Computer Power and Human Reason, San Francisc, W.H. Freeman & Co.

[Wil75] **Winston, P.H. (1975).** "*Learning Structure Description from Examples*", The Psychology of Computer Vision, P.H. Winston (Ed.), New York, McGraw-Hill.

[Wil87] **Wilson, S.W. (1987).** "*The ANIMAT Path to AI*", From Animat to Animats; Proceedings of the First International Conference on the Simulation of Adaptive Behavior, J. A. Meyer and S. W. Wilson (Eds.), Cambridge, MA, MIT Press / Bradford Books.

[Win84] **Winston, P.H. (1984).** "*Learning and Reasoning by Analogy*", Journal

of CACM, Vol.23 (12): pp.680-703.

[WK90] **Weiss, S.M. and I. Kapouleas (1990).** "*An Empirical Comparison of Pattern Recognition, Neural Nets, and Machine Learning Classification Methods*", Proceeding of the 11th International Joint Conference on Artificial Intelligence, Michigan, USA, pp.781-787.

[WL90] **Widrow, B. and M. A. Lehr (1990).** "*30 Years of Adaptive Neural Networks: Perception, Madeline, and Back-Propagation*", IEEE Proceedings on Neural Networks, Vol.48 (9): pp.1415-1442.

[WW88] **Widrow, B. and R.G. Winter (1988).** "*Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition*", IEEE Computer Magazine, (March): pp.25-39.

[WZ97] **Ward, K. and A. Zelinsky (1997).** "*Learning Mobile Robot Behaviors by Discovering Associations Between Input Vectors and Trajectory Velocities*", Proceedings - Poster Papers of 10th Australian Joint Conference on Artificial Intelligence.

[YB94] **Yu, W. and Z. Bein (1994).** "*Design of Fuzzy Logic Controller with Inconsistent Rule Base*", Journal of Intelligence and Fuzzy Systems.

[YP93] **Yaseh, I. and B. Priel (1993).** "*Design of Leveling Loop for Marine Navigation System*", IEEE Transactions on Aerospace and Electronic

Systems, Vol.29 (2): pp.599-604.

[YS97] **Yamauchi, B., A. Shultz, et al. (1997).** *"ARIEL: Autonomous Robot for Integrated Exploration and Localization"*, Proceedings of 14th International Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference, pp.804-805.

[ZB90] **Zhao, Y. and S.L. BeMent (1990).** *"A Heuristic Search Approach for Mobile Robot Trap Recovery"*, SPIE, Vl.1388 (Mobile Robots V): pp.122-130.