The design and intelligent control of an autonomous mobile robot

Robinson, Stephen David

The Design and Intelligent Control of an Autonomous Mobile Robot

Stephen David Robinson BSc.
School of Engineering, University of Durham.

A thesis submitted in accordance with the requirements of the Council of the University of Durham for the degree of Doctor of Philosophy (Ph.D.).

December 1996

Abstract

This thesis presents an investigation into the problems of exploration, map building and collision free navigation for intelligent autonomous mobile robots.

The project began with an extensive review of currently available literature in the field of mobile robot research, which included intelligent control techniques and their application. It became clear that there was scope for further development with regard to map building and exploration in new and unstructured environments.

Animals have an innate propensity to exhibit such abilities, and so the analogous use of artificial neural networks instead of actual neural systems was examined for use as a method of robot mapping. A simulated behaviour based mobile robot was used in conjunction with a growing cell structure neural network to map out new environments. When using the direct application of this algorithm, topological irregularities were observed to be the direct result of correlations within the input data stream. A modification to this basic system was shown to correct the problem, but further developments would be required to produce a generic solution. The mapping algorithms gained through this approach, although more similar to biological systems, are computationally inefficient in comparison to the methods which were subsequently developed.

A novel mapping method was proposed based on the robot creating new location vectors, or nodes, when it exceeded a distance threshold from its mapped area. Network parameters were developed to monitor the state of growth of the network and aid the robot search process. In simulation, the combination of the novel mapping and search process were shown to be able to construct maps which could be subsequently used for collision free navigation.

To develop greater insights into the control problem and to validate the simulation work the control structures were ported to a prototype mobile robot. The mobile robot was of circular construction, with a synchro–drive wheel configuration, and was equipped with eight ultrasonic distance sensors and an odometric positioning system. It was self–sufficient, incorporating all its power and computational resources.

The experiments observed the effects of odometric drift and demonstrated methods of re–correction which were shown to be effective. Both the novel mapping method, and a new algorithm based on an exhaustive mesh search, were shown to be able to explore different environments and subsequently achieve collision free navigation. This was shown in all cases by monitoring the estimates in the positional error which remained within fixed bounds.

Acknowledgements

Not through tradition but genuine appreciation must I first thank Professor Phil Mars for his supervision and all round support. Phil's wit and infectious enthusiasm for all things academic and social have proved invaluable over my time in Durham and I owe him a great debt of gratitude.

Thanks must be extended to the Engineering and Physical Sciences Research Council whose financial support is also gratefully acknowledged.

I would also like to thank Dr Jim Bumby for his advice and the donation of the robot's ultrasonic sensors. Thanks to Peter Baxendale and Jim Swift for their help relating to the computer board and the Xilinx FPGAs.

For their advice, and excellent construction of the mobile robot I wish to thank Brian Blackburn and Roger Little of the mechanical workshop. I would like to thank all the members of the electronics workshop, especially Peter Friend and Ian Hutchinson for their humour, advice and assistance. For their computing assistance I would also like to thank Jonathan Spanier, Mathew Jubb, and John Glover.

I am deeply indebted to all my friends, especially Martin Bradley and Mark Docton, for their enthusiasm, zest and camaraderie for without which Durham would certainly have been a lesser place.

Finally I would like to thank my mother, father (who tirelessly proof read this work), and family, for their ever present support throughout all my endeavours.

Declaration

I hereby declare that this thesis is a record of work undertaken by myself, that it has not been the subject of any previous application for a degree, and that all sources of information have been duly acknowledged.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview of Autonomous Mobile Robotics

Man has long dreamed of the possibility that one day he might be able to create artificial creatures so complex that they could rival animals or even human beings. This dream can be traced through myths of antiquity, to the elaborate clockwork automata of more recent centuries, and eventually to the burgeoning field of present day robotics [1]. Embedded at the forefront of this technology remains the aspiration to develop autonomous robotic machines which can interact intelligently with new surroundings without the need of human intervention. Both the unfulfilled nature of these goals and the benefits of working solutions ensures this is a current area of active research.

It is appropriate to define, in this context, the use of terms such as 'intelligence' and 'autonomous mobile robot'. Any single definition of intelligence encompassing all its widespread meanings would be controversial. For the purpose of this thesis it will be used to denote that ability to vary behaviour in response to changing situations, requirements and past experience [2]. The inclusion of 'autonomous' and 'mobile' draws distinction from fixed manipulator robots and other systems which are not entirely self–contained. Autonomous mobile robot research is therefore concerned with self–sufficient vehicles which can exhibit intelligent behaviour when faced with new conditions. This is in contrast to the study of 'automatic guided vehicles' (AGVs) which examines the problems of robots which have more defined objectives within structured environments.

The diversity of the research effort stems from the broad range of potential applications for which mobile robots can be used. In industry there is a need for robots to transport goods around unstructured environments. In hospitals similar robots are being developed for transportation. Other obvious applications involve robots operating in hazardous environments where it is unsafe for humans to work. These conditions arise in the chemical and nuclear industries where it could be important to map out contaminated areas. In the offshore industries most submersible vehicles are tethered, but this limits the robot's range, and tethers can become snagged. Therefore there is a need for free–swimming vehicles which are self–contained and have some intelligence [3]. The use of robots in space is of active interest especially as the communication time lag created by the speed of light increases over distance. It then becomes more important for the robots to share some of the control burden.

The major contributors to the research have come from backgrounds of physics, mathematics, electrical engineering, mechanical engineering and computer science. Each different approach has focused on issues considered to be important to a particular application. The diversity of activities has ranged from the practical aspects of propulsion involving legged motion, to the artificial intelligence problems of representing and manipulating knowledge in a useful manner. At the lower physical level of endeavour such systems can be shown to work or fail within relatively defined constraints. However in regard to the comparative examination of intelligent systems there exists no such universally regarded benchmarks. This results not from a lack of cooperation but a lack of agreement of importance of issues, originating from the diverse set of applications. There is still fundamental disagreement over the best strategy for which intelligent systems can be constructed. Some researches favour traditional artificial intelligence techniques, often referred to as the planning approach, whilst others attempt to dispense with any form of internal representation often called reactive control. Until systems emerge which are generally recognised as being effective solutions to robotic problems it must be important to consider all the potentially intelligent techniques available for use by autonomous mobile robots.

One core component of mobile robot systems is that of positioning with respect to some wider frame of reference. No robot can usefully operate if it cannot position its task within its frame of influence. This is the problem of navigation and requires, not only, robot positioning within a frame of reference but also the ability to move from one location to another. Many methods have been used to simplify the difficulties of this problem often involving the modification of an environment to suit a robot. Other solutions rely on providing a map for the robot which includes features which the robot is known to be able to determine in advance. However the most challenging aspects of the problem occur when a robot has to build up its own information integrating new knowledge into existing representations. This active area of research is still in the developmental stage and represents the main topic of this thesis.

1.2 Summary of Thesis

The initial aim of this work was to investigate the field of autonomous mobile robots in order to find an area of research for which further investigation was required. Chapter 2 presents a review of the current state of the research field. This provides a critical summary of intelligent control algorithms followed by their application as described in the literature. The chapter reaches the conclusion that the problem of exploration, map building and navigation without collision must be further addressed.

Chapter 3 commences with a discussion of the potential merits of the use of simulation at the start of the design cycle. The design strategy and an overview of the simulator is then explained, including a description of the mathematical modelling of the physical systems. The potential structure for the robot controller is also described in greater detail with respect to the problems of exploration and mapping.

Chapter 4 examines the background to biological as well as artificial neural networks as a potential solution to environmental mapping. A neural network solution to the problem is described and then tested in simulation. Problems associated with this method are examined and comparative tests are undertaken on two modifications. The conclusions from this work are drawn and suggestions to make this method a generic solution are presented.

Chapter 5 examines a novel mapping method based on the placement of new nodes outside the original map. Following the detailed description of this method, network parameters are described which can be used to monitor the growth and aid in the exploration process. An exploration strategy is described based on this approach which is then simulated in an environment with obstacles. The resulting map is then navigated by the simulated robot using a shortest path search technique. This work is summarised and a decision is made to port the systems onto a real mobile robot.

The design of the prototype mobile robot is described in Chapter 6. The design is subdivided into mechanical, electronic and computational systems which are then described in greater detail individually. Emphasis is placed on the reasons behind the design decisions.

The basic robot tests are described in Chapter 7. These were conducted to produce an accurate assessment of the actuator and sensor systems. Not only are the various sub–systems calibrated but the interactions of these elements during operation are observed. Only from this detailed knowledge can the actions of the robot controllers be fully understood.

Chapter 8 describes the experiments in exploration, mapping and navigation which were performed on the mobile robot. This was undertaken to compliment the previous work performed in simulation. The experiment examined robot re–synchronisation and the exploration and navigation of two different mapping techniques. In the final section conclusions are drawn from the experimental work.

The thesis ends with Chapter 9 which describes the conclusions that this work has provided. Finally suggestions for further work are made.

Chapter 2

Review of Current Mobile Robot Research

2.1 Introduction

Over recent years, the field of autonomous mobile robots has produced an increased volume of literature relating to task achieving agents which require minimal human interaction. This chapter collates this information and presents a survey of the methods and results reported to date and hence, those aspects which require further attention can be determined. To do this the present range of intelligent algorithms will be examined, followed by their results as robot controllers. Only then can the advantages and disadvantages of each system be considered in order to build an objective picture of the aims and success of the present research. This chapter finishes with a critical analysis of the literature in an attempt to extract those ideas which appear to have the most potential for implementation.

2.2 Review of Control Algorithms

This review introduces some of the fundamental problems associated with the control of mobile robots and then broadly covers the relevant aspects of neurocontrol, fuzzy logic, genetic, and reinforcement algorithms. The research is highly diverse and often incorporates many different control strategies. It is therefore not possible to fully categorise each instance into a particular discipline, but it is useful to indicate the general motivation. The section has been subdivided into categories, reflecting the range of reactive through to planning architectures. However this should provide an insight into the tools that are being applied to the problems of intelligent robot control.

Attention has recently been focused on the application of intelligent control to problems not solvable by conventional control techniques. Mobile robot control involves complex interactions between the controller, the environment and the uncertainty involved. In these dynamic, non-deterministic and previously unknown environments, it is insufficient for a controller to be purely static and pre-programmed. To be robust, a controller must be flexible and adaptive, utilising on–line learning. Hence, intelligent control must be considered as the use of general-purpose control systems, which learn over time how to achieve goals (or optimise) in complex, noisy, non-linear environments whose dynamics must ultimately be learned in real time [4].

More generally mobile robots are examples of *embedded systems* which must interact with the environment for the duration of their working life where the environment is considered to be everything which is not the robot, and is often dynamic in nature [5]. Embedded systems take sensor information and in combination with internal state, attempt to evaluate the best action. The internal state consists of the robot's memory information and may incorporate all previous sensor-actuator data. Although internal states are not necessary (pure reactive controllers), they allow the system to distinguish between identical sensor information by temporal assessment.

Dependent on the use of internal state, mobile robots can be categorised between the two extremes of *reactive* and *planning* controllers. In the past, Artificial Intelligence (AI) techniques were proposed for the logical control of robotics applications. They require a world model at a level of definition so that all lifetime tasks can be incorporated in the analysis. This prior knowledge is unrealisable in any real world situation, and to compound the problem the computational capabilities cannot cope with these large models. Another unattainable assumption is that the world interface will be precise enough for this large internal world model to remain synchronised with that of the world. It is therefore more useful to consider a looser definition of a planning controller.

Planning can be considered to be any sort of internal construct which constrains or guides the robot. This causes problems of maintaining the validity of the internal state information. Reactive control attempts to overcome this by minimising internal state's duration and quantity invoking short computational sequences. Some researchers call for the eradication of any formal planning [6], whereas others argue for the smooth integration of a traditional symbolic planner [7]. This is currently an important aspect of research and it is central to the control problem.

2.2.1 Neurocontrol

The ability of animals to thrive in the world and produce a rich set of behaviours can be attributed to their neuronal mechanisms. Neurocontrol is defined as the use of well specified neural networks (artificial or natural) to emit actual control signals [8]. Hence, control solutions using neurocontrol attempt to mimic the success which natural structures produce. Real nervous systems decompose the problem in parallel onto many neurons both temporally and spatially and are able create complex responses. To model nervous systems at a simplified level, Artificial Neural Networks (ANNs) have been developed. This development and its application in control systems aims to replicate those aspects of biological systems that conventional techniques cannot adequately cope with. These include the ability to integrate multiple sensor data both spatially and temporally, reject noise throughout the whole system, and be generally fault tolerant.

Research into the use of artificial neural networks was originally derived from observations of neural mechanisms where simplified mathematical models of brain cells are combined into networks. However, it is the underlying mathematical construct of a network built up of differentiable functions which is important, and hence ANNs are also referred to a connectionist systems, reducing biological ambiguity. Although research is highly diverse, it is possible to point to three elements that ANN's possess [9]. To determine information throughput they require an organised topological network of processing elements. A method of encoding information or training is required and also a method of recalling information.

The smallest functional unit in an ANN is a processing element (PE) and is shown in figure 2.1. An input vector to the node is multiplied by a weights matrix and summed together before being acted upon by a threshold function to give an output. In this manner an infinite input range can be mapped into a bounded output, of which the most commonly used threshold function is the sigmoid.

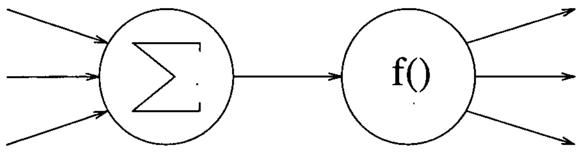


Figure 2.1: A single processing element

With greater numbers of PE's linked together the topological combinations increase exponentially, so it becomes useful to consider collections of neurons called fields (layers) and their connections both internal and external. This gives rise to three distinct types of connection. Intra field, or lateral connections, connect neurons in a specific layer. Inter field connections allow outputs from one layer to be used to drive the inputs of another. Recurrent connections introduce at least one feedback loop into the forward path of data flow.

Learning Mechanisms

For any system to learn it must gain knowledge. Learning takes place between the two extremes of being guided by a knowledgeable teacher, and pure discovery (using information redundancy). These correspond to the categories of supervised and unsupervised learning schemes respectively. In some situations a fully knowledgeable teacher is non realisable and a method of reinforcement learning is used.

Supervised Learning In supervised learning a network's desired and actual responses are compared to provide an error value which can be used to adjust the system. Error correction learning methods use this error vector to update the weights in the network. This requires all the information at the moment of training and is therefore is not satisfactory for on–line learning. Common examples of this technique are the multi layer perceptron (MLP) using backpropagation of errors for training, radial basis functions (RBF), and the cerebellar model articulation controller (CMAC).

Multi layer perceptrons do not require system models but learn data associations. To construct an ANN model of the inverse dynamics of a system, the network would be trained using samples of input and output data. In mobile robotics applications it is not often feasible to build up a mathematical model of a system and therefore this method offers a solution. However, it is slow to converge (train) which is computationally expensive. An important factor with MLPs is the inherent *global* non–linearity in all the layers which causes an error surface with potentially many local minima. Hence, although a desired function can be approximated by an ANN, training techniques cannot be proven to converge on the global minima. Another limitation is that associations cannot be independently manipulated without retraining. This is because the threshold functions act globally so that the memory is distributed and highly interrelated (a input output association is affected by many parameters).

The literature reports MLPs being used to map the inverse dynamics of the world model, simply connecting sensors to motor actuators [10] [11]. Other projects have incorporated ANNs into a larger control scheme and in this respect they have been used for sensor mapping, data confidence [12] and function approximation [11].

Radial basis functions and CMAC are also trained in the above manner but avoid some of the problems that beset MLPs. This is primarily because the threshold function acts *locally* (the output is active only over a range of the input value) so that only a small number of parameters affect network output for a given input. This allows local functional relationships to be changed without disturbing pre-taught actions. The output layers in these networks use a linear threshold function which form only one global error minima, hence fast and proven convergence. Since convergence can be shown for a given set of radial basis functions, network error minimisation becomes the problem of basis function placement.

Reinforcement Learning Reinforcement learning becomes more applicable as the density of reward information that can be assigned for given output decreases. When scalar reward values can be derived for an output, methods such as the adaptive heuristic critic (AHC) can be used. An example of this method successfully being employed is by Gachet [13] where reinforcement is only available from performed actions. Gachet uses a neural network to implement the AHC, but in general, other reinforcement algorithms can be used. These techniques are further discussed in the reinforcement learning section.

Unsupervised Learning A controller which receives no performance feedback from the environment about its previous actions must learn in an unsupervised manner. It must utilise redundancy in the input output data to self organise discovering patterns, features, regularities, correlations and categories.

Hebbian learning is an example of an unsupervised learning technique based on the correlation of values of two PE's. Hebb describes this method of learning in his book *Organisation of Behaviour* which is reported in [14].

> When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency as one of the cells firing B is increased.

The method is often integrated into larger ANN structures between specific nodes or layers. In this way action correlations between pre-encoded behaviours can be learned on-line [15]. Another variation of the basic learning strategy is explored by Heemskerk [16].

Recurrent Networks Recurrent networks fall into two broad classifications, those of simultaneous recurrence, and time lagged recurrence [8]. Simultaneous recurrent networks use an iterative relaxation technique and converge to stable static patterns. They can be considered as a static mapping method. Time-lagged recurrent networks (TLRN) offer greater computational power at the cost of more complex adaptation. In a TLRN each neuron at time t can use as an input any output value of neuron at $(t-1)$. Events occurring over a period of time greater than that within lag -1 can be represented in exponential decays and bucket brigades.

Recurrent networks exhibit the property of short term memory. If presented momentarily with a stimulus it will effect the response of the network over a period of time. They can also estimate parameters whose value varies slowly over time. Attention has also focused on the construction of deterministic finite state automatas using recurrent neural networks. However, the theory of these systems remains unclear at the present time. Until this is further advanced it will not be possible to accurately determine the benefits or limitations of their use.

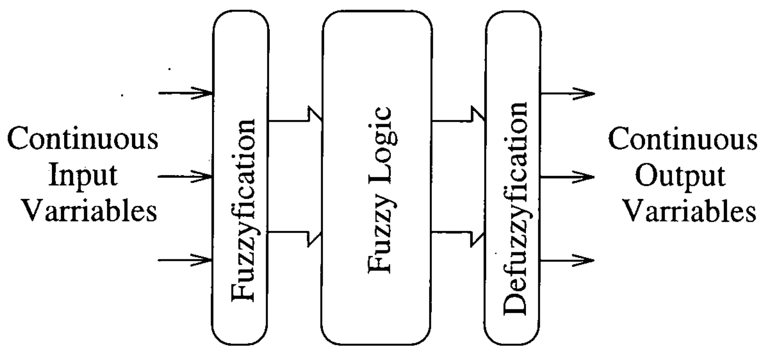


Figure 2.2: The fuzzy controller

2.2.2 Fuzzy Logic

Fuzzy logic Control is a knowledge based control strategy that allows a more abstract *rule based* model of a system to be used in an embedded controller. A control application will be described by rules relating sets of input situations to sets of output controls. A controller simply 'fuzzifies' the input variables and then performs logical operations on the fuzzy sets, as directed by the control rules, these are then 'de-fuzzified' to produce control outputs (figure 2.2).

Each fuzzy set possess a membership function which maps any input variable to a membership value (or Grade value) of that set. The sets form *fuzzy partitions* of the input's domains, as can be seen in the three set of low, medium and high in figure 2.3. The membership function used for these sets is bell-shaped, but other functions can be used including monotonic, triangular and trapezoid. These sets overlap in coverage allowing interpolation at the rule boundaries. The grade value is then used to determine the extent of action rules based upon that input. In an example case with no logical operators (for illustration simplicity) the following rules apply.

IF u is Low THEN take action A_1
IF u is Medium THEN take action A_2
IF u is High THEN take action A_3

For a given value of u we can compute the membership values for the three sets, Low, Medium and High. From this value we then use an inference process on the three actions A_1, A_2 and A_3 as shown in figure 2.4. Although this technique multiplies the membership value and action set, other methods such as thresholding can be used. Finally de-fuzzification is achieved by extracting information contained in the resultant manipulated sets into a single output at a given instant (y_{out}). One such method of achieving this is by taking moments of area as shown in figure 2.5.

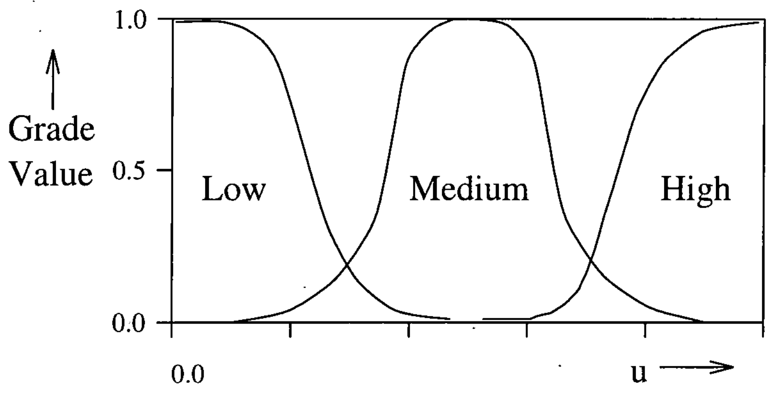


Figure 2.3: Fuzzy set partitions

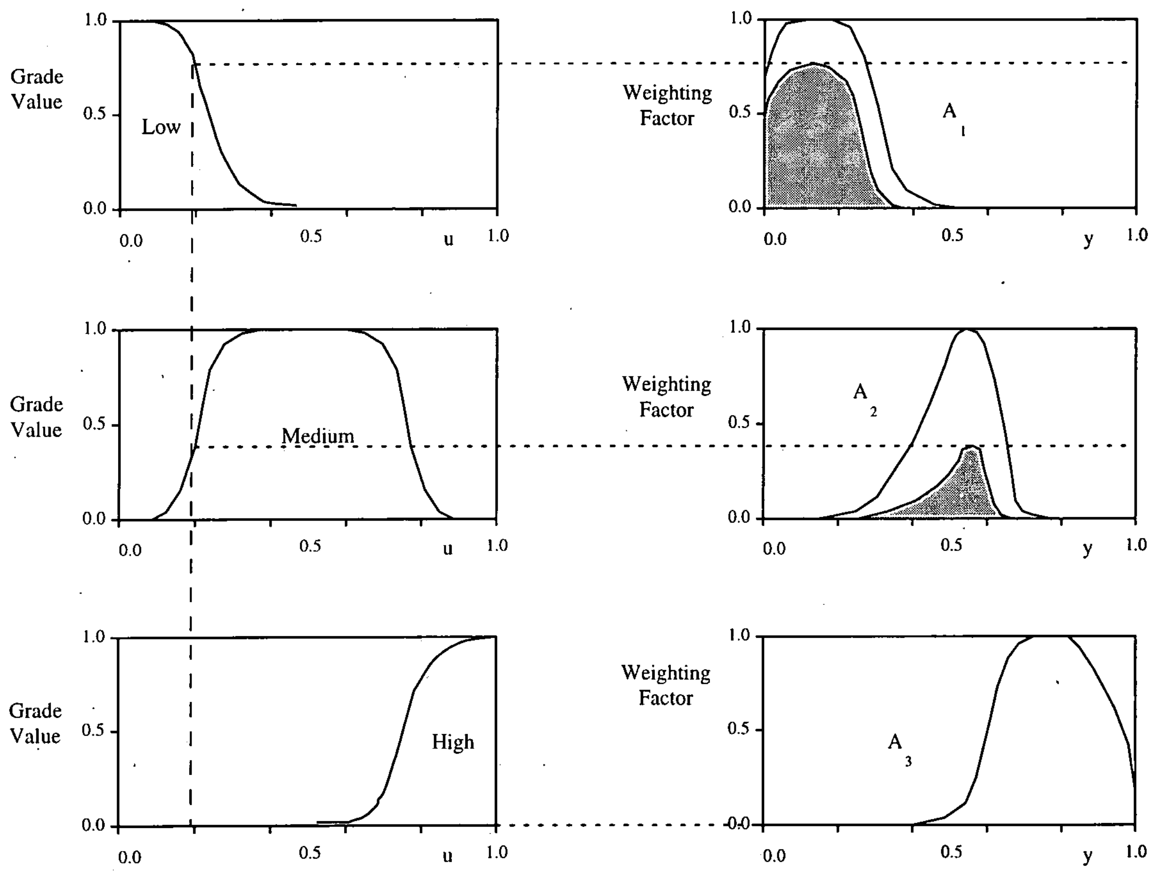


Figure 2.4: The inference process

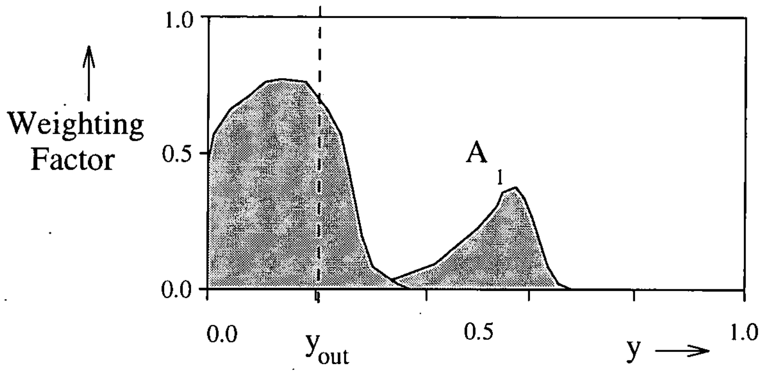


Figure 2.5: De-fuzzification

Fuzzy logic's non–linear control is achieved by manipulation of the fuzzy sets with logical operators. In concurrence with Boolean logic, operators such as *AND*,*OR* and *NOT* are used on the sets. The *AND* operator takes the minimum of the membership functions, the *OR* operator takes the maximum of the membership functions, and *NOT* corresponds to taking one's complement of the membership functions.

Sofge and White [4] describe fuzzy logic and neurocontrol in the broader field arguing that fuzzy logic is an innovative subset of 'old AI' and neurocontrol is an innovative subset of classical control. Both fuzzy control and neurocontrol deal with non–linear variables of bounded range [0,1] and there governing equations are similar. Hence, neurocontrol and fuzzy adjoin (leading to the idea of neuro–fuzzy control), linking the original disciplines. It is therefore possible to incorporate some or all of these methods into a control system.

2.2.3 Reinforcement Algorithms

Reinforcement algorithms are important for the control of mobile robots as they learn on–line using more abstract goal information. Reinforcement learning systems act to improve their performance over time, given an evaluative feedback signal (reinforcement) derived from previous actions. It is self evident that the upper limit of performance is determined by the quality of reinforcement signals received, which is often referred to as the credit assignment problem. This can be broken down into two areas of the *structural* and *temporal* credit assignment. Structural credit assignment deals with the distribution of credit across states, whereas temporal credit assignment determines actions, responsible for credit, over time.

Reinforcement learning systems receive only *scalar* values as a *reinforcement* feedback and therefore it is often referred to as learning with a critic, as opposed to supervised learning. There are two basic types of reinforcement, that which is immediate, relating to the last state–action pair, or delayed, where reinforcement is received only at a particular goal state. This provides only *evaluative feedback* of the performance with respect to a particular goal but does

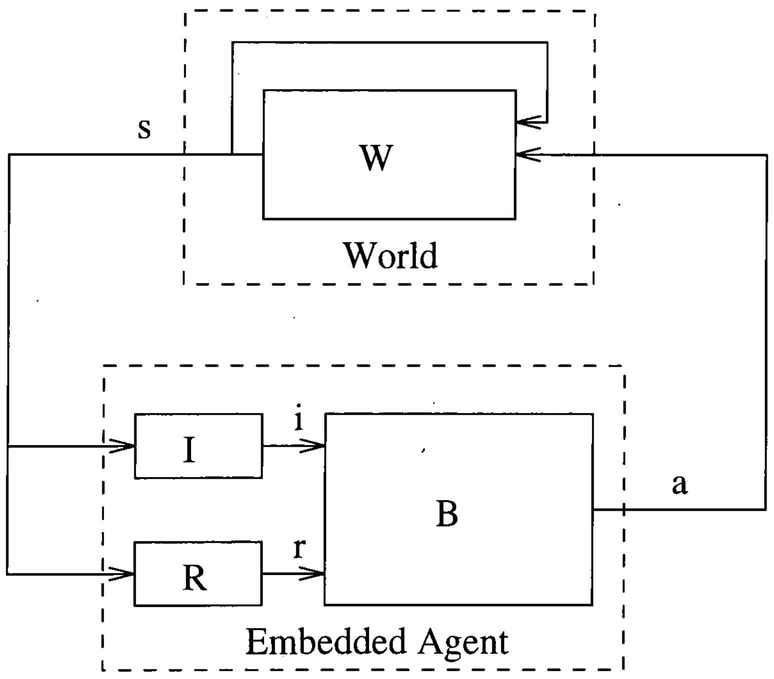


Figure 2.6: The reinforcement learning framework

not provide direct error information of the system's internal representation [17]. There is a fundamental compromise which exists in reinforcement learning systems between exploitation and exploration. Any system that *only* receives appraisal from a performed action cannot possess certain knowledge of rewards associated with actions not performed. Therefore in any situation, an agent can exploit the knowledge it has gained, or explore a different action to attempt to find a better alternative. Systems biased to exploiting will perform better in the short term, and systems with an exploratory bias should converge better in the long term.

Consider the application of reinforcement learning to a navigational task for a mobile robot operating in two dimensions. The inter-relation between the robot and the environment, or reinforcement learning framework, can be viewed as in figure 2.6. In this example the world *state* s would be the position of the robot in a static environment. The robot can change its position, the world state, through actions performed a. A task might involve moving to a particular destination which would require the robot to sequence actions to manipulate the world into the corresponding *goal* state. The agent must utilise information i received from sensors which map (I) the world state guided by reinforcement r. This reinforcement signal is manufactured from the world state through the application of credit assignment. The robot must therefore attempt to maximise its cumulative reinforcement received. An example of an immediate reinforcement scheme would be to return reward value dependent on how near it is to an optimal path. However, if a reinforcement signal was only available at the goal state then a method which used delayed reinforcement would be required.

Immediate reinforcement algorithms

Probability vector algorithms including the Stochastic Learning Automaton (SLA) use reinforcement signals to attribute a probability distribution over all possible actions. This system is based only on the reinforcement signal for a particular action and not on sensor information. In effect only the mapping R is used and it cannot associate locally optimal actions with particular states of the world as there is no internal state access to sensor information. Examples of this method are the Linear Reward Penalty(L_{RP}), Linear Reward Inaction (L_{RI}), and Interval Estimation (I_E). The method can only search for the global best actions based on probabilities, hence it can get caught in non optimal states or *absorbing states.*

Associative algorithms utilise the input function I as well as the reinforcement values, allowing them to associate action probability vectors with states. It is not sufficient to use a separate stochastic learning automata for each state as this would not generalise across actions. To be able to generalise, a parametrised distribution function is required of which the most common implementation is to use a neural network. The main advantage of associative algorithms is that they are able to associate optimal actions over whole classes of world state. They also generalise from past interactions with the world but it is not always possible to scale them to more complex problems.

Delayed Reinforcement

The above algorithms require a constant and accurate feedback of reward after each action has been taken in the environment. Tasks faced by mobile robots are often defined in terms of goal positions or end states into which an environment must be manipulated, prior to which no reward is attained. One method used for these problems is that of *temporal difference* methods which include the adaptive heuristic critic and a form of dynamic programming called Q–learning. [18]

Q–learning discretely partitions the world into a finite set of states s and then learns the value of every action a performed from that state. The function $Q(s, a)$ is built up which returns the expected discounted reward of taking action a in state s given that optimal actions are taken thereafter. As the world is explored, reinforcement values become dispersed throughout the states and converge to the correct value, as the environment is exhaustively searched. The method is computationally expensive and the Q function must be re–learned for new goals.

The adaptive heuristic critic (AHC) learns the function which attributes expected future reinforcement to each input state. The algorithm stores two vectors, one for expected reinforcement and the other which measures the frequency that it has been in that state (the states activation). When no reinforcement signal is present, all expected reinforcement values decay with respect to their activation and a *discounting factor.* With the application of a

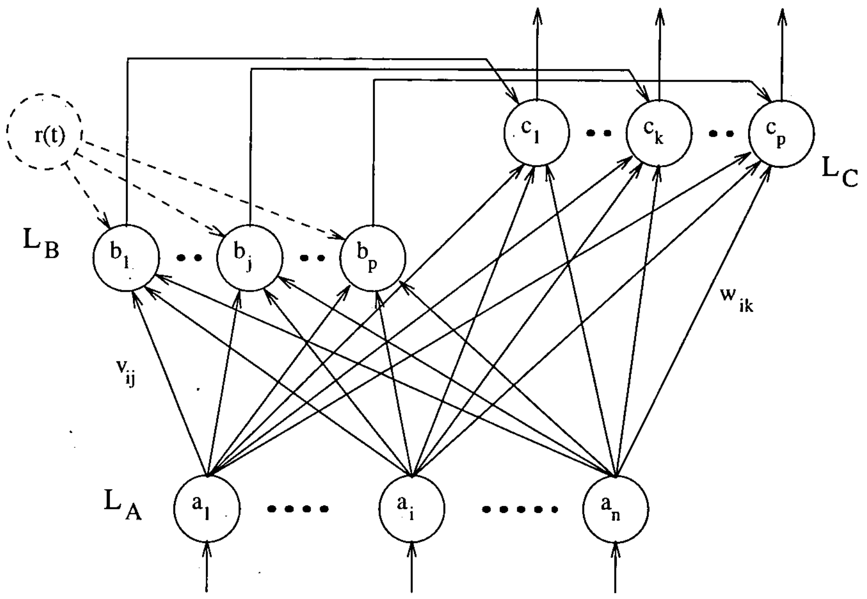


Figure 2.7: The adaptive heuristic critic

reinforcement signal, the expected values are increased dependent on their activations. Therefore, reinforcement is propagated back through the actions which were more active, prior to a global reinforcement, spreading reward through time. The AHC method converts the global reinforcement problem to a local one which can then be solved by most immediate learning algorithms. Figure 2.7 shows a neural implementation where the expected discounted future reward values are stored by the layer L_B neurons and the activation value is based on values in the L_A and L_C layers.

The primary aim for reinforcement learning algorithms is to be able to learn in real time. However, the complex learning algorithms can cause slow convergence to optimal behaviours which can be directly related to the time taken for each iterative step. The operation of each step in time and data size are often bounded, and these systems are referred to as being *strictly incremental.* For real control applications, these algorithms must be used in dynamic environments and their capability to operate will primarily depend on their transient activity.

2.2.4 Genetic Algorithms

A Genetic Algorithm (GA) is a method which drives a population of points in a search space using a set of operators to maximise a given performance measure or *fitness* [19]. The algorithm is not guaranteed to find the optimal solution but will converge to near optimal solutions. The members of the population are individually evaluated and the best are carried into the next generation, proportionate to performance. Genetic operators are used to recombine existing members, aiding crossing over of beneficial data strings. Data mutation are

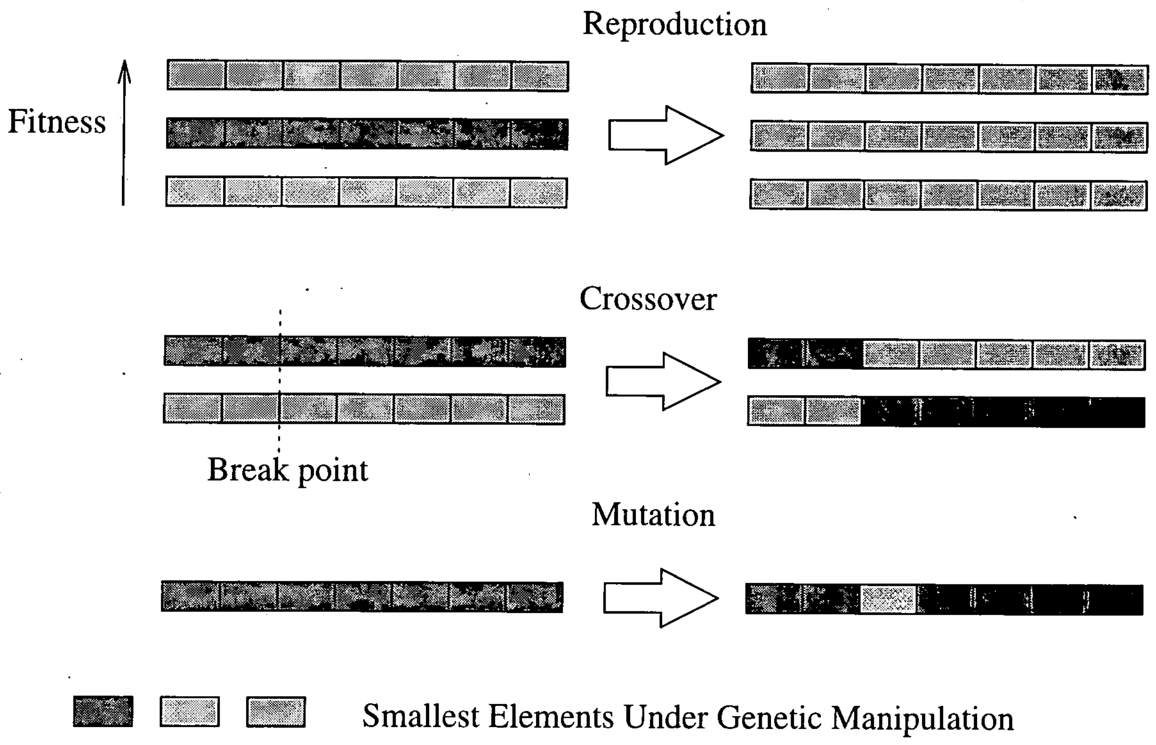


Figure 2.8: The genetic operators

also used to expand the range in the search space.

Points in the search space correspond to a string of data elements (conventionally bits in a bit string) and the set of all points is called the breeding pool. The bits are analogous to 'genes' and the strings attempt to imitate gene structures or 'chromosomes'. Initially each individual is tested in the problem domain to evaluate its performance or *fitness*. This information is then used to determine the members that will be subject to the *genetic operators* and those will be discarded. The operators used most commonly are *reproduction*, *crossover* and *mutation* and their action on small string is graphically shown in figure 2.8.

The *reproduction operator* selects the fittest individuals and copies them to keep the population size the same for each generation. This is done on a probabilistic basis proportionate to fitness (often referred to as the roulette wheel) which ensures a smooth selection process. The members which are not selected are discarded. The *crossover operator* selects a pair of strings at random, picks a random break point along its length and then swaps the data strings over. The *mutation operator* selects randomly both individual and element in it, to be mutated. This stops the algorithm from *premature convergence* where improvement is halted because all individuals have the same value for a specific gene. It also prevents the search from being sparse.

The genetic algorithm can be viewed as critic based. With the overall performance being a compromise between exploration and exploitation in the search space. Where the reproduction operator acts in an explorative fashion utilising known performance and the crossover operator explores the space for possibly better alternatives.

2.2.5 Summary of Control Algorithms

The algorithms described above, comprise some of the latest ideas in intelligent control, because they have the ability to vary their behaviour in response to varying situations. This makes these algorithms of particular importance for mobile robots, but at this stage it is not possible to determine which methods show the greatest potential for future results.

Having looked at the various intelligent algorithms alone, it is important to assess how they have been used in mobile robot experiments. In the following section these algorithms will be seen within the present context of research, as reviewed from the searched literature.

2.3 Review of Robot Designs

This section presents a critical review of the work which has been reported in the literature by researchers in the broadly definable field of autonomous mobile robot control. It is not always possible exactly to categorise many of the reported control architectures because of their hybrid nature. However, this section is subdivided to indicate the spectrum of projects from reactive to planning controllers.

2.3.1 Reactive based Control

Moorman and Ram [20] describe a parametrised schema based reactive control system called ACBARR (A Case BAsed Reactive Robotic system). This method is built upon a set of reactive sensor–motor actions or *schemas* whose parameters are contained in a set of interchangeable behaviours or *cases*. Each case is divided into three parts, general information, parameter ranges and applicability information. The cases are instigated dependent on the match between their applicability and the world state. A library of robust cases was built up by a human operator empirically observing the simulations.

The simulations demonstrated the effectiveness of the ACBARR system to get out of situations a purely reactive strategy could not (notably a boxed canyon) whilst still maintaining reactive advantages. However the simulations relied upon sensor information that was highly rich and possibly unrealisable for an in situ mobile robot.

It can be shown that a reactive controller will perform sequences of actions if the environment provides the correct sensor stimuli. Based on these ideas Nilsson [21] discusses Teleo–reactive programs. T-R sequences consider chains of states and actions that lead to a predetermined goal. For each state a control loop is built such that its action moves the system further down the chain towards the goal, hence teleo (distance) reactive. Inputs to the T-R modules can be taken from different sources, thus allowing different architectures. Nilsson provides no experimental data but indicates the potential mobile robot applications as these ideas are developed.

2.3.2 Neurally Inspired Robots

Franceschini *et al.* [22] have taken inspiration for neural connectivity from insects. They have successfully simulated and built a mobile robot that moves through a static environment and reacts to the relative motion. This passive control system is modelled on that of a housefly, which does not possess a visual cortex, indicating that most of the visuomotor control is achieved in parallel, and at a 'low level'.

The described robot possesses a compound eye which is loosely based on that of the insect, implemented in parallel and analogue electronic units. Arrays of units called elementary motion detectors (EMDs) detect motion of the visual flow field, which are connected to the panoramic compound eye located around the periphery of the circular frame. Two reactions are integrated into the robot, target attraction and object repulsion. However, the basic reactive control technique can cause the robot to become caught in local minima. Although this system will not work for a dynamically changing environment or when the robot is at rest, it demonstrates a fusion of sensory motor information for control in real time.

Heemskerk *et al.* [16] describes an approach to controlling a toy car in which behaviours, described as sequences of procedures, are used to train a neural network. Distal schemas (the externally observable behaviours) are used to create the proximal schemas (the embodiment of the control) such that by carrying out the proximal schemas, the distal schemas emerge.

Four light detectors are used as the input layer to the controlling neural network. The second layer uses lateral inhibition so that all the logical combinations of the inputs model to a specific active winner node. The connections between this layer and the two output motor control nodes are adjusted by the CALM learning rule (Categorisation And Learning Module). Under supervised learning the vehicle is directly controlled and the CALM algorithm performs Hebbian learning. Further work is being carried out to implement a reinforcement distal supervisor.

Methods using adaptive neural networks to condition actions are explored by Pfeifer and Verschure [15] by using the Distributed Adaptive Control (DAC) paradigm. Sensors are connected to an avoidance layer and an approach layer by variable connection weights. These are connected to a motor command layer by fixed weights to trigger predetermined responses. The avoidance and approach layers have an inhibitive connection so that obstacle avoidance remains a higher priority. The mobile robot uses a tactile bump sensor, a target sensor and a range finder. The network implements a form of on–line Hebbian learning that progressively associates the range finder data to the bump and target information and hence the predetermined responses.

Experiments were conducted in simulation and demonstrated the robot's ability to correlate the range finder data with the other sensors in such a way that a combined emergent behaviour forms from past experience. There is no distinction between learning and performance phases as the algorithm for updating the weights forces an equilibrium for stable environments which will change correspondingly to the environment.

Research has gone into the use of feed forward networks (FFNs) and simple recurrent networks (SRNs). Meeden *et al.* [11] uses a mobile robot in a small environment with a light source to evaluate the performance of these networks. All the control networks are trained by a modified version of the complementary reinforcement backpropagation algorithm (CRBA) where data can be incrementally built up during the training phase.

To maximise the number of comparative tests between networks, Meeden initially used a simulator with the percentage punishment as the performance index. Various methods were used in an attempt to increase the temporal abilities of the SRNs. These included the use of copying previous values of the hidden units and motors to the inputs, and training the network to attempt to predict subsequent sensor values. The most complex controller tested included auto-association of the input values to the output values.

Tests were carried out on two relatively simple tasks in a confined space. That of *avoid and move* with the goal of moving without collision and *light as food* where a light source indicated a food stimulus. The *avoid and move* experiments indicated that the sensory feedback from past motor states was more important in decreasing punishment than the tactile collision sensors or light sensors. Prediction and contextual memory were of benefit, whereas auto-association made no difference. The *light as food* experiments demonstrated that the robot, once trained to seek the food source, exhibited hierarchical behaviours.

Nagata *et al.* [10] sets out a control architecture comprising two feed forward neural networks. A 'reason' network was used to correlate input sensor signals to actuator outputs, and to be able to coordinate responses, a second, or 'instinct' network, incorporated a short term memory on its inputs taken from the reason network and some of the sensors.

The aim was to encode two different types of behaviour in two separate robots, those of seek and escape. Initially the neural networks were modelled on a computer and a range of stimuli and required outputs were fed to the networks. Using a method similar to back propagation the connection weights were changed. From this, the robot's behaviour was simulated and the useful stimuli-output conditions were used to update the network. In the next stage, the network simulator drove the actual robot so that its actions could be further tested. Finally the fixed connection weights were downloaded onto the mobile robots.

The experiment successfully demonstrated a neural networks potential to effectively control a real-time mobile robot. The network interpolates from the set of input and output relationships which were used to program it. However because of the fixed weight connections this particular use of a neural network cannot learn new information once it is in situ.

One attempt to embed a neural control structure in VLSI is outlined in [23]. A resistive grid is used for path planning, a nearest-neighbour classifier for localisation using range data, and a sensory motor network for obstacle avoidance. For path planning, a hexagonal resistive grid, shaped to the environment, is used to empirically compute optimal paths to goals so that wall collisions are avoided. The grid is implemented in VLSI and a gradient descent is performed between the start and goal positions where the voltage difference is applied. For localised positioning of the robot at a particular instant, a single layer neural network takes its inputs from a 360° infra-red range finder and outputs via a 'winner takes all' network to best match a node in the resistive grid. To avoid obstacles, an avoidance module takes relative velocity information and uses a low level network to directly control the motors. It is hoped to be able to teach the obstacle avoidance network to associate appropriate motor action with different sensory inputs.

To date the path planning and localisation module have not been implemented in VLSI but run in software on a SUN 4 workstation linked to the robot. Using this method they have reported real time control in static environments but comment that operation in dynamic environments could only be achieved with hardware implementation.

The use of custom designed neural VLSI has been investigated by Peacock and Bolouri [24]. Simulations of the VLSI stochastic network are run on a PC and used to control a mobile robot. The architecture used is a three layer feed forward neural network with off-line training. Infra-red, ultrasonic and tactile sensors have been interfaced so that it can avoid obstacles whilst being attracted to, or repelled from, a moving source. Moderate success has been reported.

COLUMBUS is a mobile robot with a global aim of maximising its knowledge of the environment [12] and follows on from his work in simulated environments [25]. It is equipped with a time of flight sonar system and two neural networks to interpret the data and limit the sensor noise. The sensor interpretation network takes previous sonar readings and interpolates to give a value of estimated reward for a given position. The confidence estimation network then returns the expected error in the result of the interpretation network. These results are processed to construct an exploration utility function which maps the environment. From this, gradient descent can be then used to determine exploration paths.

The approach requires an accurate determination of the position of the robot at any instant, and due to cumulative errors this cannot be accurately predicted. To overcome this, recent data was compared against that predicted by the global model using an error squared function. The local model could then be matched to the global one by reducing this cost function. This system could only operate in real time by using a remote link to several SUN workstations performing the computation.

Mitchell *et al.* [26] has taken a reinforcement learning policy and combined it with predictive neural network to reduce the number of examples required for such a network to learn. The evaluation function Q(s,a) measures the cumulative future expected reward, given the present state s and the next action to be taken a. Once this function is learned a robot can select such actions so as to maximise its reward. Mitchell's explanation based neural network (EBNN) uses a neural network as an action model to predict the next state given a previous state and action. With this differentiable network model, EBNN constructs *target slopes* by extracting the *derivative* of the final reward with respect to features of the state s. From this information, the evaluation function can be described with reduced sampling as the rest of the information about the environment was encoded in the action model.

Changes have been made to make EBNN more applicable to mobile robot applications. The values of the target slopes were derived from the approximate action model and therefore their accuracy can be computed by comparing the observed with the predicted values. This is then used to weight the learning of these slope values. To avoid *the problem of suboptimal action choices* Watson's Q-Learning has been employed. EBNN has been demonstrated to work in simulation and allows domain knowledge to be transferred between learning tasks.

A method for the coordination and integration of primitive reactive behaviours using a adaptive heuristic critic is described by Gachet *et al.* [13]. Six behaviour primitives are defined including such as goal attraction, perimeter following (contour left), perimeter following (contour right), etc., where each behaviour produces a time variant value of speed and direction. These six vectors each have an attributed coefficient which is controlled by a fusion supervisor module and the summed values are used to control the robot's wheels. They term this architecture and Adaptive Fusion of Reactive Behaviours (AFREB).

The fusion supervisor is a neural network adaptive heuristic critic (AHC). Filtered information from ultrasonic sensors is input to the network and the output values determine the coefficients. Through the use of a simulation the AHC was trained to produce emergent behaviours by combinations of the primitive behaviours. The robot in simulation and in empirical tests was shown to perform *surveillance*, *go to goal* and *follow a path* without collision in a constricted environment. They are attempting to improve the input's resolution using a Kohonen Neural Network.

The use of recurrent neural networks evolved using a genetic algorithm has been explored by Yamauchi and Beer for the learning of reactive and sequential behaviours [27]. The paper examines the application of continuous time Hopfield recurrent networks to three areas of sonar data recognition, one dimensional navigation, and sequence learning.

Real sonar data from a full scan around the perimeter of two different objects was used to test the networks. Networks were then evolved using eight fully interconnected neurons.

After 15 generations a network was found to work in simulation and could on a real mobile robot correctly categorise in 17 out of 20 trials. The navigation exercise placed an agent in a one dimensional 'world' containing a landmark and a goal which were not close enough to be sensed at the same time. The agent had to detect whether the landmark was between it and the goal, or on the other side and then use this information to move to the goal. This was achieved by evolving three submodules. One module categorises the environment and dependent on that information it triggers one of the other two modules to seek the goal. They have also demonstrated the abilities of recurrent networks to learn sequences of bit streams from reinforcement signal. Although they report that three bit sequences have been learned, their genetic algorithms search space increases exponentially with the number of neurons.

Although the use of genetic algorithms for evolving a successful network can be achieved, it is limited to off–line learning. The applicability of GAs to real time learning is not evident nor discussed.

2.3.3 Robots based on Reinforcement Learning

Mitchell, Keating and Kambhampati at the University of Reading have experimented with simple 'insect' robots [28]. These mobile robots possess simple ultrasonic sensors and low computational power and are designed to act in reactive fashion to the environment. They describe a learning strategy based on a fuzzy automaton algorithm. They build up five automata based on classes of ultrasonic input i.e. obstacle close to left sensor, obstacle far from left sensor etc. Each automata builds up a probability distribution across the set of actions based on the reinforcement from the external critic.

A simulation was initially used to experiment with the automata's learning parameters and aspects of reinforcement. Many simulations were run initialising the automatas with random probability values. The robot was shown to learn to move around its environment without collision when all five automata were used. however, as expected, with less than five automatas, the system cannot converge to the desired behaviour.

Ashwin Ram and Juan Santamaria [29] have designed a reactive schema architecture whose parameters are continually refined by a learning and adaptation module and is called SINS (Self-Improving Navigation System). The motor schemas used were *avoid static obstacle*, *move to goal* and *noise* (random motion). The sensor data is converted to four vectors that characterise the environment and then associations between these cases and output parameter values are made. Using a reinforcement method the number of cases are built up and used as a library for adapting future parameter values.

The empirical comparisons were all made by computer simulation and demonstrated SINS improvement over fixed and random schema systems. Experiments were carried out to find optimal values for the learning module's parameters. It was also shown that the architecture was relatively robust to environmental changes.

Kaelbling [30] describes the use of a reinforcement algorithm called *interval estimation.* This technique produces an *action map* which maps all internal and sensor inputs to an action. The experiments were entirely simulated and the full input information was only five bits with only three actions to be mapped to. The Interval estimation records the number of times an sensor action has occurred and the number of times a positive reinforcement has exceeded a predetermined threshold.

The experiments qualitatively validated the learning algorithm. However, this method cannot cope with delayed reinforcement and must record every possible input state, which is unrealistic for more complex systems. They point to Q learning and other approaches to overcome these difficulties.

Mahadevan and Connell [31] investigate Q learning, a temporal reinforcement strategy, combined with two different methods of propagating reinforcement across states. The two methods used were that of statistical clustering and Hamming weight. The tests are primarily carried out in a behaviour based architecture as pioneered by Brooks [6] but comparisons with monolithic architectures are shown. The results are presented for both simulations and real robots where the task chosen was that of pushing boxes in a playpen. This problem requires that an autonomous robot locate boxes in an irregular shaped room and then push the boxes to the walls.

The robot described had ultrasonic and infra-red sensors. Eight ultrasonic sensors faced radially from the circular frame, each having a two bit resolution of 'near' and 'far'. Although this reduces informational input (reduced precision), the abstraction process reduces the error in the resultant data (increased certainty). The infra-red detector acted as a forward bump detector and a motor current monitor indicated if the robot was in some way stuck. The Hamming algorithm requires that all possible states are stored hence a reduced input resolution was chosen so that the state space could be diminished. This was overcome by the statistical technique where the robot learns a set a clusters, each with an associated Q value, for each action.

The results suggest that individual behaviours can be effectively learned, sometimes outperforming the hand coded agents. They also demonstrate that by breaking up the problem into hierarchical layers the reinforcement tasks are simplified, thus improving performance. Overall it was shown that reinforcement algorithms work substantially better in a subsumptive architecture and that the clustering improved over Hamming as a structural credit assignor.

Dorigo and Colombetti [32] describe the design and interaction of a modularised architecture based on reactive control. They use reinforcement learning to translate into agent actions, a stipulated demand. Each module or *classifier system* is composed of a performance system, an apportionment of credit system and a rule based discovery algorithm. The performance system includes a set of *classifiers* or sensor–action pairs with an associated credit indicated by the apportionment of credit system. To introduce new classifiers into the performance system a genetic algorithm selectively recombines and replaces low credited ones. The classifier system modules were combined into various architectures including monolithic, flat and multilevel hierarchical.

The results have shown that complex interactions with the environment can be achieved, including certain type of sequential behaviour where enough sensor stimulus has been perceived. It was also shown that with 'robot shaping' the trainer could remain relatively abstract. Adaptation to an environment was shown to be aided by the genetic algorithm and speeded up by correct choices of architecture.

Dorigo and Colombetti elaborate on their proposed ideas in [33] to include discussion and experimentation on the sequencing of behaviours. Reactive controllers can only perform sequences of actions if the sensory stimuli lead the behaviour, otherwise the agent must utilise some form of internal state. These are explored and implemented as a memory bit accessed by a coordinating classifier system.

The task set for the robot was to move backwards and forwards between two points. This environment does not directly indicate which point is the present goal, so the robot must develop self initiated sequences to cope with this. The simulated results show that non reactive behaviour sequences could be learned but highlighted complications in determining reward values for actions that could have more than one interpretation. To overcome this, previous reinforcement values were stored in an attempt to indicate to the system which goal was being sought.

2.3.4 Genetic Based Controllers

Koza investigates a genetic evolved architecture [34] and compares it against a hand-coded subsumption architecture reported elsewhere. The paper focuses on a wall following behaviour to be implemented in a software simulation. A set of basic *functions* that can be linked to *terminals* (sensors, actuator or other functions) is laid down for the genetic algorithm to work with. Squares are placed around the environment's perimeter and the fitness indicator is determined by the number of these intercepted in a run. It was this fitness function that caused the development of the structure. The algorithm also involves reproduction crossover of chains of functions and terminators.

The best of the generations program that performed the wall following behaviour, when inspected, could be seen to be similar in action to the subsumption architecture against which the algorithm was being tested. Although this method produced a controller of the same ability as a hand coded one, the evolutionary stage could not be directly implemented on a real robot.

In Koza's book Genetic Programming [35] he again takes a problem reported by other researchers and uses genetic methods to solve it. The undertaken task is that of box pushing and was reported by Srihar Mahadevan [31] (reviewed above). The fitness function being the sum of the distance between the box and the wall summed over the test time. Koza reports the success of the GA to find a solution which matches that of Mahadevan, but then proceeds to compare both methods based on false assumptions. He assumes that during optimisation of the control algorithm there is access to accurate and complete world data and that an unlimited number of trials can be performed. Both these are unrealistic and Mahadevan does not assume them.

2.3.5 Fuzzy Logic Systems

Saffiotti describes the use of fuzzy logic to coordinate the responses of a set of behaviours [36]. Sensory information is processed and mapped into the *local perceptual space* centred upon the mobile robot. Also, indicators called *artefacts* created by higher level planning are added to provide global information for the behaviours. These artefacts allow strategic (e.g. sequential) goals to be specified which will be acted upon in a reactive manner. Each behaviour's responses are then multiplied by an activation level dependent on its contextual applicability, and this eliminates potential local minima associated with reactive control.

Simulations and empirical techniques were used to evaluate the system. They have shown the ability of the robot (called Flakey) to move through previously undefined environments avoiding obstacles. Flakey came second in the first AAAI robot competition in San Jose.

2.3.6 Behaviour Based Control Paradigm

Brooks's original work is outlined in [6] where he uses the ideas of subsumption architecture for mobile robots. The physical robot used had a ring of twelve Polaroid sensors and two Sony CCD cameras. The three control levels used were avoid obstacles, random motion, and a directed exploratory mode. The results indicate that this reactive control system works, but also reveals problems of local minima.

Eustace, Barnes and Gray at the University of Salford (UK) have used a behavioural based control system to produce co-operation between physically separate mobile robots [37]. Four strategy levels are used in total and these are SELF, ENVIRONMENT, SPECIES and UNIVERSAL. The SELF and ENVIRONMENT levels search out battery charging locations and avoid obstacles respectively. The SPECIES level coordinates interaction between individual robots. The UNIVERSAL level is task oriented. Each level takes in sensor data weighted according to level relevance and outputs a motion vector which used to compute the total response.

The task is for the robots to move a pallet supported from beneath by two robots at either end. The robots use force and torque sensors connected on the capture head, which holds up the pallet, to interact in the task. Through simulation and real demonstrations this system has been shown to avoid obstacles without dropping the pallet. The main consideration reported is that of optimising the values of importance assigned to each level.

2.4 Summary

One of the first observations that must be made concerning the research field of mobile robots is that there is a large and diverse number of applications. There is little consensus on any benchmark problems, which in turn leads to a correspondingly diverse set of robot experiments. Although this makes any kind of quantitative comparisons between projects nearly impossible, a qualitative assessment can be made.

It is not unsurprising that there is such diversity. Often the higher level control aspects come from ideas born from Artificial Intelligence, itself a relatively new discipline. Many of the sensory systems that can be used on mobile robots are also in development and arguably, the computational requirements for these projects are only just becoming realisable. In such a state of flux and with each advance, new possibilities open up, leading to a discontinuous research effort.

It is possible to see from the literature one aspect which has not been fully addressed and which must form the basis for further research. This is the problem of map building and navigation. In order for mobile robots to be able to demonstrate intelligent behaviour they must first be able to understand their own spatial constraints. An area of particular and active research interest is that of a robot exploring an unknown environment. Such a robot must be able to integrate newly acquired knowledge into previously stored maps, whilst coping with the inherent noise.

Columbus, designed by Sebastian Thrun, attempts to satisfy these criteria. It uses only time of flight sonar readings and builds up a representation of its environment. A minimum of prior knowledge is used with the robot's main aim of efficient exploration. Effective exploration is achieved by the robot storing predictive reward and expected error from the sensor data, thus allowing it to judge which areas are less well explored. The primary focus of the research has been on the development of a working physical robot which demonstrates that relevant problems have been tackled. However, it is unclear as to whether uncertainty between the positions of locations could cause problems after long periods of exploration.

The work carried out on the SRI robot Flakey has demonstrated that it can interact with the real world. Although the robot does not construct as detailed information about its immediate environment as Columbus, it able to navigate through it to non-immediate goals. However, Flakey demonstrates a method of smoothly integrating higher level plans to a local navigational strategy. This continues to the idea of defining objectives in a more abstract and controllable fashion.

The above chapter has examined the large area of research which is currently being undertaken in the research field of autonomous mobile robots. From a detailed assessment of the literature, the problem of map building and navigation has been targeted for active consideration, and will be examined in more detail in the following chapter.

Chapter 3

The Design of a Mobile Robot Control Structure in Simulation

3.1 Introduction

To be able to further develop the ideas of a map building mobile robot, the design methodology must be examined. Although an eventual realisation of the design of a working robot is required, this is not the only test bed with potential for experimentation. With the current range of available research computers, it is possible and even advantageous to simulate the full mobile robot design prior to a prototype being constructed.

One of the major benefits that simulation offers, is the ease of accessibility of all the parameters of the robot's control system during the test runs. Not only does this allow swift elimination of errors in the system, but speeds up the development time of the control algorithms during the initial design phase. The complexity of both the control system and the environment models can be incrementally increased and this in conjunction with the relatively short run times leads to a quick and fluid development process.

A major criticism of simulation work is that it does not offer realistic experimental conditions, and therefore the results obtained have little absolute value. This criticism holds only when the physical dynamics of the system have not been properly considered and poor system models are used. Hence, if good models which more accurately mimic reality have been implemented, it is certainly possible to conduct important comparative tests between control systems. However, at this stage, it is not possible for the many interactions between the robot and its environment to be *fully* modelled. In order to fully validate the control systems in absolute and independently comparable conditions the controller would have to be ported to a real robot. However, in the earlier stages of development the use of simulation work is highly important.

Although the final aim is to develop a working mobile robot, this is not necessarily the best way to approach the initial design of the control system. It takes considerable time to develop a working prototype, giving rise to long delays before any control aspects can be considered. Also, even though it is possible for the robot environment to be simplified to aid initial development, there is a basic level of competence which any control system must possess. As with all physical processes, tests conducted with a real robot take time to conduct, including the downloading and uploading of run time information, which can lead to lengthy basic tests. For these reasons, it was considered important to design and test the initial control systems in software before being ported to a real mobile robot platform.

3.2 The Software Simulation Strategy

The software was developed within a UNIX environment because of the combined advantages of processing power and display development software. This development software to produce a graphical user interface, which is required for displaying the robot's information, was based on the X11 windowing system. This is based on a client server protocol, allowing the software to operate over a network upon available machines. It is also possible to run this over a network on a PC equipped with suitable software. With the use of a suitable tool–kit (XView) the simulation package could be developed under the standard C programming language.

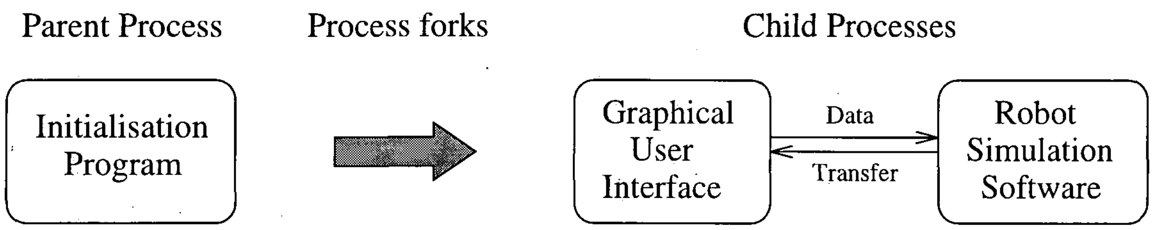


Figure 3.1: Initial set–up of the main simulation software

As with any UNIX environment it is possible to have more than one process, or program running simultaneously and this can be achieved by the *fork* function. When the program is initially executed it splits, or forks, into two separate processes, one of which is used for the robot simulations, and the other for the graphical user interface (Figure 3.1). These programs are linked by a data channel, or pipe, which allows the programs to transfer information. The graphical user interface handles all the simulation parameters and displays all the results, as required by the user. Whereas the simulation software mimics the actions of a mobile robot, and returns data and any requested information.

The use of the fork command to split the programs up into separate entities was primarily used to increase software reusability. It is often important to analyse the streams of data being produced by the robot simulation software and this is possible by simply replacing the graphical user interface with a suitable program. The information flow of the data processing

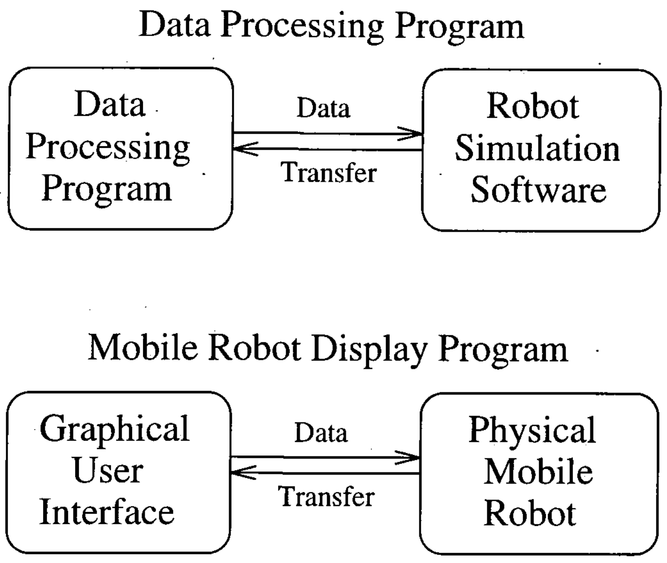


Figure 3.2: Software reuse

program is shown at the top of Figure 3.2. Such a program can run the simulations many times in order to assess the effects of changing specific parameters.

The lower half of the Figure 3.2 shows how the graphical user interface can be reused for displaying the information from the physical mobile robot. This both aids the development process and allows interpretation of the robot's actions based upon its informational state. Hence the graphical user interface offers a tool for examining a real robot in greater detail.

3.3 The Graphical User Interface

A graphical user interface was chosen for displaying the information produced in the simulation run by the robot. This form of display has the advantage of being able to show large amounts of positional data in a clear and concise manner. A basic requirement is that the software be able to show the environment, and the robot's motions within it. This main program window from which all the other functionality is called, was called the mobile robot simulator, and is described in more detail below.

Figure 3.3 shows the main simulation window. It comprises four selection buttons mounted above the main display window, with the simulation run time and a path clear button beneath. The environment is defined in terms of an occupancy grid with each element either an object or a space. The resolution of this environment is 64 by 64 blocks, however this changes dependent on the environment loaded in. The path of the robot can also be seen as a meandering line in the free space. In this example the robot starts off its path from the middle left hand side of the maze. This simulation has been stopped at an arbitrary time to show a typical robot path.

A design feature is that the robot is shown as a point source which can move to any point in the free space. This can be considered as a representation of the real environment where the walls have been enlarged to account for the radial size of the robot. Such a representation has the advantage that all connected free space regions are traversable without reference to the width of the robot. Also when the traversable maps are shown, any mapping errors can be easily seen as they cross an object region.

Three of the four buttons at the top of the display access menus. The file button has an associated menu from which maps can be saved and loaded to the screen, also new environments can be loaded. The robot button calls up a range of parameters associated with the robot which can be easily changed. The simulator button allows the simulations to be stopped, started or reset. From here other simulation parameters may be accessed, including the simulation run time and environmental size. The last button is for the display menu and is used to open the various data display methods, which are described in more detail below.

The main simulation display window is called up from the main display menu is and shown in Figure 3.4. This is used to display the the robot's stored information in as clearer form as possible. An example of a mapping topology is shown to illustrate the type of information that needs to be displayed. Here, the position of the map can be shown with relation to the environment, with only the environment and the topological links displayed.

The buttons at the bottom are present in the later versions of the simulation software which are used in Chapter 5 to allow access to the larger quantities of data stored in the mapping structure. The first three buttons toggle on and off the environmental objects, topological links, and the nodal range data, for easier visual inspection. The vectors menu allows associated vector information about nodes to be displayed.

It is often useful to examine properties of the networks during the development. If this is to be achieved whilst indicating the relative positions, some form of three dimensional display technique must be used. As the development of such software is time consuming, an independently written software program was used. By accessing the main display menu button, the data is written out in a format that this Xss program can display, and the program is started.

Figure 3.5 shows the image that is produced. The environment is shown all at one level in the x–y plane with the scalar of each node being indicated by its height in the z plane. The whole image can be rotated and translated to permit viewing at all angles. This method is useful for determining which network parameters most accurately reflect network properties.

During the testing of the simulated mobile robot many parameters need to be changed at the start of each new simulation, and to avoid having to recompile the program each time, data entry panels were designed. Figure 3.6 shows the simulation parameters entry panel.

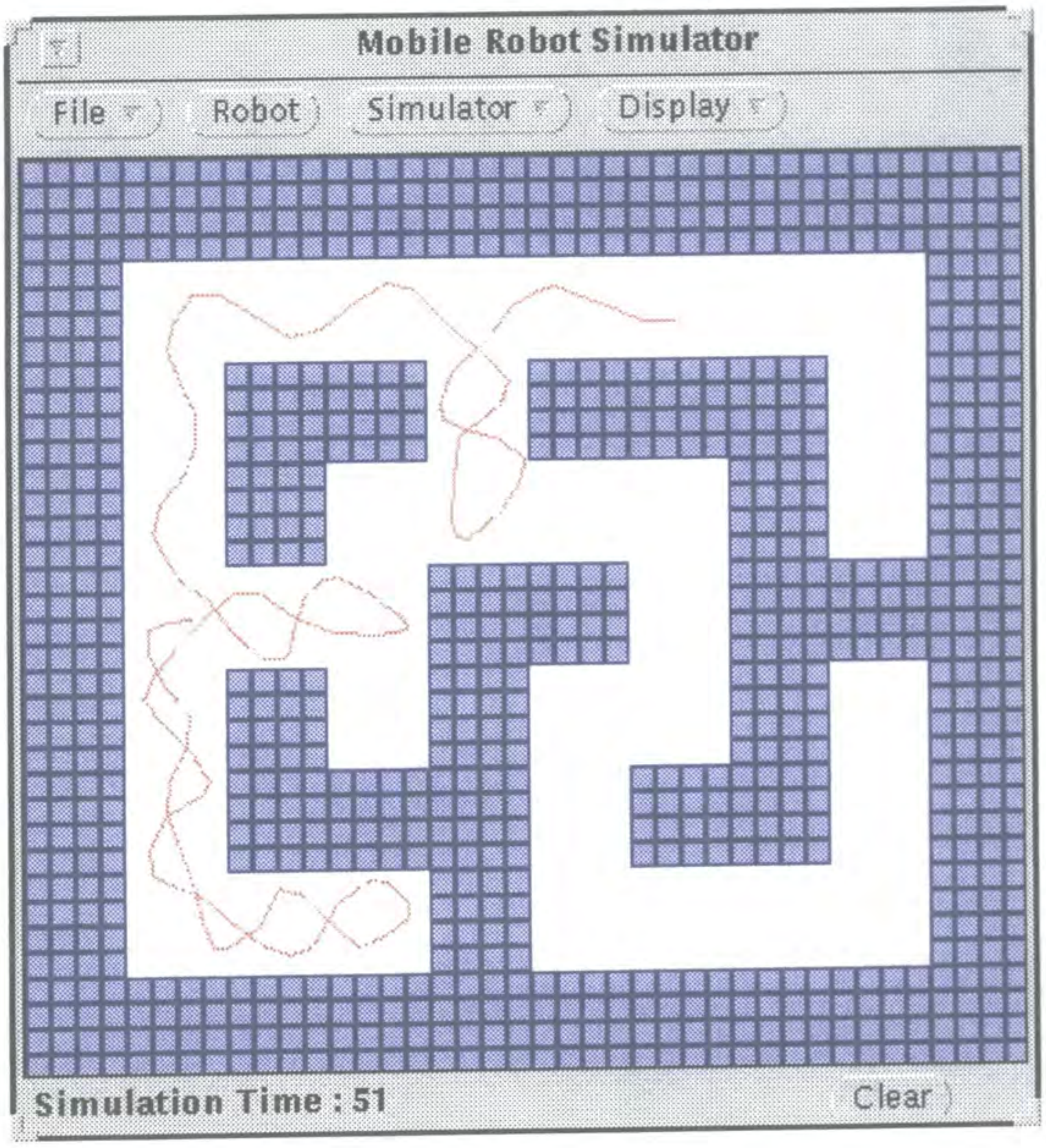


Figure 3.3: The mobile robot simulation window

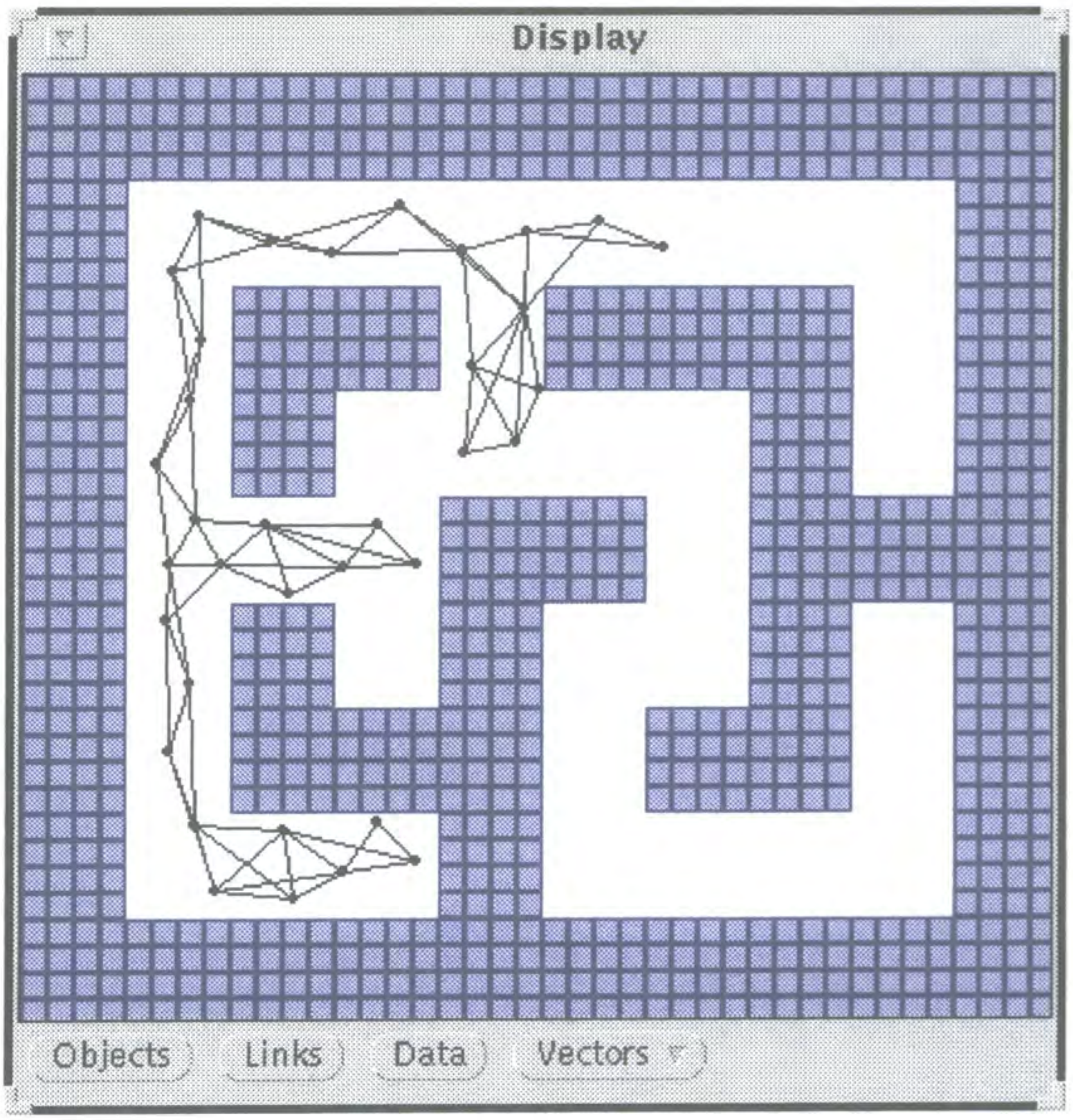


Figure 3.4: The data display window

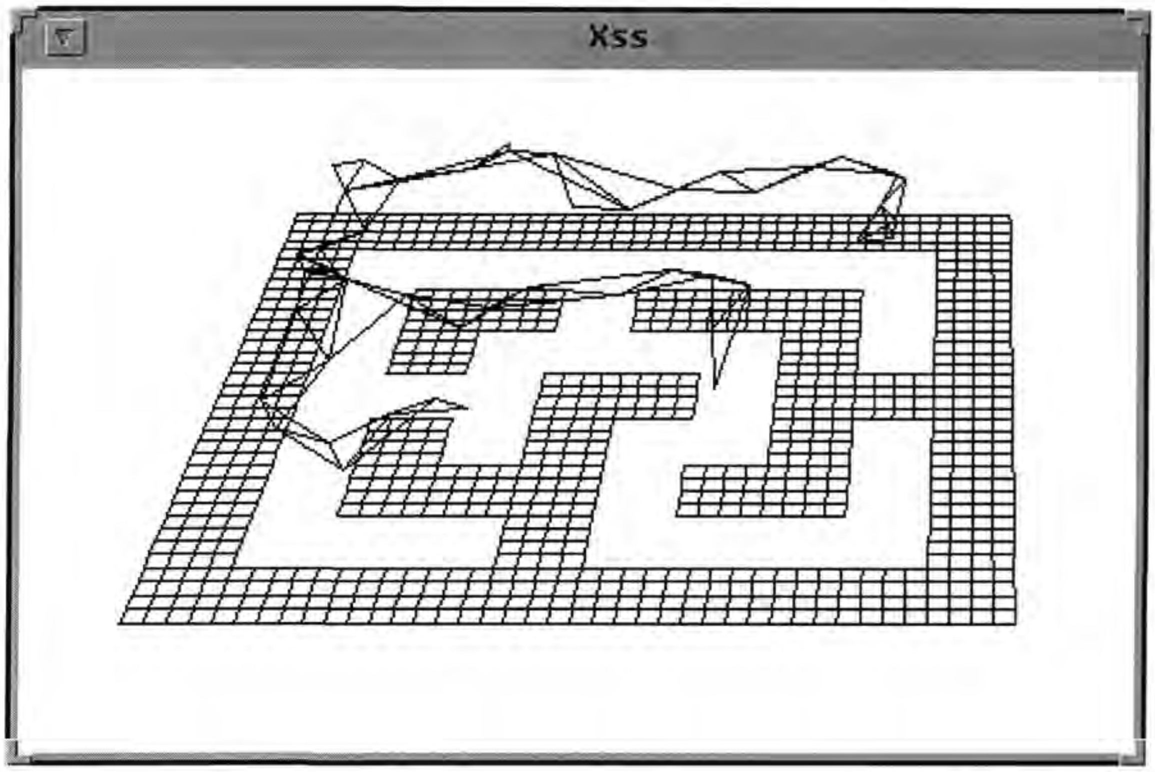


Figure 3.5: Three dimensional network display

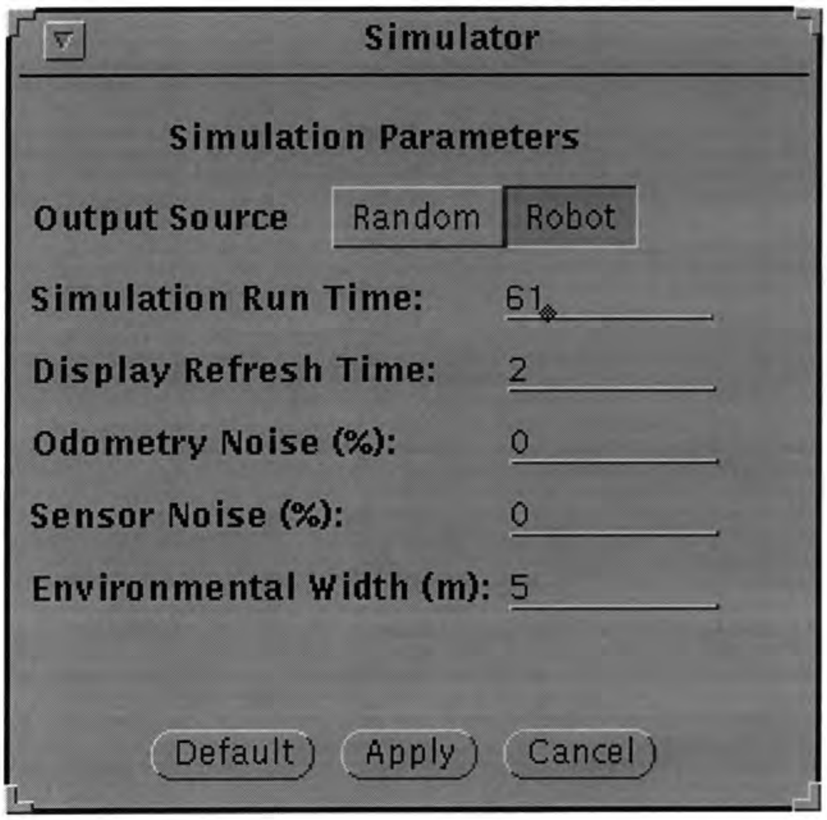
Examples of different forms of data entry can be seen here. The first is a choice of simulation output to be fed to the robot controller; random environmental sampling, or that produced by the robot. The other scalar parameters can also be easily changed. There is a similar data entry panel for the robot's control parameters, which is not shown.

The above describes the functionality of the graphical user interface, which is one element of the simulator. Described below are the methods used to simulate the actions and interactions of the robot with its environment.

3.4 The Robot Simulation Software

The robot simulation software can be described in terms of three main elements. These are the robot controller, the robot chassis, and the environment which the robot is situated within. The interactions of these components can be seen from figure 3.7. As with a real mobile robot the control algorithms would be executed in software but here, all the physical systems must be simulated in software.

The robot controller is the control architecture of the robot, and takes values from the sensors and returns an actuator demand. In order to more accurately mimic real systems the



Simulator

Simulation Parameters

Output Source: Random [✓ Robot]

Simulation Run Time: 61

Display Refresh Time: 2

Odometry Noise (%): 0

Sensor Noise (%): 0

Environmental Width (m): 5

Figure 3.6: Simulation parameters entry panel

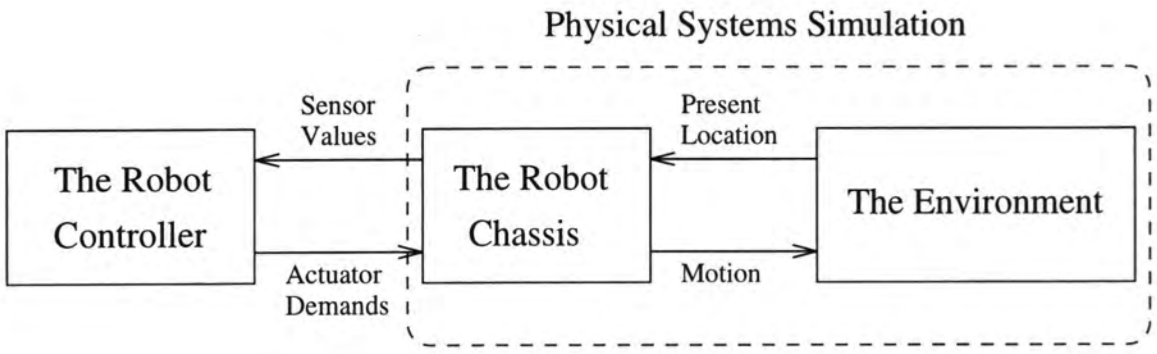


Figure 3.7: The main simulation elements

sensor values are periodically sampled whilst the actuator demands are updated. The rate at which this can be performed can be changed in the simulations, but was initially selected to be at half second intervals. The control architecture is described in greater detail in 3.6.1.

The rest of the robot simulation software attempts to reproduce the responses which a control architecture would receive if mounted on a real robot. To achieve this both the robot chassis, and the robot's interaction with its environment must be modelled. Therefore, the general aspects of the robot's mechanical design must be determined so that it may be modelled to an acceptable level of accuracy. Similarly the type of sensory systems must be defined before software models can be constructed and the environment with which the simulated robot interacts through its mechanics and sensors must also be defined. These methods by which these systems have been constructed are more clearly stated in the next section.

3.5 The Physical Systems Simulation

Before the physical systems can be simulated, the general properties of the robot mechanics have to be defined. The entire mechanical structures need not be completely defined at this stage, just the general modes of interaction of the sensors and the drive mechanisms.

3.5.1 The Robot's Sensor Systems

The sensory systems that were selected, as might be expected, are task dependent. As the defined task is that of map building and navigation the sensors that are most applicable are concerned with the measurement of spatial distance, and of distance travelled.

For the measurement of open distances, the *time of flight* (TOF) of a short pulse of energy from the sensor to target and back again is the most commonly used method. Although pulses of light can be used to gain fast data with a high angular resolution, the high speed electronics make it prohibitively expensive. Cheaper and more widely available ultrasonic devices are available which offer realistic range measurement systems.

Odometry or dead reckoning are terms used to describe the measurement of distance travelled by a wheeled vehicle. This is achieved by the integration of all the robot's movements as monitored by its wheel encoders. This type of sensor system is cost effective and widely used.

3.5.2 The Robot's Drive Mechanism

Before the drive mechanisms can be modelled, the best drive mechanism for this application must be selected. There are a range of possible configurations which have been used in mobile robotics application, however the most suitable is problem specific. Important factors are manoeuvrability, design complexity, and wheel slippage, causing errors in the odometric systems.

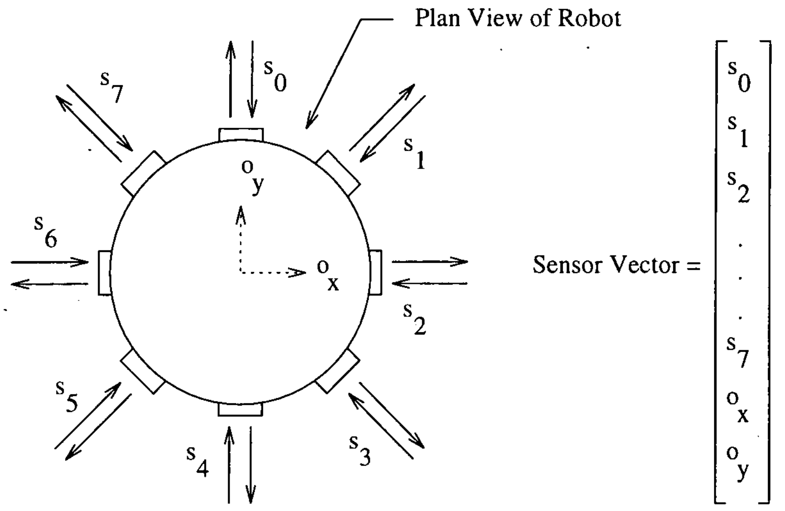
The most common form of mobile vehicle is of course the car. This drive configuration is termed Ackerman steering where the front wheels must point at certain angles in a turn to avoid wheel slip. As with the tricycle drive system where only one wheel at the front is used for steering, both these systems cannot turn on the spot. With this inability, such systems would require control solutions outside the intended scope of this work and hence neither were used.

Differential drive systems have two wheels mounted on either side of a robot platform with one or two castors for stability. The advantages of such a simple design are the ease of construction and relative low weight. The main disadvantage for this application is that the platform changes its orientation dependent on its direction of travel. This would increase the complexity of the sensory systems which would be required for correction. Also the drive wheels can be subject to wheel slippage, leading to errors in the monitoring of the robot's position. Tracked vehicles, such as those used by military tanks, are essentially similar. Although they perform well over a range of surfaces, they rely on *skid steering* which inherently leads to large errors in the monitoring of the robot's position.

The synchro–drive configuration is based upon the idea that all the wheels can be synchronously pointed in any angular direction and also rotated to provide propulsion. As only the wheels are angularly rotated, the main body of the robot stays at a fixed angle. Therefore the angle of the sensory systems mounted on the robot are independent of the robot's direction of travel. During the robot's motion, because all the wheels apply the same force there is reduced slippage and improved dead reckoning accuracy. Although this system is more mechanically complex, it provides many advantages which lead to its selection. The most preferable plan view shape of the robot is circular, as its mobility is constant, irrespective of orientation.

3.5.3 The Physical Systems and their Simulation Models

The preliminary physical design is of a circular plan view robot with a number of TOF distance measures equidistantly mounted around its periphery. The number of these sensors is a compromise between full angular coverage against firing rate, as each sensor requires a certain interval to be able to detect its own ultrasonic pulse. To attain complete angular coverage requires 24 sensors but this would require too much time for a full sample, therefore initially only 8 sensors were used. Figure 3.8 shows the concatenated sensor vector that is regularly sampled by the robot controller. The simulation model must produce a vector of the eight range distances $(S_0, \ldots, S_7)$ combined with the odometric position estimate (O_x, O_y).



Plan View of Robot
s_0
s_1
s_2
s_3
s_4
s_5
s_6
s_7
o_y
o_x
Sensor Vector = $\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ \cdot \\ \cdot \\ \cdot \\ s_7 \\ o_x \\ o_y \end{bmatrix}$

Figure 3.8: The robot's sensor vector

In order to model the motor drive system two components were considered. Firstly it was assumed that a demand angle could be issued to orientate the wheels and that this motion would be instantaneous. Even though this is unrealistic, the control algorithms do not make rapid turning movements, and it was therefore acceptable.

The forward propulsion was expected to be developed by a small DC motor which could be easily modelled as a voltage controlled device. Assuming there was no slip between the wheels and the ground, the total force (F) applied by the robot on the floor could be determined as shown in equation 3.1 as derived in appendix A. This force changes dependent on the supplied voltage V_{dc} and the forward velocity of the robot (u). The constants are the gear train reduction (G), a constant that includes the number of poles and the number of turns in each winding (K), the flux per pole (Φ), the armature resistance (R_a) and the robot's wheel radius (r). These constants can be calculated from the motor's specifications along with the preliminary gear train assembly.

$$F = \frac{GK\Phi}{rR_a}\left(V_{dc} - \frac{GK\Phi}{r}u\right) \tag{3.1}$$

A motor with an integral gearbox was chosen so that with a wheel radius of 5 cm and a 1:1 gear train the maximum velocity of the simulated robot would be just above 20 cm/sec. This velocity would correspond to the full 12 volts being applied to the motor (V_{dc}). A drive motor power factor can be defined as the applied terminal voltage (V_{dc}) divided by 12. Therefore the actuator demands from the robot controller are simply an angle and drive motor power factor.

To calculate all the values to be used in the equation 3.1 the specifications of the motor combined with gearbox were taken from the manufacture's data. It is 12 volt motor with a maximum continuous operating current of 0.493 amps producing 600 mNm of torque. It has an integral gearbox of 100:1 with an angular velocity of 40 r.p.m. at 12 volts. Its terminal resistance is stated at 10 ohms. The motor has permanent magnets and therefore $K\Phi$ is constant and can be deduced from the motor's standard operating conditions. The geared down torque T_g is related to the rotors torque T by: $T_g = TG$. Where T can be described: $T = K\Phi I_a$. Hence substituting 0.493 amps for I_a and 0.6 for T_g into equation 3.2 we get $K\Phi = 0.01217$. Hence the total force applied by the robot on the floor can be re–written as in equation 3.3.

$$K\Phi = \frac{T_g}{GI_a} \tag{3.2}$$

$$F = 2.4341V_{dc} - 59.247u \tag{3.3}$$

Knowing the direction in which the wheels are pointing and also the force that is being applied, the dynamics of motion are computed using Newton's Second Law ($F = ma$), where the mass (m) was assumed to be 3 Kg. From the acceleration (a), the velocity and position of the robot can be computed at any instance. In the simulation, if the robot collides with any objects its velocity is zeroed.

The environment is stored as a two dimensional occupancy grid within another file, so that alterations can be easily made. This information is used along with the robot's position to determine the values of the ultrasonic distance measures, at the regular sampling times. If required, white noise can be added to the sensor distance values to attain a greater model realism. The simulation software possesses the exact position of the robot within the environment, however, this is not used for supplying the odometric position to the robot controller. Instead there are a separate position estimate which is constantly updated and can be made subject to accumulative errors if required.

3.6 The Robot Controller

Described above are the physical systems and their simulation models, by which the robot controller can sample a value of the sensors at constant intervals and update a demand action to be undertaken by the robot in the next time step. The general approach to the control architecture and the various forms of mapping are described in more detail below.

3.6.1 The Controller Architecture

Before examining the individual components which are required to build up the control system, the overall control strategy must be more clearly defined. This framework must

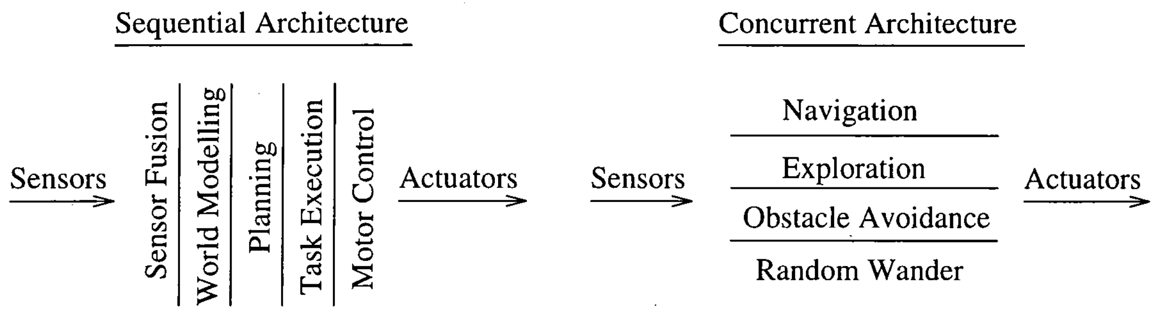


Figure 3.9: Sequential and concurrent architectures

provide simple solutions which are computationally easy to implement, as many of the early mobile robots had large computational burdens and suffered slow performance. The system selected must be flexible enough so that different algorithms can be implemented and tested against each other. This would also allow any algorithm to be incrementally build up in sophistication, without having to design a new interface with the sensors and actuators.

Such a system was first described by Brooks [38] who called his control framework a subsumption architecture. He recognised that the robots which were able to perform any useful tasks at that time were designed to be based on the ideas of the Artificial Intelligence community. These control systems take the sensory input and perform sequential operations on it. The main drawback of this approach is the large amount of computational effort that is required for each motor control output. All the sensor information must be fused together and used in conjunction with the robot's world model before any response can be determined. Therefore, these systems can be slow in their actions. However, other robot experiments suggested that a faster system response could be obtained by having concurrently operating systems that produce reactive reflexes.

Brooks combined these different methods by attempting to decompose the tasks that the robot performs so that they could be run concurrently. This means that each of the concurrently running layers, or behaviours, cannot be held up in its processing by the other systems. Not only does this allow fast reflex responses by lower levels but allows for the complexity of the system to be incrementally added. Figure 3.9 shows the architectures of both a sequential and a concurrent control system.

The important aspect of the concurrent design is the way in which the actions that are produced by each of the layers are combined into one unified control output. Brooks's solution to this problem was to prioritise the importance of each of the layers, then the actions of the higher layers would simply override (or subsume and hence the name) the responses of the lower layers. In essence, the problems of sensor fusion have been dispensed with at the cost of requiring some form of actuator output fusion.

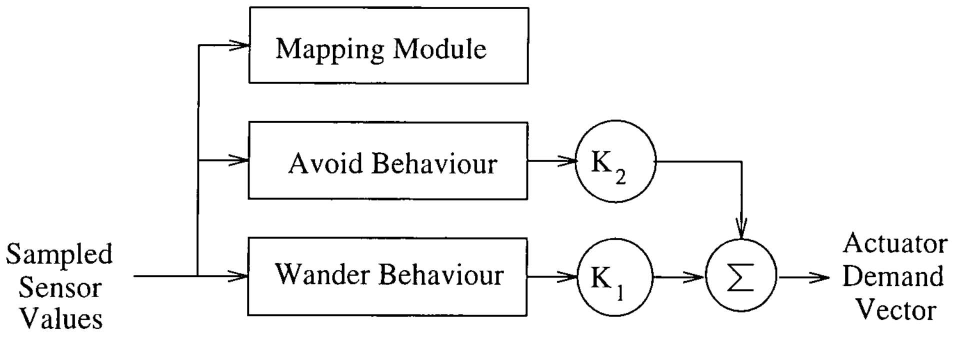


Figure 3.10: The control structure

It is however, not obvious how subsumption could scale up directly to incorporate more complex layers. In certain circumstances it might be to the robot's disadvantage for the higher layers to override some of the reflex actions, and in such a case the upper layer would either have to simulate the reflex, or take information from the lower levels. This would cause unnecessary computation or the invalidation of the distinction between the layers, in the above cases respectively. There are some problems with the system when it is scaled up, but it does offer a system that allows fast responses to control inputs and which can be incrementally built up in complexity. This is why it was chosen for the control architecture. Below the design chosen for the simulations is described in more detail.

Figure 3.10 shows the controller architecture. The sampled sensor values form the inputs to the three modules. The avoid and the wander behaviours produce vectors which are used to form an actuator demand vector. The mapping module in the figure is passive and can only record information from the sensors. Although, it might be useful to get feedback from the mapped data to aid in the directing of the robot, in the initial stages of testing this was avoided. This allowed more accurate assessment of different control structures on the same repeatable paths, because when feedback is used the paths are no longer comparable. In the more advanced work, which is reported in Chapter 5, the network layer takes a non passive role and more directly controls the robot's output.

The wander behaviour produces a vector output which attempts to direct the robot away from its present path. This is achieved by monitoring the angular direction of travel of the robot, and then outputting a vector which is an angular deviation away from this. The size of the deviation is random, but is within a specified tolerance. It was found empirically that too large a value of this tolerance caused the robot to move slowly and erratically throughout the environment. Whereas too small a value forced the robot's path to be near straight. A tolerance of 36 degrees in either direction gave good results.

The avoid layer produces an output vector based on the ultrasonic range information which attempts to drive the robot into areas of perceived open space. For each of the range sensors, a repulsive vector is calculated, and then all the vectors are then added together. The vector is inversely proportional to the distance recorded by the sensor, and its angle is diametrically opposite to the direction which the sensor faces. Therefore when the robot is close to objects, large repulsive forces are created which direct the robot away.

The vectors are multiplied be gains K_1 and K_2 for the wander and avoid behaviours respectively. This allows the effects of both behaviours to be balanced to improve the robot explorative path. The vectors can then be simply added together to produce a combined control response.

3.6.2 The Mapping Module

The fundamental aim of the work is to design a robot which moves around unexplored areas building up maps for navigational purposes. In order to achieve this the robot must be able to know where it is, where it wants to go, and finally work out some path to this destination. Such a robot requires the ability to estimate its own position with respect to the information that it has already gathered. Borenstein [39] categorises the problem into two sets of partial solutions, that of *relative* and *absolute* positioning methods.

Odometry is an example of a relative positioning method which provides distance information from the wheel encoders. This system is self–contained and provides good short term data, however the error constantly increases unless an independent reference is used.

An absolute position measurement system produces information not relative to its own state, but to that of another system. The more simple methods require the environment to be modified, however the more challenging problems arise when no changes are allowed.

The methods of *Active Beacons* and *Artificial Landmark Recognition* both use a modified environment. In the first method, three or more transmitters are used for triangulation by the robot. These can be light or radio sources placed at known locations. When passive indicators are used instead, the problem becomes that of artificial landmark recognition. Examples of landmarks are bar–codes, but other objects can be used.

Natural landmark Recognition triangulates from distinctive features. The environment is not changed, but the positions of these landmarks must be known in advance. In *Model Matching* the current sensor information is compared to a world model, or map. If a match can be found, then a positional estimate can be made. The map matching domain covers the current research areas of updating previous maps in dynamic environments and the development of new maps to cover unexplored areas.

Of the absolute positioning methods, only the map matching technique can be employed for map building in unknown environments. However, this can be used in conjunction with relative positioning methods, particularly with that of odometry to improve accuracy. The main advantage of map matching is its use of naturally occurring structure. The production of maps also enables path planning and navigation through environments which have local minima traps in which many reactive strategies fail. Also the robot can learn and improve its positioning accuracy by exploration. The disadvantages are that there must be enough stationary features for the robot to be able to perform proper map matching. The level of mapping detail is dependent on the tasks that are expected to be achieved. Also this method presents important sensory and computational burdens. At the present time map based positioning is at an early stage of development, restricted to laboratories and simple environments.

Central to map based positioning is the idea of some form of internal representation of the world. There are two major types of representation that are commonly used, and these are *Geometrical* and *Topological* maps. Geometrical maps attempt to position objects within geometric relationships to one another. Examples of such are occupancy maps, line maps and polygon maps, where all the features that are recorded are positioned with respect to one arbitrary frame of reference. The major difficulty arises from attempting to match local maps with the global map when there are error uncertainties in position. The geometrical description takes no account of these errors which can lead to false descriptions of positional relationships. In order to reduce this problem the main focus becomes the attempted elimination of errors at the sensing stage.

The topological approach builds up maps, not with respect to a global frame of reference, but by describing the local relationships between observed features. This method therefore produces a graph like structure with the nodes (the observed features) being linked together by edges (their localised relationships). These relationships can be achieved without positional estimates, and unlike the geometrical approach, it is possible to build up a topological map without positional information. This therefore eliminates the conflict between the robot's estimated position and its position within the map. However, maps with no positional information cannot be used for optimal global path planning.

The inability of geometric mapping explicitly to take into account the uncertainty of positional relations between nodes can be tolerated when the problem is constrained by a known environmental map. However, in an exploratory phase the position of the robot can only be estimated, which makes the discovered relations between new features also estimates. In such situations any mapping technique that explicitly reflects this level of estimation, must have a greater potential for accurate mapping. It is from this observation that it is suggested that some form of topological mapping must be superior for explorational knowledge acquisition.

3.7 Summary

Software simulation is important in the preliminary design of a mobile robot. It allows for quick assessments of the basic elements early in the design cycle. It forces design considerations to be made as to the basic type and provisional construction which is best suited for the task, and requires assessments of different sensor systems and drive mechanisms.

There is also a need in the design process to have effective tools for data visualisation which can only be adequately provided by a graphical user interface. However, such systems must be complemented by good systems modelling to achieve realistic results. The accuracy of this modelling cannot reproduce the true nature of real interactions and it is therefore envisaged that the developed controller designs must eventually be implemented on physical robots.

Although there are many possibilities for control architectures, it has been argued that a layered control system is the most suitable framework for development. This solution combines computational efficiency with ease of development. The selection of architectures and the determination of a mapping module leads to an assessment of the fundamentals of different mapping strategies. From this it is suggested that topological forms of mapping should be more fully investigated.

This chapter has examined the role of simulation and described both a software simulator and outlined the basic design of the mobile robot. However, a mapping technique which has the advantages of topological maps is required. Such a strategy and the simulations of this make up the body of the next chapter.

Chapter 4

Simulations of an Artificial Neural Network Mapping Robot

4.1 Introduction

This Chapter examines the background to biological as well as artificial neural networks which possess mapping strategies. Based on this, a neural network based solution for use as a mapping module is then described and tested. Problems associated with this implementation are examined and comparative tests are carried out on two modifications. Conclusions of this work, to make the neural network solution viable for use as a mapping module, are then presented.

4.2 Natural and Artificial Neural Networks for Mapping

After man, the most effective autonomous systems which depend upon their own abilities to explore, map and navigate are living organisms. They provide proof that such functionality is possible, but how are these actions seemingly so easily achieved? Much research has been carried out in Biology in an attempt to answer this question. One particular study is reported here which examines rat navigation by modelling neurological systems. Rats were chosen, as they can exhibit fast learning of environments, and can form sub–optimal paths towards goals.

Burgess *et al.* [40] initially detail the current thinking in neurophysiology as to how mapping information is stored in a rat's brain. Many researchers believe that a structure called the hippocampus is involved with a rat's ability to locate itself. This structure is composed of many cells which only fire when the rat is in a particular portion of its environment. These *place cells*, with their associated firing fields, densely cover the environment and are strongly influenced by sensory cues. They allow the rat to distinguish between different locations and have been shown to be built up rapidly when a rat enters a new environment.

It appears from this study that the rat's nervous system is using localisation based on some form of phenomenological representation. This is similar to a topological map which does not consist of place cells, but localised areas of recognisable features. The implication is that a rat's mapping systems are far closer to topological rather than geometrical maps.

Burgess *et al.* proceeds to describe a neurological model for navigation which is then used to simulate the movements of a rat. The model is composed of several layers of cells, the first being a set of cells which take information from cues in the environment. The next layer is composed of the place cells which form their firing fields from the layer below. The cells in the final layer combine the firings of many place cells to provide larger firing fields. Goals and obstacles can be associated with these larger firing fields which can then be used for navigation. The primary aim of this work is to validate the proposed neurological model as opposed to achieving the best navigational strategies. However, it does provide strong clues as to how rats achieve exploration and navigation.

Zimmer [41] combines the ideas of a topological map with an artificial neural network method of building the map. He terms his map a 'Qualitative Topological Map' which is a graph style topological map with positional information associated with each node. Zimmer recognised that a topological map is very similar to some forms of self organising maps, which perform unsupervised learning. In particular a variation called the Growing Cell Structure, which can increase its mapping size by introducing new nodes into its topology. This was to form the basis of the robot's automatic mapping generation system.

The type of robot problem that Zimmer describes is the exploration of a controlled environment based on using simple passive sensors. The sensors that the robot uses are 24 peripherally mounted light sensors, 24 touch sensors and an odometric position estimate. The environment included obstacles and light sources, similar to studio spotlights, to provide distinctive illumination. In both simulation and on a real robot Zimmer showed that the robot could produce topological maps of the environment subject to the correct selection of network parameters.

The main premise of this work is that an environment can be categorised on the basis of light and dark regions. Although, these make for simple sensory systems, a particular set of lighting schemes is not a fixed property of an environment. It is also unrealistic to introduce directional lighting sources which allow the all environmental locations unique sensor vectors.

The combination of both a topological mapping framework and that of a self organising artificial neural network require more detailed examination. However, such work must be undertaken using more realistic sensory systems operating in more realistic environments, where the problems of non unique location vectors, are present. The next section describes the growing cell structure algorithm in greater detail. This is then followed by simulations in which active sensors, and environments with non–unique location vectors are incorporated.

4.3 Description of the Growing Cell Structure Algorithm

The method tested out in this section is based on a graph structure in which the nodes represent points of data storage about a localised regions and the links between them indicate neighbouring regions. The algorithms which are used to build up the maps are based on the work of Bernd Fritzke [42]. Fritzke describes a self organising network for unsupervised learning, which he calls a Growing Cell Structure (GCS). The algorithm is designed to construct a suitable network structure and size, recording the statistical properties of an input data set. This is achieved by inserting (growing) and sometimes removing nodes (cells) in a network. Although this method was originally designed to map a given set of data, the aim is to use it to build up internal data maps of an environment during active exploration. For the rest of this section the original algorithm will be discussed, followed in the next section by its adaptation to be used in the mobile robot domain.

The Growing Cell Structure is based on nodes and connections between nodes. The nodes store an n th dimensional vector along with information which is used in the growing phase of structural development, also each node keeps track of its immediate topological neighbours. As opposed to the Kohonen network, the structure is made up of k dimensional simplexes (when $k = 2$ this produces a triangular mesh). Simplexes were chosen for their minimal complexity and that the number of connections increases linearly with k. However, as the mobile robot operates in a two dimensional environment only networks of two dimensional simplexes (meshes composed of triangles) will be considered.

$$\tau_{new} = \tau_{old}\,(1 - \alpha) \tag{4.1}$$

Each of the cell's vector weights can be considered as the position of the node in the input space (as defined by the number of dimensions of the cell's vector weights). The algorithm attempts to place nodes in this space according to the presentation of information to the network, with topologically close neighbours having similar signals being mapped to them. To be able to do this, each node stores information apart from the vector weights. All cells have a counter value labelled τ, which allows the statistical nature of the input data set to be computed. Each time a node is selected for adaptation the signal counter is incremented whilst all the other values of τ are decreased (see equation 4.1) and these values are normalised to yield the relative signal frequency or h. The reason that all non selected cells have their values of τ decreased is that over time the cells move around in the input space, which affects the accuracy of the value of relative signal frequency of that region. This region that the cell occupies (the region of the input space which maps to this cell) is defined as the *Voroni Region* and has an associated volume (F_c). Figure 4.1 shows the Voroni regions of a $2D$ network mapped onto a $2D$ input space. This splitting up of the input space into subregions is called a *Voroni tessellation*, and although in this example the Voroni tessellation is in two dimensions, in general they are n dimensional hyper-volumes, where n is the dimension of the input space.

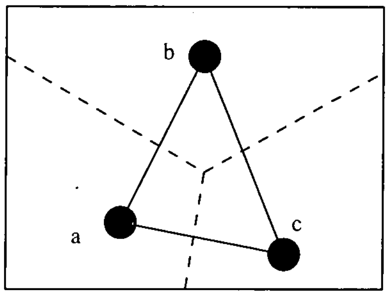


Figure 4.1: Voroni regions formed by a three node graph

The GCS initially starts with a three cell structure with the input vectors randomly initialised over the domain of the input space and the values of τ set to zero. The network is trained by presenting vectors from the input data stream and then adapting the network for each. This is achieved by locating the best matching unit (*bmu*), which has the minimum Euclidean distance to the presented vector. The weight vector of the *bmu* is adapted towards the inputted vector by an amount ϵ_b. The topological neighbours of this cell (as denoted by the links to other cells) are then adapted towards the input vector by an amount ϵ_n. The value of ϵ_b is selected to be larger than that of ϵ_n so that the influence of a data presentation is distributed over a larger region of the network. Figure 4.2 graphically shows the adaptation of one of the cells and the movements of both it and its neighbours in the input space. The adaptation is completed when the values for τ are updated.

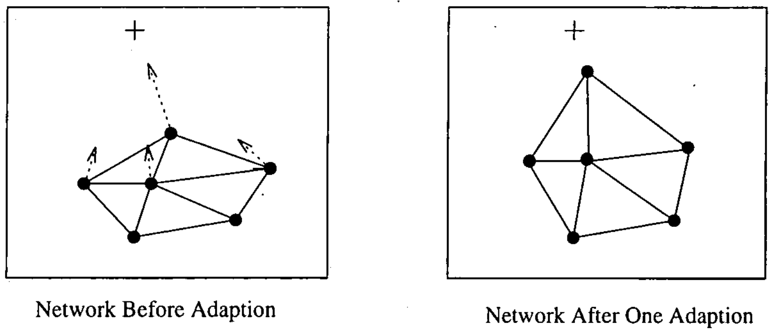


Figure 4.2: Adaptation of a node

With each adaptation the network moves from the initial random positions to a dynamic equilibrium based on the distribution of the input data. Following a constant number of adaptations (λ), the network is then 'grown' by one cell being inserted. Where the node is inserted depends on the desired structure of the network. The algorithm described below attempts to cover the input space with a nodal density representative of the statistical properties of the input data stream. The insertion process begins by selecting the node with the highest computed relative signal frequency (cell q). The topological neighbours of cell q are then searched to find the one with the largest Euclidean distance in the input space, and this is called cell f. The new node (cell r) to be created is then inserted between these two nodes (each element in cell r's vector weight is initialised to the average value of the respective elements in cells q and f). Then the new cell is linked up topologically to cells common to both q and f so that the network remains consistent as a mesh of simplexes. Finally the values of τ are redistributed to reflect the density of information that each node would have received were it to have been in its new location from the start. This involves the calculation of the density of presentation of input data in the input space and hence uses the values of Voroni field stored by each of the nodes. The insertion process is shown in figure 4.3.

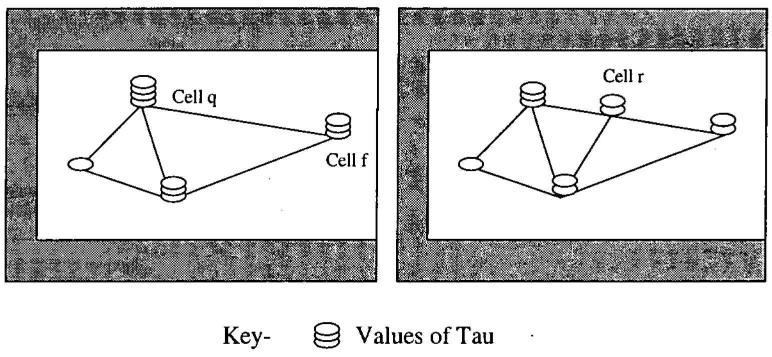


Figure 4.3: Insertion of a new node

The process of network growth can be seen in figure 4.4. The four diagrams show the progressive growth of a two dimensional network onto a two dimensional data set. The nodes are shown as black dots with their topological links displayed between them. Their positions are a graphical representation of their respective weight values, the input data set is indicated by the white areas from which the input data is taken randomly. The network at an early stage forms an approximation of the data set and with repeated training it forms in increasingly matched mapping of the data.

The GCS algorithm is based around a certain number of repeated adaptations followed by insertion of a cell. The total cycle is then repeated until a certain density of nodes has been attained. However, in some situations node insertion by itself is insufficient correctly to map an input data source. A prime instance is when the input data is split into separate regions of positive probability density. In such a case, nodes can be placed in locations of zero probability. This is mainly caused because the network at each stage in the growth process represents the data set at the level of definition allowable by a limited number of cells. Hence at the start of the growth process, nodes can be inserted in zero probability density regions between established nodes. These can stay in those regions and corrupt the validity of the

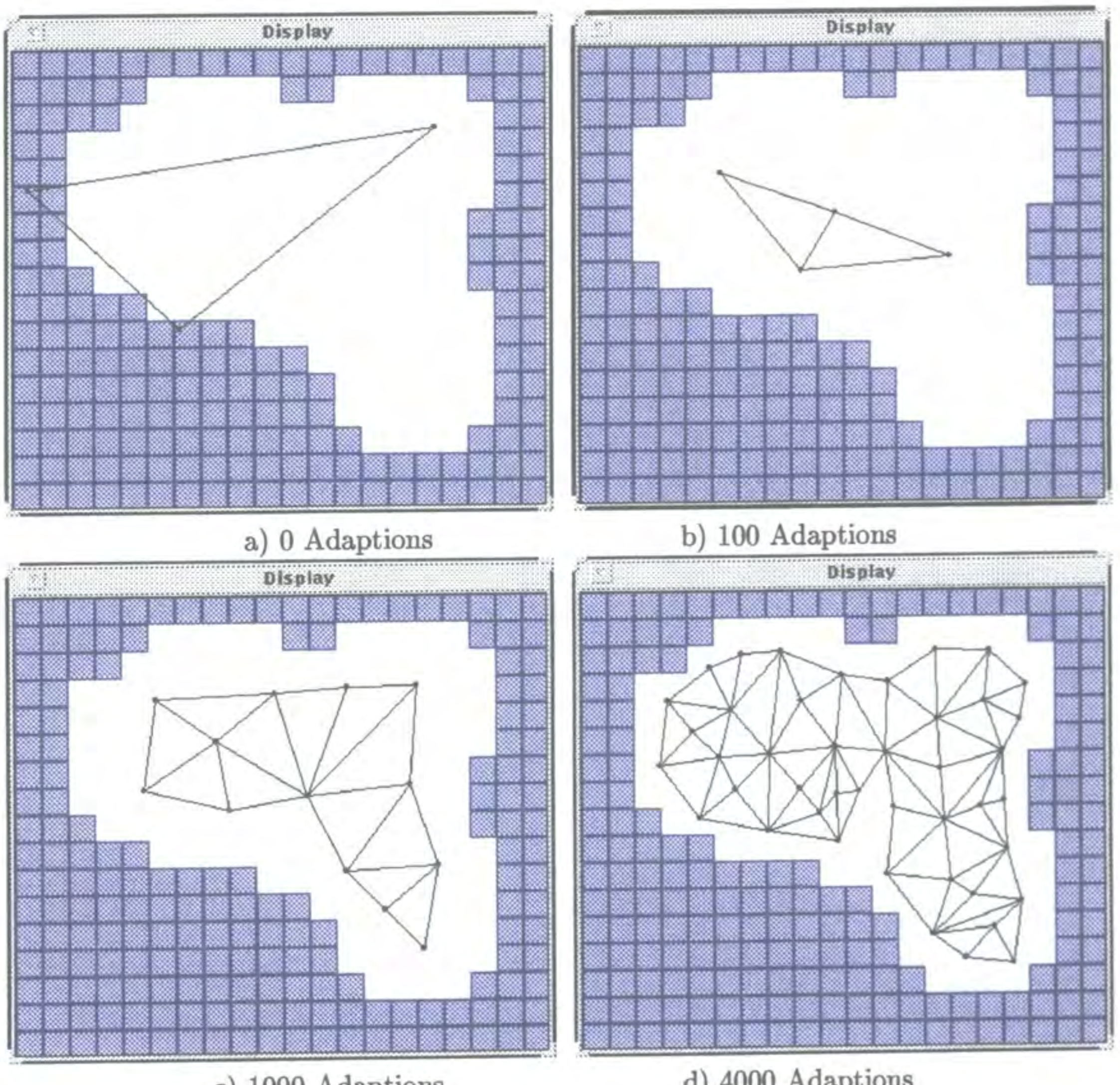


Figure 4.4: Development of a $2D$ growing cell network to a $2D$ data set. The model parameters were $\lambda = 100, \epsilon_b = 0.06, \epsilon_n = 0.002, \alpha = 0.05$

final network, and it is therefore necessary to delete nodes in this original GCS model. To assess which nodes are potentially in regions of zero input probability distribution, the node's relative signal frequencies are monitored. These relative signal frequencies together with the voroni field sizes give an indication of the probability of input signals being mapped to a particular node in the network. When this probability drops below a pre-set threshold the node is deleted.

With respect to the desired properties of an internal data structure as discussed above the growing cell algorithm has many useful features. It is computationally efficient with the ability to increase or decrease in both content and structure. The links between nodes indicate similarly matched nodes or regions, and for a mobile robot this can be used to indicate the topology of the environment and hence produce a map. There are however some assumptions in the design of the growing cell structure which do not hold for the mobile robot application and these must be addressed.

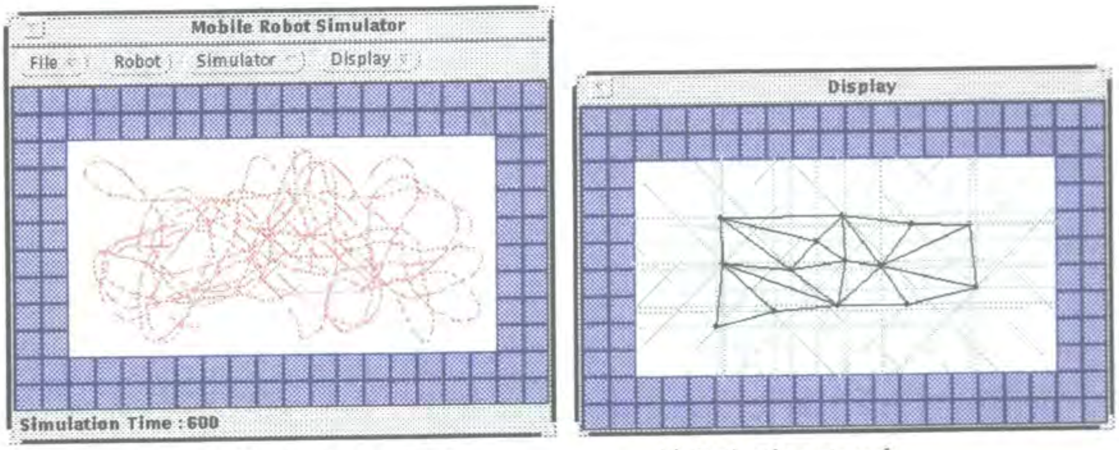
4.4 The Growing Cell Structure used in the Mapping Module

In the previous section the growing cell algorithm was described. The production of a good quality topological map was shown from a stream of statistically independent samples. In this next section the GCS will be trained from data taken from a mobile robot's sensors, in order to generate a topological map of the environment.

The first robot control system which was tested used the simplest form of growing cell structure network. This simply took the stream of data vectors from the robot's sensors and used them to construct a network. The output from the control system did not use any information which was held by the network, the control being produced by the combination of two behaviours. In effect the GCS simply monitored the movements of the robot in the environment and made adaptations to learn this. As the control system makes no use of the information in the map to direct the exploration, the control system must use a method of moving within the environment designed to maximise the area covered uniformly. The particular control system used in these tests included information from only the wander and avoid layers and hence the paths could not be altered by using information from the built up map.

The growing cell structure starts with three cells which have to be initialised with data. The cells were not initialised with random vector weights but were instead located in a circular ring around the robot prior to release in the environment. The robot has no prior knowledge of the environment or the vector values that those positions would possess, so the values for initialisation were chosen to be those that the robot was detecting.

The first experiment was conducted in a rectangular grid, with the robot moving with both the wander and avoid behaviours in operation. The sensors in this experiment are being



a) Path of the robot b) Robot's network

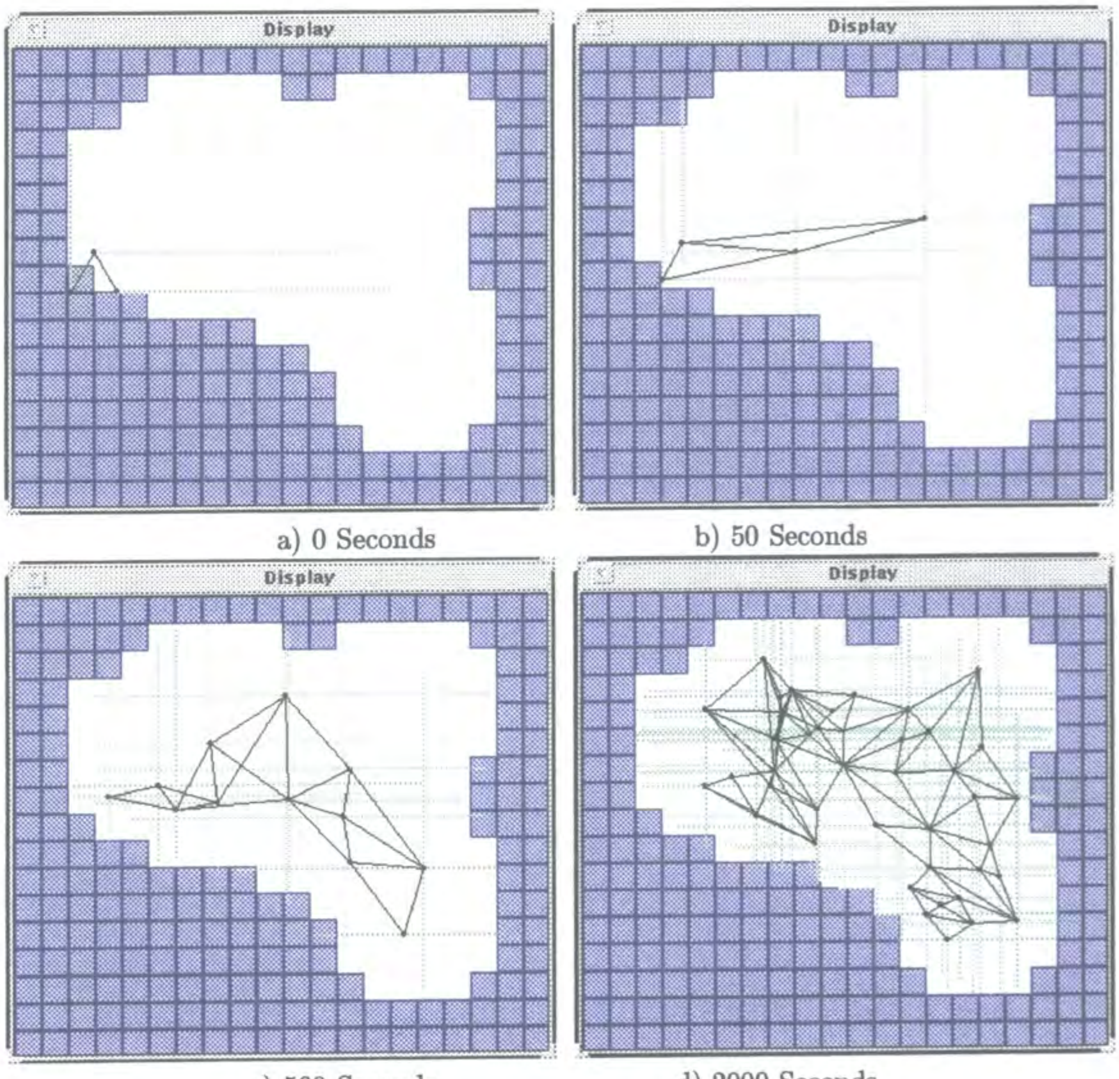
Figure 4.5: Network growth of a mobile robot using simple GCS

sampled at half second intervals, so in total the network shown in figure 4.5b has received a total of 1200 input vectors shown only once each. The faint lines on this diagram emanate from each of the cells and graphically show the information stored at each node. The robot shown in this diagram has used eight range finder sensors, but similar results are obtainable for other numbers of sensors. The path the robot has traversed in the environment is also displayed in figure 4.5a. It can be seen from this path that the movement in the environment covers the area well, even though it is non directed.

The growth of this network in combination with a robot's movements in the environment indicate a good matching both in cell values and network topology. At this stage of experimentation, no noise affects have been included. The diagram shows that this form of mapping could have potential as a mobile robot's internal data structure.

The next experiment compares the growth of a network mounted on the mobile robot, with that obtained by randomly sampling data in the environment, to assess the differences in the growth and resultant output. Figure 4.6 shows the growth of the network at the same stages of adaptation as was the network shown in figure 4.4, as the sampling rate in the test was every half a second. The robots shown, possessed four range sensors, each without noise disturbance, and drove under the control system as in the prior experiment.

It can be clearly seen from the diagram that the network growth is inferior to the network which has had data randomly sampled over the whole region. If the growth in the network is compared at the time interval of 500 seconds (corresponding to 1000 adaptations), the network of figure 4.4 can be seen to be more fully developed and with a more even topology. In figure 4.6c the left hand side of the network is showing signs of twisting but the right hand portion is growing in a more even manner. These disturbances persist into the final network structure at time 2000 seconds (d), with the final topology showing signs of both good and



a) 0 Seconds b) 50 Seconds

c) 500 Seconds d) 2000 Seconds

Figure 4.6: Development of the GCS during exploration of the environment. The model parameters were $\lambda = 100, \epsilon_b = 0.06, \epsilon_n = 0.002, \alpha = 0.05$

bad areas of development. Using the described control system, these results are typical of an average run, however the difference between networks is diverse. The diversity can be seen when the robot is forced to proceed down different trajectories (seeding different values of the wander behaviour's random number generator).

It is suggested that two differences that lead to the network's misshape are the correlation in the input data stream, and the inability for a robot with mass to cover the full area (producing a less rich data set). This would suggest that a suitable solution might be developed by reducing the correlation of the received data stream, or by using feedback from the network to force the path to have an inherently lower correlation value.

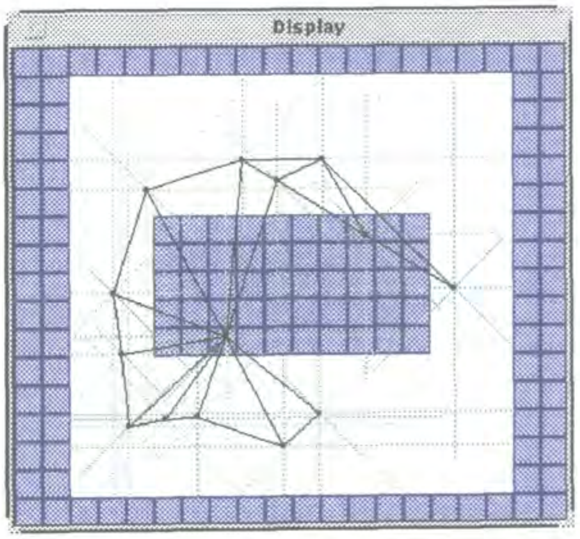


Figure 4.7: Problem mapping

One other major problem for the GCS is its handling of fragmented shapes. Figure 4.7 shows the network growth over a complex shape. It can be seen that there is a tendency for the network to develop toward the centre of the distribution, and this results in nodes and connections over the central object. As the robot moves, the network does not have enough nodes to adequately record the information, which leads to mapping failure. This is because the development of the network is only linked to the number of samples it has been trained with, not to the area of the environment that has been covered.

4.5 Problems with the Growing Cell Structure for Mapping

Simulations using a mobile robot as a data source for the production of growing cell structures have shown effects which have not be reported in the original paper by Fritzke [42]. Twisting and distortion of the network have been observed and it is hypothesised that this is a direct result of the correlation in the input data stream from which the network has been trained.

The following experimentation tests out this hypothesis.

The growing cell structure which has been discussed by Fritzke, takes an unknown input data stream and models this through unsupervised learning. The algorithm creates a topological map which clusters data with similar characteristics onto topologically close regions on the map. Fritzke extensively discusses the algorithm for the creation of the mapping, and shows a few examples of the network's ability to fit an input data source. One of the features of the network's growth is its ability to record the probability density of the input data stream in nodal density, so that after training, each node has an equal probability of being selected. However, although the network should record the probability distribution, it is assumed that successive signals are independent of each other.

If the growing cell structure is to be used to map information from a mobile robot, then the particular statistical properties that this type of data source produces must be considered with respect to the algorithm. The mobile robot takes in data from its sensors in a periodic fashion (dependent on the number of sensors, this could be expected to take around half a second). We can treat this as an n dimensional vector, having components from the distance measures and the odometric calculations. Each successive data vector from the robot will therefore be similar to the last vector and hence the data is sequentially correlated.

The correlation of the input data stream from the mobile robot in an environment can be considered to be the combination of two effects. Firstly, the robot's motion within the environment dictates the lower bound on the correlation of the input data, if it is considered over the whole of the robot's run. If the robot moves slowly from one location to the next with respect to the environmental size, the sequential correlation in the data stream will be far greater than if the robot covers a larger area in the same number of samples. Hence, the amount of correlation from different control strategies will vary significantly. The second factor that determines the correlation of the data in the input stream is the order in which the data is presented to the network. If we have the full run of data from the mobile robot then the order of presentation of data can be altered to minimise the correlation of the data stream. In practice it would be computationally expensive to store all observed data, but by storing only recent data it would be possible to use this to reduce the correlation of the data.

Our prime interest is to form an even topological map in the two dimensions of x and y. It is proposed that the effects of input data which is sequentially similar will be investigated, and suitable methods to combat this problem will be examined. It is important to investigate a suitable method to assess this correlation.

4.5.1 Measuring Correlation of the Sensor Data

These experiments are based on the belief that network growth in a growing cell structure can be affected by the sequence of information in the input data stream. It is therefore important to produce some form of measure that usefully indicates the sequential similarity of a sequence of data. This measure of self similarity is the opposite of the assessment of randomness, and therefore methods which have been used to assess randomness will be used.

Initially a path in an environment will be chosen and then different measurements for randomness will be then used to assess their suitability. Figure 4.8 is the path of the robot in a rectangular environment showing as squares the points at which the robot took a sample in the environment. This path produced three thousand data samples which would be used to train a network, these can be used to assess the correlation in this data set.

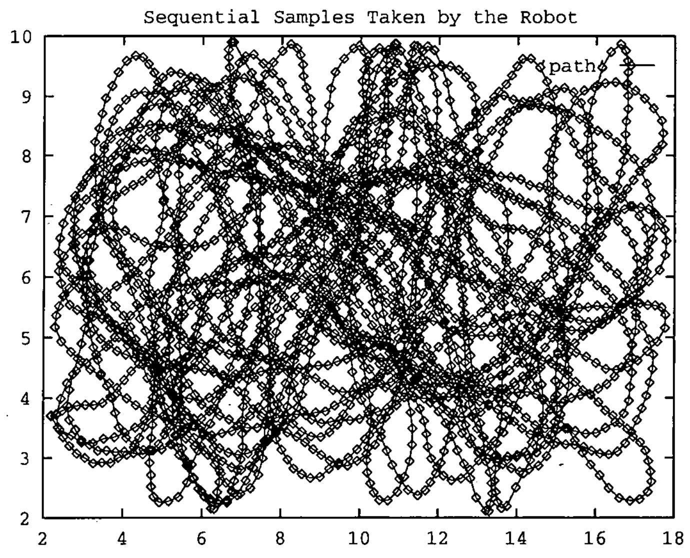


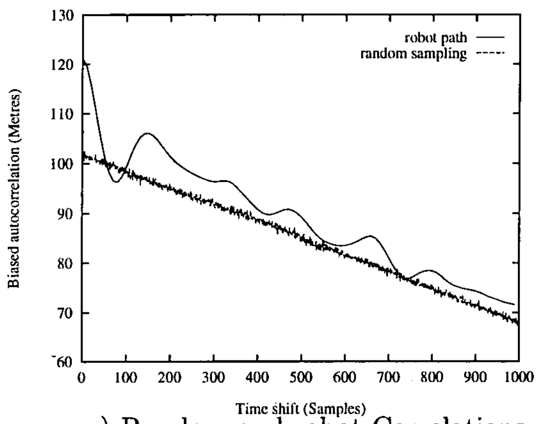
Figure 4.8: The path of the robot with sampling points

Different methods of assessing the randomness of the sequence of data must be compared on this example path. The measures of autocorrelation, auto-covariance and correlation coefficient will be compared. The parameters of x,y, and distance from the origin d will be also compared for use.

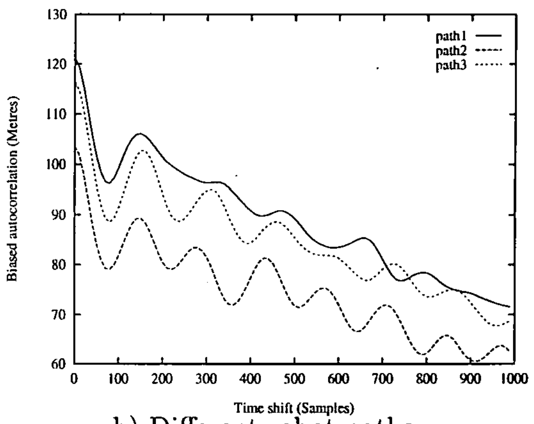
$$R_{XX}(t_1, t_2) \triangleq E\{X(t_1)X(t_2)\} \tag{4.2}$$

If we wish to examine the similarity of the data in the x direction we must consider the sequence of values of x from the robot's path, or input data set $X(t)$. The autocorrelation of $X(t)$, denoted by $R_{XX}(t_1, t_2)$ is the expected value of the product $X(t_1) \times X(t_2)$ (equation 4.2). Given that from the path we have a sequence of samples, we can assess the average correlation between data points at a specified separation. For larger values of separation, there are less product pairs and the average becomes less reliable. If instead of creating the average by dividing the sum by the number of pairs, the sum is divided by the number of samples, this biases the function. This biased autocorrelation function reduces the misleading correlation values produced at larger separations, where the average value becomes less reliable.

The biased autocorrelation function was used on the sequence of x values from the path (shown in figure 4.8) to produce the graph of figure 4.9 a). The x axis of the graph indicates the value of t_2, t_1 being zero. The robot's correlation values are shown against those produced by a stream of random samples in the environment (created by the C language function rand()). Figure 4.9 b) shows two others paths in the same environment, and their values of autocorrelation. The values of autocorrelation show large differences between the three paths taken. It appears at first that some of the robot paths have, on average, lower correlation values than that of randomly distributed values. This is because the average values of x have been assumed to be the same, but for a short robot run this is not necessarily true. Therefore, different runs in the same environment can produce differing plots of the autocorrelation function. Hence the autocorrelation function is not a good comparative measure.



a) Random and robot Correlations



b) Different robot paths

Figure 4.9: Comparison of unbiased autocorrelation function for data sequences

The auto-covariance function eliminates the average values of the data set so that a more accurate comparisons can be made. The equation for calculating this function is shown in equation 4.4, using the value of mean μ_X computed from equation 4.3.

$$\mu_X(t) \triangleq E\{X(t)\} \tag{4.3}$$

$$C_{XX}(t_1,t_2) \triangleq R_{XX}(t_1,t_2) - (\mu_X(t))^2 \tag{4.4}$$

The paths that produced the autocorrelations\graphs in figure 4.9 were repeated and the auto-covariances were calculated and recorded in figure 4.10. It can be seen from the graph that the mean values have been removed and the similarity between different paths can be more accurately assessed. The shape of the graphs will be further discussed later.

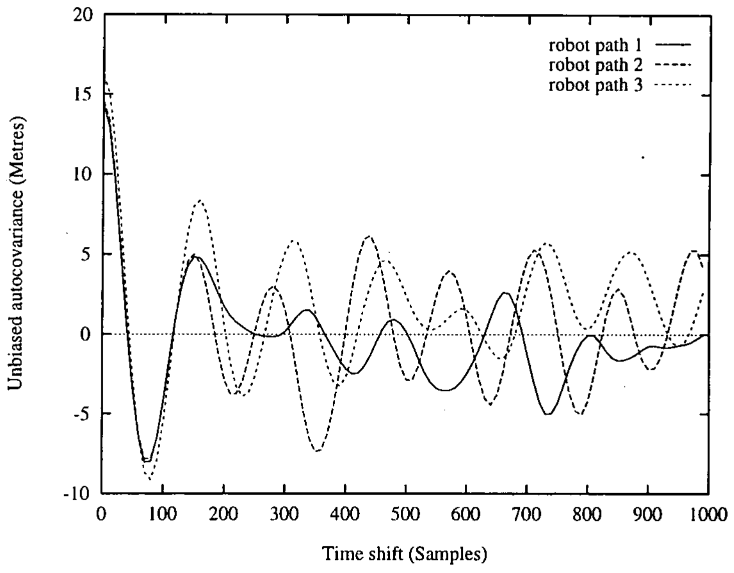


Figure 4.10: Unbiased autocovariance for three different robot paths of 3000 samples long

To check that the number of samples that were being used was not affecting the results, differing run lengths were tested. Figure 4.11 shows the unbiased auto-covariance for four run lengths, indicated by the number of samples taken. It can be seen from these, that only when the run length approached the value of the τ maximum, were the results badly affected.

The function that has been discussed can be normalised so that different environments can be compared on the same graph. This function is called the correlation coefficient and allows for the comparison of data sets with differing variance values. Hence, this function can be used to compare different correlation factors, and it can be used to compare factors between different environments. This function is calculated from the equation 4.5. Figure 4.12 shows a comparison between the x,y,and total distance d correlation coefficients. Also displayed on the same graph is data stream taken from a random sampling in the environment.

$$r_{XX}(t_1,t_2) \triangleq \frac{C_{XX}(t_1,t_2)}{\sqrt{C_{XX}(t_1,t_1)C_{XX}(t_2,t_2)}} \tag{4.5}$$

To assess the reason for the nature of the graphs produced so far, from the correlation methods, the production of the examined path must be understood. The path is typical of those generated by the motion of the robot under the influence of two behaviours; those of avoid and wander. The wander behaviour creates a force vector intended to produce a path for the robot that would meander but not produce Brownian motion (Brownian motion tends to move slowly from a given place). The wander behaviour in each instance produced a vector that deviated uniformly randomly from the direction of travel by an amount $0.1 \times \pi$. The avoid behaviour simply produced a repulsive vector from the sensor range measures. These behaviours combine to drive the robot backwards and forwards in the environment. It is

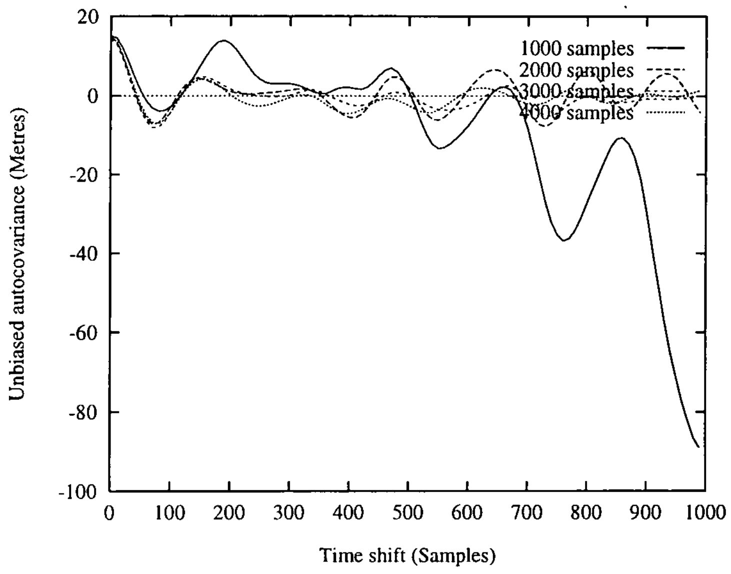


Figure 4.11: Variation of autocovariance for different path lengths

this tendency for the robot to wander to and fro, that leads to the oscillating values for the correlation. As the comparison between samples becomes further apart (greater values of τ) the correlation decreases. It can also be seen that for random presentations of the data to the network, the correlation values are near zero (the same training sets that produced good topologies).

It is hypothesised that the distortions which have been observed with the GCS algorithm are a direct result of the correlated nature of the presented data. Therefore, reducing the correlation of the input data stream should lead to better network growth and twist free topologies.

The next section looks at two different methods that could be used for the reduction of the correlation in the input data. The first is based on a shift register with a number of taps, and the second re-presents the data randomly from a short term memory.

4.5.2 Reduction of Correlation using a Shift Register

This section looks at the potential benefits which could be gained from the use of a tapped shift register for the de-correlation of the data stream presented to the GCS. Each time the robot takes a complete sampling of its sensors, this is fed into the shift register (as shown in figure 4.13 where each block represents a full sample). At intervals of m samples apart, copies of the data blocks are taken to be used to train the network. In effect this method produces n times as many training vectors for the network. The method relies on the assumption that after m samples the statistical dependence between samples is sufficiently small so that the correlation does not interfere with itself.

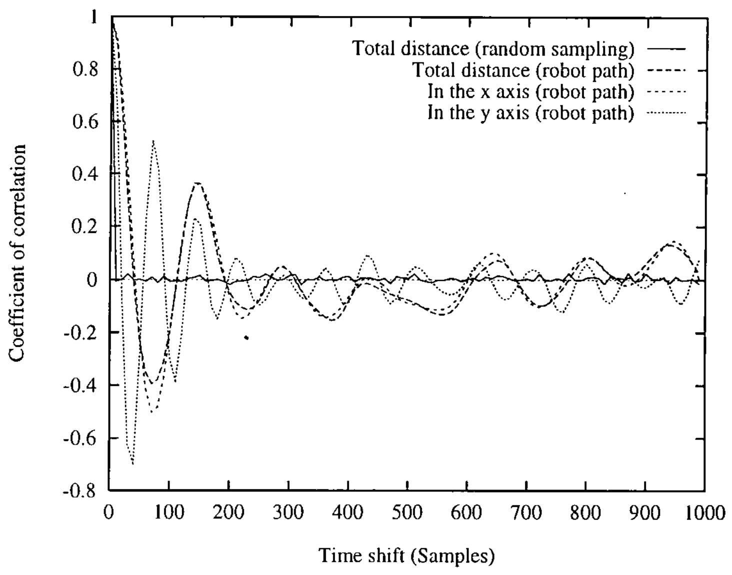


Figure 4.12: Comparison of correlation of x,y and total distance measures

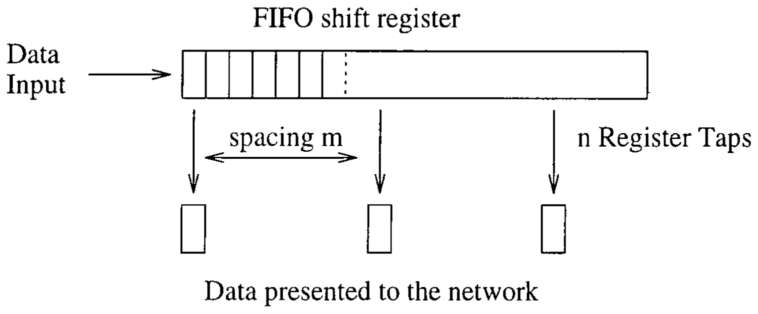


Figure 4.13: Shift register used for de-correlating incoming data

The information which is used to adapt the network is a newly formed stream of data, and it is the correlation between terms in this data stream that will be assessed. One factor inherent with this design is the increase in the number of data samples that are created. Although, this might imply a faster training time, the underlying data only covers the same area of the environment. Therefore, for comparative tests the network lambda value should be multiplied by n, so that the network creates the same number of nodes per unit time independently of the number of taps.

To investigate the de-correlating effects of the shift register, the data produced by the robot in a rectangular environment was used. This input data was passed through the shift register, and subsequently analysed for correlation. For the first test, and to compare the results with a theoretic analysis, a simple two tap register was used. This register was set to have a delay of 200 samples between the taps. This means that for the first 200 samples out of the shift register, there will only be data from the first tap. Figure 4.14 shows the input

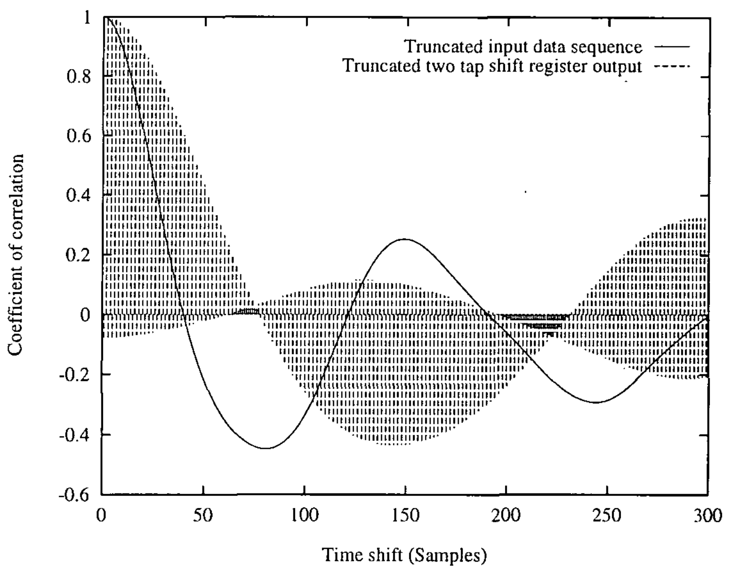


Figure 4.14: Comparison of the input and output data from the two tap shift register

and output data's correlation between successive samples. Each data sequence was truncated by the removal of the first 200 samples so as to test the validity of the method. It can be shown that for even values of shift τ that the value of output correlation r_{yy} corresponds to half the shift for the original sequence r_{xx} (see equation 4.6). For the odd shift values the value is more complex but is described in equation 4.7. This assumes that the sequence is ergodic, and the sequence under analysis is sufficiently long.

The results from the figure 4.14 matches that which would be expected from the equations. With more taps the graph of the coefficient of correlation becomes more complex. Figure 4.15 shows the correlation produced by a 400 long shift register with ten taps.

$$r_{yy}(\tau) = r_{xx}\left(\frac{\tau}{2}\right) \tag{4.6}$$

$$r_{yy}(\tau) = \frac{1}{2}\left[r_{xx}\left(\left|m + 2 + \frac{\tau+1}{2}\right|\right) + r_{xx}\left(\left|m + 1 - \frac{\tau+1}{2}\right|\right)\right] \tag{4.7}$$

Although this method breaks up the correlation function from being continuous, it remains unclear as to whether this produces better results for the development of network growth. In some instances, the development of the growth has not been noticeably improved. For example, using a ten tap 400 long shift register, the growth of the network in the environment used in section 4.4, is not improved. The method's beneficial use at this stage of development is inconsistent. This might well be attributable to the repeating nature of the correlations against the shift τ. Although it might spread out the correlations, there is no mechanism to reduce them.

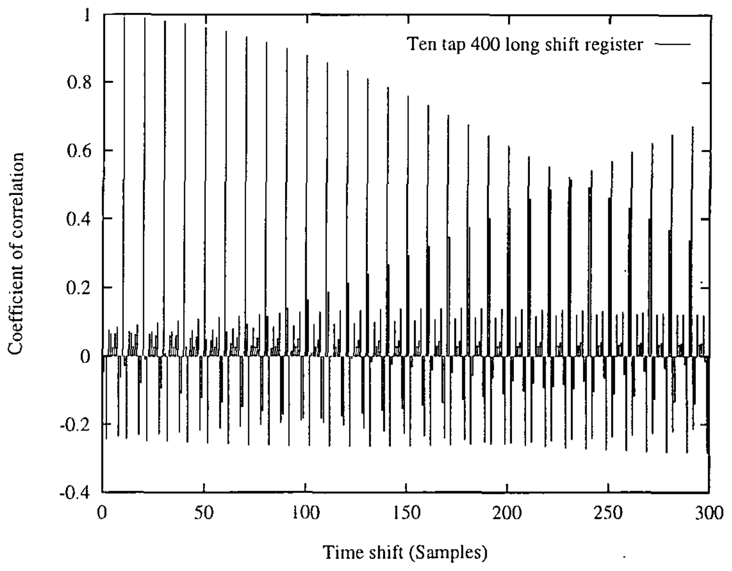


Figure 4.15: Output correlation from a ten tap shift register

4.5.3 Reduction of Correlation using Random Sampling

This method uses a shift register style memory bank of length m. Data from the robot's sensors is fed into the register at each sampling instance, and then a block is selected at random to be the output (see figure 4.16). The block is selected by using a random operator to give a uniform distribution of selection. This method unlike the one above, produces the same number of output vectors as it takes in, and hence the correlation plots can be directly compared.

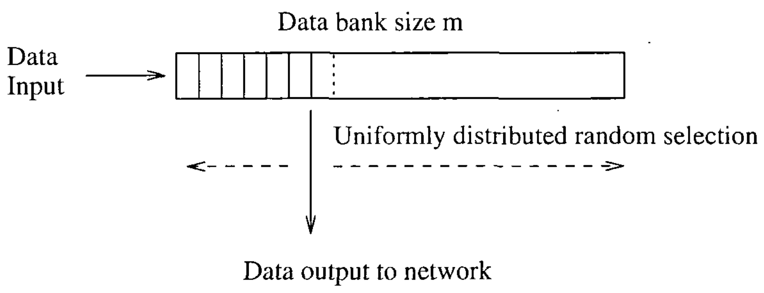


Figure 4.16: Random sampling used for de-correlating incoming data

Figure 4.17 shows the correlation produced by different memory lengths on the path of the robot in a rectangular environment. This method shows a gradual decrease in the correlation between data points as the separation between them increases. This becomes more marked as the memory length increases. For a memory length of 200 samples the correlation becomes constant for differing separations of τ. This method appears in the experiments so far to

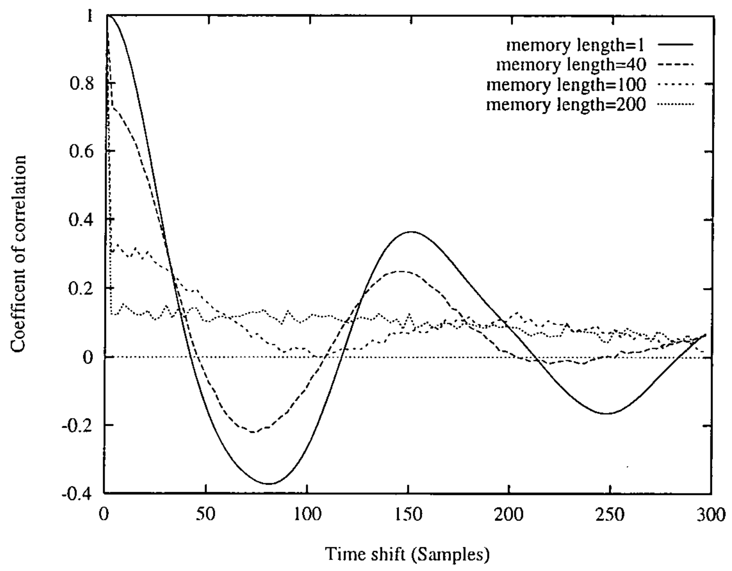


Figure 4.17: Output correlations for different sized memory banks

produce consistent de-correlation of the input data steam. The networks which have been produced by this method similarly show a better growth than those without it.

Figure 4.18 shows the development of a network using a 400 long memory store, for the same length of time and in the same environment as used previously in section 4.4. Therefore, the network can be directly compared to that produced without the memory store. It can be seen that the network has grown evenly and fitted itself to the underlying structure of the enclosure. It can be seen that the network is located mainly in the central regions of the environment, unlike the network produced by random sampling. This results from the robot's path within the space which more densely covers the central regions.

The results reported above seem to support the claim that network distortion is directly a result of correlation within the input data stream. This method of using a memory and random selection de-correlates the data and leads to a uniform network growth.

4.5.4 Conclusions on the Growing Cell Structure

The GCS was originally designed for unsupervised learning of high dimensional mappings from a statistically independent data source. The beneficial factors for using the system are its ability to cope with high dimensional data streams and the automatic generation of map topologies. The problems which have been experienced come from the particular restrictions that are inherent for a mobile robot. That is the highly correlated nature of the sequential samples from the robot's sensors, and the requirement to have an accurate mapping in just two of the dimensions.

The requirement for this accurate mapping conflicts with the mobility of the GCS nodes in

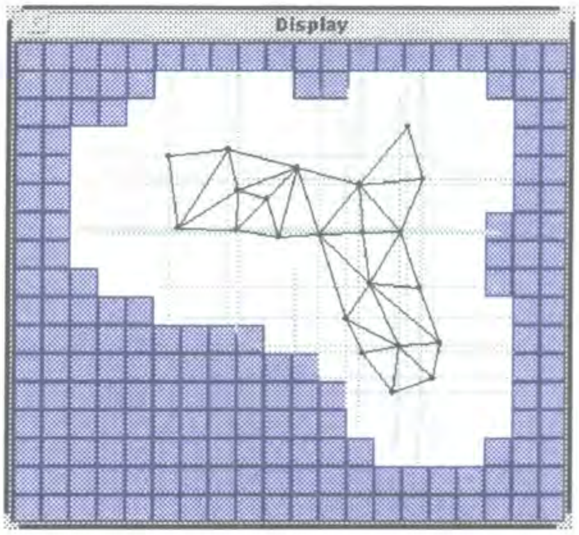


Figure 4.18: Development of a GCS structure using a 400 long memory after 2000 seconds. Network parameters were $\lambda = 100$,$\epsilon_b = 0.04$,$\epsilon_n = 0.004$,and $\alpha = 0.05$

the input space. If the network is trained directly from the robot data the nodes are dragged by local factors in the path, and the nodes and connections between them become unchecked. For a reliable solution the problems which corrupt the GSC must be more accurately defined.

The GCS needs uncorrelated data over the full input domain. This is impossible for the defined problem as the robot cannot know the bounds of its environment. Its environment is defined as unknown at the start of the robot's run. The use of the memory bank in the last experiment, showed a method of reducing the correlation in the data stream for a given environment. This reduction in the correlation produced a good map of the environment, where previously only poor results had been obtained. However, for different paths and more complex environments, the size of the memory and other parameters would need to be changed. These parameters would have to be adaptive, and a generic solution to this problem has not been yet been developed. Such a method would require yet greater algorithmic complexity.

Generic, and therefore reliable solutions to the problems appear to be far from being developed, whilst the complexity of the system keeps increasing. It is now argued that simpler methods can be used to build up maps in static environments, overcoming many of the difficulties involved in the use of the GCS algorithm.

4.6 Summary

This Chapter examined the problem of exploration of an unknown static environment and subsequent collision free navigation. The use of a software based simulator for comparative testing in conjunction with a phenomenological (or free space) mapping control system has been argued.

The growing cell structure neural network method has been examined in detail and its use as the basis of a robot mapping system investigated. Although the network structure has been shown to develop with statistically independent input data, poor network growth has been observed when trained directly from a robot. More complex environments have been shown to exasperate the problem, leading to failure to produce usable networks.

A corrective solution which uses de-correlation of the training data has been proposed, and two potential methods have been experimentally tested. Only one of the methods is shown to be useful in the de-correlation of the data. This method has then been used to successfully train the network with data from the robot run.. However, the development of this approach into a generic solution requires considerably more algorithmic complexity. It is therefore argued that a simpler method for the construction of the network is possible, which will be developed further in the following chapter.

Chapter 5

Simulations of a Novel Mapping Method

5.1 Introduction

The Growing Cell Structure develops a network structure in stages by the insertion of nodes between older nodes. The newly created nodes inherit, or interpolate the information possessed by the older nodes. This type of network growth is in effect *interpolative growth*, with new nodes being created within the domain of the network. For such a network to develop and cover larger areas in the input space, nodes must be dragged from their positions so that new nodes can be formed behind them. Not only is this disadvantageous for the data stored in the nodes, which must be re–adapted to match that of the new locations, but the validity of the connections between the nodes becomes unchecked.

The novel method proposed attempts to overcome the difficulties described above by employing *extrapolative growth*. Instead of new nodes being created within, forcing larger numbers of nodes to be moved, old nodes will remain in their created positions. Therefore with stationary nodes their informational content need not be changed and the connections between them need not be reassessed. New nodes will be deposited by placing them outside the network structure, with connections being made to existing nodes. This method relies on the incoming data stream being sequentially correlated in (x, y), because the formation of a new node and its connections are determined by a distance threshold. If a data sample was presented which was spatially distant then the links formed to it would be erroneous, which as previously discussed, is an important property of data produced by mobile robots. Extrapolative methods would not have to retrain already covered space, and therefore present a method which should converge on a solution in a shorter space of time. The novel approach to be described is based on such an extrapolative method.

5.2 A Novel Topological Method for Mapping

The use of a graph representation of free space is still used in this method, offering the flexibility and potential for data efficiency that a grid method cannot. Also the network that will be developed will be based upon a mesh of triangular elements linking up the nodes. This has been chosen over a completely free structure so that a greater connectivity will be produced, giving path finding algorithms using the network a richer potential for paths. The structure and the data of the network is still that used in the previous GCS algorithm, receiving information from ultrasonic sensors mounted around the robot's periphery. Similarly each node records its position relative to some datum (This is normally the start of the robot run). The following description relates the growth of the network to the robot motions in the environment.

Initially the robot is considered to have been placed in an unknown environment. Similarly the robot has a network which is initialised with three nodes and also three connections between them. The triangle of nodes is located so that its centre is where the robot starts off, with each node being spaced evenly apart, forming an equilateral triangle. Other than the positional data of the nodes, the information contained with respect to the sensors is initialised to that of the robot's current values. This is the internal configuration of the robot prior to each new training run.

The robot can then start moving around in its environment, continually taking odometric readings and sensor values. The robot's position is used to find the nearest node in the network. The sensor values of this node are updated by adapting them to the new sensor values by an amount ϵ. However, the position of this or any other node is not changed.

As the robot moves around in the new environment, its distance from the nearest node is continually being monitored. When this value increases above a defined threshold *delta*, it is presumed that the robot is in a new area of which its network should store information. A new node is created, containing the values of the data the robot is detecting at that time. As well as creating a new node, the connections that link it up must also be formed. The network being composed of triangles and nodes requires that each node must have two connections to it (forming the apex of the triangle). The first assumed connection is that to the nearest node from which the distance exceeded the threshold delta. A third node must be selected and this is chosen to be that neighbour of the near node which is closest to the new node. A new triangular structure is created with the three selected nodes as vertices. The network has now developed by one node and two connections (or one triangle), the distance delta is zeroed and the robot moves on till the distance increases above that of the threshold again. This mechanism allows the robot's internal network of information to develop, covering the robot's known area with nodes spaced out according to the delta parameter. The creation of a node in the network and its relation to the environment are shown in figure 5.1.

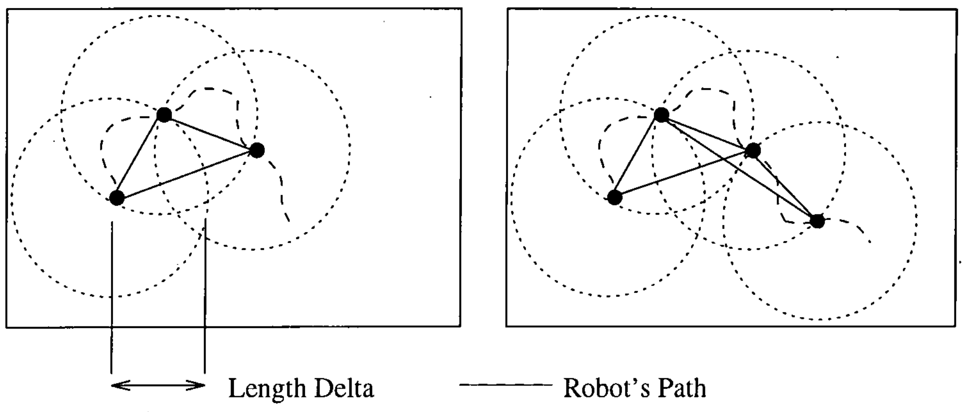


Figure 5.1: The creation of a new node in the network

The above mechanism is a method for the spacing out of the nodes in the environment, but does not explicitly connect all the nodes within that environment. The connections that are developed between the nodes are a result of the insertion process, and do not necessarily link up the nodes in a localised region with the best topological structure. A search algorithm which was efficient in covering the area with nodes would not necessarily create a good topological structure. If the connections are to be used to abstractly store information regarding the layout of the nodes and traversable paths between them, additional links must be made to update the structure of the network.

Each link between the nodes represents the potential for the robot to move without collision. Therefore, if while the robot is moving around within the network and it passes from one node to the next which are not linked, these are then linked up. For the following description the previous node, and the present node that the robot is attached to will be termed nodes A, and B respectively. For the insertion of a new connection between nodes A and B a new triangle must also be introduced. The third node required to link in the triangle was chosen to be a neighbour of the past node, so node C was chosen as the neighbour node of A, which was closest to the node B. Hence, potentially, two new connections are introduced for each linking operation.

5.3 Parameters for Monitoring Network Growth

For a robot to drive around and then use its networks information afterwards to update or re–examine its environment, it must have some means of monitoring the completeness of its knowledge. To find out these parameters, which are important to the development of the network and the accuracy of which that network maps the environment, we must look at the fundamentals of the mapping. It is proposed that the three fundamental factors, are *environmental coverage*, *network connectivity* and *network error*. Each factor provides an

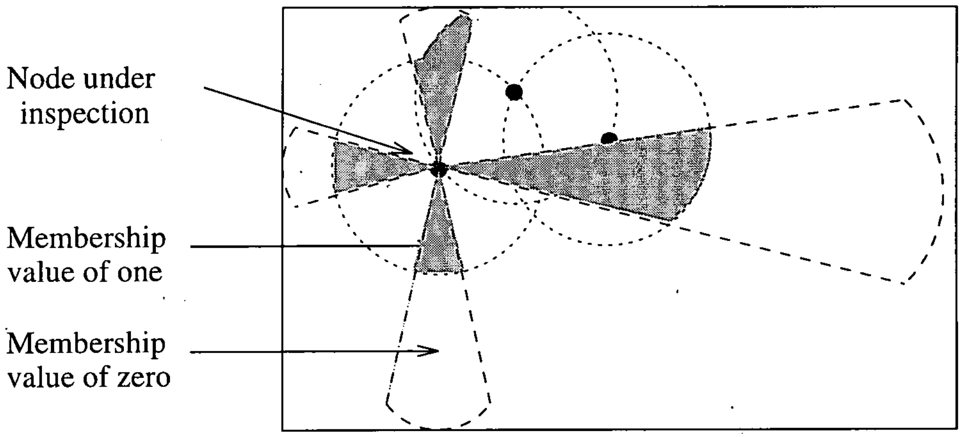


Figure 5.2: Calculation of coverage from sensor range measures

independent cost function by which to monitor the success of the resulting network.

5.3.1 Network Coverage

An important factor is the amount of knowledge that the robot stores in its network about the environment. A ratio of the area mapped, over that which the robot can observe, gives an indication of the robot's coverage of the environment. This is one of the parameters which was used and it was termed *coverage.*

The most basic element in the map is the node which is considered to cover an area of the environment. This area is dependent on the proximity of the nodes as determined by the factor delta. Each node also stores information about ultrasonic range measures giving an indication of the free space that can be detected at that point. Each sensor value can be considered to be a segment of a circle whose angle is dependent on the physics of that particular sensor, and of radial length which is the distance measure itself. For a given elemental area within that sonar sweep there can be assigned a value of membership to the total network. This *membership value* is a reflection of the amount that particular element is covered by the network. In the simplest case this value is either 0 or 1. If the element is within a distance delta of a node it is assumed to belong to the network and a membership value of 1 is returned, otherwise 0. Figure 5.2 shows the regions of a node's sonar sweep 'covered' by other nodes.

The area that is covered within a sensor segment is computed in a piecewise manner. Each elemental area is multiplied by its associated membership function and these are integrated over the total area of the sensor. The values for coverage are computed for an individual node by summing the covered area for all the sensors pertaining to that particular node, divided by the total area that the node covers. This normalised fraction gives an indication of the level of coverage around the node. For a network as a whole, the coverage can be deduced by computing the ensemble average of the nodes coverage.

This method is computationally expensive, but gives a clear indication of the state of the robot's network with respect to the nodes locations. The results show, as the robot creates more nodes within the environment, the value of the network coverage increases. This value tails off to a maximum as the nodes in the network maximally cover the area. Although the value does not usually reach unity, the value reached is relatively high.

5.3.2 Network connectivity

Although the coverage values give a good indication to the nodal distribution in the environment, it includes no information as to the quality or quantity of the connections which exist between the nodes in the network. If the connectivity of the network structure is to be used for the planning of navigable paths in the environment, it would be beneficial if the robot could monitor the network structure that is being developed. However, there is no simple method of producing a cost function for uniformity in a graph topology.

A method is proposed that could be used to assess this function. However this must be regarded as an *ad hoc* approach, and of a developmental nature. The method compares the Euclidean distance of nodes that are in each others sonar 'beams' with the topological shortest distances between them. For a given node N_i, the network is searched to find if any other nodes that are in the sweep of N_i's sonar range measure. The A* algorithm is then used to find the minimum topological route between these nodes, which is then compared to the actual Euclidean distance.

The above method will reveal discrepancies in the network structure which cannot be observed using purely the coverage method. For example, in the case where branches of the same network wrap around an object, the nodes can 'see' each other, yet with a close nodal proximity this is not shown up in the value of the coverage. This information is not reliable enough to form the basis of an automatic method of restructuring the network, but is proposed to give feedback as to the quality of the developed topology. The robot could incorporate this error function for directing its search process.

5.3.3 Network Error

One of the goals which has been defined for the robot, is the ability to be able to detect changes that have occurred in its environment. This is based on whether the robot's network information is still correct. This error could be manifested in the distribution of nodes (coverage), validity of its links (connectivity), or the values of range information that are possessed by the nodes. This last form of error measure can be used by the robot to detect

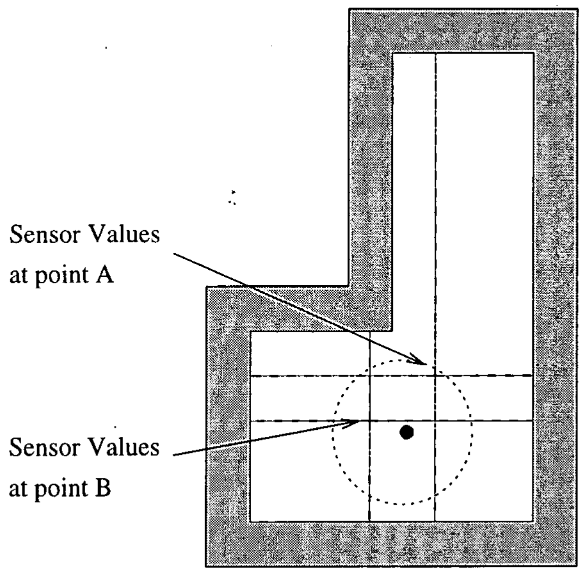


Figure 5.3: Discontinuous interaction between node and environment

changes which can influence both of the other sets of network information.

The network of nodes represents a sampling of the environment. Hence it is only possible accurately to assess the error in a node's information if the robot is guided to its exact location. This is a restrictive method, so some form of interpolation must be used, or a method of correction for localised position displacement from the node. The latter was chosen for the preliminary tests because of its computational ease. Each time the error value was to be calculated, the robot's sensor vector would be biased by a difference vector, which would then allow for a direct comparison to the nodal value.

The nodal error measure, described above, can be adversely affected by the discontinuous nature of interaction of sensor measures with the environment. Nodes contain information relevant to a region in the environment. The above method of error calculation assumes that the sensor values corresponding to the node are continuous over its domain. Figure 5.3 shows such a case where one of the sensor values changes dramatically within the node's domain. This erroneous value for the sensor will corrupt the error term derived for this node, indicating a network error and swamping the other sensor's information.

For the robot to be able to use the comparison of its stored and current sensor values to calculate any error in the estimated robot's position, elimination of erroneous sensor comparisons was carried out. This was achieved by rejecting from the calculation any error term which exceeded a threshold value. Hence large discontinuities would not swamp the valid small error terms of the other sensors.

5.4 The Robot Control System

The problem for the robot has been explicitly defined. The robot is expected to drive around, building up information about its environment. Once the robot has sufficiently mapped out the environment it should switch out of the learning phase. The robot would be expected to be able to navigate around the environment without collision. However, if the robot detects any change in the environment, it will be expected to update its knowledge of the environment. This goal can only be achieved in incremental stages from tested building blocks.

The simplest method of integrating the new mapping strategy into a robot control structure would be to use it with a robot wandering without using information from the network. A suitable control output could be created using a wander and avoid behaviour, which would, over time, move the robot around the environment in which it was placed. The network has no influence over the control of the robot but instead attempts to build up a map of the environment. The network always lays down nodes in a uniform manner, but the connections can be affected by the path that the robot has taken. In more confined spaces, the network produces more even topologies, where as in larger spaces the network is more distorted.

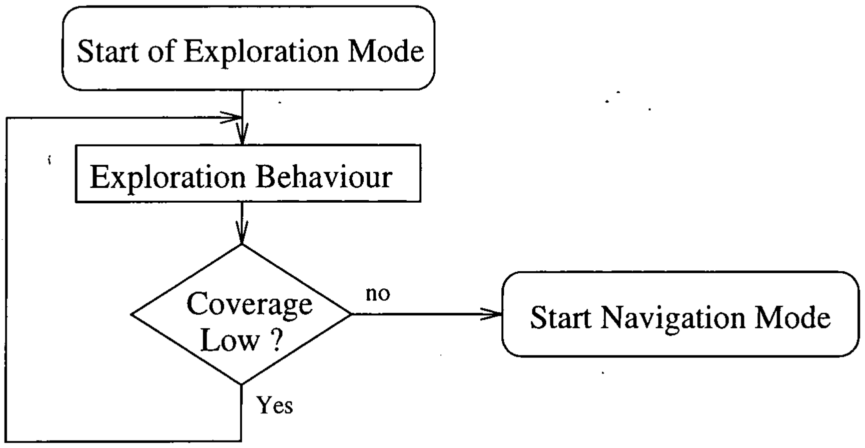
The main drawback of such a simple strategy, is that the robot has no method by which to monitor the development of the network which truly reflects its state. In order to increase the effectiveness of the robot's control method, feedback from the developing map should be used to direct further actions. The following describes one such method which uses the coverage property to direct the robot's actions.

5.4.1 A Navigation Strategy using Coverage

The coverage value gives the robot an indication as to the detected area of the environment which has been modelled. This can now be used to switch the robot's action between an exploration and an exploitation strategy. The exploration phase could be random or directed to search in area of less knowledge. The exploitation phase allows the robot to move within its environment utilising the map and thereby checking its validity.

Figure 5.4 shows in the form of a flow chart the sequential steps the robot must undertake. Initially the robot is configured so that it starts in the exploration mode. The robot is presumed to have no world knowledge at this point. The simplest form of exploration behaviour is the random wander, but more intelligent search processes could be used to reduce the time taken before the search is complete. It is assumed that once a sufficiently high value of coverage has been attained within the network, the robot has successfully explored its environment. Once this has been achieved, the robot then switches to the navigation phase.

The navigation process works by comparing the structure of the network to the robot's movements within the environment. Therefore the robot must attach itself or lock onto a



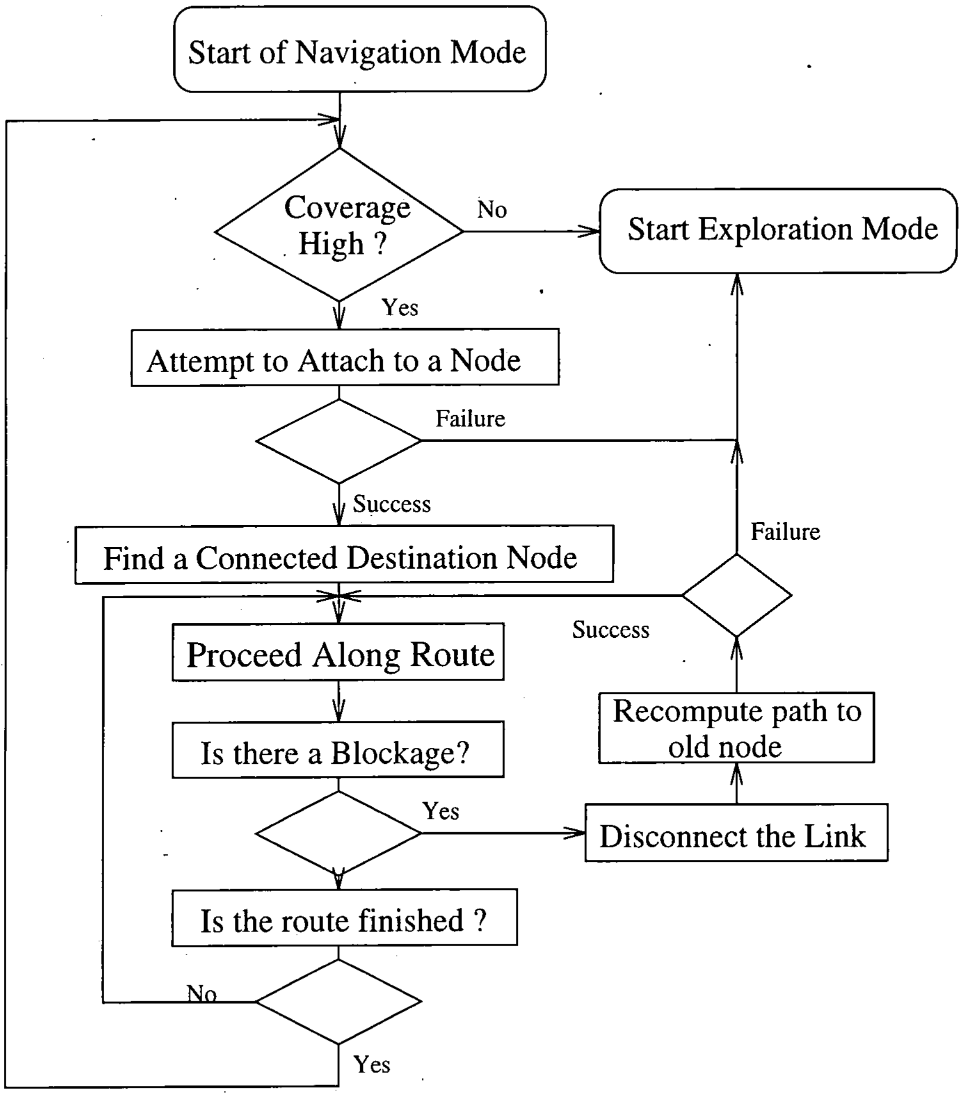


Figure 5.4: Flow diagrams for coverage control strategy

node. If this is not possible the robot returns to the exploration phase to move it away from the localised obstruction. Once attached, the robot can determine a destination node within the map. This could be selected on the bases of frequency of visitation, thereby directing its motion evenly around the network. Once the route to that node has been determined (this is performed by using the A* search algorithm), the robot can proceed along this path. If the robot successfully navigates to the destination node, the process is started again. However, if the robot encounters an obstacle along the route to the node, the network structure must be adapted by removal of this connection.

The obstacle is detected in this strategy by monitoring the velocity of the robot during the navigation. If this is detected to be zero for a period of time, then the robot is presumed to be stuck. The link that the robot is presently navigating down is removed, and a new path is then computed to the same destination goal as before. If this proves successful then the robot will continue its path towards the goal. However, if the network has been split into sub regions rendering navigation via the map impossible, then it reverts back to exploration.

One of the problems of this approach is that it relies only on coverage as a measure of the network's quality and completeness. However, the coverage parameter cannot give any information relating to the network structure or the errors stored by the nodes. The main problem is that topological connections are not sufficiently developed by the time the coverage parameter reaches its peak. Hence, areas of the map can be badly connected even though the robot is in its navigation phase.

Although the technique can cope to an extent with changes within the world, the robot does not actively search for errors. The robot makes no use of discrepancies between present sensor values and recorded values.

5.5 Simulation Results

The previous section has described in detail a novel method for the construction of networks which can be used for mapping an environment. This section reports on simulations which have been used to test these ideas, looking at potential problems and corrective actions that can be taken. In the first experiment we observe a robot using the new technique to build up useful maps in an unknown environment. A problem is reported for which a modification in the network linking is required, and a successful solution is described. The resulting network is used in conjunction with a graph searching algorithm, to direct the robot's actions within the environment. The robot control strategy which is used is shown to combat errors in the network, and able to relearn previously false information. Finally the control strategy is demonstrated in its ability to learn and solve a maze based problem.

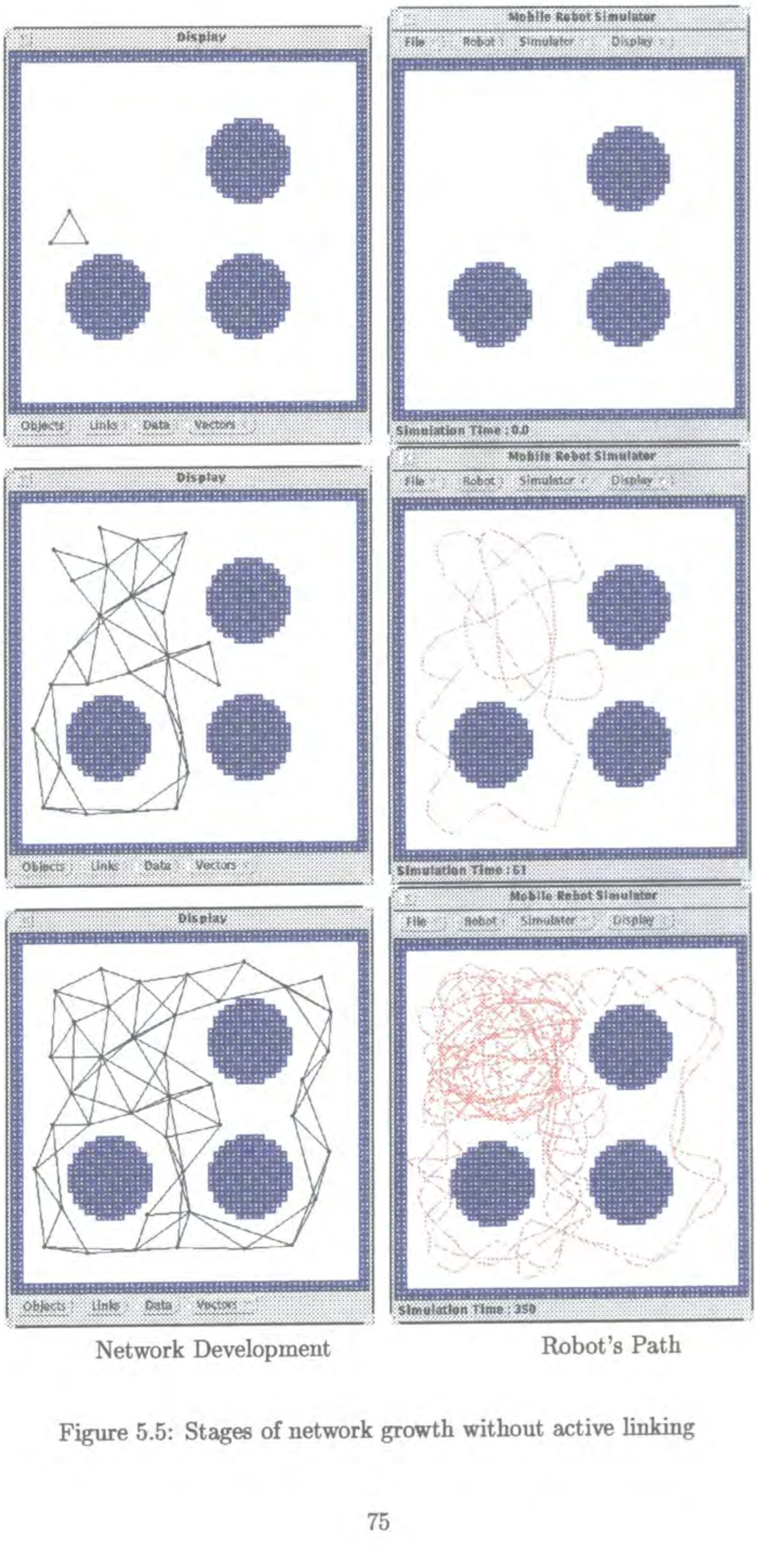
5.5.1 Basic Map Development

This first experiment looks at the application of the basic novel approach for the construction of a map. In this test the mapping module is not used to control the robot's motion but simply to build up information from observing the robot's sensors. The robot's motion is produced by the combined action of a random wander and obstacle avoidance behaviours. This same strategy was used in the previous tests on the GCS algorithm, and has been described in greater detail. The environment in which the robot has been tested has been designed to include straight edges, sharp corners, rounded objects, and is in nature asymmetric. Through the use of three circular objects the open space is broken up so that a mapping method must join up the disconnected arms of its developing map.

Figure 5.5 shows the development of the robot's spatial map as the robot explores the environment. The figure is split up into three time periods, each of which displays the network and the robot's path so far. The left hand windows show the robot's internal model of the environment by projecting the network into the environmental positions of each node. The right hand diagrams show the robot's path in the environment up to that moment in time. The first pair of diagrams show the start conditions with the robot's network being made up of one triangle with its centroid located at the initial position of the robot. The second pair of diagrams show the robot after it has moved within the environment for a period of sixty seconds. The map which has been produced covers entirely the region of the environment that has just been explored. This demonstrates the map's ability to create a relatively detailed topology from a single explorative path. It can also be seen that the map has joined up around the bottom left hand circular obstacle as the robot traversed around it in a clockwise direction. The link was created as the robot detected that there was no direct connection between two consecutive nodes which it had passed over. The final set of diagrams shows the network which has been created over three hundred and fifty seconds. Although the network that has been formed is accurate over the area that the robot has moved, it is obviously topologically incomplete because of the absence of a link between the two right hand circles. This is arguably a problem for the robot's directional control, but in this simplest case there is no feedback from the network to direct the robot's motion. However, the networks data contains a discrepancy which can be used to rectify the problem.

5.5.2 Map Development With Active Linking

As described in the previous section on the theory of the novel approach, the environmental connectivity can be used to get a comparative measure of the topological, as compared to direct distance. This can be used selectively to connect up nodes that are physically located within close range, but have long path lengths through the map. This assessment of nodal connectivity was undertaken at periodic intervals of fifty seconds of simulation time.



Network Development Robot's Path

Figure 5.5: Stages of network growth without active linking

The assessment process examines each node in turn, and for a given node, the range measures that are directed radially outwards are compared to the positions of known nodes. If the nodes are within the range measures scan, the network and topological distances can be computed. Given that there exists a large enough discrepancy between the two measures, and the node is physically close (within a distance of three delta lengths), a link is added.

Figure 5.6 shows this linking mechanism applied to that same robot path as before. It is obvious from the last frame that this method has been able to create a more accurate map than that produced by the original method. It can be easily seen that network has developed a correct link between two separate regions even though the robot has not directly explored this. If the rest of the topology is examined there are some other differences. One of the problems associated with network re–linking are the chances of introducing an illegal link. An example of this can be seen in the bottom left hand corner of the network, where a link cuts across the circular object. This particular error is minor and, as will be shown, can be removed by the robot checking the validity of the network later.

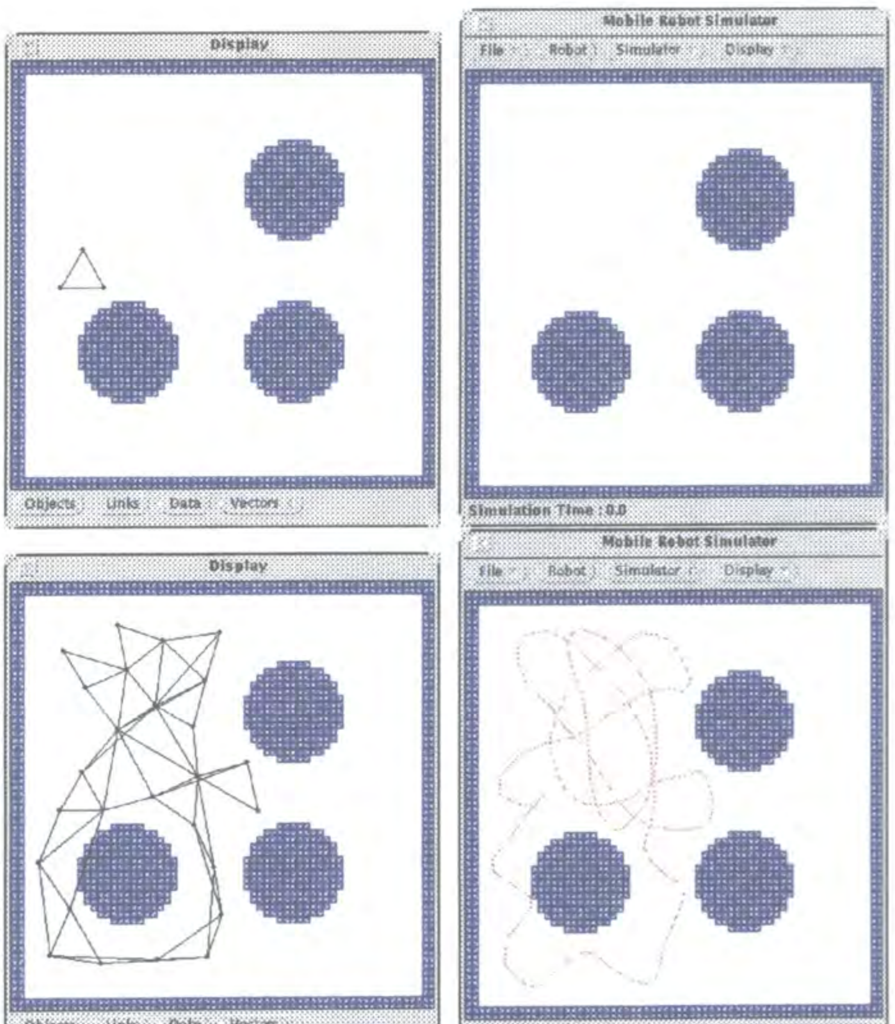
Linking has been shown to offer a mechanism by which the robot can use the information stored so far to rectify discrepancies within the map. In this simplest case where the robot cannot use the information of the network to direct the robot's path, the direct alteration of the network is required. However, as this problem can also be considered as part of exploration, the discrepancy information could be used to direct the robot's motion. This remains a possible area of research if topological discrepancies become a problem in further work.

Although this method has been shown to be beneficial in the above example, there must be reservations for its use as the basis of network construction. This stems from the interpretation of the sensor information which is used to develop hard links. Any noise in the sensors data, coupled with misalignment of the growing network could lead to unstable network construction. In such situations, it is better to use the information for exploration. Errors can be checked against the environment, rather than potentially error prone network data. The extent of these problems will only become apparent when tests using noisy information are performed.

5.5.3 Navigation using the Constructed Map

Shown in figure 5.7 is the map built up using the active linking mechanism during the construction phase. The map is now to be used to direct and guide the path of the mobile robot. The robot starts its movement from the top right of the window and the diagrams show the robot planning and driving to two sequentially selected goal locations.

In the first diagram the robot starts from the top right and is required to plan a path to the bottom left. For demonstration purposes this was selected by the user, but destination



Network Development

Robot's Path

Figure 5.6: Stages of network growth with active linking

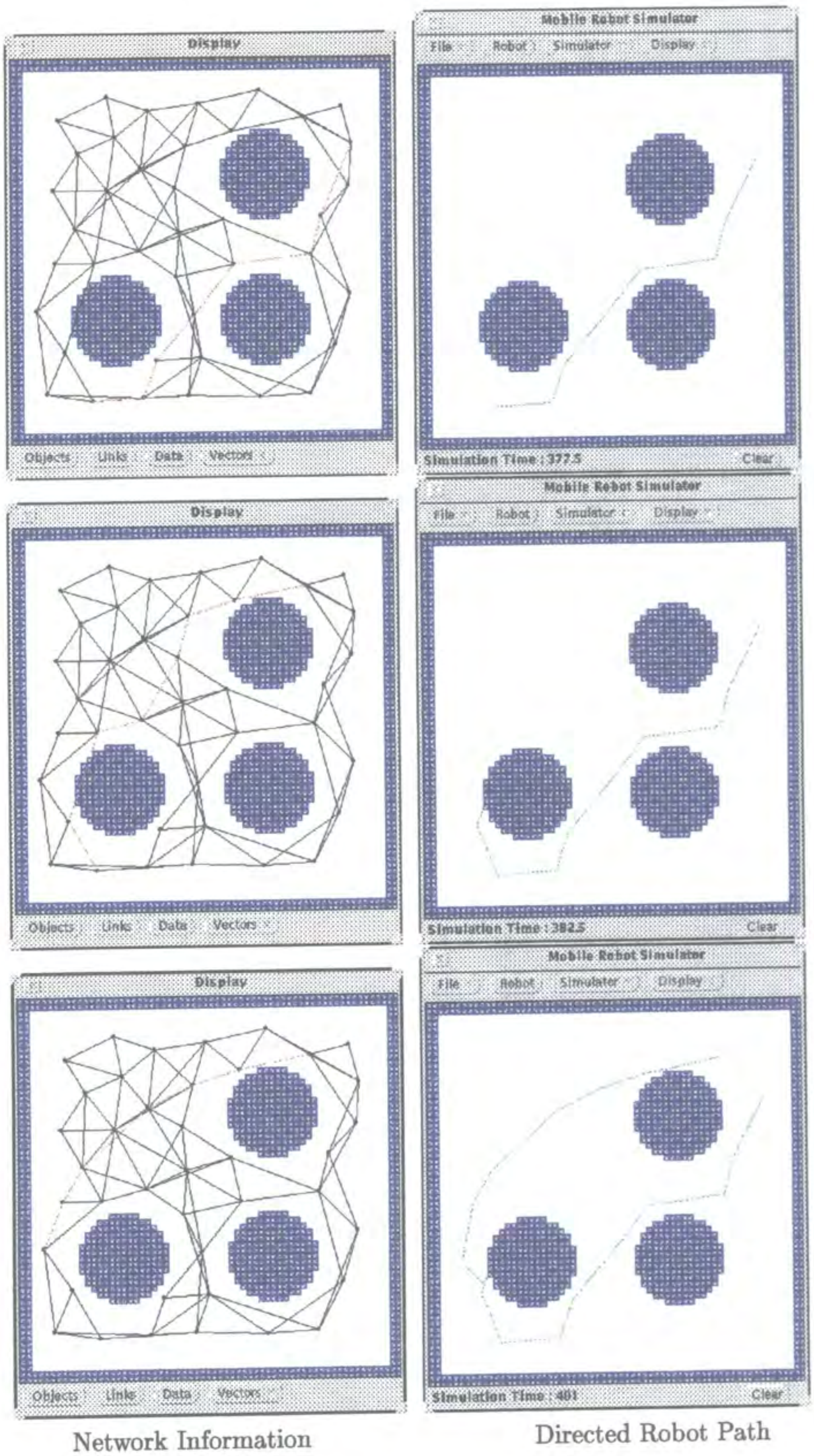


Figure 5.7: Path planning and corresponding robot motion

nodes are more normally computed by the robot. As the start and end nodes are known, the robot can use the A* algorithm on its network to produce a shortest path which is shown in the top left hand diagram. The guidance system then uses the positional information of the nodes to enable it to move along this course. This loop of actions has been reported on in the previous section. It can be seen from the diagram that the robot has achieved collision free navigation in the environment.

After the robot has traversed the first path, it is then requested to drive back to a new node which is close to the original start location. This will demonstrate that the path planning algorithm can produce new shortest routes and that the system can cope and correct topological errors within a network. The planned path to the new goal node can be seen to pass through a link which will force the robot against an obstacle, and as the path is traversed, this is what happens. When the robot has collided, the control system senses no motion and removes this illegal link. The robot then computes its path to the goal again, and the result is a successful movement combined with a corrected map.

5.5.4 Map Construction and Navigation within a Maze

So that the network development and the use of the A* algorithm could be more extensively tested another test was designed. The previous test environment contains open spaces and is not confined, so a maze was chosen to test the method in more restrictive circumstances.

The search method used, incorporated some of the information produced by the coverage algorithm. This information is in the form of a vector that can be computed for each node and gives an indication of the location around the node where there is least coverage. This was used to encourage the robot to continue its movement into unexplored locations.

The first pair of diagrams in figure 5.8 show the robot building up the network map which it will then used to navigate. In this example the environmental width was eight meters, and the robot's value for delta was set to forty centimetres. The time taken for the robot to build up a complete map of the maze was 350 seconds, after which the robot could then use the map for navigation.

The second set of diagrams show the robot traversing the network using the built up map. The robot in this example starts in the right hand side of the maze and is requested to move to the bottom left location. The A* algorithm computes the shortest route and the robot simply drives down that path.

The ability of this novel approach to create a useful map within a confined environment is demonstrated. Here the ability of the system to construct a map in a unknown location and subsequently use that map to produce collision free navigation is demonstrated. The robot control system is shown to be resistant to limited errors within the network and be able to correct them.

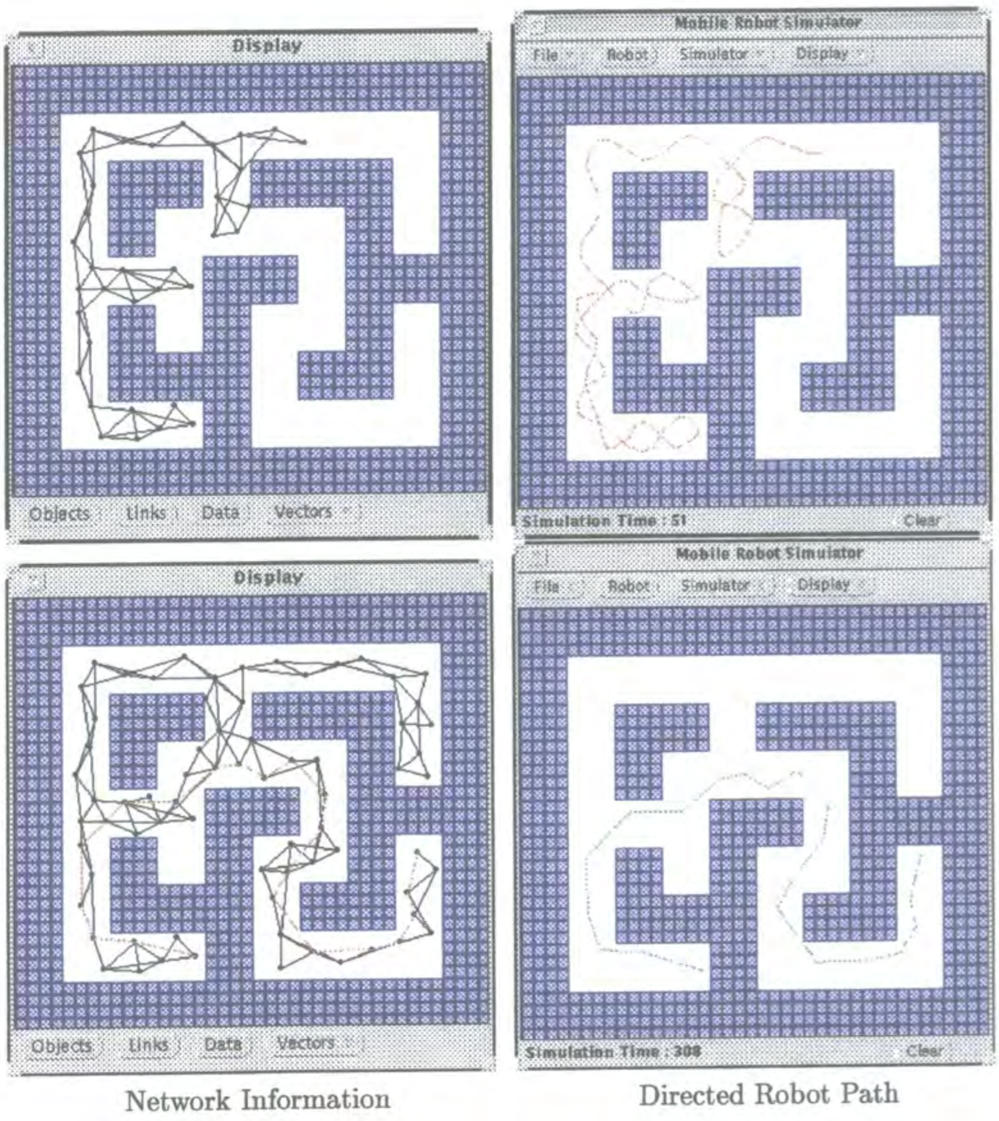


Figure 5.8: Path planning and corresponding robot motion in a maze

5.5.5 Conclusions on the Novel Mapping Method

The novel method, in comparison to the GCS, is embedded more directly in the environment. The creation of nodes is dependent not on number of training samples, but on the spatial position of the data values. At the creation of a new node its data is taken directly from the sensor values recorded at that position rather than being interpolated from the network. The fixed nature of the nodes and the connections between them limit the topological errors that can be introduced later.

In all of the experiments the network had been completely developed within only 350 seconds corresponding to just 700 samples. Another advantage is that at each stage in the creation of the map the developed network is correct and can be used for navigational purposes.

One of the problems associated with the exploration of more complex mazes is directing the robot to the unexplored regions. In this maze experiment a simple form of direction was used to drive the robot when it was in the location of an unexplored region. This could be integrated in a control strategy with the fact that the partial map is navigable. The controller would navigate the robot to the edge of an unexplored area where the coverage vector could be used to move the robot in the correct direction for searching. This would then be an efficient search strategy.

5.6 Summary of the Novel Technique

A novel method is proposed to form the basis of a mapping system for a mobile robot. This method is shown accurately to map different environments with far less algorithmic complexity than the growing cell structure. The system is shown to be capable of exploring an environment and creating a topological map from the information. This map is then used by the navigation module for collision free navigation. The method is shown to be resistant to minor topological faults, being able to correct these errors.

A method of monitoring the network's coverage of the environment has been described, which produces a vector which can be used to direct the robot search. The vector is used in conjunction with the wander and avoid behaviours to increase the sophistication of the search process. This method has been successfully employed in a maze, with the robot mapping out the entire area. The robot was then able to plan paths and navigate to any desired location.

5.7 Summary of Simulation Work

A control strategy capable of generating topological maps and then subsequently using them for navigation has been shown to work in simulation. However, these simulations did not include accumulative errors that develop in odometric measurement systems. Neither were the errors that are inherent to ultrasonic range measurement incorporated. Both these types of errors are hard to accurately model, as they develop from the complex interaction of the robot with its surroundings.

The simulation work has provided a productive way of building up the control algorithms which can be used on a mobile robot. However, for further insights to be gained, the complexity of the modelling of the simulator must be increased. This presents problems not only of accurately mimicking the interaction of the robot with its surroundings, but also of computational time for each simulation. It is suggested this approach will give diminishing returns for the inputted research effort.

In order to continue the design of the control algorithms, the systems were then ported to a real mobile robot. Solutions to mechanical, computational and algorithmic problems will need to be developed. The design of the robot and its physical construction are reported in the next chapter.

Chapter 6

The Robot Design

6.1 Introduction

There are many benefits to be gained from building a prototype robot to test out the control designs ported from simulation. Such real test situations offer greater challenges to control systems and allow for validation of the simulation work. It is always possible to construct new environments which push the limits of the robot's abilities, forcing the need for better robotics systems. Finally, only real systems present the best objective situations for researchers empirically to compare their designs.

This chapter describes the design of a prototype mobile robot and includes the considerations taken to achieve it. To facilitate a greater understanding of the various systems and their interactions, the chapter has been subdivided. Initially, the mobile robot as a working entity is described. Following this overview, the design is subdivided to reflect the different aspects brought together in the complete robot. The first section details the mechanical system with the design considerations that were taken. Next, the electronic systems are examined. This section is split into modules, each examining in detail particular subsystems. The last section describes the embedded computer system.

6.2 Global Overview

The design goal is to build an autonomous robot which is as compact as possible with sufficient sensory and computational abilities. There is a balance in the level of complexity of the control systems against the larger physical size of robot which would be required to house them. This can only be achieved by determining the level of complexity and then proceeding to design the most compact robot around this.

The requirements for the robot's control system are an extension of the work that was undertaken in simulation. Of primary importance is that the robot be mobile and autonomous.

This means that all the power and computation requirements for the full duration of any robot run must be self contained. The physical frame of the robot must also have room for mounting the sensors and be able to have sufficient power to supply these and the motor drive systems.

After considering all these factors, which are described in their respective sections in more detail, it was decided that the robot should be of a circular design with a diameter of no more than 40 centimetres. The circular shape means that it has exactly the same manoeuvrability in all directions. It was also decided that the robot should have the potential to move at a reasonable speed. However it is important not to have too powerful a drive system in an experimental mobile robot which must be resistant to control failure. It was therefore decided that a speed which related to a robot length per second would be sufficient. Hence the top speed designed for initially was 40 cm/s. Figure 6.1 shows the photographs of the completed mobile robot taken from the side and above.

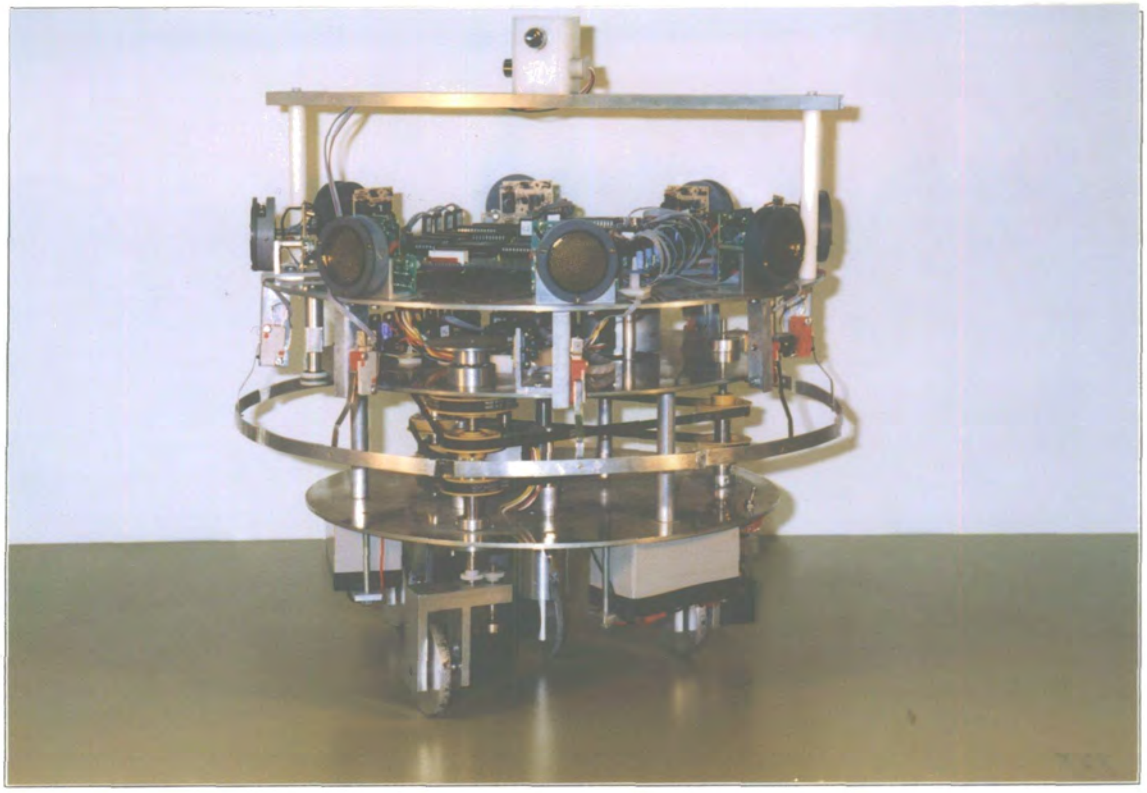
To simplify the description of the robot it is useful to split the design into three broad subsections of mechanical, electronic and computational systems. Figure 6.2 graphically illustrates the interaction of these systems in the working mobile robot. Each subsystem is at a different level of abstraction from the environment and therefore has different design requirements.

The mechanical section covers the basic design of the physical robot frame. This includes the mechanical linkages for the robot's actuators, their dynamics and the positioning of all the robot's sensors. The positioning of the batteries and the circuit boards are also included. The electronics section examines the interface hardware needed to implement the sensor systems, however this does include some low level computational routines. Finally the computational section details the computational aspects of the design.

6.3 The Mechanical Chassis

There are many different mechanical chassis configurations which can be used for a mobile robot drive system. These have already been outlined in Chapter 3 in section 3.5.2. The design selected as being the most preferable for the robot's chassis was the synchro–drive. This was due primarily to the fixed orientation of the robot's frame irrespective of the direction of travel. There are also benefits for the accuracy of odometry with reduced wheel slippage. Overall the advantages of this system outweigh the greater mechanical complexity required to construct a working system.

In the synchro–drive configuration three or more wheels are used to propel the robot. The orientation or direction of forward motion of all the wheels, which are mechanically linked, are in parallel. This synchronism in their directions gives rise to the configuration's name and means that the robot can move in any direction without changing the orientation



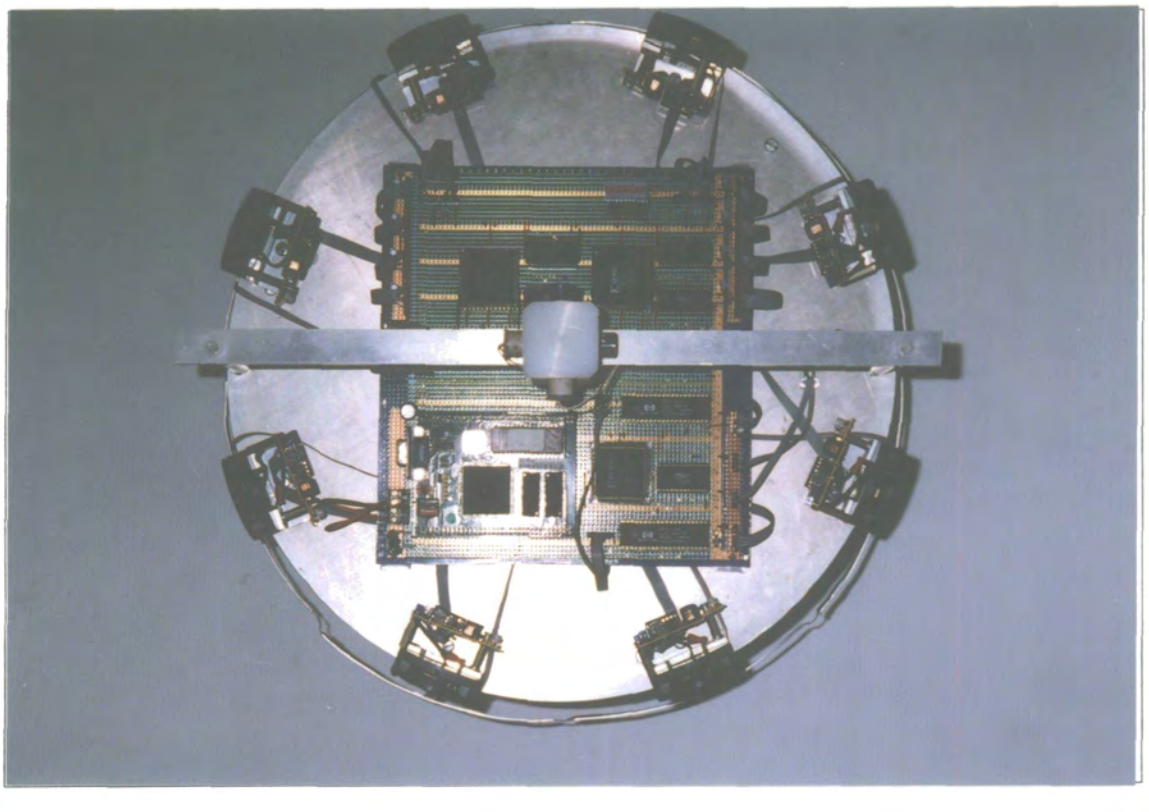


Figure 6.1: Photographs of the mobile robot

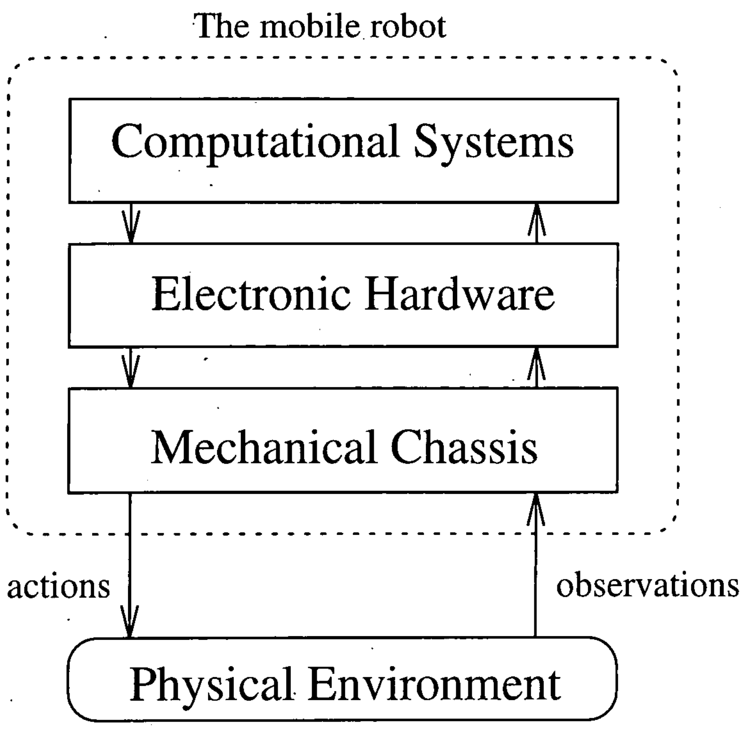


Figure 6.2: Interactions of the component subsystems

of the chassis. This has the major advantage that any sensor mounted on the chassis does not have to correct for the direction of robot travel as in differential drive systems. The forward rotation of each of the wheels is also mechanically linked and this provides a more even propulsion which reduces the effects of wheel slippage. The particular form by which this synchro-drive has been implemented is now described in greater detail.

Figure 6.3 shows the complete robot as viewed from the side, a height of about 40 centimetres. The base of the chassis consists of the wheel assemblies and the battery housings. Between the lower and the upper fixing plates, the drive and steering belt transmissions are housed. The design uses a head assembly which can be rotated, allowing greater sensory observation even though the robot frame's orientation is fixed. Apart from the encoders for positional and angular feedback, all the sensors are mounted on this head assembly. This includes the ultrasonic range detectors, the collision switches, and the magnetic field sensor. This magnetic field sensor is mounted on a bridge to keep it physically far from any onboard magnetic fields.

At the base of the robot there are four wheels in contact with the ground. Three of these are used for propulsion and are equally spaced out, forming a triangular base. The fourth is mounted in the centre and provides odometric information. Although many mobile systems take information from encoders mounted on the same wheels that produce the motion, this configuration cannot detect slippage. Such slippage could be caused by the robot over

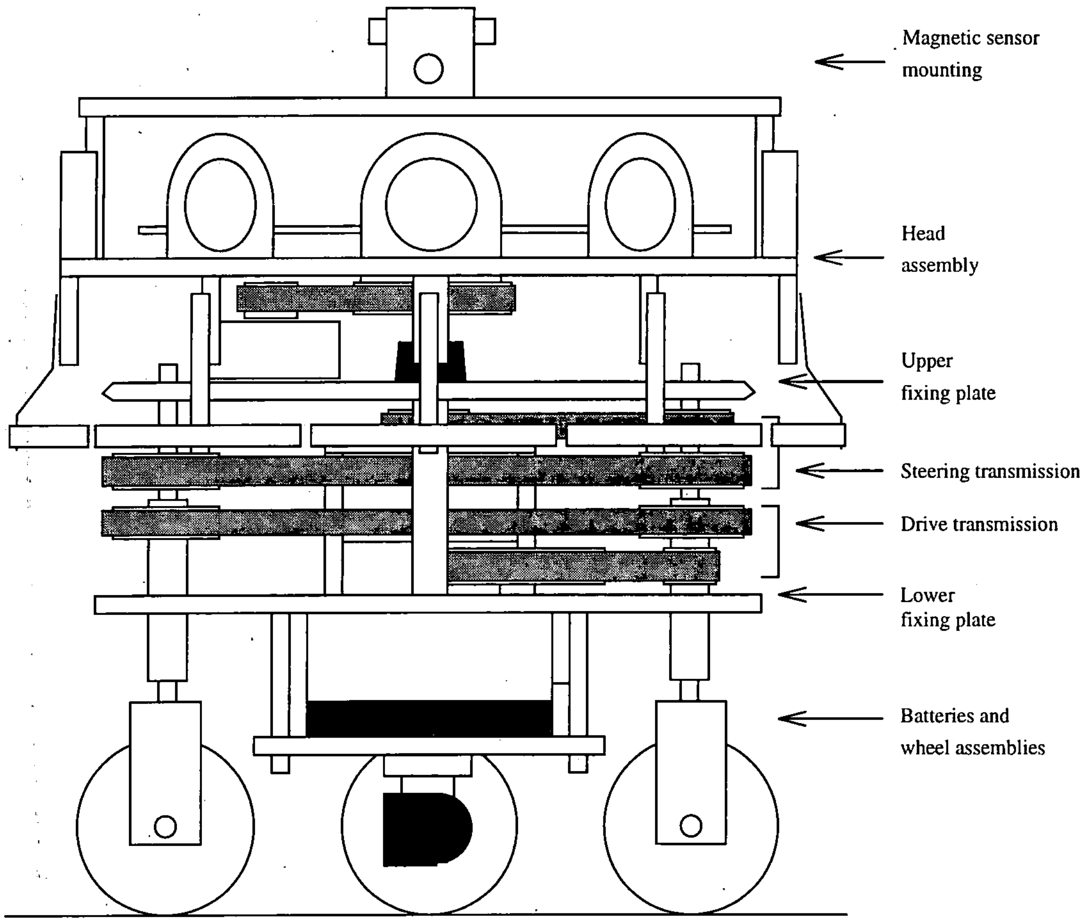


Figure 6.3: The robot chassis

accelerating or decelerating, or by driving the wheels when motion is impeded. This is a new design for a synchro–drive configuration in that it uses the synchronous turning of the wheels to be able to turn a fourth central wheel to point in the same direction of travel as the robot. By mounting an encoder directly on this wheel, only one encoder is required to determine the forward and backward odometric distances. As only one encoder is used this can be used to control the drive motor in a feedback loop which then takes into account the actual position of the robot on the ground.

There are three lead acid 12 volt batteries used for the on board power supply. These are mounted around the periphery of the lower fixing plate to increase the stability of the design. Between the two fixing plates, there are the main pulley systems for the drive and the steering transmissions. The lower set of belts are linked to the drive motor mounted beneath the fixing plate, and the upper belts are linked to the steering motor mounted between the fixing plates. Above the upper fixing plate is the head assembly. This is mounted on the upper fixing plate by an external slide ring that has been cut into the edge of the plate. The head assembly is then mounted on posts which run on guide wheels. The head assembly is rotated by a motor with a belt transmission onto a centrally mounted pulley on the head.

The three drive wheel assemblies have the same basic design, which consists of an inner shaft fixed to the wheel assembly and an outer shaft which drives the wheel's forward motion. Figure 6.4 shows the drive shaft (e), and the steering shaft (c) in greater detail. In all these figures, light grey moving parts indicate a linkage to the drive system, whereas darker grey signifies linkage to the steering system. The top of the steering shaft spins in a bearing mounted in the upper fixing plate (a), and is rotated by means of a pulley (b) that is driven by a belt (d). This is a notched timing belt which does not slip over the pulley so that all the wheels will remained aligned after they have been correctly set–up. The outer shaft is fixed with respect to the lower plate by bearings (f). This allows the pulley on the inner shaft to be mounted close to the top of the outer shaft therefore stopping this inner shaft from falling out of the upper bearing. The rotation from the outer shaft is transmitted to the wheel by two gears (g) which drive a 90 degree gearbox (j) which in turn drives the wheel (i). The outer gear (g) must be fitted with bearings so that it smoothly rotates rather than being tangentially thrust into the wheel housing (h). The wheel itself is mounted with a bearing to reduce frictional losses.

The wheel is mounted offset from the central axes of the shafts to reduce friction when turning. Instead of the wheel spinning on its own vertical axis, it describes a circle around the shaft's central axis. During a stationary turning manoeuvre, the inner shaft rotates but the outer shaft remains fixed. This causes the drive transmission gears (g) to act as a planetary gear system, causing the wheel to rotate. As this gearing ratio is 1:1 this leads to the wheel rotating forward once for each angular revolution. Hence, the wheel must be positioned with its own radius offset from the shaft's axes for it to describe a circle centred on these axes.

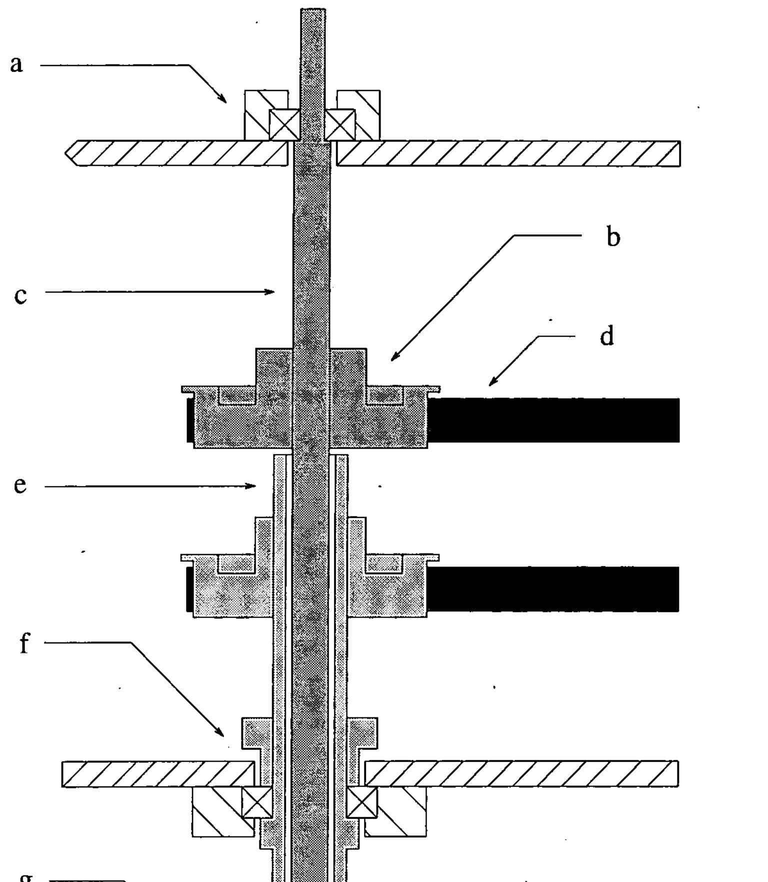


Figure 6.4: The drive wheel assembly

This assembly has the advantage that when the wheels are turned, the drive pulleys are not forced to move and therefore require no clutch. However, when the wheel is providing a propulsive force it creates a torque on the steering shaft because it is mounted at an offset. This must be countered by a holding torque applied to the steering shaft to stop the wheels being forced off course.

The central wheel is for the odometric readings and is angularly synchronised to the other wheels. Figure 6.5 shows the mechanism in greater detail. Unlike the other wheel assemblies, the odometry wheel has only one shaft, which is hollow (d). This is secured by two bearings, one in the upper plate (b), and the other in the lower plate (e). The assembly is rotated by a pulley (c) linked to one of the three drive wheels steering shafts. At the bottom of the shaft is the main housing (f) that allows only vertical movement of the odometry assembly (h). This assembly is sprung mounted so that the odometric wheel remains in contact with the ground at all times. Two pins which run in a slot in the housing (g) have been screwed into this assembly to prevent it from rotating in the housing. They also limit its vertical travel length, and thus prevent it from falling out. The assembly provides a mount for a thin aluminium wheel (i) which has bearings and a rotary encoder mounted on it (j). The electrical connections from this encoder are routed through the hollow shaft to appear at the top of the upper plate (a). The odometric wheel unlike the drive wheels is not mounted to the side of the shaft's axis. This is because during a stationary turn of the wheel assemblies, the odometry wheel must not register any forward motion.

In order to describe both the drive and the steering transmission systems, the robot is sectioned and viewed from above. First the lower section which provides the linkages between the drive motor and the drive wheel assemblies shall be described, and this is shown in figure 6.6. There are three main fixing pillars between the upper and lower plates to keep them in line, and beneath these are mounted the three battery packs. By mounting these low down and far apart on the robot, this not only lowers its centre of gravity, but increases the inertial resistance to it toppling over.

A DC motor was selected to provide the propulsive force for the robot's drive system. DC motors offer low weight for their mechanical power output, and can be easily controlled to provide variable speeds. A 12 volt medium duty DC motor was selected with in integral gearbox providing a rated output of 60 r.p.m. at 300 mNm torque. With a gearing up ratio of 1:2 in the transmission to the 3 cm radii wheels, the velocity of the robot at rated load would be 0.377 ms^{-1}. To work out the acceleration time taken for the robot to reach its maximum velocity, the work done to accelerate the robot's body (5 kg) to this speed, must be deduced using equation 6.1. The motor power can be calculated from the motor's rated load torque using equation 6.2. For this preliminary assessment of the motor's capability, it is assumed that the developed motor power is constant, independent of speed. From this value, the time taken to accelerate the robot to its maximum speed is calculated from equation 6.3. This

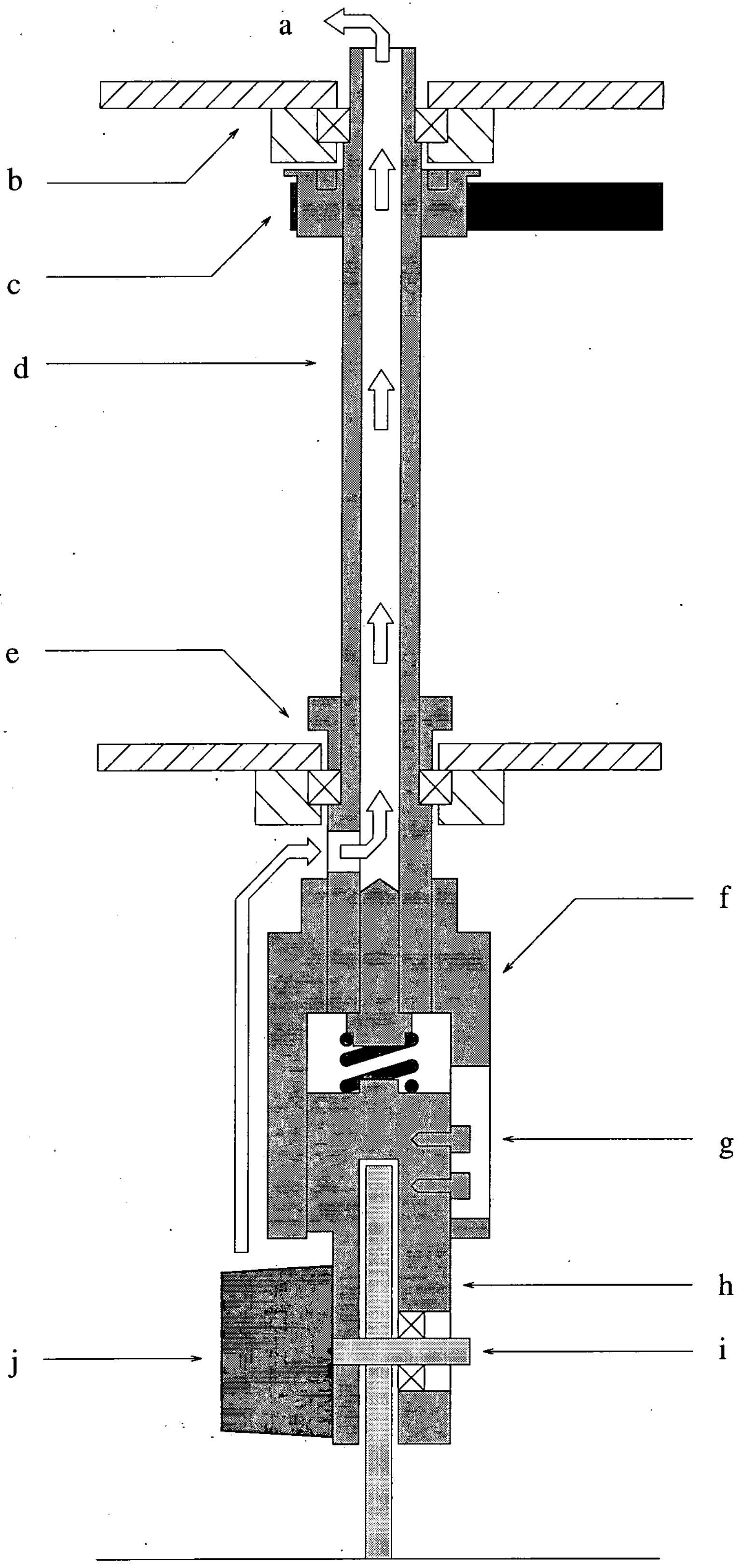


Figure 6.5: The odometry wheel assembly

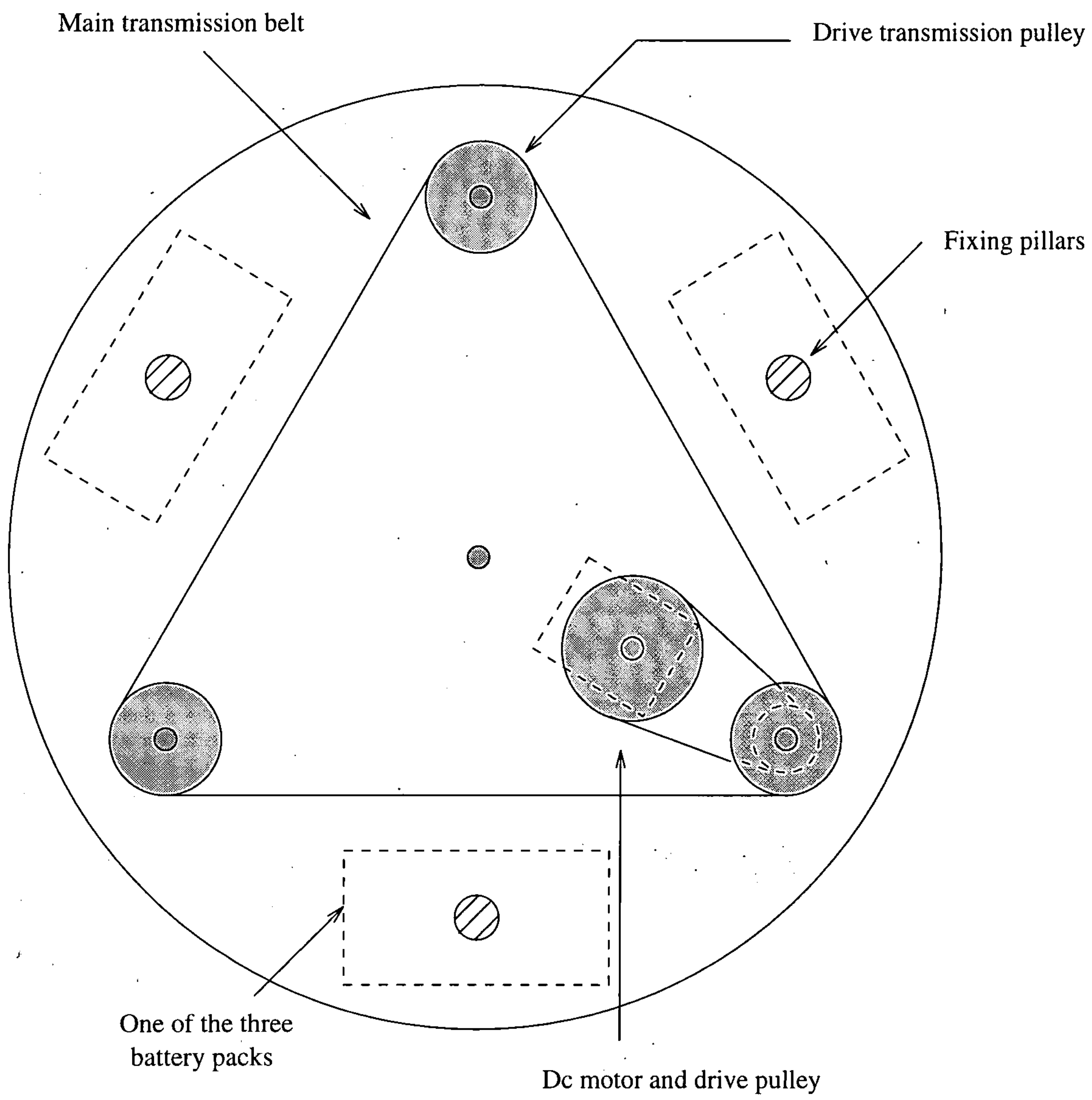

Main transmission belt
Drive transmission pulley
Fixing pillars
One of the three battery packs
Dc motor and drive pulley

Figure 6.6: The drive transmission

gearing ratio and motor selection gives a maximum robot speed of 38 centimetres a second and an acceleration time to this speed of under a fifth of a second. This motor selection provides adequate performance in both maximum velocity and acceleration times.

$$\begin{aligned} \text{Work Done} &= \frac{1}{2}mv^2 \\ &= 0.355 \text{ (Joules)} \end{aligned} \tag{6.1}$$

$$\begin{aligned} \text{Motor Power} &= T\omega \text{ (at rated load)} \\ &= 1.884 \text{ (Watts)} \end{aligned} \tag{6.2}$$

$$\begin{aligned} \text{Acceleration time} &= \frac{\text{Work Done}}{\text{Motor Power}} \\ &= 0.19 \text{ (Seconds)} \end{aligned} \tag{6.3}$$

From the above analysis it can be seen that the motor must be geared up to drive the wheels. This is achieved by using a motor to drive shaft pulley selection of 80 teeth to 40 teeth respectively. The transmission gearing ratio form the drive shaft to the wheels is 1:1, which satisfies the gearing requirements. Finally, a large timing belt is used to transmit the drive force equally to the drive pulleys of each of the wheel assemblies. The two timing belts can be seen in figure 6.6. The drive linking the motor to the drive shaft is in the bottom left hand corner and the main transmission belt forms a triangular loop around the three drive pulleys.

The steering transmission system is similar to the drive transmission. The three pulleys, each mounted on the inner shafts of the wheel assemblies, are linked by a large timing belt to provide a turning force (see figure 6.7). When this system is set–up, each of the pulleys must be adjusted so that all the wheels point in the same direction. There is also a pulley belt link to the central odometric wheel so that its angular movement is linked to the rest of the wheels. A stepper motor was chosen to provide the motive power for the orientation of the wheels. The primary reason for this choice is that a stepper motor has a holding torque which resists motion. As has been described above, when the robot moves forward it generates a torque on the steering shaft which must be resisted, otherwise the wheels would change direction. Therefore, the use of a stepper motor has a dual purpose, to provide this holding torque whilst the motor is in forward motion, and to orient the wheels.

Different stepper motors have different step angles. This is the change in rotor's angle caused by advancing the excitation phase sequence. The stepper motor chosen had a step angle of 7.5 degrees and gearing down ratio of 7.5:1 was selected, so that of each step of the motor the wheels would move by 1 degree. The strength of a stepper motor is defined in terms

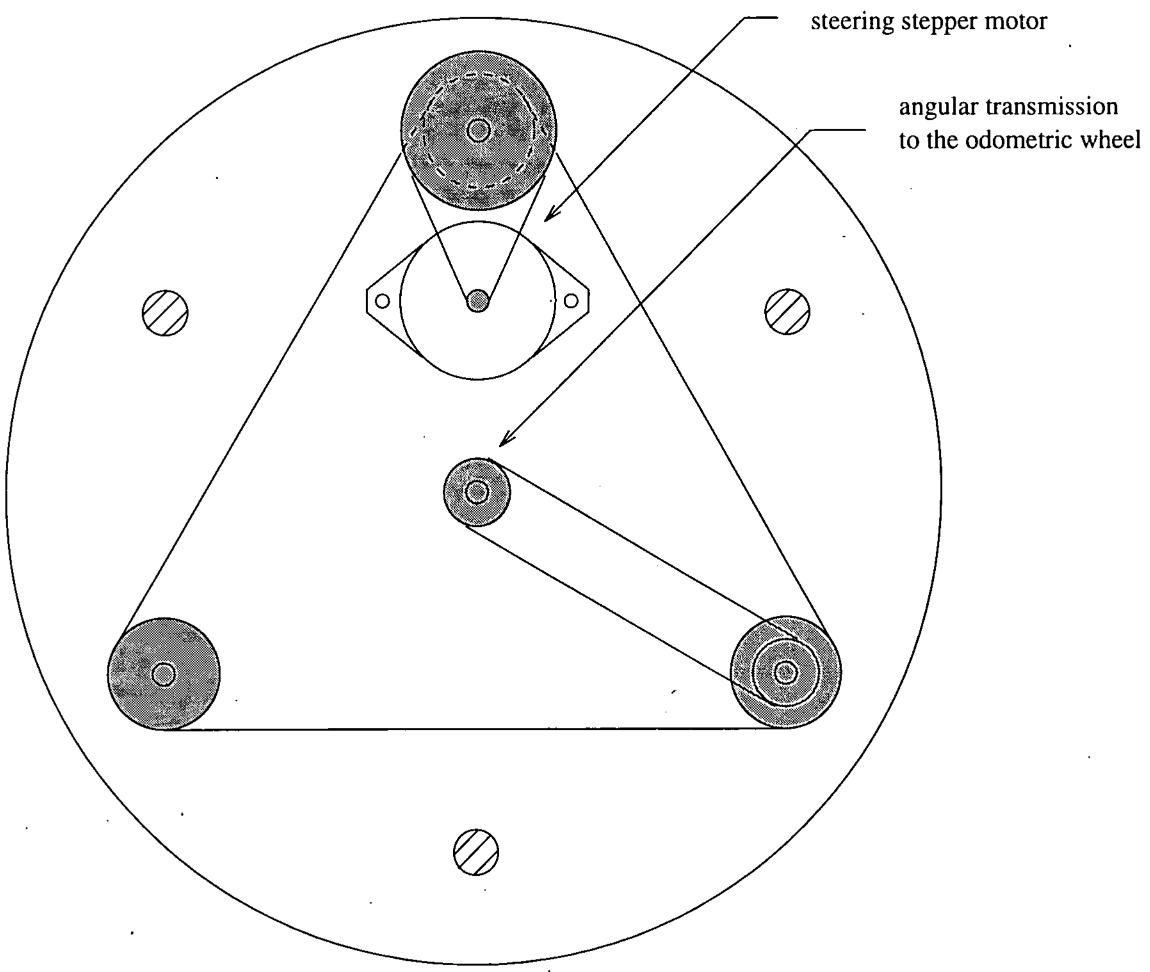


Figure 6.7: The steering transmission

of its holding torque. This is the maximum static value of torque that the excited motor can resist. From this value it is possible to determine the 'pull out' torque, or the maximum torque that the motor can develop. For low inertial, high friction systems, the maximum 'pull out' torque has been documented at 0.71 times the holding torque of the motor [43]. It was empirically determined that the maximum torque required by the steering system from the motor, was 57 mNm. This turning force significantly depended on the surface that the robot was on, for smooth floors the values were much reduced. The value of 57 mNm was obtained from a test on a floor tile similar to most office environments. A stepper motor was therefore chosen that had a holding torque of 155 mNm which gives a 'pull out' torque of 110 mNm ultimately giving a factor of about 2 safety. The stepper motor is mounted on the lower fixing plate, however, a 360 increment encoder is mounted in the upper fixing plate and is directly connected to the stepper motor's shaft. This produces information as to the orientation of the wheels.

Figure 6.8 shows the mounting on top of the upper fixed plate, for the head assembly. The outer edge of the upper fixing plate has been bevelled to form two slopes of 45 degrees to the horizontal from both the top and the bottom of the plate. This forms an external slide ring on which guide wheels, or journal bearings contact. There are three such bearings mounted on posts on the head assembly by which it is supported. One of these posts is eccentrically fixed to allow for minor adjustments and the removal of the head. On another of the posts is mounted a small horizontal plate. Near the edge of the rim is mounted an infrared (IR) emitter receiver pair, whose beam the horizontal plate interrupts when at the limit of the head's rotation. This allows the head to orientate itself with respect to the lower robot chassis. Mounted on the opposite side of the IR sensor housing is another emitter receiver pair which monitors the wheel angle index. This disk is attached to the inner shaft of one of the wheel assemblies that protrudes through the upper fixing plate. This sensor can therefore be used to determine the angle of the wheels with respect to the lower robot chassis. The head plate is driven round by a stepper motor identical to that used for the steering transmission. The gearing down ratio is the largest that can be achieved with the range of pulleys used and this was 10:1. Hence the head can moved in increments of 0.75 degrees. The pulley attached to the head assembly was fixed by bolts in order to form a central hole through it. This allows electrical connections to be routed from the head assembly to the lower levels. The power and drive circuit board is mounted on the upper fixing plate and contains all the power electronics for the drive systems as well as the power supplies for the control circuit board and the ultrasonic sensors. This physical separation of the control and the power electronics reduces problems of electrical noise.

The head assembly is the main location for the robot's sensory systems and it computational electronics which is shown in figure 6.9. Both systems are quite extensively electrically interconnected and it is therefore simplest to mount them together. The aluminium disk of which the head is constructed also provides some shielding from the power systems below.

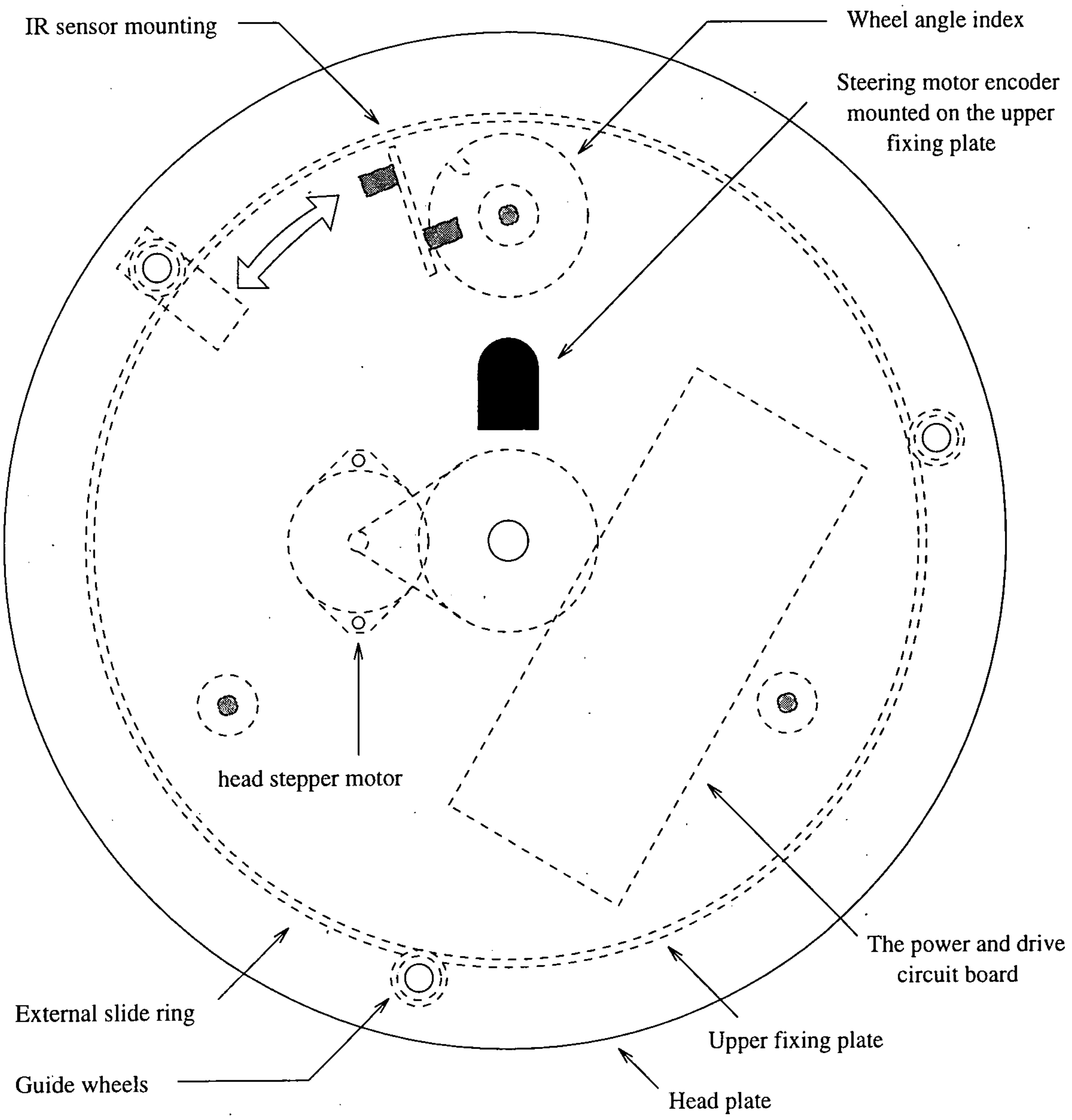


Figure 6.8: The mounting for the head assembly

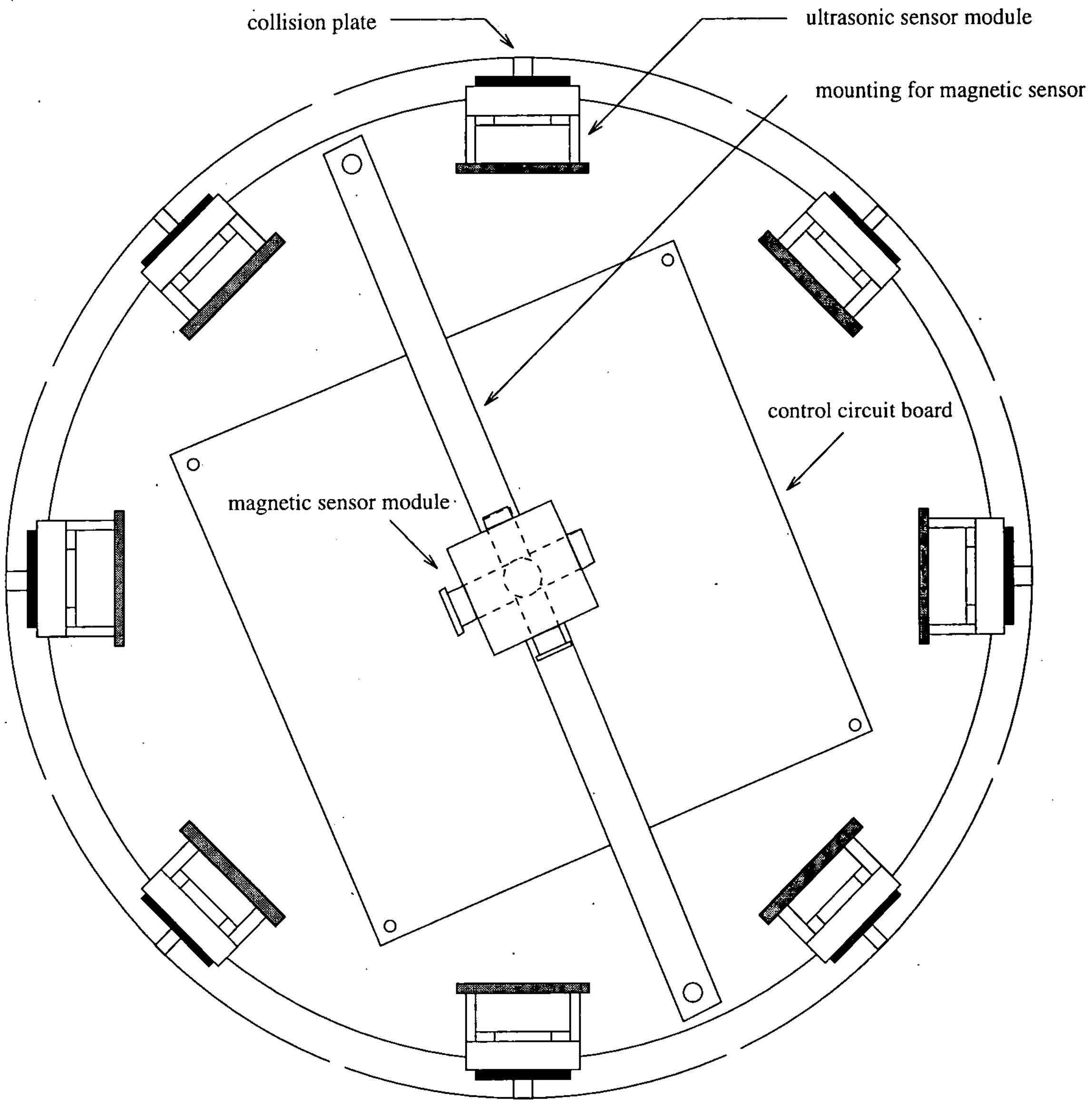


Figure 6.9: The head assembly

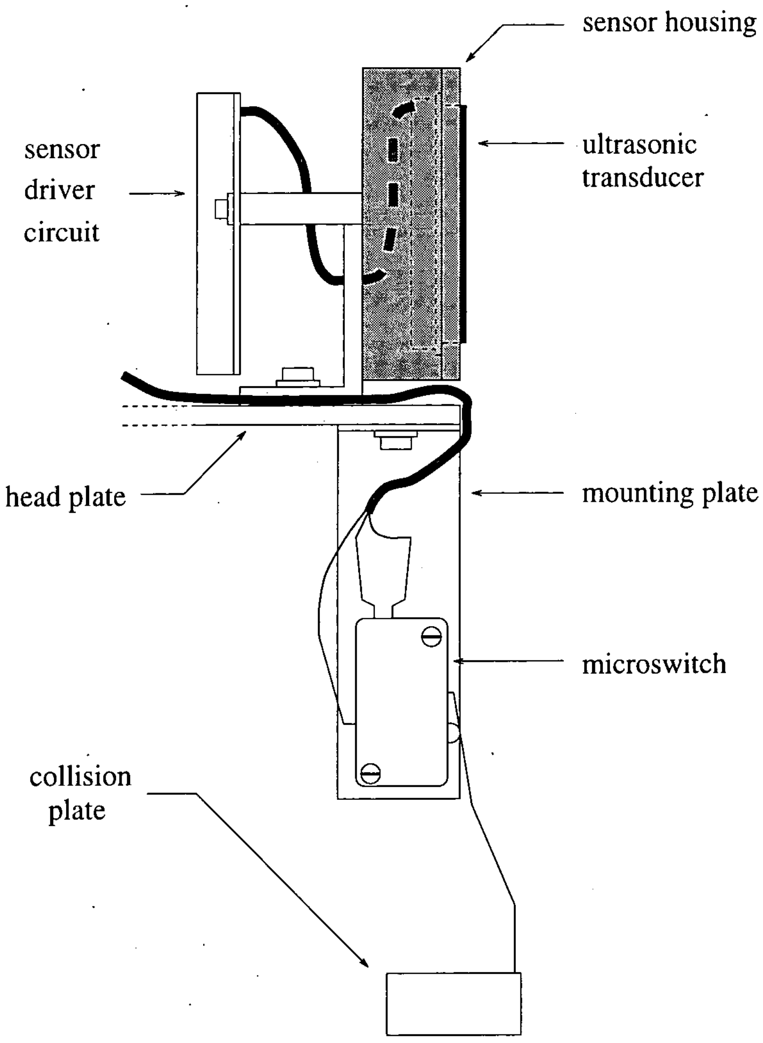


Figure 6.10: The ultrasonic sensor housing and collision detector

Equidistantly positioned around the outside of the disk are the eight ultrasonic sensor housings. Each comprises an ultrasonic transducer, and a sensor driver circuit. Beneath each is a collision detector attached to the underside of the head disk. Each detector is made up of a collision plate attached to a microswitch and this is shown in figure 6.10.

Above the control circuit board is the mounting for the magnetic sensor. This is an aluminium bridge that is positioned to be physically far from all the electrical systems that could distort the external magnetic field. The two sensors which are used to determine the magnetic field are mounted in a plastic block at right angles to each other. This block is fixed to the centre of the bridge.

6.4 The Electronic Hardware

The electronic hardware provides an interface for the computational systems to be able to interact with all the sensors and actuators on board the robot. All the electronics have been built on two PCBs, a power and drive circuit board, and a control circuit board. The details of their layouts are included in appendix C, but the following is a description of their functionality.

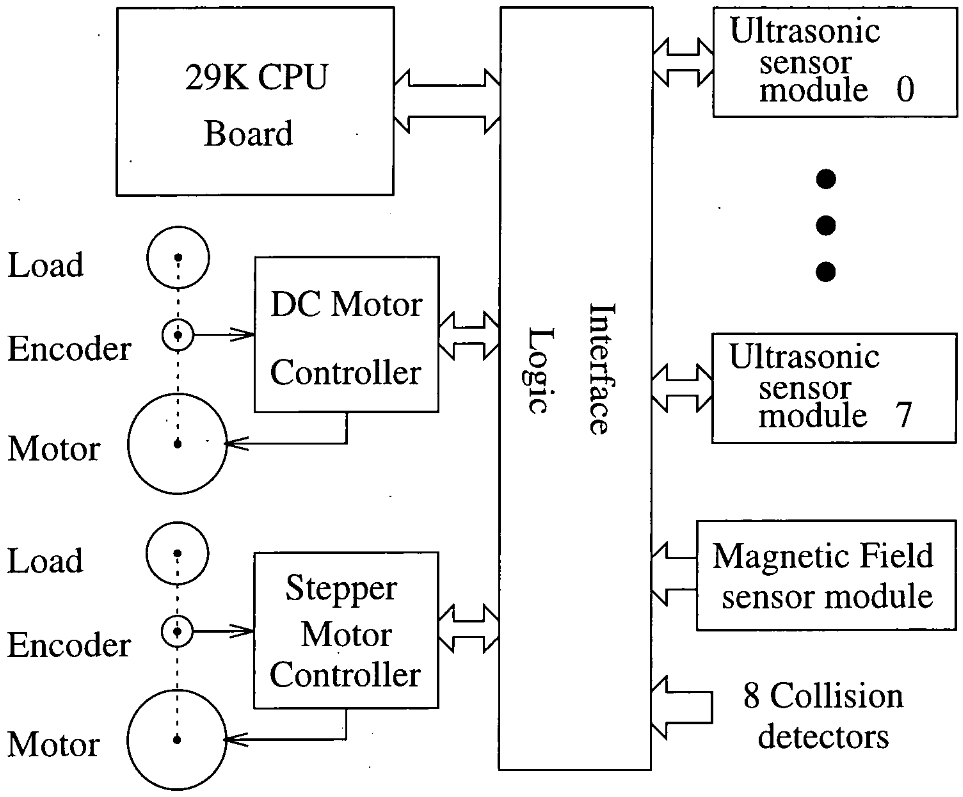


Figure 6.11: An overview of the electronic systems

In order to simplify the description of the electronics, it is useful to consider the system as composed of modules. Figure 6.11 shows the interactions of these modules, and their interface to the 29K computer board which itself is described in section 6.5. The design utilises programmable logic for interfacing and for the control logic required within the various modules.

Both the DC drive motor and the stepper motor use separate feedback controllers. These controllers are ICs which encapsulate a feedback control loop and are designed to interface to computer systems. Each IC monitors the appropriate encoder and produces motor outputs dependent on how it has been programmed to act by the computer. By using separate feedback controllers, both the computational and interface burdens are reduced for the main processor, whilst benefits are made from the fast hardware specific control.

The sensory systems are also modularised. There are eight ultrasonic sensor modules, one for each of the sensor driver circuits. Each module incorporates control logic that interfaces these driver circuits to event timers. Under the scheme that the sensors will operate, two event timers are required for each sensor. It was decided to have this functionality in hardware rather than software because the interrupt capability of the 29K board was limited. The magnetic field module and the eight collision detectors were also linked to the computer by the interface logic.

6.4.1 The Programmable Logic

Logic circuits are required to interface the various modules to the computer board. Also there is a substantial logic requirement within the modules themselves. One option is to hard wire logic elements from standard IC packages. This has the two disadvantages of low gate densities and hard to change designs. It is now possible to use programmable logic devices which have high gate densities and allow the routing of the designs to be changed. The generic name for all channel–routed, user–programmable logic devices is Field Programmable Gate Arrays (FPGAs). One such device is the Xilinx FPGAs which have their gate routing configurations downloaded onto the IC each time it is to be used. This leads to fast development times and rapid elimination of errors. The Xilinx based FPGAs were therefore chosen to form the basis of the robot's logic requirement. The particular architecture that Xilinx uses in its FPGA is termed Logic Cell Array (LCA), which is trademarked.

The particular Xilinx IC that was chosen was determined by the availability of the devices and the software used for configuring the logic. The device selected was the XC3030 which combines a high logic count of 100 Configuration logic Blocks (CLBs) with 80 user configurable input/output pins. However, as the logic capability and the number of pins used on each device increases, the routing problem becomes more complex. The routing is performed by software, but with greater complexity, the final design can have significant electrical signal delays. In order to supply the logic requirement for the robot and reduce delay problems, three devices were used in total.

After the Xilinx devices are switched on, they must have their logic configurations downloaded. This can be achieved in different ways dependent on the mode that the IC is set–up in. An advantage of these ICs is that they can be 'daisy chained'. Only one IC is directly connected up to the computer board, from which all the other devices are automatically configured. The detailed connection diagram for the three Xilinx FPGAs is contained in appendix D which also describes the signals required by the devices to load each byte of data in. After the three ICs have been configured the programmed logical configurations become active and are described below.

Figure 6.12 shows the three Xilinx ICs and their logical interconnections with the rest of the systems onboard the robot. The first Xilinx IC 0 produces all the computational

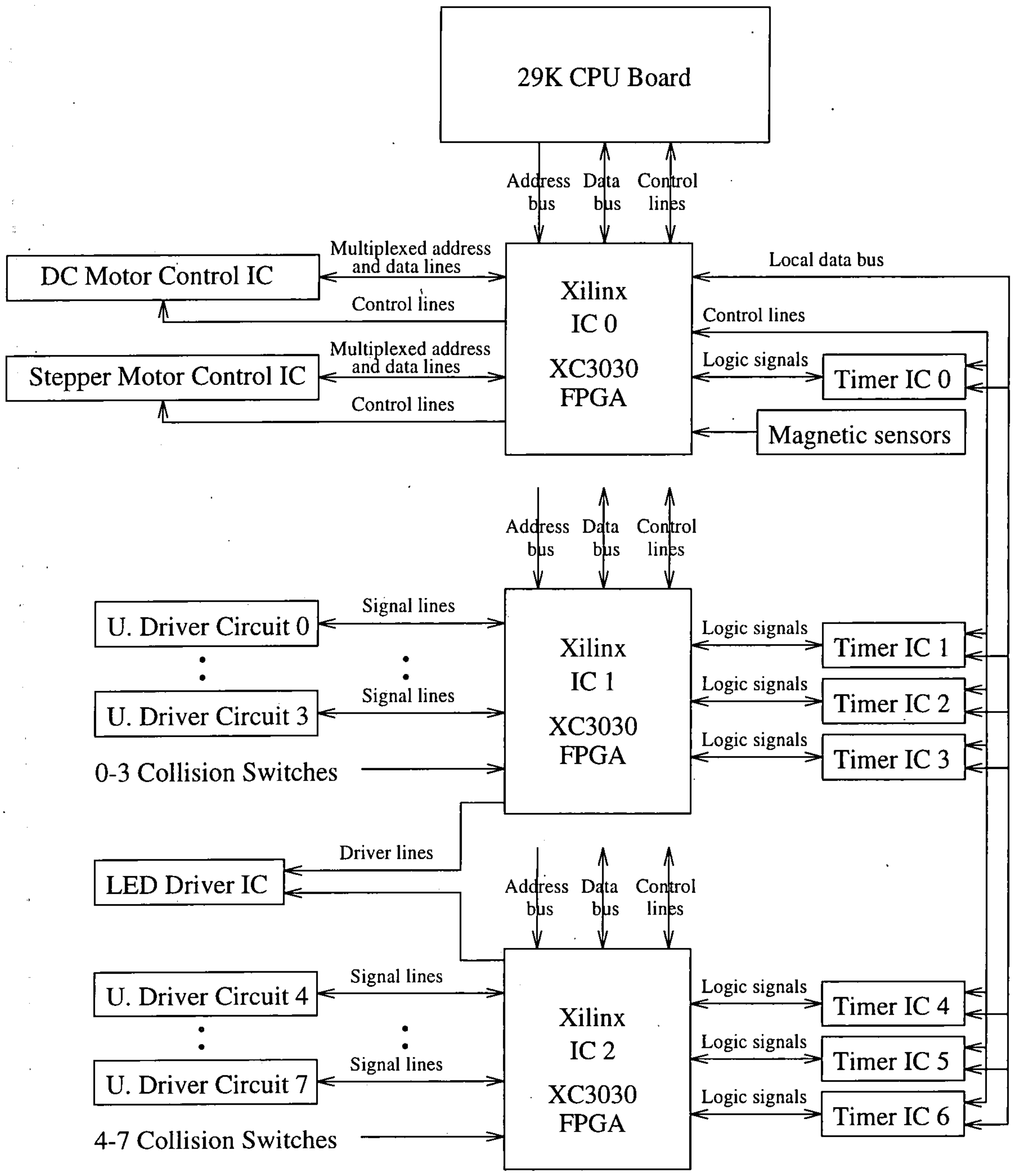


Figure 6.12: The system's logic layout

interfaces to the other ICs. The main interface is with the motor control ICs as they require a multiplexed address and data bus which the 29K computer board does not support. Therefore the standard address and data busses must be latched at the appropriate times, and signals to sequence the motor controllers must also be constructed. This IC also produces a local data bus and control signals for all the timer ICs. This allows the computer board to be able to access all the data registers on these devices. Each timer IC has three down counters, which can be programmed to operate in different modes. These counters are linked up to the other electronic systems using the logic. Timer IC 0 is used for timing the pulses from the magnetic sensors and therefore the logic to link the sensors up to it is included within the Xilinx IC 0.

As a method of dividing the logic up between the Xilinx ICs, the eight ultrasonic sensor modules are split into two groups, that of 0 to 3, and 4 to 7. The eight collision switches are similarly divided. All sensors 0 to 3 and 4 to 7 use the Xilinx ICs 1 and 2 respectively. Each ultrasonic sensor module requires two timers, and therefore three timer ICs are linked to each Xilinx IC, leaving one down counter free for other purposes. Also each Xilinx IC provides four output lines for the eight bar LED display.

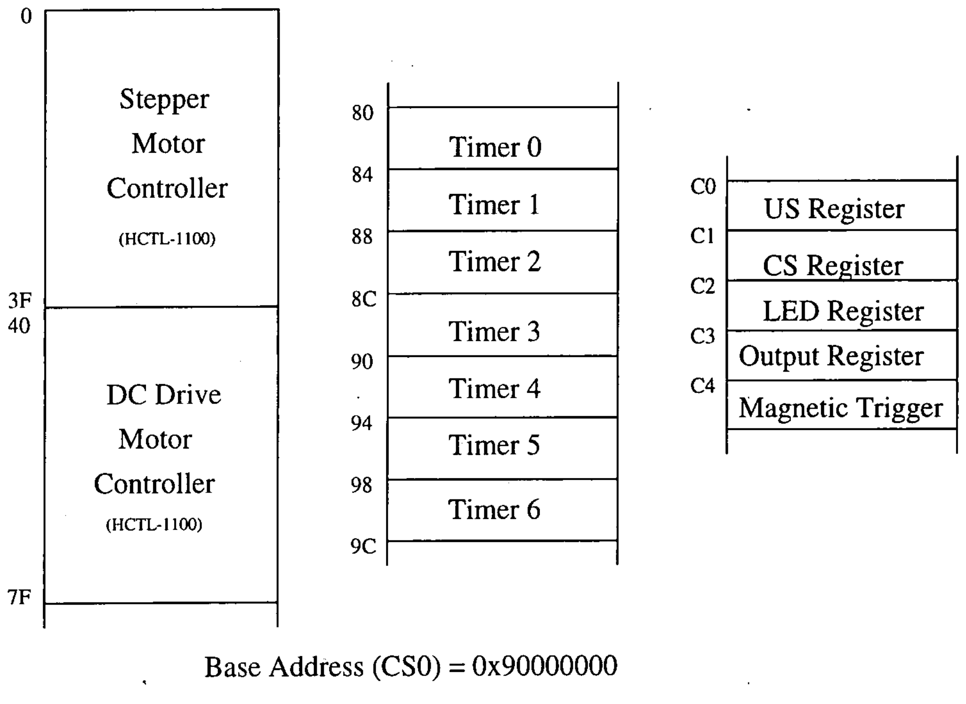


Figure 6.13: The address map

The Logic interface allows the various components in the system to be accessed by the computer board as peripheral memory locations. Figure 6.13 shows the memory map of the peripheral systems as seen by the computer. The 29K computer has five peripheral chip select regions and the robot systems map into the base of region 0. This is termed the chip select 0

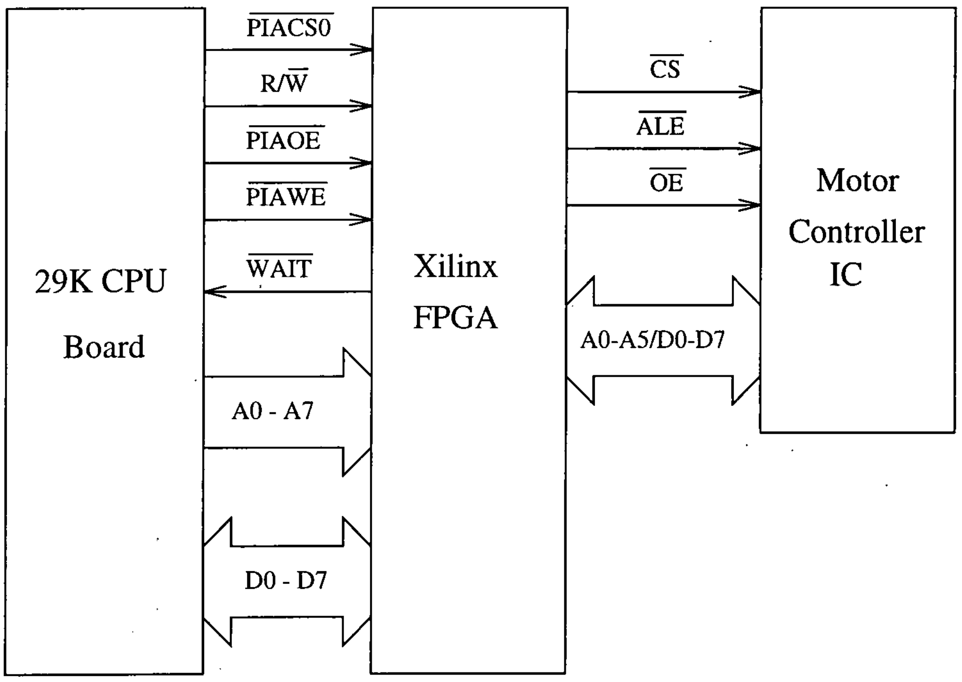


Figure 6.14: The motor controller interface

region and has a base address of 90000000 (Hex). As this computer is a 32 bit processor, the addresses listed, refer to the word offset from the base address, and not the byte address. To obtain the byte address, the word address is simply multiplied by the number of bytes in a word, which is four. Both the motor controller ICs have 64 registers and therefore occupy the first 128 addresses. Each timer IC has four addresses which are located between 40 and 7F (Hex). There are also five system specific registers which are located from address C0 (Hex) upwards, which are described later.

6.4.2 The Communications Interface to the Motor Controller ICs

The motor controller ICs used a multiplexed address and data bus, which is not supported by the 29k computer board, as it uses separate address and data lines. The motor controller ICs also require special control lines that signal different phases in the read and write cycles. An interface was designed based on the timing diagrams of the read and write cycles of both systems.

Figure 6.14 shows the connections to and from the Xilinx IC on which the programmable logic was used for the design. The $\overline{\text{PIACS0}}$, $\overline{\text{PIAOE}}$ and $\overline{\text{PIAWE}}$ are the Chip Select, Output Enable, and Write Enable lines respectively, of the Peripheral Interface Adapter (PIA). This is a 29K external area which can be accessed in the same way as memory. The R/$\overline{\text{W}}$ line indicates a read cycle when high and a write cycle when low. The $\overline{\text{WAIT}}$ line is used for

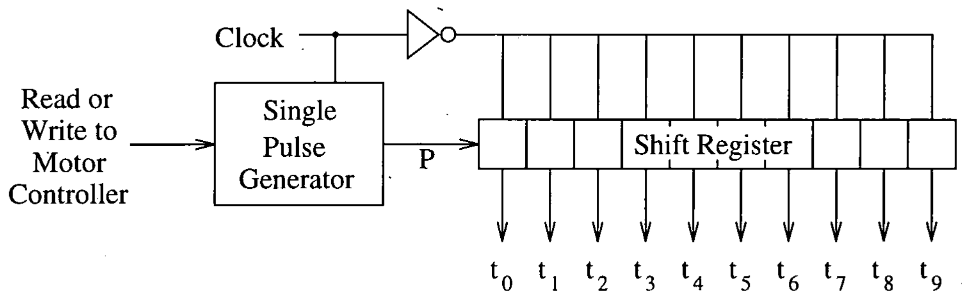

Clock
Read or Write to Motor Controller
Single Pulse Generator
P
Shift Register
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9

Figure 6.15: The motor controller pulse generator

forcing the processor to wait for external devices during a read/write cycle. Although only the lower six addresses are used by the motor controller, all eight are required for each motor controller to be mapped into the right area of memory. The motor controllers use three separate control lines each. The $\overline{\text{CS}}$ line is the Chip Select line and indicates a read or write cycle is to start. The $\overline{\text{ALE}}$ is the Address Line Enable and indicates that a valid address is on the address/data bus. The $\overline{\text{OE}}$ is the Output Enable and is used to signal to the IC when to latch data onto the bus.

To be able to interface the devices, the fixed read/write pattern from the 29K computer had to be converted to a sequenced latching of address and data information onto the multiplexed bus, with corresponding signalling. To achieve this, a shift register based solution was devised. This is based on a single pulse being triggered at the start of a read/write cycle which then shifts down the register under the control of a clock (as shown in figure 6.15). As the pulse (P) propagates down the register, it activates and deactivates set/reset flip–flops that are linked up to the signals t_0 to t_9. The outputs of the flip–flops control the sequencing of data and address bus latching as well as creating the control signals. The clocking for the shift register is in anti–phase with the rest of the system to ensure a stable sampling of the shift register.

Figure 6.16 shows the activations of the taps over time with the clocking signal used for the flip–flops. From the flip–flop outputs, the read and the write cycles are generated. Both sets of signals are generated simultaneously in the hardware, however only one set is used as gated by the R/$\overline{\text{W}}$ data direction signal. In both cycles the 29K computer must be delayed from executing any further instructions and therefore a $\overline{\text{WAIT}}$ signal is generated. At the end of this wait period when the computer's CPU is released, it deactivates $\overline{\text{PIACS0}}$ clearing all the flip–flops and registers.

To create a multiplexed address and data bus, the interface uses a communications bus internal to the Xilinx IC. On this bus, the address and the data can be latched. The internal latching of the address and data to this bus is controlled by the Address Enable (AE) and the Data Enable (DE) signals. In the write cycle the CS is activated as the CPUs addresses

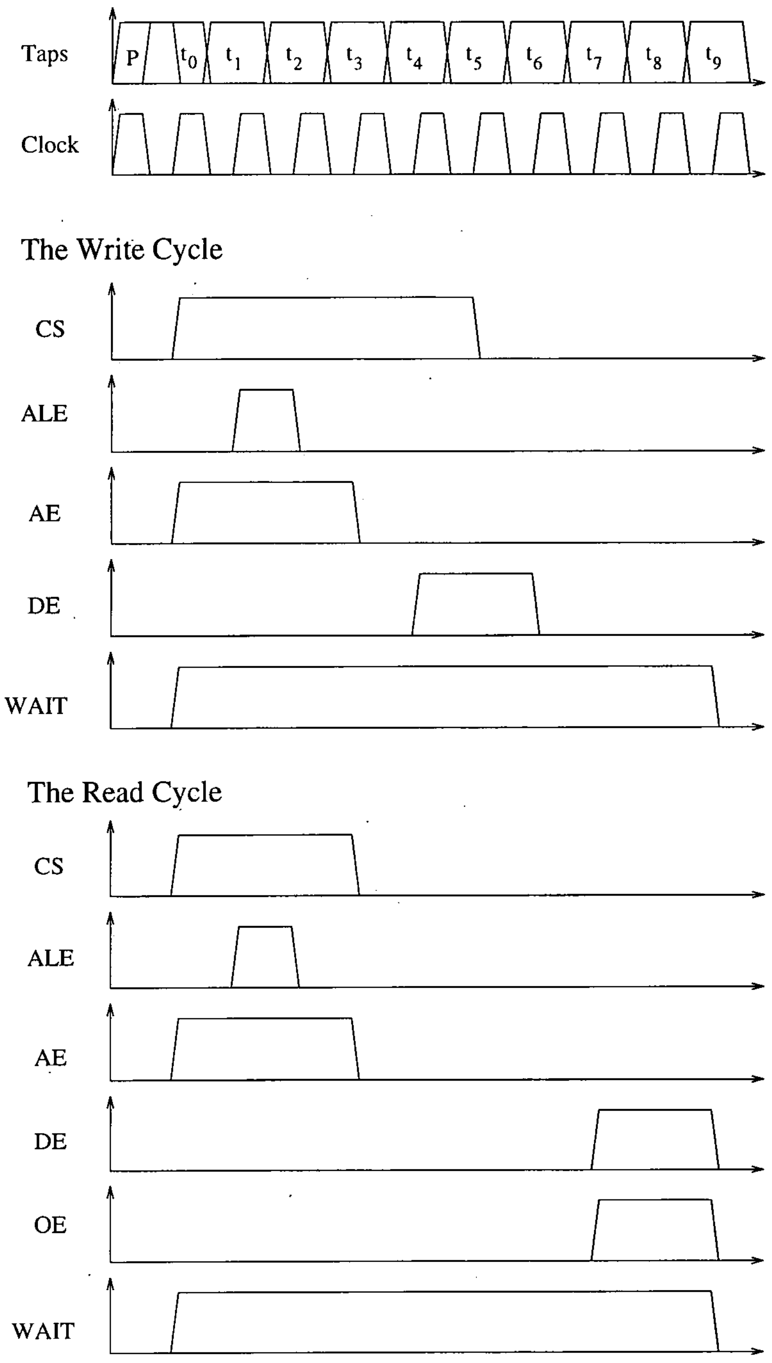


Figure 6.16: The motor controller timing diagram

are latched through to the motor controller IC. ALE is activated to force the controller to sample this address (ALE is shown here as the flip–flop's output, not the its inverse $\overline{\text{ALE}}$ which is output to the controller). After AE becomes inactive the data bus is latched through (DE goes high). When CS is deactivated the motor controller IC samples the data into an internal latch. The cycle is completed when WAIT is deactivated. For the read cycle, the CS is only activated during the address reading phase. After the address has been read, the device specifications require a delay before the data can be read. This is finally done by latching the data from the controller IC to the CPU board and then deactivating the WAIT line.

6.4.3 The DC Motor Controller

The DC motor controller system comprises a controller, motor and a feedback encoder. This is different from other mobile robot odometry systems in that it uses a feedback loop which includes the ground movement beneath the robot. Most encoders are mounted on the drive shaft, and therefore cannot take these effects into account.

The motor controller employed for both the drive and stepper motors was the HCTL-1100 manufactured by Hewlett Packard. This is a general purpose motor control IC that can be used to control either DC motors or stepper motors. Therefore using these controllers reduces the computational burden of digital motion control on the main processor.

The motor controller ICs require quadrature phase encoders. These rotary encoders have two sets of slots which are phase displaced by 90 degrees. From these two channels, the direction as well as the angular distance can be computed. Another benefit of this system is its improved positional accuracy resulting from the ability to determine which quarter of each slot–mask element the encoder is at. This method of counting in quarters is termed quadrature positioning and its units are quadrature counts. A high resolution encoder was chosen in order to maximise the accuracy of the odometric system. The encoder has 500 slots per revolution which with a wheel radius of 3 cm, gives a resolution of just over 10 quadrature counts per millimetre of robot travel.

The overall system layout for the DC drive motor can be seen in figure 6.17. The controller's command output to the motor is in the form of a pulse width modulated signal (PWM) and a sign/direction signal (sign). The pulse width modulation signal is a pulse train whose mark to space ratio varies dependent on the drive power required by the motor, and the sign indicates the drive polarity. Both these signals are further processed by interface logic to be suitable for powering an 'H' Bridge power amplifier. The power amplifier drives the DC motor which, through the transmission, drives the three wheels. The feedback signals of channel A, and B come from the rotary encoder that is mounted on the odometer wheel in the centre of the robot. The inclusion of the ground movement in the feedback loop allows the odometry system to avoid errors caused by slippage when over accelerating or braking. Also

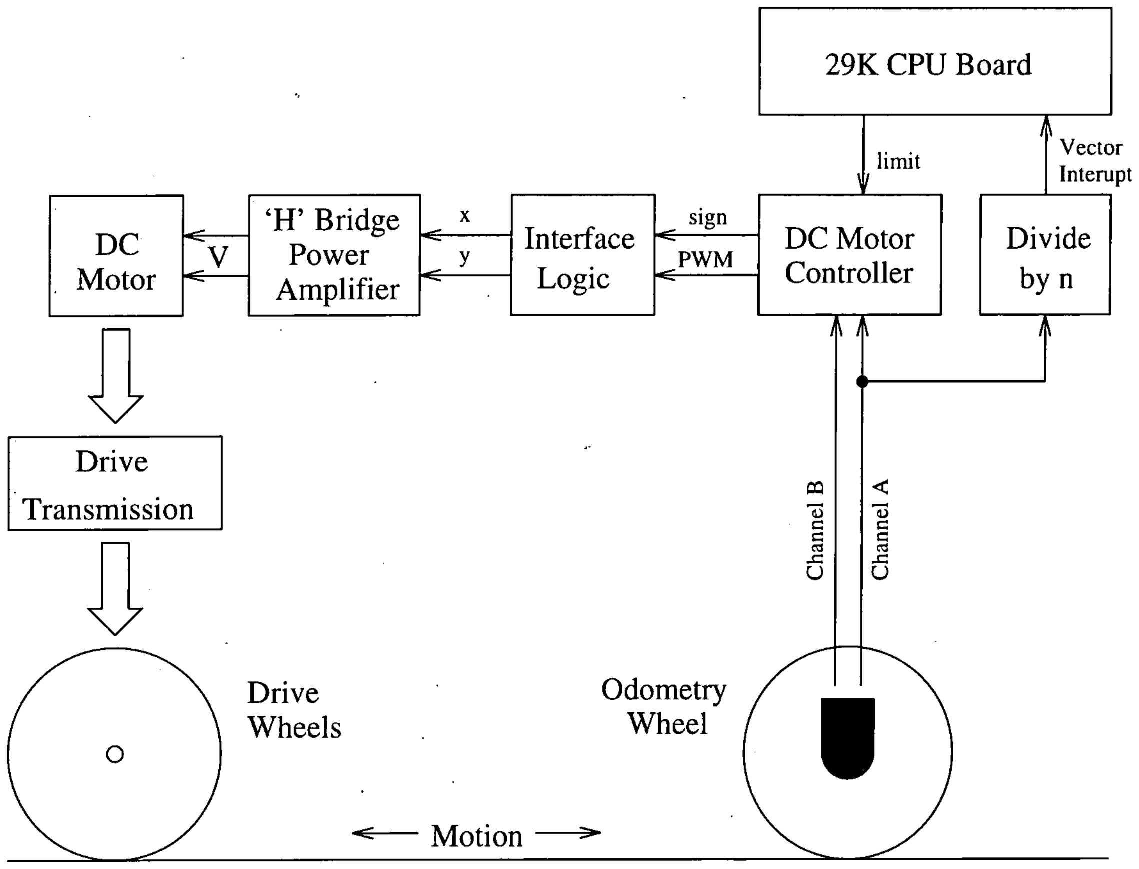


Figure 6.17: The motor controller feedback loop

if the robot is pulled or pushed, the feedback system can correct the movement by driving back to the original location.

The 'H' bridge configuration of amplifier was selected as it allows a motor to be driven in both directions from a single positive power supply. In order to create the correct driving signals which are the PWM signals in each direction, two AND gates were required and are shown in figure 6.18. These 5 volt logic signals are amplified by two BC183L transistors to drive the bases of the power transistors (see figure 6.19). The x and y signals fire alternate transistors in the 'H' bridge which then form a voltage across the motor (M).

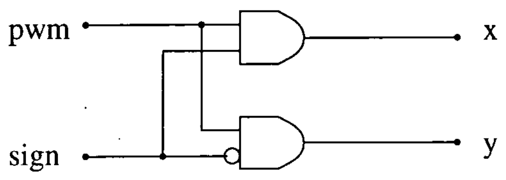


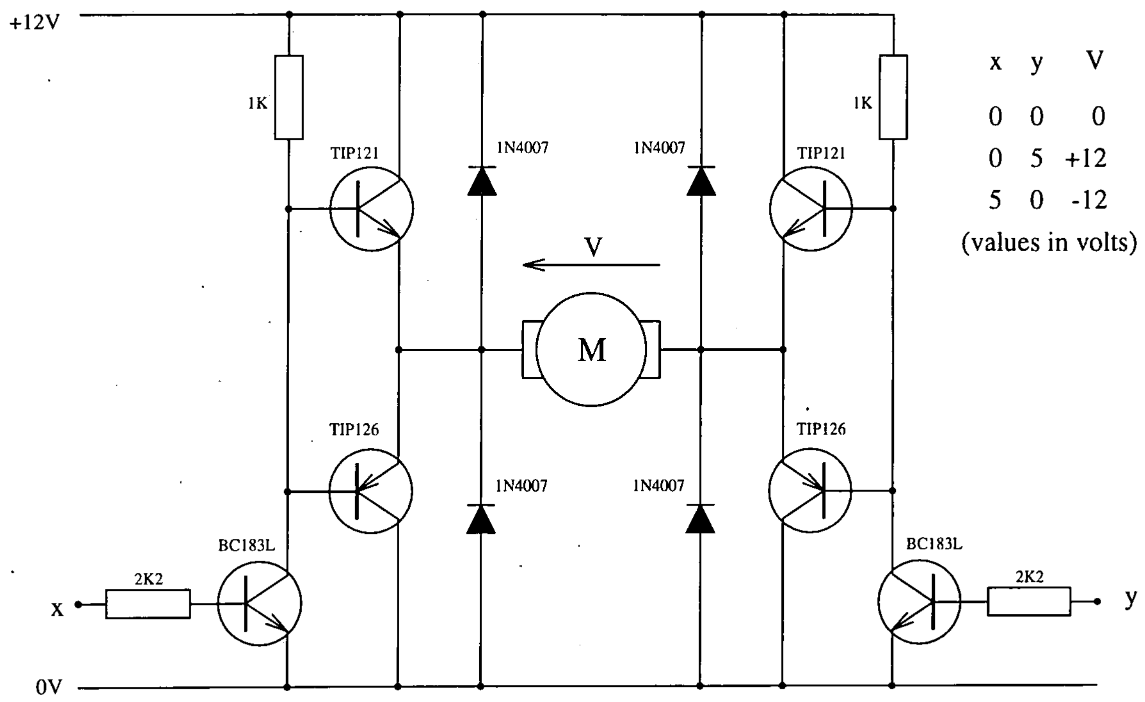
Figure 6.18: The interface logic

There is a limit pin on the motor controller which when activated causes the motor command to be set to zero. This is used by the CPU board to allow automatic shut down of the motor in the event of a collision. There is also a vector interrupt that is generated from the channel A of the encoder. This is used by the CPU board for automatic generation of the robot's odometric position. After every n pulses, the CPU is interrupted and takes information from both the drive and stepper motor controllers to get forward and angular positional information. These vectors can then be integrated to give the robot location.

6.4.4 The Stepper Motor Controller

The stepper motor controls the angular position of the wheels and must also provide a holding torque whilst the drive motor is in operation. To provide an accurate positioning system another HCTL-1100 general purpose motor controller was used to form a separate control system. However, unlike the DC motor, the motor controller also provided the sequencing for the four phases of the stepper motor.

The stepper motor has 48 full steps per revolution which leads to a small angular difference between phases in the torque cycle. To ensure that accurate information about the motor's position in its torque cycle was fed back to the controller, the encoder was mounted directly on the motor's shaft. In order for the controller to provide the correct phase sequencing for the stepper motor, it requires an encoder with an index pulse which can be aligned to a known location in the motor's torque cycle. This also means that the encoder's number of quadrature counts per revolution must be exactly divisible by the number of motor steps per full revolution. Hence every revolution of the index pulse always occurs at the same location



x	y	V
0	0	0
0	5	+12
5	0	-12

(values in volts)

Figure 6.19: The 'H' bridge power amplifier

within the motor's torque cycle. The encoder that was selected had the maximum obtainable resolution that fitted this requirement of 360 slots per revolution.

For the commutator in the motor controller to function correctly, the parameters that define the motor's torque must be determined. It was determined that the motor would be operated in half step mode to increase the accuracy of the control. One of the parameters that the controller requires is called the ring register and must be programmed with the number if quadrature counts in the torque cycle. As 96 half steps corresponds to $360 * 4$ quadrature counts, and there are eight half steps in a torque cycle, the value of the ring register is 120. Figure 6.20 shows the phase sequencing over a torque cycle. The commutator's sequencing is programmed in terms of the number of quadrature counts for which one phase is active (x), combined with the length of overlap between adjacent phases (y). For the half stepping mode, both these values are set at 15 quadrature counts. The Index pulse must be aligned to the start of the commutation cycle, and this is performed by energising the last phase of the motor (phase D) and then fixing the encoder index pulse to this position. However, to perform fine adjustment of this alignment between the electrical and the mechanical torque cycle the offset register is used.

There are four outputs from the motor controller to the four phases. These can be combined with the PWM signal to form a drive signal for the motor. By using the phase signals in conjunction with the PWM information, the motor phases are powered only when required and not continuously as in most stepper motor applications. This offers significant power

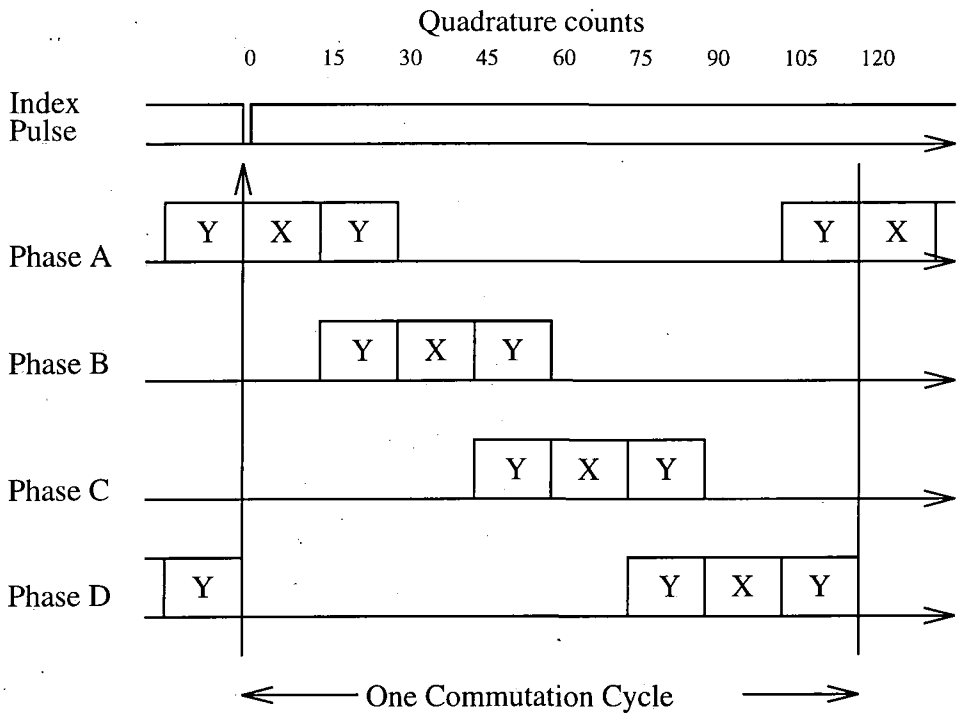


Figure 6.20: The stepper motor commutation cycle

savings which lengthens the robot's operational time. Figure 6.21 shows the combination of these signals by means of AND gates. Figure 6.22 shows the power electronics that take these signals which drive the four transistors and power the four phases of the stepper motor.

6.4.5 The Ultrasonic Ranging System

Of all the information gathering systems on the robot, the most important are the range gathering sensors. These are required to gather distance information from which the robot can infer structure about its environment. A cheap and readily available type of sensor which is used extensively in mobile robot applications are the Polaroid sensors. These are time of flight devices which emit a ultrasonic pulse of sound whose round trip from sensor to object can be timed. The simulation experiments suggested that about eight sensors would be required for this control application, but there is always a balance between angular resolution in the sensor system and rapidity of firing. More sensors give this increased resolution but the total sensing time is lengthened, as each sensor has a unique firing duration. It was therefore decided that eight Polaroid sensors would be used. However, if a greater angular resolution is required, the head assembly can be rotated so that more readings can be taken at different angles.

The Polaroid sensors handle the analogue electronics which are required to operate the

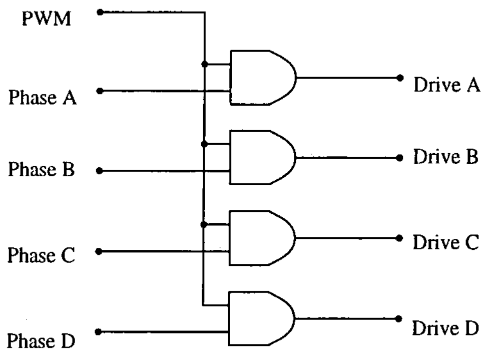


Figure 6.21: The logic interface

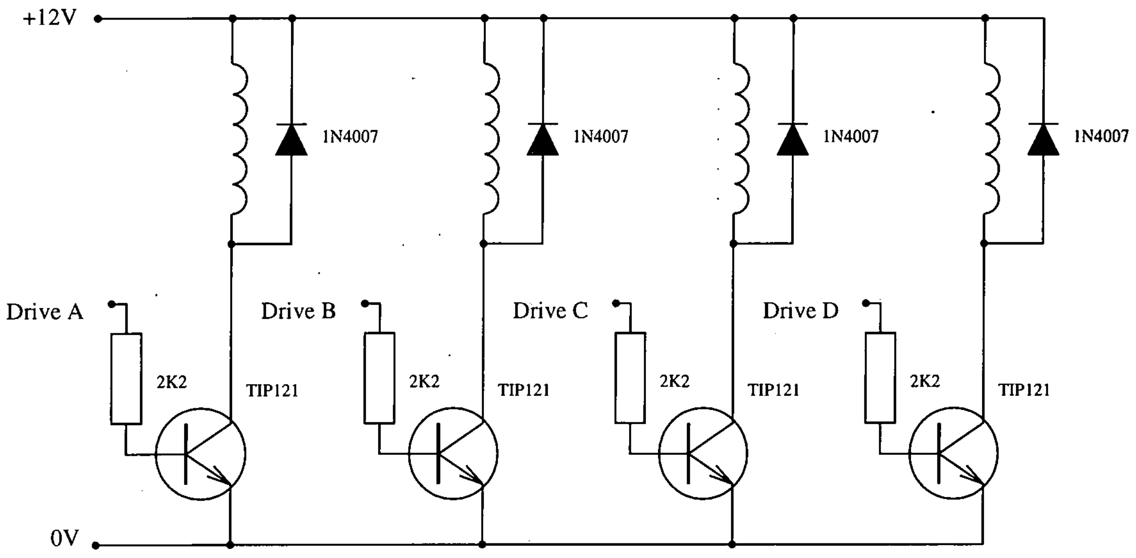


Figure 6.22: The stepper motor circuit diagram

transducers, but the electronics for timing and interfacing the sensors to the computer had to be developed. These were developed with reference to the functionality of the 29K computer board. This computer board does not support extensive interrupt facilities, and hence any sensor system must not overburden the main processor. The system must also be versatile enough to be able to implement a sensor firing scheme called Error Eliminating Rapid Ultrasonic Firing (EERUF [44]), which was developed by Borenstein to reduce errors in ultrasonic sensors. This is based upon each sensor being fired after a specified delay which alternates between firings. Although it increases the complexity of the firing system, it can detect crosstalk errors which lead to erroneous distance readings. To satisfy these conditions the system was based in hardware, which offers fast control and accurate event timing and is also easily accessible by the computer board. The computer would only be required for light interrupt handling leaving its CPU free for other processing.

The aim of the EERUF scheme is to be able to use standard ultrasonic sensors and eliminate the errors associated with crosstalk. The basis of this method is that the time of flight of an ultrasonic pulse is constant over a fixed distance irrespective of when that reading is measured. However, the consistency of crosstalk effects depends on the relative firing times of interfering sensors. Therefore by altering the times at which sensors are fired it must be possible to determine which sensors are experiencing crosstalk and which are not. The actual EERUF synchronises the sensors to a fixed repeating period. Figure 6.23 shows this firing scheme in operation for two sensors A and B. The sensors are only allowed to fire after a fixed delay time from the start of the period. These specified delays are alternated between successive firing periods. The diagram shows the effects of a crosstalk path that exists between sensor A and B. Sensor B's own ultrasonic pulse is still in flight when A's pulse is received by the sensor. Only by observing another reading is it possible to determine that B's distance is alternating in length and therefore it must be erroneous.

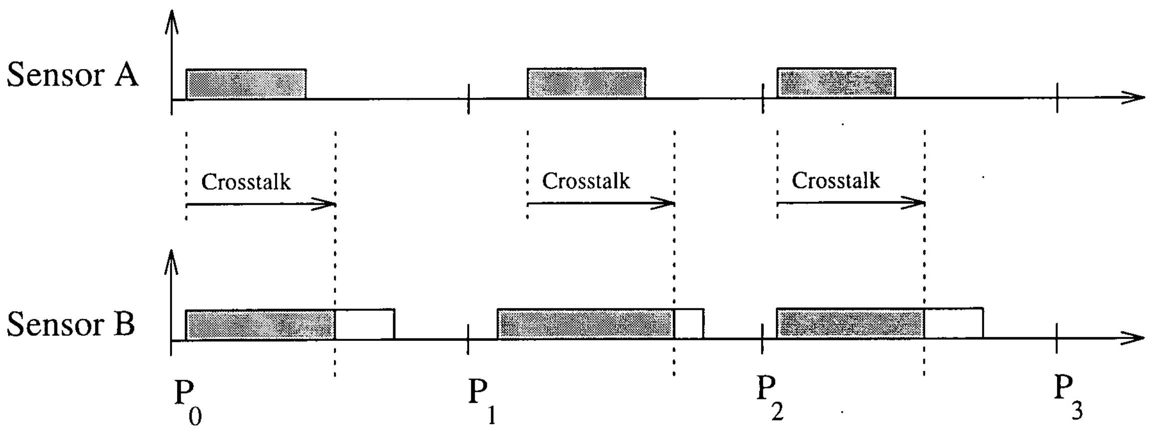


Figure 6.23: The error eliminating rapid ultrasonic firing scheme

The method by which this system was implemented was to have two event timers dedicated to each sensor. One of the timers stores the delay time and the second timer records the time of flight of the pulse. There was also a main period timer that produces the synchronising periods. The power to all the ultrasonic sensors can be switched on or off in software, so this can be used to reduce the battery drain when they are not operating.

The ultrasonic sensors are set–up as an interrupt driven system thus freeing the processor for other tasks. To operate the sensor system, the hardware is initialised and a period counter is set to the number of firing rounds plus two. After the period timer has been started the system is purely interrupt driven. Figure 6.24 shows the sequencing of the sensor firing broken up into periods. These periods are created by the period timer which can be programmed with a delay interval. At the end of the delay, the timer causes an interrupt in which all the sequencing routines are operated. This delay interval can be repeated by simply writing a new delay time to the period timer again. At the start of the first period after the sensors

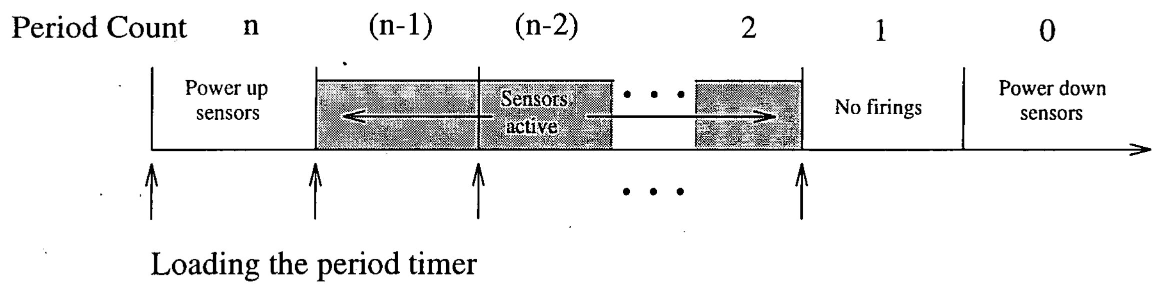


Figure 6.24: The sensor firing period timing diagram

have been powered up, they are allowed to settle. Then the firing rounds are performed with each sensor system being triggered at the start of each period. At the penultimate period interrupt, the sensors are not fired so that sensors still operating from the previous period can finish. At the last interrupt the sensors are powered down, the period sensor is not re–loaded and the sensing is over.

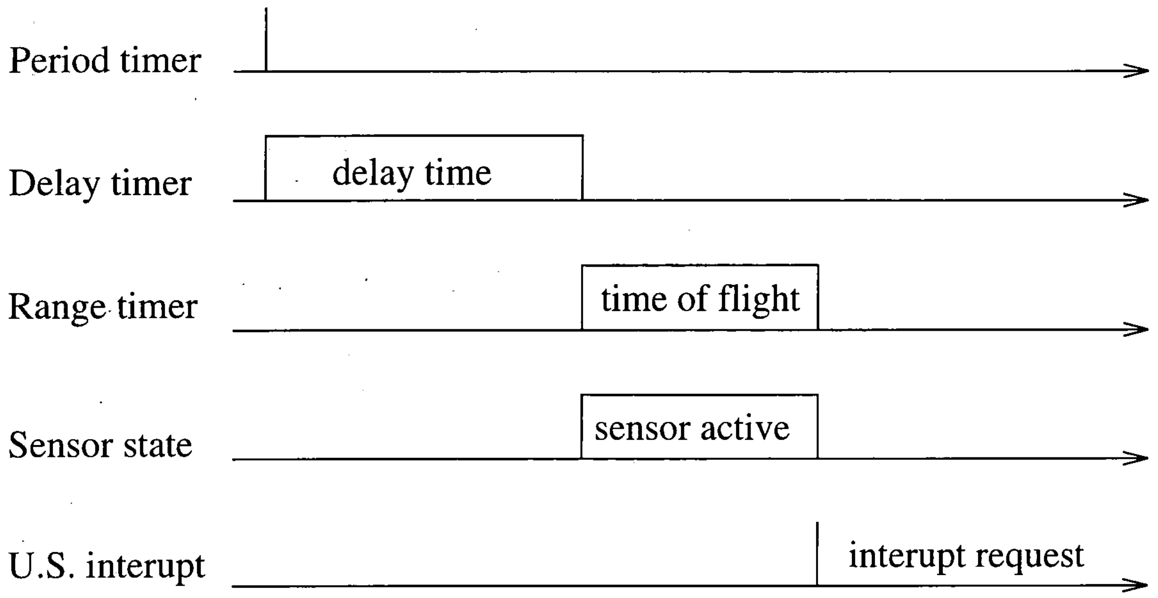


Figure 6.25: The ultrasonic sensor timing diagram

At the start of each firing round all the delay timers are programmed with their own odd or even delay times dependent on the period count. As soon as the value is programmed, this starts off the delay timer, and the rest of the timing is controlled by hardware. Figure 6.25 shows the timing diagram for an individual sensor. When the delay timer is finished it triggers both the range timer and the ultrasonic sensor circuit. The range timer is a down counter programmed with a maximum sensor time, and if the sensor does not return an echo value before this finishes, an interrupt is requested. In normal operation when the sensor receives the incoming echo pulse it stops the range timer and requests an interrupt. As well as requesting an interrupt, a bit is set in the Ultrasonic Sensor register to indicate which sensor called it. Each bit corresponds to one sensor module and they are arranged in ascending order with the LSB corresponding to sensor module 0. The interrupt request calls an interrupt service routine which examines the U.S. register to determine which sensor requested an interrupt. The range time for the sensor is then recorded in a global structure allocated to store the full sensor data. After this, the range timer is reprogrammed with the maximum range time. The hardware controlling the sensor is reset by writing a 1 into the bit of the calling sensor in the U.S. register. Then the sensor is ready for the next firing interval.

After all the sensors have concluded firing, a software routine is called which takes the raw timer information from the global data structure and processes it to determine which sensors have measured valid distances, and which have been affected by crosstalk. This yields the eight range distances and a confidence value for each of them.

6.4.6 The Hardware Implementation of the Ultrasonic Ranging System

The overview of the total ultrasonic sensor system is shown in figure 6.26. The sensors are split into eight identical sensor modules. Each module handles the timing and the low level signalling required to interact with the Polaroid sensor driver circuit. The driver circuit has been modified so that it can be operated in its normal mode, and in a short range mode which can decrease the minimum range of the device. The selected mode is applied to all the sensors and therefore is directly set by the short range enable line. The other direct control line from the computer board is the ultrasonic sensor enable which also controls the power to the sensors. The main register which is used to assess the state of the sensors and reset them as required, is the Ultrasonic Sensor Register (U.S. Register). If a bit is set in this register it indicates that the corresponding sensor module has requested an interrupt. If a bit is written to this register, it resets the hardware in that particular sensor module. The values of this register are combined by means of an AND gate which is used to trigger a level sensitive interrupt line on the computer board.

The timers that are used in the ultrasonic ranging system are down counters which can be programmed to operate in different modes. The period timer is one of these down counters operated in mode 4 which is a software triggered strobe. Figure 6.27 shows the three connections to the counter and their interactions in this mode. In the example a value of 6 is written to the counter (c=6 on the $\overline{\text{WR}}$ line). Once the value has been written to the counter the down count begins based on the clock (CLK). The counting can be halted by a low value on the GATE pin and when this value is raised, the counting begins again. The OUT line which is normally high, strobes low when the count value reaches zero. Hence the period timer interrupt on the computer board is set to be inverted edge sensitive.

The schematic diagram for each of the sensor modules is shown in figure 6.28. This circuit

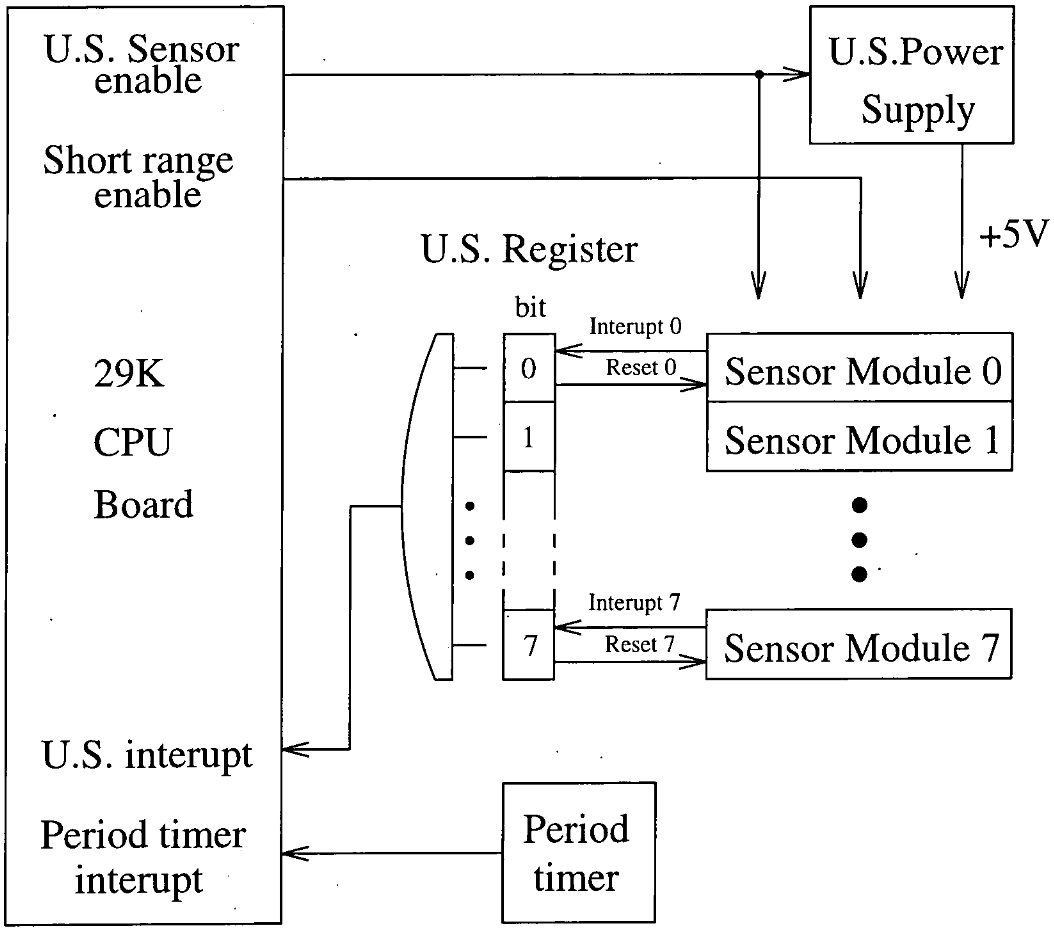


Figure 6.26: The schematic diagram of the sensor system

coordinates the delay timer, the firing of the sensor, and the timing of the flight of the echo pulse. The delay timer is programmed by software as described previously. On the completion of its count, OUT strobes low and if the sensor enable is active, causes the Set/Reset latch to be set (this latch is actually implemented using a JK flip flop for synchronising purposes). The output of the latch sets the initialise line INIT to become active. This starts both the sensor circuit operating and sets the GATE on the range timer high, which enables the range timer to start counting. An interrupt is caused by either the sensor circuit detecting an echo pulse and setting ECHO high, or by the range timer counting down to zero and pulsing OUT low (the range timer is pre–programmed with the maximum range time). The interrupt is latched into the U.S. register and also resets the hardware ready for another delay timer output.

The delay and range timers are designed to meet different objectives, the delay timer for versatility in time duration, and the range timer for accurate timing To solve this problem the timers were run off different frequency clocks. The counters are 16 bit and therefore have a maximum count of $2^{16} = 65536$. The delay timer's clock is 62.5 kHz, whereas the

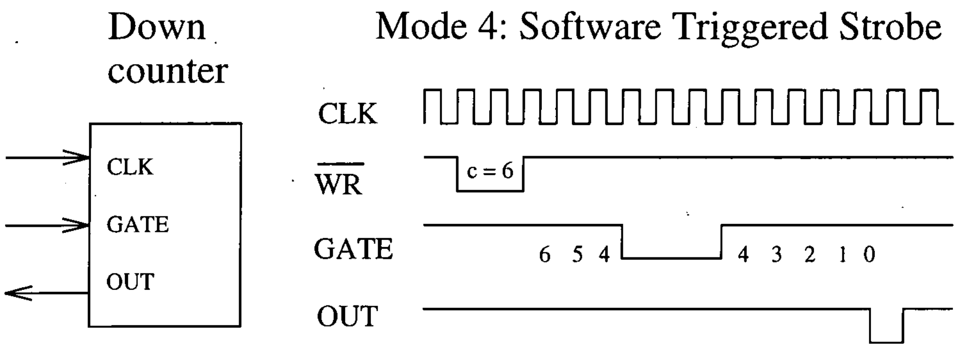


Figure 6.27: The down counter's operation in mode 4

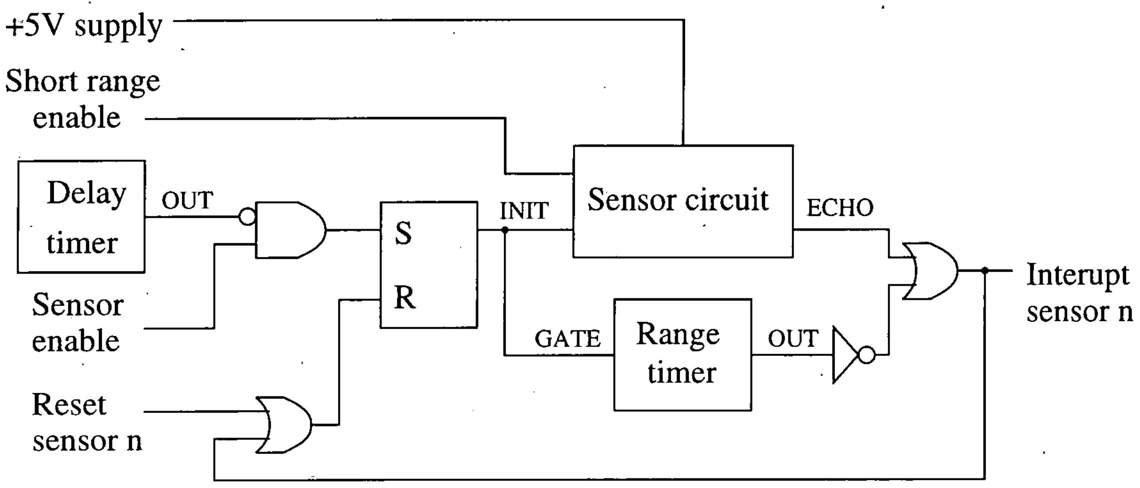


Figure 6.28: The schematic diagram of the sensor module

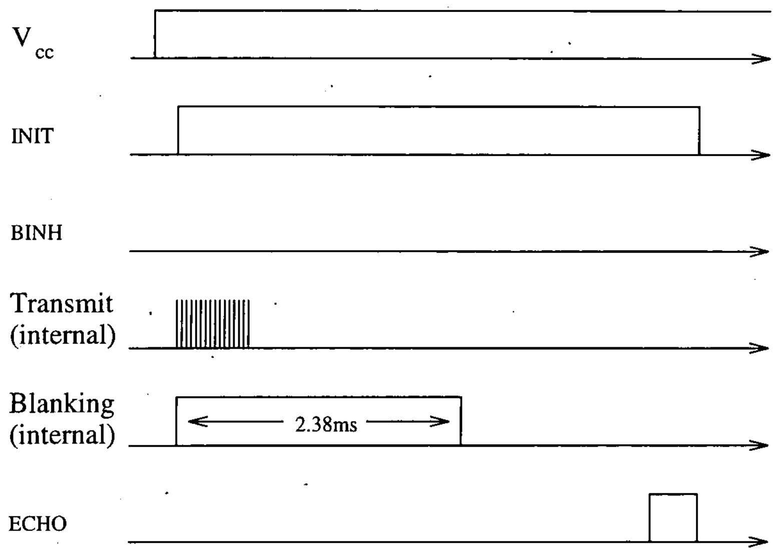


Figure 6.29: The Polariod sensor operating cycle

range timer's clock is 250 kHz, which gives maximum timing intervals of 1048 ms and 262 ms respectively. Hence, there is a greater accuracy in the range measurement, but more versatility in firing times for the delay.

The schematic diagram for the sensor circuit is dependent on the operation of the Polaroid sensor driver circuit. In order to understand the signals that are required to modify the circuit for short range operation, the method of operation of the Polaroid circuit must be reviewed.

The Polaroid ultrasonic ranging module is an active time of flight sensor which is widely used in mobile robotics predominantly due to its ease of use and low cost. The module comprises one circuit board and a transducer which is used as both a transmitter and a receiver for the ultrasonic pulse. This pulse frequency and duration is controlled on the circuit board and is set at 49.1 kHz. The minimum and maximum ranges specified for the module are 20 cm to 10.5m respectively.

A typical operating cycle is shown by a timing diagram in figure 6.29. After the circuit has been powered up by supplying 5 volts to V_{cc} the sensor is ready to fire using the initialise signal (INIT). Taking INIT high triggers the 16 pulse transmit signal which drives the firing transistor of the transducer. There is an internal blanking system which stops the receiver circuit from operating until the transducer's oscillations have abated. Unless the Blanking Inhibit (BINH) is activated this duration is fixed to 2.38 ms which limits the shortest measurable distance. The module automatically increases the gain of the receiver amplifier over time to compensate for the loss in echo strength. When the return echo exceeds a threshold in the receiver, the module indicates this by driving ECHO line high.

For the circuit to be useful for measuring shorter distances then 20 cm, some additional

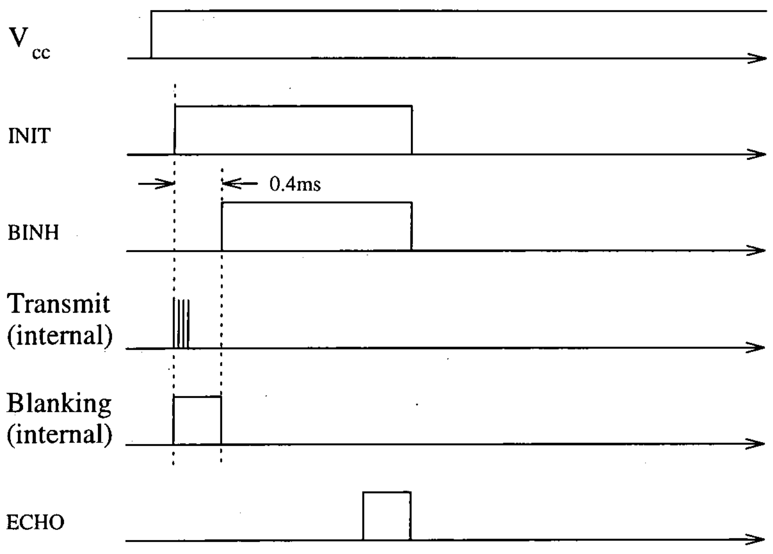


Figure 6.30: The Polaroid short range operating cycle

circuitry must be used. It is possible to reduce the length of the transmitted burst and reduce the time taken for the oscillations in the transducer to dampen down. This can be achieved by replacing the pulse circuitry with a custom made design. Figure 6.30 shows a short range operating cycle where the number of transmit pulses has been reduced from 16 to 4. The BINH line is then asserted 0.4 ms later which forces the internal blanking off. This is the shortest reliable time for blanking that was empirically found. The short range system allows distances to be measured down to about 7 cm.

In order to switch easily between operating the sensors in long range, or short range mode, it was decided to reproduce in hardware the circuit on the sensor for creating the transmit pulses. These reproduced transmit pulses were then wired in place of the originals on the driver circuit board. This enables the number of pulses to be easily controlled, and allow synchronising of the blanking signal to them.

Figure 6.31 shows the schematic diagram for the circuit that drives the Polariod sensor driver circuit. When the INIT line goes high, indicating the start of the transmit, a clock running at the frequency of the transmit pulses is started (50 kHz). There is a separate clock for each module so that the clock is synchronised to the asynchronous INIT signal. The clock drives a counter which is used to sequence the events. If the short range enable line is high, the blanking inhibit circuit activates BINH 0.4 ms after the INIT pulse, as determined by the counter value. The pulse generator circuit gates the clock signal to produce the transmit firing pulses (XMIT). If the short range enable line is high, then only four pulse are gated through, if low then sixteen pulses are allowed to pass. The XMIT line is replaced on the Polariod sensor driver circuit for the original firing pulses.

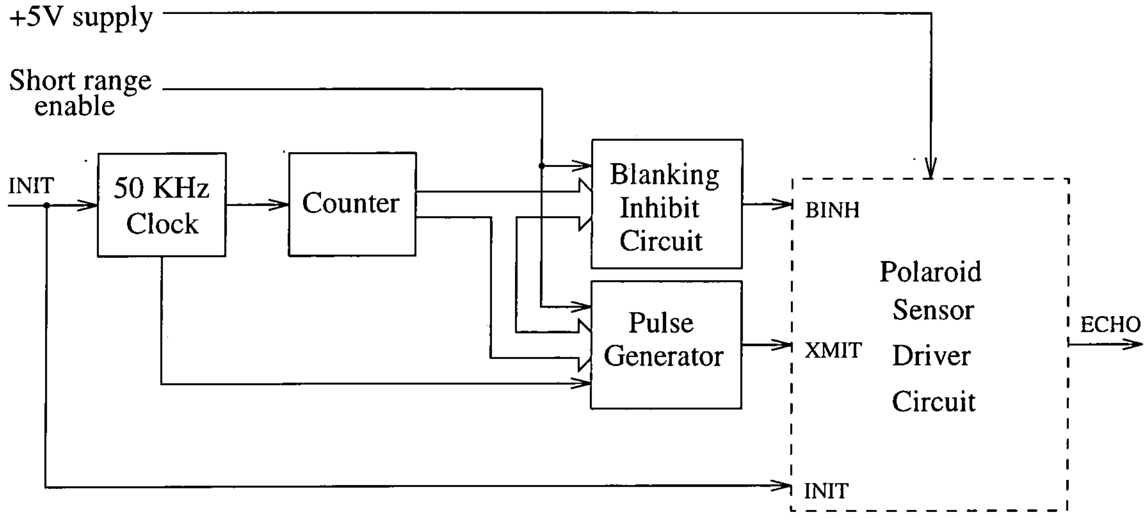


Figure 6.31: The sensor circuit

6.4.7 The Magnetic Sensor System

One of the major error modes associated with mobile robotics is positional drift over time. The x, y drift can be compensated by reference to sensors, such as range measures that directly relate to these coordinates. However, rotational drift errors would affect the range measurements and are harder to differentiate from standard sensor noise. Therefore it is useful to have some form of rotation sensor that could be used as a reference.

The most obvious sensor for this is a magnetic compass, which senses the direction of the earth's magnetic field. The commercial electronic compass systems are expensive, so a cheaper alternative which still gives accurate values was devised. The system is based on a new low cost saturable core magnetometer sensor produced by Speake & Co. Limited. Each sensor costs £14 and produces a pulse train whose frequency is inversely proportional to the magnitude of the field strength. The non-linearity of the sensor is documented in [45] at 5.5 per cent over the range $\pm 0.5 oersteds$, which relates to the earth's field strength of about 0.5 oersteds in the UK. By mounting two of these sensors perpendicular to each other, the ambient magnetic field strength and direction can be determined.

The most accurate method for monitoring the two sensors, is by counting the number of pulses from each sensor over a fixed interval. This can be easily achieved by using three down counters, one for controlling the interval duration, and the others for counting the pulses. The counter that was used to create a timing interval was operated in mode 1 which is shown in figure 6.32. The counter is initially written to, with the length of the interval over which the value of OUT will be low. After this has been set, it can be repeatedly triggered with a pulse to its GATE.

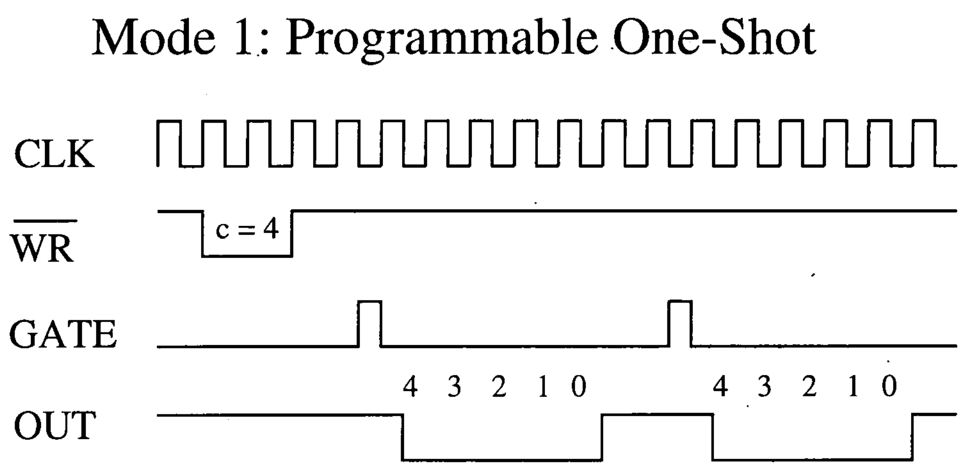


Figure 6.32: The down counter's operational mode 1

Figure 6.33 shows the circuit schematic of the magnetic sensor hardware. The counter 0 is used for the interval timing and is operated in mode 1. When the magnetic trigger line is pulsed high, OUT 0 goes low for the pre-set time interval. This magnetic trigger pulse is created in the hardware whenever the hardware trigger address is written to. The inverter drives both GATE 1 and GATE 2 high therefore allowing counter 1 and 2 to count the pulses from sensors A and B respectively.

The sequence for operating the magnetic sensor after the counter 0 has been initialised, is to write a known value to counters 1 and 2 and then to write any value to the magnetic trigger address. After a short interval, the values of counters 1 and 2 can be read. The number of pulses from a sensor is therefore the original known value minus the read count. These two values can be processed by software to give the values of field intensity and direction.

6.4.8 The Eight Collision Detectors

There are eight collision detectors, each of which is mounted beneath an ultrasonic sensor transducer. When the robot collides with an object it is important for it to be able to respond immediately. The collision system has been implemented to create software interrupts that permit a fast response. Figure 6.34 shows the schematic diagram of the system. Whenever a collision switch is activated it sets the corresponding bit in the Collision Sensor register (C.S. register). The AND gate drives the C.S. interrupt line high causing a collision sensor interrupt. Figure 6.35 shows the collision detector circuit which is composed of the microswitch collision sensor, and two NAND gates. This provides a 'de–bounce' for the switch, and an interrupt signal which can be reset by software.

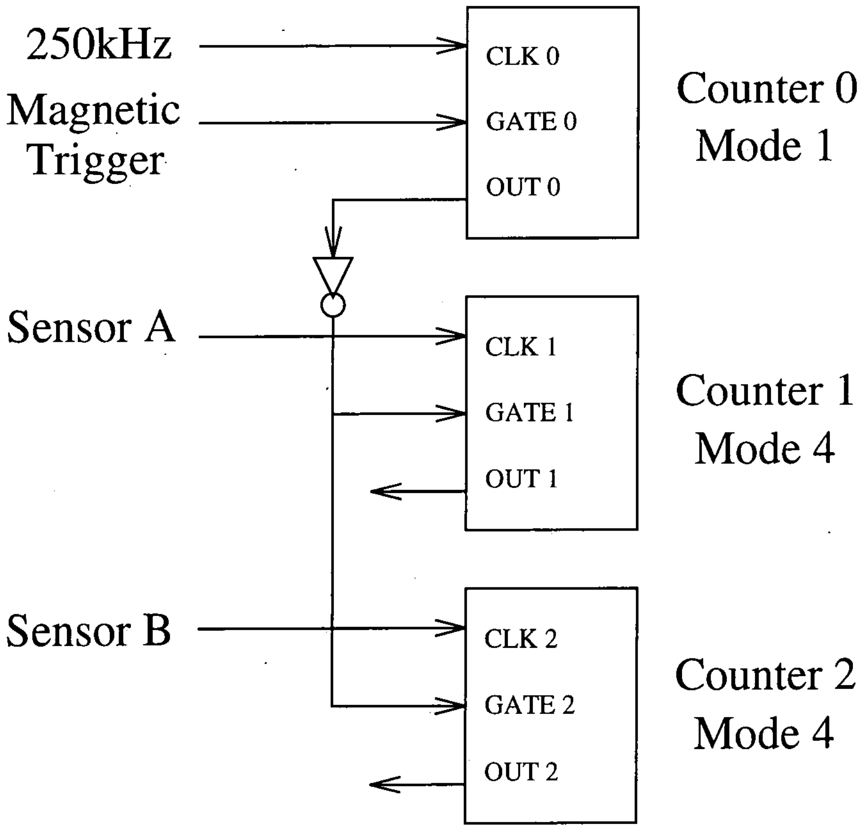


Figure 6.33: The magnetic sensor schematic

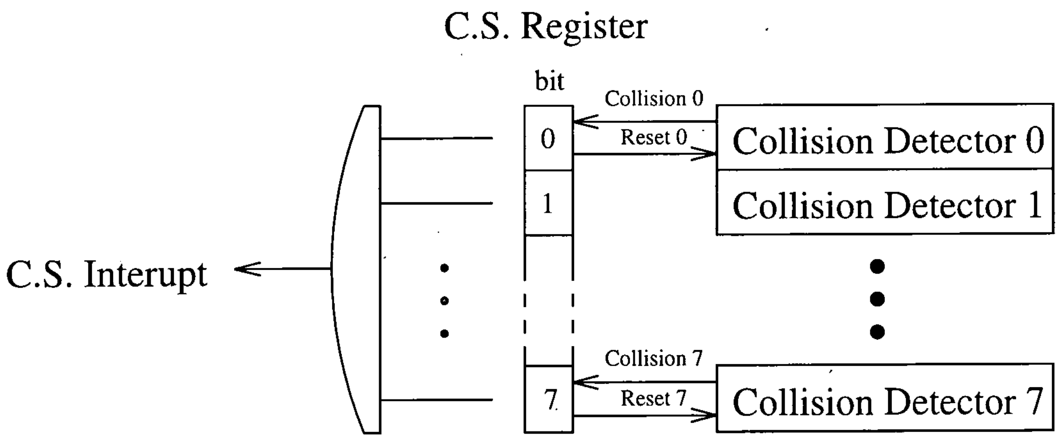


Figure 6.34: The schematic diagram of the collision system

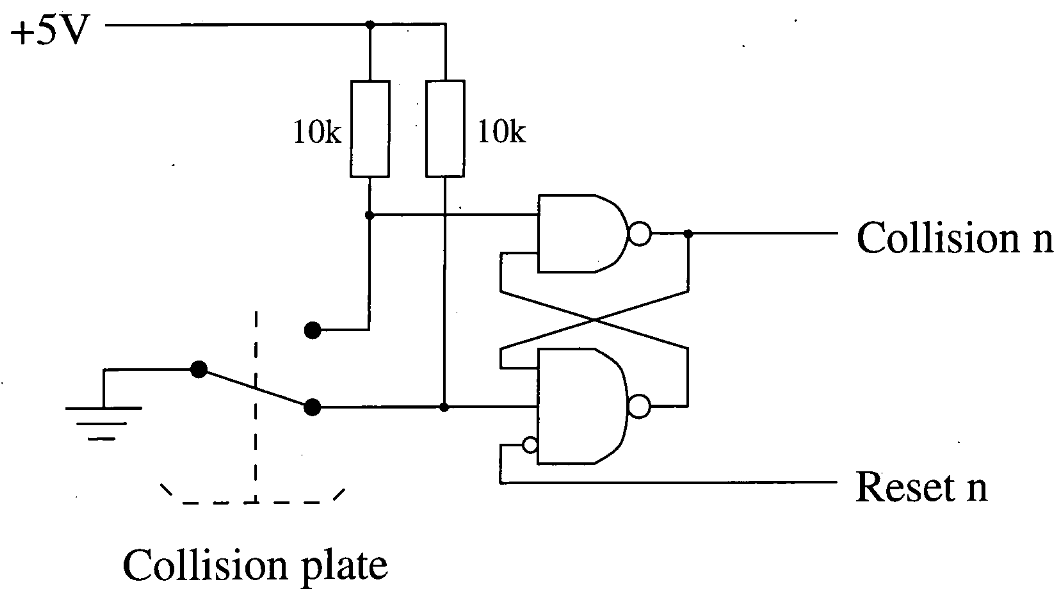


Figure 6.35: The collision detector diagram

6.4.9 Other Hardware Facilities

There are two output registers which are called the LED register and the output register, at PIA addresses C1 and C2(Hex) respectively. The LED register, as its name suggests, is used to drive the bank of 8 light emitting diodes on the control board. The other output register is of more general purpose. The lower 4 bits are used to drive the four phases of the stepper motor that rotates the head assembly and the upper four are unused. The LSB of the register activates phase A and setting any of these bits high causes the corresponding motor phase to be powered on. These can be sequenced in software to drive the motors.

There is also a red button mounted on the control board which can be monitored by software. This provides a limited form of signalling to the robot control system when it is operating autonomously. It can be used to indicate to the robot to commence a task.

6.5 The Computational Systems

The robot control system is based around the SA-29200 Demonstration Board [46]. This is an Am29200 RISC microcontroller–based system which is built on a single wallet sized card. It has EPROM for starting its operating system, 1 Mbytes of Dynamic RAM (DRAM) organised as 256K × 32 bits and an RS–232 serial communications port. Also there is an expansion interface which allows full access to the microcontroller's signals. This demonstration board offers a small, low power consumption, high performance embedded computer, which comes with software allowing it to be operated from IBM–AT compatible computers. Of the available computer systems its combination of functionality of a microcontroller design with ease of use, prompted its selection.

The demonstration board comes with debug monitor software which can be used to download programs over the serial line (RS–232) and execute them. The standard rate for data transferral is 9600 baud. However the EPROM which contains this development software has been reprogrammed with software which operates at 38400 baud. This has the advantage of quicker download times and faster communications. An important aspect for the autonomy of the robot is that the serial communication line can be disconnected and then reconnected without causing any system or communication errors. This can be done as long as no communication is attempted during the period of disconnection.

All programs that are to be executed on the board must be first compiled on the host computer using a cross–compiler. The serial line must be connected from the host to the board when the board is powered up. The MiniMON29K monitor software which is supplied with the system is then started. This establishes communication with the board after which the executable program can be downloaded and executed. If required the program can suspend communication to the host, the RS–232 can be temporarily disconnected, leaving the robot to operate autonomously. The RS–232 can be reconnected later for downloading back to the host any run time information.

6.6 The Summary

This chapter has examined in detail the design and considerations behind the construction of the prototype mobile robot. After showing why it was necessary to transfer the problem from simulation to a real robot, the breakdown of the design into physical, electronic and computational elements was described. Although highly interlinked, each of these elements could be considered to function at a different level of abstraction.

The description of the mechanical chassis covers the reasons for the selection of the syncro–drive assembly to the full descriptions of all the mechanical assemblies. In the electronics section the low level computer interfaces are examined, followed by descriptions of the systems behind the robot's actuators and sensors. Finally the robot's 29K computational systems are overviewed.

So far only a description of the robot systems has been documented. In the next section many of the robot's systems will be subjected to basic tests to assess their performances. This commissioning of the robot allows the design requirements to be validated. This will also generate in–depth knowledge of the functions and potential interactions of the many sub–components of the system.

Chapter 7

The Robot Hardware Assessment

7.1 Introduction

Before any of the experiments on the robot could be conducted, it was necessary to undertake an accurate assessment of it sensors and actuators. This was carried out, not only to calibrate the various sub-systems, but to develop a greater understanding of the physical interactions of the robot with its environment. Only through a detailed knowledge of this interaction could the subsequent actions of a robot controller be fully understood.

The first set of tests examined the basic mobility of the robot including its performance on different floor surfaces, and an assessment of the odometric system's accuracy. Subsequently the active range measurement system was investigated, looking at the effects of crosstalk between sensors and the ability of the Error Eliminating Rapid Ultrasonic Firing (EERUF) scheme to reveal this. Another factor which affects the capability of these sensors to operate effectively is the reflectivity of the environmental walls. It was therefore necessary to take relative differences between wall surfaces into account. Finally, the magnetic field sensors were calibrated and adjustments were then made to improve their measurement accuracy.

7.2 The Assessment of the Robot's Mobility

There are two general purpose motion control ICs used for controlling the movement of the robot within its environment. One of these creates the drive signals for the stepper motor, which controls the direction of travel of the robot. The other IC controls the DC motor which provides propulsion. These controller ICs are highly versatile and can be operated in different modes dependent on the specific application. These modes, where applicable, will be described below. The full specifications for these devices are contained in the technical data [47].

The most basic mode of operation is the *Position Control Mode* which performs point to point movement between the actual position and a given a command position. This is achieved internally on the IC by calculating the positional error and then applying full digital lead compensation to create a motor command output. The selection of the compensator's parameters is control situation specific, which can lead to different sets of parameters being used for different modes of operation.

There are also two modes for the control of the motor's velocity. The first of these methods is the *Proportional Velocity Control.* In this mode a 16 bit command velocity is compared with the actual computed velocity, and the motor command is driven in proportion to the error between these values, using the gain value K. The second is the *Integral Velocity Control Mode* which internally implements velocity profiling through position control. In this mode both a 16 bit velocity command can be specified along with an 8 bit desired maximum acceleration. The fundamental difference between the two modes is that, in integral velocity mode, there is zero steady state error in the velocity. However it is more difficult to attain closed loop stability.

The final control mode is the *Trapezoidal Profile Mode* which is a point to point positioning mode. In this mode a desired acceleration, maximum velocity and a final position are specified and the controller performs velocity profiling to move to the final position. If the maximum velocity is reached, the profile is trapezoidal with the velocity being accelerated and decelerated within the desired acceleration limits. Otherwise if the maximum velocity is not reached, the profile is triangular. The trapezoidal profile is implemented internally, using the integral velocity profile mode, and therefore there is no steady state error in the velocity values. There can however be steady state error in the final position.

Not all the modes are used by the controllers of the DC and stepper motors. The stepper motor that controls the point to point positioning of the wheels is a relatively high friction low inertia system which can attain maximum angular velocity in a short period of time. It is therefore more suitable to control this with the position rather than the velocity control mode. Also because of the inertia and friction there would be no advantage in using trapezoidal velocity profiling. However the use of the different modes for the DC motor would be very application specific. In this system where large distances might be covered, the inertia is a more important factor than the friction. Therefore the use of position, velocity, and velocity profiling become important. This is especially so in the case of velocity profiling, where maximum desired acceleration limits can be specified to avoid wheel slippage.

In both the position control, integral velocity, and trapezoidal velocity profiling modes full digital compensation is performed by the controller. This compensator $D(z)$ has the form of equation 7.1, where K is the digital filter gain, A is the digital filter zero and B is the digital filter pole. Each of these values has a range of 0-255 and can be individually programmed into the controller. The selection of these parameters is therefore dependent not only on the desired response of the system but also on the actual system itself.

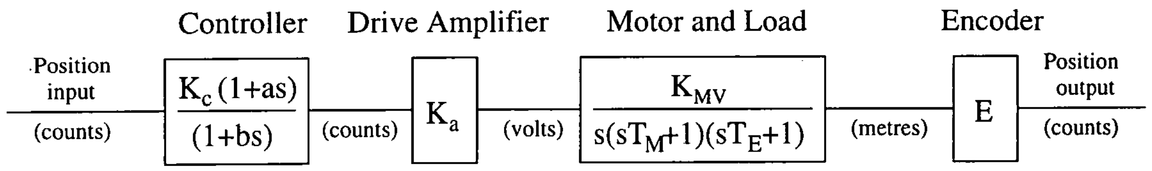


Figure 7.1: The transfer function of the system

$$D(z) = \frac{K\left(z - \frac{A}{256}\right)}{4\left(z - \frac{B}{256}\right)} \tag{7.1}$$

In the case of the steering mechanism it is difficult to model the system adequately as it is highly dependent on floor surface friction and the robot's motion. To compound this problem the stepper motor's speed saturates to its maximum value in about 10 ms. It was therefore decided to use the controller as a proportional controller by setting A and B equal to zero with a gain selected empirically. Low values of gain would not keep the wheels close to the forward direction when moving, however too high values caused oscillations. A gain value of 40 was selected which was below the oscillatory value for both smooth and carpeted floor surfaces but which was high enough for wheel alignment.

The DC motor and its associated system could be far more profitably modelled. To be able to determine the open loop transfer function, the components in the system model had to be calculated. As the sampling time is small it was decided to model the system in the s-plane utilising analogue design techniques. The following determination of the open loop transfer function is as suggested in technical data for the controller [48]. In the literature the controller's parameters were calculated by the combination method. It was decided however that the design would be by examination of the root locus in the s-plane followed by a mapping into the z-plane.

The open loop transfer of the controller and the system model is shown in figure 7.1 which starts with the lead compensator's $G_c(s)$ input, which is the position command. $G_c(s)$ is shown in equation 7.2 and has a zero at $-\frac{1}{a}$, and a pole at $-\frac{1}{b}$ with a gain of K_c. Here the output of the controller is the PWM motor command which forms the input to the H bridge power amplifier.

$$G_c(s) = \frac{K_c(1+as)}{(1+bs)} \tag{7.2}$$

The transfer function of the power amplifier can be determined by relating the average voltage produced by the H bridge to the mark to space ratio or duty cycle of the corresponding PWM wave–form. The calculation of the gain is shown in equation 7.3 from the maximum and minimum outputs of the amplifier are ±12 for duty cycles of ±100 counts.

$$
\begin{aligned}
K_a &= \frac{[\text{Maximum Voltage Output}] - [\text{Minimum Voltage Output}]}{[\text{Maximum Duty Cycle}] - [\text{Minimum Duty Cycle}]} \\
&= \frac{[12] - [-12]}{[100] - [100]} \\
&= 0.12
\end{aligned}
\tag{7.3}
$$

The transfer function of the motor $G(s)$ which is driven by a voltage source is dependent its mechanical time constant T_M (seconds), electrical time constant T_E (seconds) and its gain constant K_{MV} (metres/volt-sec). Normally the units of the gain constant are rads/volt-sec. However the model of the motor and load and the subsequent encoder have been described in terms of distance travelled over the ground in metres. The mechanical time constant of the robot was experimentally calculated by measuring the time taken for it to reach 63% of its full velocity which was 0.15 seconds. As $T_E \ll T_M$, the T_E term was neglected simplifying the equation further. A value for K_{MV} can be defined as the velocity of the robot when the DC motor is being driven with 1 volt across its terminals. Taking the maximum velocity of the robot to be around 30 cm/s at 12 volts makes $K_{MV} = 0.025$. Equation 7.4 shows the derivation of the motor's transfer function $G(s)$

$$
\begin{aligned}
G(s) &= \frac{\text{Position Output}}{\text{Voltage Input}} = \frac{\theta(s)}{v(s)} \\
&= \frac{K_{MV}}{s(sT_M + 1)(sT_E + 1)} \\
&= \frac{0.025}{s(0.15s + 1)}
\end{aligned}
\tag{7.4}
$$

The final element of the open loop transfer function E is the encoder which converts the distance travelled by the robot in metres into quadrature counts. E was calculated to have a value of 10575 (counts/metre). Therefore the transfer function of the system G_p without the controller is shown in equation 7.5.

$$
G_p(s) = \frac{31.725}{s(0.15s + 1)} \tag{7.5}
$$

From these transfer functions it was possible to design different controllers using standard s-plane design techniques to produce a range of responses for test purposes. After the controller has been selected, the analogue controller can be mapped into the z-plane to be implemented on the digital controller. The mapping was performed by using Tustin's Bilinear Rule shown in equation 7.6. With such a fast sampling time $T = 0.002048$ (seconds) and dealing with low frequencies the effects the use of pre-warping would be significantly less than the errors introduced by having to quantize the values of A, B and K into the range 0-255. Therefore no pre-warping was performed.

$$s = \frac{2(z-1)}{T(z+1)} \tag{7.6}$$

The above analysis assumes that the system is linear, but this is not the always the case. The main non-linear effect is caused by the voltage limits within which the amplifier must supply the motor. Lesser effects are backlash in the drive transmission to the wheels, slippage of the wheels, and static friction. Therefore, although the linear modelling of the system cannot give adequate account of all the system's behaviour, it can provide useful information during predominately linear operation. An important example would be the final stage of positioning control when the motor is not operating in saturation.

Having modelled the system the required response must be determined. One of the most basic requirements of this control system is that it should be able to reposition the wheels given a small positional disturbance. This condition can occur when the robot is on a frictional surface and attempts to turn the direction of its wheels using its stepper motor. Under such circumstances instead of the wheels processing about a circle beneath the robot, this extra friction fixes the wheels, and the robot's frame then gyrates with respect to the ground. This unwanted effect can be eliminated by a correctly tuned control system for the DC motor. Consider the case when the robot attempts to change the direction of its wheels by turning them clockwise. If there is sufficient friction the wheels will not move in a circle but instead force the robot to rotate about the point of contact with the ground. This rotation will be clockwise and force the encoder wheel to rotate around in a circle beneath the robot. As this encoder is the feedback monitor for the drive control system, this disturbance of its position will cause the DC motor to be powered to counter this backward motion of the encoder. Hence the drive motor will provide forward motion, forcing the wheels off their point of contact with the ground, and causing them to rotate beneath the robot. If this response is fast enough the encoder will be forced back towards its initial position and hence the frame of the robot will be actively fixed in place. The wheels now describe a circle beneath the robot being driven by the DC motor to overcome the effects of friction. This however relies on the drive control system having a fast response.

In the tests three different control set–ups, or cases, were used to produce different responses in the system. They have been numbered 1 to 3. Case 1 is a purely proportional controller with A=0, B=0 and the minimum gain of $K = 1$. From the calculated theoretical model of the system this produces closed loop poles at $s = -3.33 \pm 6.46j$ with a damping factor $\zeta = 0.46$ and natural frequency $\omega_n = 7.27$ rads/s where $K_c = 0.25$. Which should give a percentage peak overshoot of $\hat{p} = 100 \exp^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}} = 19.6\%$ and a 5% settling time

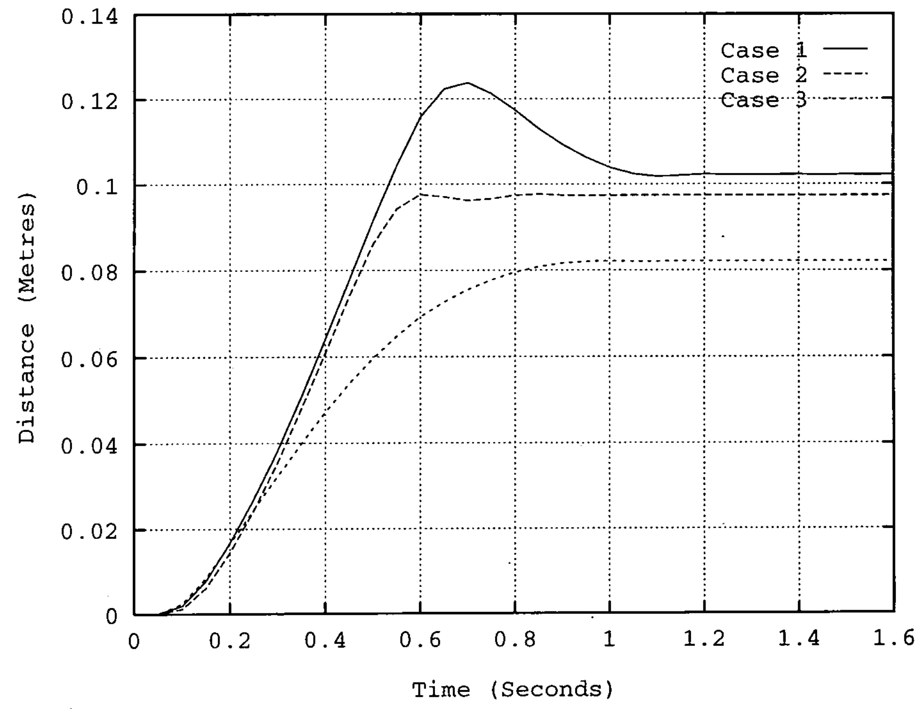


Figure 7.2: The robot's step response at different gain values

of $t_s \approx \frac{3}{\zeta\omega_n} = 0.90$ seconds. Hence the range of possible values of K means that only an underdamped purely proportional controller can be created.

To compare the performance of the real system with the model, a step input of 0.1 meters was given to the closed loop system with the controller set–up as case 1. As the system has been modelled as a second order system, estimates can be made as to the rise time $\hat{t}_r = 0.166$ seconds (using simulation), percentage peak overshoot $\hat{p} = 19.6\%$ and time to first peak $\hat{t}_p = 0.486$ $(\frac{\pi}{\omega_d})$ seconds can be predicted and compared to the recorded values. The step response of the robot under the control of case 1 is shown in figure 7.2 and from this the following values were taken: $t_r = 0.48$, $p = 24\%$ and $t_p = 0.64$. Although t_r is greater than $\hat{t}_r$ this could be explained by saturation of the motor's response, however $\hat{t}_p - \hat{t}_r = 0.166$ is approximately equal to $t_p - t_r = 0.16$ which is in the linear range. The shape of the recorded response is very similar to that produced using the control software package CODAS. One observed difference in the tests is that the final value is not as predicted, but this is probably related to static friction and backlash. Hence it appears that the results from the system model correlate well with the observed response in the linear region.

In case 2 a lead compensator was designed to improve the performance of the proportional controller. As the motor has two poles one at the origin and one at -6.66, the controller's zero was placed at a value of -10.0. The controller's pole which must be positioned further to the left for a lead compensator was provisionally placed at -2000. This configuration bends the root locus in a circle around the zero and allows for decreased settling times at higher values of damping factor and therefore smaller values of overshoot. With values $A = 251$, $B = 88$ and $K = 82$ we get $K_c = 0.2979$ and poles at $s = -6.49 \pm j4.59$ giving $\omega_n = 7.95$ and $\zeta = 0.816$. This gives theoretically estimated values of a 5% settling time of $\hat{t}_s = 0.46$ seconds and a peak overshoot of $\hat{p} = 1.18\%$. This can be compared against the empirical results shown in the step response of figure 7.2 indicated as case 2. It can be seen from the test that the response is significantly faster than case 1 with a much reduced overshoot. However there is still some steady state error that could well be attributed to the non-linear effects and the real system being higher than a second order system.

Finally a value of gain for the above system $K_c = 0.0836$ was selected to produce a critically damped response. Keeping $A = 251$ and $B = 88$ the value of K_c relates to $K = 23$, this corresponds to closed loop poles at $s = -3.927$ and $s = -4.512$. It is possible to calculate the rise time based on analysis of the effect of the two dominant poles of the system. This is defined as the time to go from 10% to 90% of the final value and was calculated as $\hat{t}_r = 0.81$ seconds. The actual response only reaches a final value of 0.082 but the rise time t_r measured between 10% and 90% of this value was 0.55 seconds. It can therefore be seen that at this lower value of gain the frictional effects that have not been modelled are starting to become significant.

In the above work the system has been modelled and three different sets of parameters have been selected to test both the accuracy of the model and the response of the system. These three sets of parameters will now be discussed in relation to the normal operation of the robot. Two types of operation of the robot will be considered; the response of the control system during a turn, and the response during trapezoidal velocity profiling.

Tests were conducted to determine the ability of the three sets of control parameters to compensate for positional disturbances, thus keeping the robot in position during a turn. The robot was placed on a smooth surface and the wheels were rotated through 180^o whilst the error of the drive control system was being monitored. Figure 7.3 shows the results of the tests for the three cases 1 to 3. One important factor to note is that as soon as the wheels stop being turned these small values of the error stay fixed. This is most likely the effect of static friction and hides the further movement of the drive controller. It can be clearly seen that the turned controller case 2 provides the superior performance. Case 1 which is the underdamped system does not adequately drive the response down, but performs considerably better than the overdamped response of case 3. This is an important result as positional errors incurred during turns will effect the accuracy of the robot's odometric estimate of position. Therefore the parameters of case 2 are the most suitable for controlling the robot's drive controller during a turn.

Another important aspect of the drive system's performance which must be investigated is the trapezoidal velocity profiling mode. This also uses the full digital compensation feedback and a set of parameters must be chosen that produces the most suitable performance. Unlike

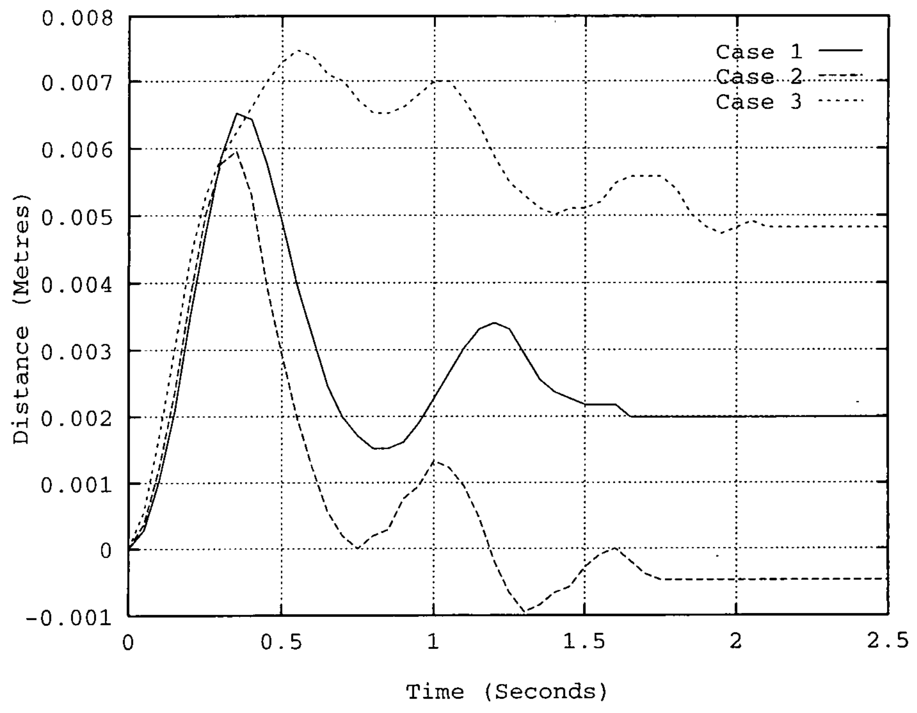


Figure 7.3: Displacement of the drive system for different controllers

the position to position control of the drive system, the trapezoidal mode is not fully documented in the literature. Details of how the maximum velocity and the maximum acceleration values are combined to set the command position are not included. However it is stated that integral velocity profiling is performed. Hence not all the information that would be required to theoretically predict the behaviour is available, and therefore the parameter selection was performed based on observation. It was found that underdamped system behaviour caused a marked oscillatory acceleration and deceleration phase. During this period the input to the control system must consist of a set of ramp functions with steadily increasing or decreasing gradients. Each change in gradient would produce a new transient response which would explain the observed oscillatory nature. However an overdamped system, although it would have a greater time lag in response, would not overshoot and would therefore produce a smoother change in velocity.

Figure 7.4 shows the results of the trapezoidal velocity profiling mode with a specified maximum acceleration of 1 and a maximum velocity of 7 (which is the largest value of the velocity at which the drive motor can operate). The distance over which the robot moved was 1.5 metres, which is long enough for the robot to reach its maximum velocity. The result of case 2's response shows the oscillatory acceleration in contrast to case 3 where the critically damped response is much smoother. Although the response is smoother, there is a greater time lag in the response. Due to the smoother acceleration and deceleration characteristics of the case 3 controller, it was selected for use when the robot was in trapezoidal velocity

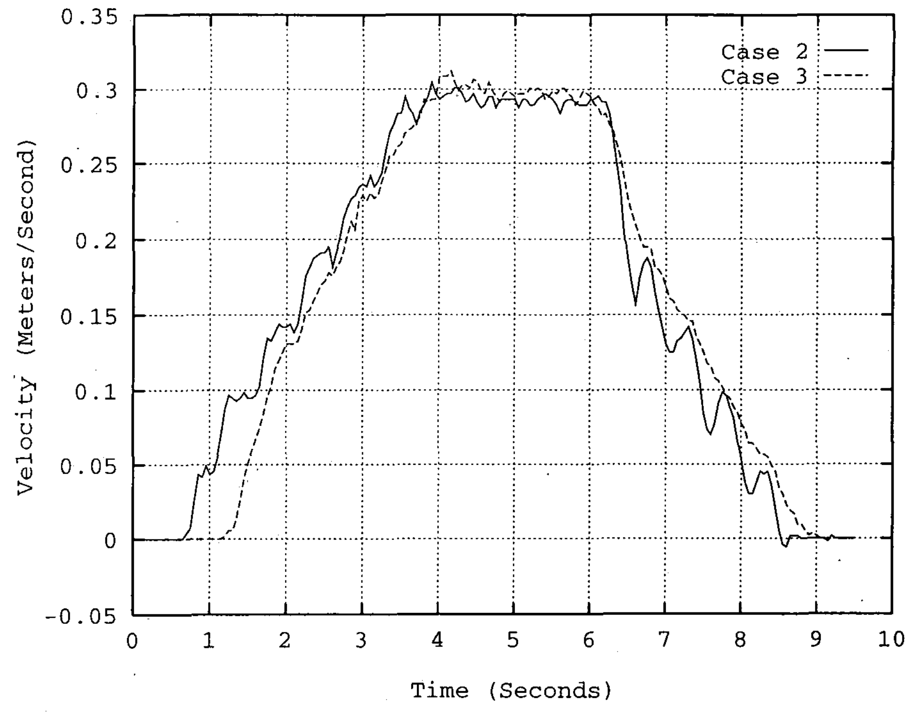


Figure 7.4: Trapezoidal velocity profiling for different controllers

profiling mode.

In summary, there are different sets of controller parameters that best suit different modes of controller operation. When the robot is using the position control mode the parameters should be set to case 2. However, when the robot is in the trapezoidal velocity profiling mode the parameters should be set to case 3.

The odometric measurement system produces a positional estimate based on the information gained from the drive and steering encoders. The simplest model for determining the position of the robot assumes that the forward distance relates to straight line motion and that the steering angle relates this angle of motion to some fixed starting datum. Both these assumptions will be examined in further detail to assess the accuracy of this model and provide suggestions and limitations for improvement.

Before any tests were conducted the wheels of the robot were aligned. Although there is little backlash on the steering drive transmission there will always be some small discrepancies in the angles of the wheels. In order to calculate this, the robot was placed on graph paper and the true direction of each wheel was drawn as a line. From this the true relative angles of two of the wheels to the first were calculated at 0.24^{o} and 0.48^{o}.

An important assumption to test is that the robot moves in straight lines during purely forward motion. In order to do this a felt-tipped pen was mounted to one side of the robot and allowed to rest on a long sheet of paper stuck down to the floor. The robot was driven

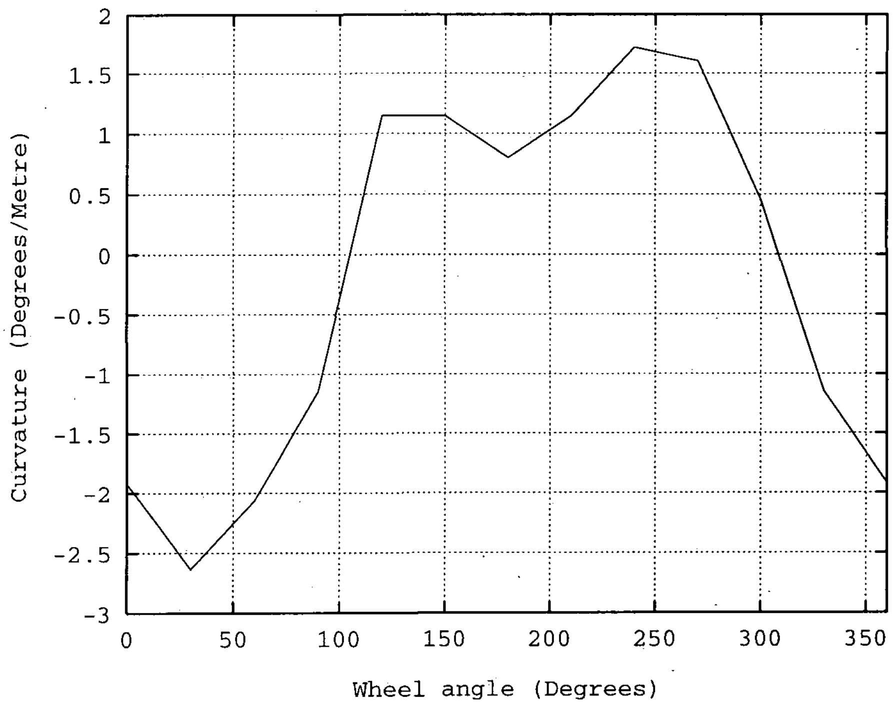


Figure 7.5: Robot curve (degrees/metre) at different wheel angles

forward for certain distance and its path could be seen on the paper. Initially the robot was driven under trapezoidal velocity profiling mode with the desired maximum velocity set at its minimum value of 1 which is approximately 4 cm/sec. From the trace on the floor the amount of curvature per metre could be assessed. This test was then repeated for different angles of direction of the robot's wheels.

The results of this test for systematic error in the drive mechanism are shown in Figure 7.5. The most striking feature is that unlike a differentially driven robot chassis (two wheels operating on either side of the robot and independently driven) the curvature of the robot's path is dependent on the direction of travel. Although the maximum curvature looks high because it has both positive and negative components, its effects are often masked by the robot moving in different directions. The major factors that could give rise to this curvature will now be discussed.

The most obvious cause for the directional curve is the misalignment of the wheels in the initial set–up. It is not simply the initial set–up which causes misalignment in the direction of the wheels. This direction is maintained by the belt transmission from the stepper motor and, as each belt has elastic properties, its extension will be related to the force that it transmits. This transmission force being that required to hold the wheel housings in directional alignment. The angular forces are caused by floor friction associated with the robot moving at a constant velocity or undergoing acceleration. To compound the problem each wheel has a different length of belt leading back to the stepper motor and therefore the distribution of forces back to this motor will cause belt extension and misalignment. Hence the misalignment will be dependent on factors such as floor friction and the accelerations of the robot.

Misalignment explains curvature in the robot's direction of travel, but it cannot in itself explain the change with respect to the direction of the wheels. This can be explained in terms of wheel slippage. Each of the wheels when it is not slipping provides a propulsive forward motion on the robot. If there is any misalignment in the wheels this will cause conflicts in the direction in which each wheel should move. This conflict will result in slippage in some of the wheels. The forward direction will then determined by the static friction from the non-slipping wheels and the dynamic friction of the slipping wheels. Which wheel will slip and which will not, is in turn dependent on the downward forces perpendicular to the surface of the floor. This force, experienced by a wheel, is related to its position beneath the robot. However, with the wheels mounted off axis, this position beneath the robot depends on its direction. Hence the downward force on each wheel can be expressed as a sinusoidal function dependent on wheel direction. Hence for different directions, different wheels slip, and the curvature of the robot's path changes.

One possible method to reduce this problem is to attempt to model the system and compensate for its effects. This could be achieved by using a 'look up table' made from detailed measurements of the robot's performance or by mathematically modelling the system and calculating any discrepancies. The advantage of a 'look up table' is that it would be fast to implement in software but it would require extensive tests over different velocities, accelerations and floor surfaces. The problem with modelling is acquiring a model which adequately models the system without being too computationally expensive. The underlying problem with both these methods is that a poor compensation scheme can introduce additional error thus compounding the problem. For this reason corrective compensation was not used in the further experiments, but this is an important sensor area for further investigation.

The above analysis has only looked at those errors which are introduced into the odometric positional estimate by systematic errors. There can be no modelling for non-systematic errors which occur unpredictably and not as a direct result of any robot actions. At some level of accuracy the odometric system must be accepted as having inherent errors which can only be coped with as part of a larger control strategy.

7.3 The Assessment of the Ultrasonic Sensor System

In this section the actual operation of the ultrasonic sensor system will be examined in order to determine the operational characteristics that can be expected. Initially basic tests will be performed on the sensors, assessing the typical distance errors and the minimum ranges of both the short and the long mode. Following this, the reflectivity of different surfaces to ultrasonic frequencies will be discussed. Finally the method of firing called EERUF, by which the crosstalk between the ultrasonic sensors can be eliminated, will be assessed.

Initially the basic performance of the individual Polaroid sensor must be examined. As a design feature the robot's sensory system incorporates two modes of operation of the ultrasonic sensors, these are the long and the short range modes. The long range mode operates the sensor as originally designed, with a minimum range of about 40 cm, whereas the short range mode is designed to operate down to around 10 cm.

The following test was designed to assess the accuracy of the two systems and their minimum ranges after calibration. The calculation of the range distances depends on a value for the speed of sound which was calculated for a temperature of 20^oC at 343.11 m/s. Figure 7.6 shows the error in the range results for the two systems as a function of distance from the target. The short range system had a minimum range of 10 cm whereas the long range had a minimum of 40 cm. It can be seen from the figure that both systems produce the same high accuracy distance values from 70 cm and above. From 70 cm and below the short range mode has an increasing error but even at its maximum is only -1.55 cm. The above results are after calibration of the sensors at a known temperature as this affects the speed of sound and hence the absolute accuracy. The temperature change indoors is not sufficient to adversely affect the results and therefore these affects were not investigated. The maximum ranges of both modes depends on the scattering properties of the reflecting surface. In fact both modes of the sensor were able to detect distances up to the documented maximum range of 10.5 metres. It was found that the long range mode would give more constantly accurate results from poorly reflective surfaces that the short range mode could not detect.

One of the problems associated with a ultrasonic sensors for range measurement is that this relies on the acoustic wave being reflected directly back from the object. This depends on the angle of incidence and the reflective surface. For a given wavelength a surface falls into one of two categories; smooth or rough. A smooth surface reflects all the incident energy specularly in a single direction whereas a rough surface will reflect the energy in various directions at different magnitudes. This property is frequency dependent, as the transition between smooth to rough occurs when the surface irregularity dimensions cause the wave–front to interfere with itself. Hence a surface that appears smooth for one frequency could appear rough to a higher frequency. To calculate the transition between rough and smooth surfaces we can use the 'Rayleigh Criterion' [49] which states that a surface is considered smooth for

$$h < \frac{\lambda}{8\cos\gamma} \tag{7.7}$$

Where h is the height of the surface irregularities, γ is the angle of incidence and λ is the wavelength of the sound source. The ultrasound is at a frequency of 50 kHz and therefore

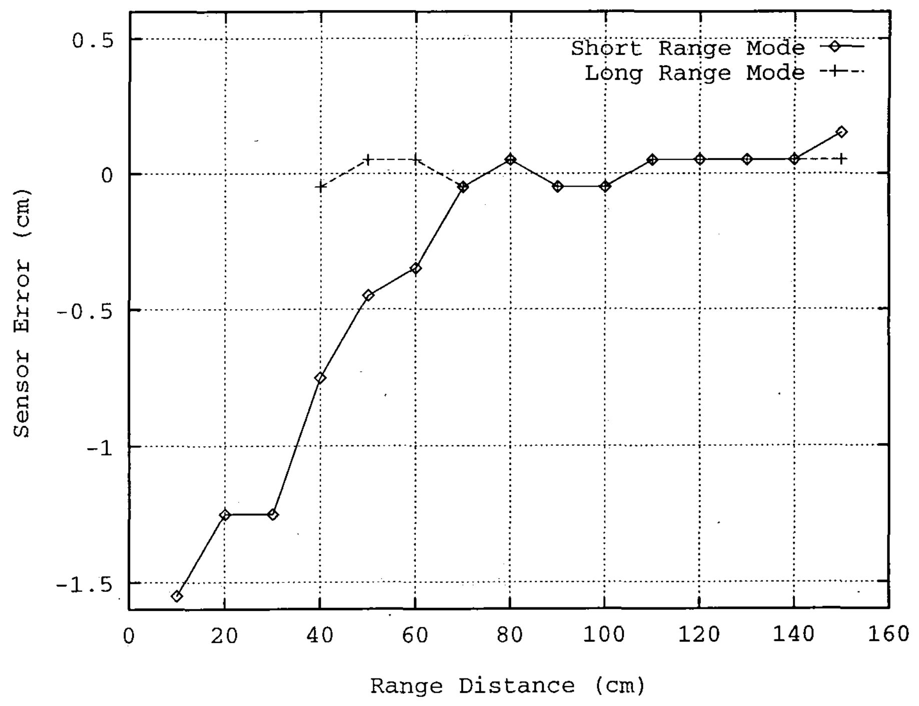


Figure 7.6: The sensor error recorded as a function of distance measured

$\lambda = 0.00686$ (m). Consequently if the surface irregularities are below 0.857 mm then this will appear smooth for all angles of incidence. A 'normal' painted wall has irregularities far below this value and will therefore be smooth for the ultrasonic sensors.

In the preliminary tests of the robot, its environment needed to be constrained to match the simulated conditions. It was important that the robot's responses could be seen in simple environments with straight walls and acoustically visible objects. Therefore these walls needed to be able to reflect at acute angles and hence be rough. To achieve this, corrugated cardboard was taken and its backing removed. This creates a surface which has regular undulations with a depth of $h = 3$ mm. Which from equation 7.7 makes a surface which will appear rough at angles of $\gamma < 73^o$. This material was used to line the environments of which the robot then used in all the main tests.

To enable the robot to detect crosstalk between its sensors or interference from the sensors of other robots the EERUF scheme was employed. The EERUF is based on firing the sensors at certain fixed times with respect to some synchronising pulse. The selection of these firing times is based on some assumptions of operation. To evaluate the system the firing times of the sensors were calculated based on the set of example values recorded in the original paper that set out this method.

Although EERUF can eliminate crosstalk [44] it is preferable to reduce the chances of this occurring. In the original paper this was achieved by introducing time lags between the

Sensor	1	2	3	4	5	6	7	8
T_{lag} (ms)	0	25	50	75	100	125	150	175
$T_{\text{wait,a}}$ (ms)	24	24	24	24	24	24	24	24
$T_{\text{wait,b}}$ (ms)	18	12	6	0	20	14	8	2
$T_{\text{fire,a}}$ (ms)	24	49	74	99	124	149	174	199
$T_{\text{fire,b}}$ (ms)	18	37	56	75	120	139	158	177

Figure 7.7: Table of firing times for the sensors operating EERUF

firing of sets of four neighbouring sensors. This was empirically found in their set–up to minimise the crosstalk errors, whilst minimising the complete firing time of all the sensors. In the experiments on this robot different time lags will be introduced for all the sensors, thus making no assumptions as to the potential interaction of the sensors.

Figure 7.7 shows the basic firing times that were used in all the robot tests. Initially the period over which the sensors are to be fired is chosen at 200 ms. Then each sensor has its associated time lag T_{lag} which are spaced at 25 ms intervals which reduces the chances of crosstalk. Added to these time lags are the alternate wait times $T_{\text{wait,a}}$ and $T_{\text{wait,b}}$ to produce the firing times $T_{\text{fire,a}}$ and $T_{\text{fire,b}}$. This produces all the firing times for a period spacing of 200 ms which are split into alternate firing phases of a and then b. Hence a given sensor fires at $T_{\text{fire,a}}$ during period 'a' followed at $T_{\text{fire,b}}$ in period 'b'. All the values are scaleable and therefore it is possible to change the period interval and scale all the firing times accordingly. Therefore it is possible to increase or decrease the firing rate of all the sensors by simply scaling the firing times whilst still keeping the ability to detect crosstalk in the sensors.

In all the experiments this method has shown itself to be highly effective at detecting crosstalk between the sensors. The most basic test before the sensors were mounted on the robot involved pointing sensors 1 and 2 in the same direction separated by half a metre. The target was a sheet of card two metres away. The firing times for the sensors were set to their delay values $T_{\text{fire,a}} = T_{\text{wait,a}}$ and $T_{\text{fire,b}} = T_{\text{wait,b}}$ therefore without their time lags. Figure 7.8 shows the set–up for the three tests to confirm that the system was detecting crosstalk between the sensors. Test a) consisted of firing the sensors at the target. From the firing times it can be seen that the sensors both fire at the same time in the 'a' periods but in the 'b' periods, sensor 2 fires before 1. In this test sensor 1 rejected its false readings but sensor 2 recorded the correct value. In test b) a piece of card was used to direct the sensor 1's pulses onto 2. Here sensor 2 correctly rejected its distance value and sensor 1 recorded a correct value. Finally in test c) the pulses were separated and both sensor recorded correct distance measures.

General results from the EERUF as tested on the robot indicate that the larger the distances measured from the robot, the more likely rejection will occur. One effect which can

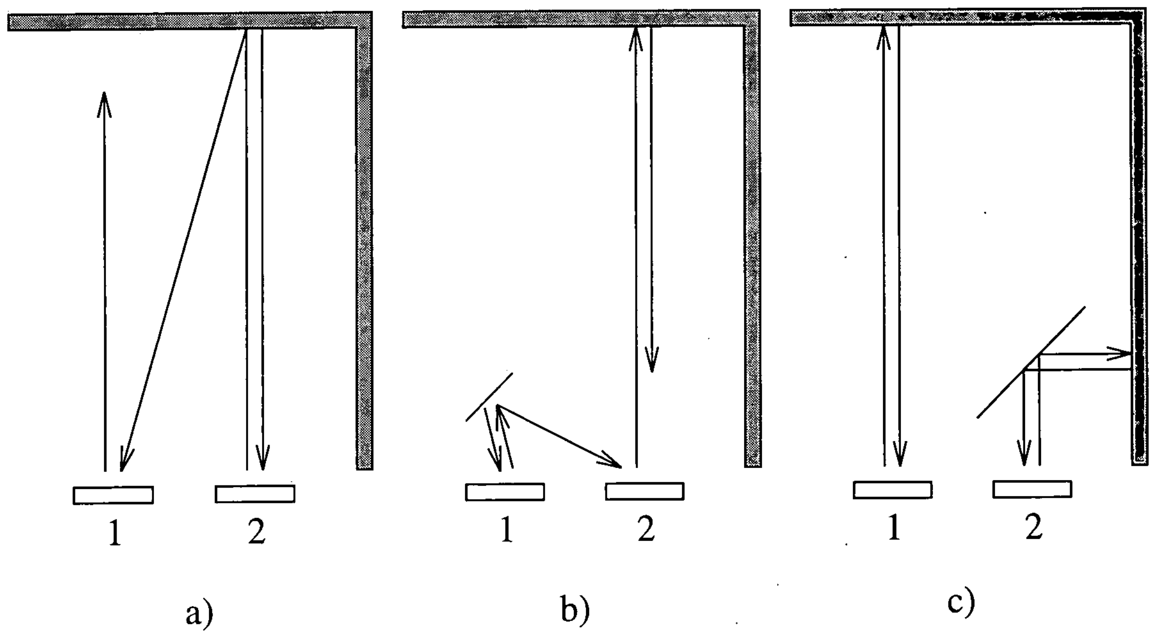


Figure 7.8: Tests for crosstalk rejection

cause this, is that residual echoes from other sensors are reflected around within the robot's environment. Any sensor which is measuring a large distance has its gain automatically increased by the electronics in the Polaroid sensor module. Therefore for larger distance measures there is a greater likelihood of the sensor detecting any residual ultrasonic reflections.

So far only static tests have been performed on the EERUF scheme, but for the robot it is important that the system can operate effectively whilst moving. This introduces extra complications, as the distances that the range sensors are measuring can be changing over time. As the EERUF method relies on monitoring the differences between sequential values of range measurement any extra change in distance over time will affect the system. Hence the faster the robot moves the greater its rejection rate becomes. However, it is possible to correct for this if the robot's velocity is known, which can be easily determined in software. From the velocity measurement and the angle of each of the sensors, it is possible to compute the expected change in the distance. This value can then be subtracted from the difference in the sequential range measurements. This method has been shown to reduce the rejection rate substantially whilst the robot is moving. Although it is reduced, it is still slightly higher than for a static robot.

The major parameter to be set on the system is the period interval over which the sensors fire. The default value for this is 200 ms, but by increasing this value the rate at which the sensors fire can be increased. However if this value is set too short rejection rates increase. Empirically a good compromise value of 250 ms was found. One option for further work that has not been examined here, is the idea of adaptive changing of the period interval based on the rejection rate. This would have the effect of dynamically tailoring the sensor firing rate to suit the local crosstalk conditions.

7.4 The Assessment of the Magnetic Sensor System

The magnetic sensor system consists of two perpendicularly mounted magnetic field sensors whose frequency varies inversely with the field strength. This frequency is monitored by the robot by counting the number of pulses for given interval (0.04 seconds). The two values for the sensors can then be used to determine the angle and magnitude of the magnetic field through the sensor unit. Described below is the method by which this system was calibrated and linearised, and shows the typical error limits on the subsequent magnetic field reading.

The sensor system was calibrated after it had been mounted on the robot and required no further adjustments. As the two sensors are attached to the robot's head, readings could be taken at 0.75^o angular interval over the head's moveable range of 315^o. The robot's head is moved by means of a stepper motor, and therefore care was taken to make sure that it was switched off whilst the sensor readings were being taken. Figure 7.9 shows the recorded number of counts from both sensors, A and B, as a function of the head angle. From the figure it can be seen that both the sensors' output consisted of roughly sinusoidal wave–forms which are not calibrated with respect to amplitude and zero offset. The response of the sensors is non-linear with respect to frequency and this can be seen as a slight elongation of the low values and sharper peaks.

The non-linearity can be corrected as described in the data sheet [45] which is shown in equation 7.8. In this equation H is the field strength for a measured period value of T, T_0 is the period at zero field and $T_{\min}$ is the period approximating to the maximum negative value of field strength in oesteds. The coefficients c_0, c_1 and c_2 are determined from the calibration procedure. As no calibrated reference value of field strength could be obtained it was assumed that the maximum values of detected field strength were 0.5 oesteds (the value of the field strength of the earth's magnetic field). This potentially introduces an error factor into the magnitude of the detected field but does not affect its angle.

$$H = c_0 + c_1\,(T - T_0) + c_2(T - T_{\min})^2 \tag{7.8}$$

To calibrate a sensor its values of the maximum, minimum and zero field counts must be read off the graph. These can be normalised by dividing by the zero field count when $T_0 = 1$. The three coefficients can then be found by substituting values H for the maximum, minimum and zero field counts and solving for c_0, c_1 and c_2. The values from the graph for sensor A were $T_0^a = 2654$, $T_{\max}^a = 3073$ and $T_{\min}^a = 2340$ giving the coefficients $c_0^a = 0.05367$,

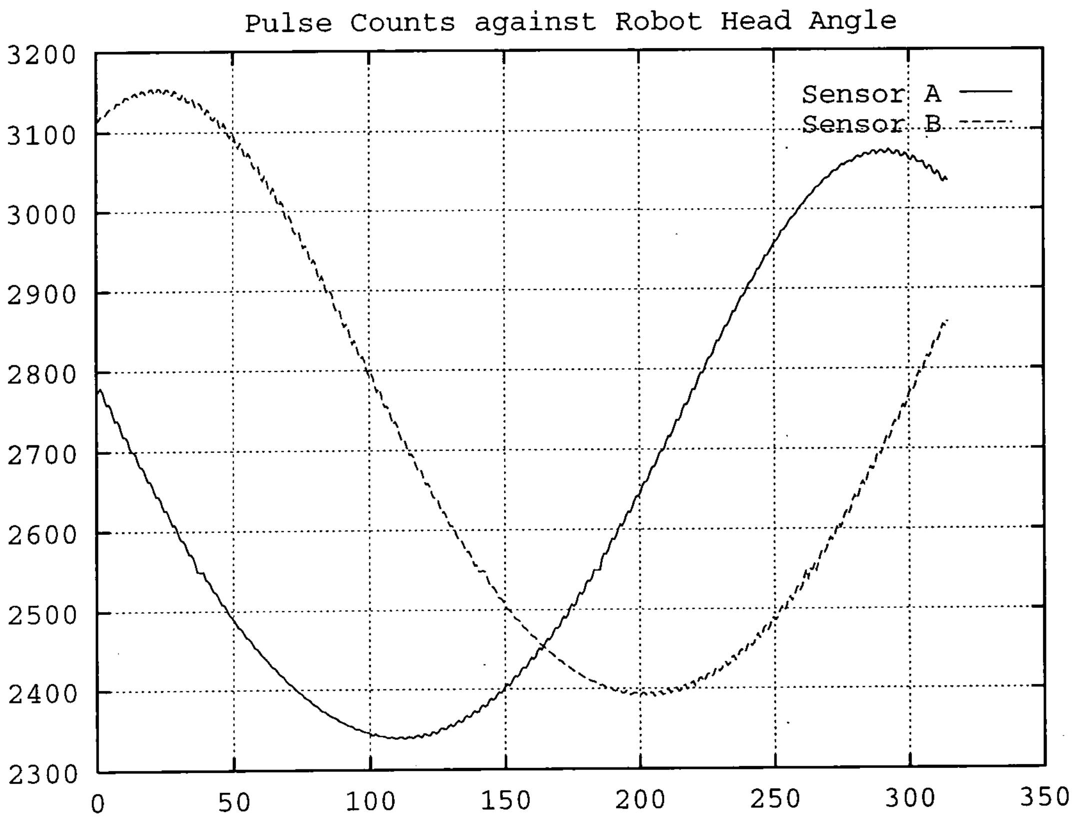


Figure 7.9: The output values from the two sensors A and B

$c_1^a = 4.6798$ and $c_2^a = -3.8345$. Similarly the values for sensor B were $T_0^b = 2715$, $T_{\max}^b = 3150$ and $T_{\min}^b = 2394$ giving the coefficients $c_0^b = 0.05360$, $c_1^b = 4.6829$ and $c_2^b = -3.9801$.

$$H^a = 0.05367 + 4.7698\left(\frac{T^a}{2654} - 1\right) - 3.8345\left(\frac{T^a - 3340}{2654}\right)^2 \tag{7.9}$$

$$H^b = 0.05360 + 4.6829\left(\frac{T^b}{2715} - 1\right) - 3.9801\left(\frac{T^b - 2394}{2715}\right)^2 \tag{7.10}$$

The values of the linearised field strengths H^a and H^b can then be found by substituting the count values from sensor A (T^a) and B (T^b) into equations 7.9 and 7.10 respectively. The results of applying this calculation to the test data is shown in figure 7.10. From this it can be seen that both sinusoidal wave–forms are phase displaced by 90^o and have maxima and minima of 0.5 and -0.5 oesteds respectively. The slight rounding affect at the base of the minima as seen in figure 7.9 has also been eliminated, and the curves have a more linear appearance. The angle of the field with respect to the robot α can be calculated using the inverse tangent as shown in equation 7.11.

$$\alpha = \tan^{-1}\left(-\frac{H^a}{H^b}\right) \tag{7.11}$$

To assess the accuracy of the final values of the sensor system its position must be compared to the angles at which they were detected. Figure 7.11 shows this error expressed in degrees as plotted against head angle. The most interesting phenomena is the repetitive undulation repeating in cycles of four. This is attributable to the effects of the permanent magnetic fields in the stepper motor used to drive the robot's head. This occurs in cycles based on the four magnetic states in the electrical cycle of the stepper motor. Without physically moving the sensor system higher up or using μ metal shielding this cannot be reduced further. The rest of the deviations are probably attributable to other magnetic influences on the sensor by the robot. These, however, are small and would be difficult to compensate for.

The magnetic sensor system has been shown to be accurate down to $\pm 2^o$ The main expected use of the magnetic sensor is for repositioning of the robot's head. This situation might occur if there was an sudden angular translation, or could be the result of slow angular drift. With its level of accuracy the magnetic sensor system could easily achieve this. Overall for the cost and simplicity of this sensor system the results obtained are very accurate and will be useful for the control of the robot.

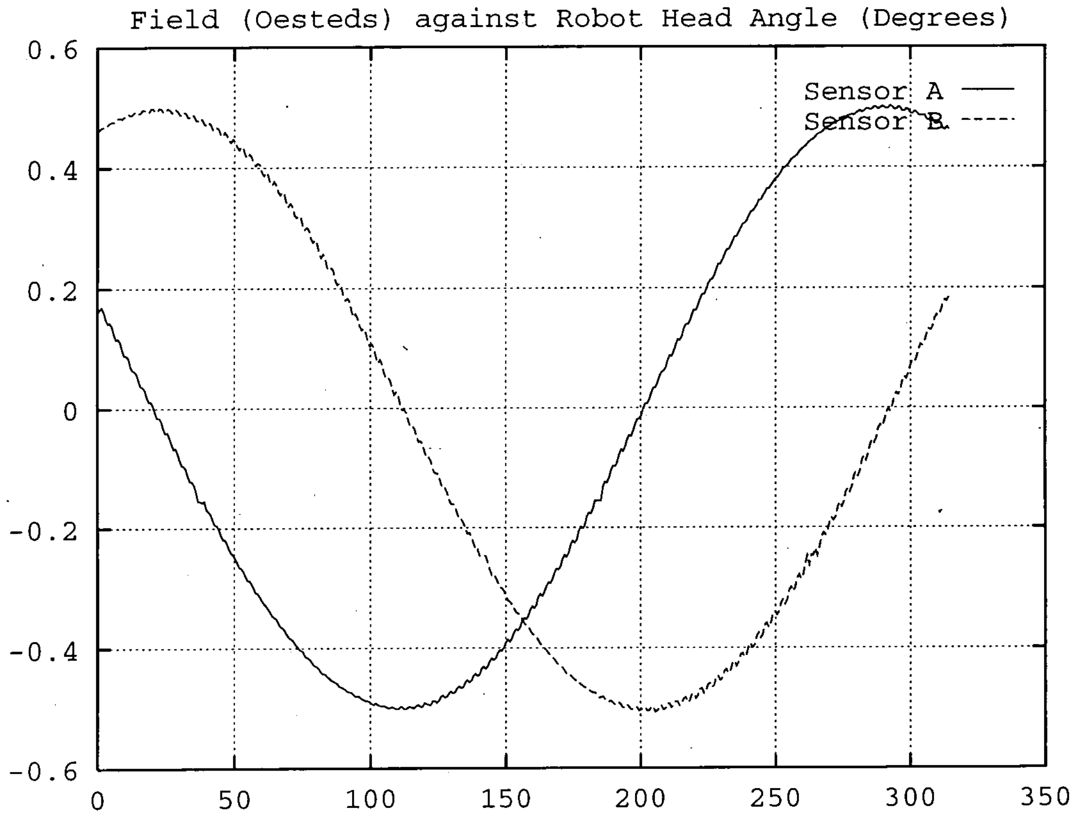


Figure 7.10: The calibrated field values from the two sensors A and B

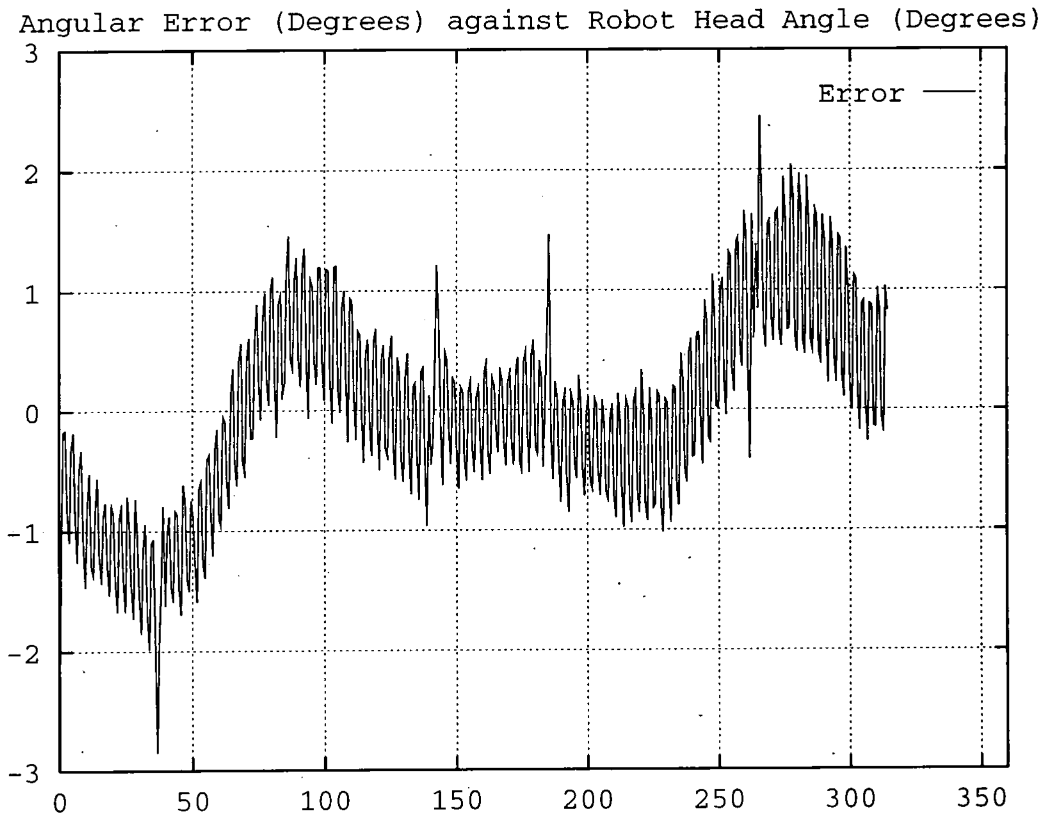


Figure 7.11: The error in the magnetic field sensor system

7.5 Summary of the basic tests

The above work has examined in detail the sub–systems that comprise the prototype mobile robot. The first section covered mobility, and the actions and interactions of the drive and steering control systems. It was shown that different control parameters were preferable for different control requirements. These can be substituted as required. An important observational result was that the robot's path was curved and this depended on the angle to which the wheels pointed. This curvature would introduce odometric error but this must be recognised as a mode of error inherent in odometric measurement.

The ultrasonic sensor system was then tested. As part of this the accuracy of the range measurements in both the long and the short modes of operation of the sensors was determined. Then the method of firing of the sensors, called EERUF, was tested to determine its ability to reject erroneous range measurements caused by crosstalk between sensors. The system was shown to be able to perform this even when the robot was moving.

Finally the magnetic sensor system was calibrated so that the non–linear operation could be compensated for, therefore improving its positional accuracy. It has been shown that even with the effects of the magnetic disturbances from the robot's chassis and electronics, the sensor is accurate to $\pm 2^o$ accuracy.

In conclusion, the sub–systems of the robot have been satisfactorily tested and their various modes of operation recorded. After these detailed interactions of the robot's mechanism were more fully understood, the experiments on exploration, mapping and navigation were then undertaken.

Chapter 8

Experiments on Robot Exploration, Mapping and Navigation

8.1 Introduction

This chapter contains the experimental results of tests on the control system used in exploration, mapping and navigation on the prototype mobile robot. The research was undertaken to complement the work previously conducted in simulation. Simulation allows swift development in the early stages of design. However the limitations of this method are exposed when the interactions between the robot and its environment become more complex. Only through testing these controllers under real conditions can more be learnt about the control of real robots.

A major factor affecting real robotic systems is noise and inaccuracy in sensors and actuators. Initially basic tests were conducted to examine methods for counteracting the problems of odometric noise in the positioning system. This was conducted using a simple three node map.

Following this, the novel mapping method which uses a behaviour based controller for exploration was investigated for its ability to map out different environments. Again the problems associated with odometric error were examined but this time with respect to the larger topologies which had been constructed.

Finally a new method of searching the environment was devised and tested in the same environments as the previous mapping methods. As with the other experiments the ability of the robot's odometric systems to keep the robot positioned after exploration were examined. The work is summarised with the main conclusions drawn from the results of the experimental work.

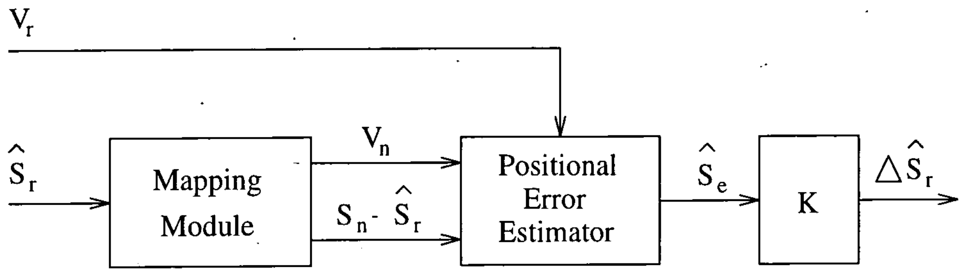


Figure 8.1: The correction mechanism for odometric drift

8.2 Re–synchronising the Robot's Map and Environment

Any mobile robotics solution that uses odometry to provide a relative positioning is subject to the problems of drift error. If this error is not reduced with reference to some other sensory system, the gulf between the robot's estimated and actual positions will render the previously gained knowledge useless. To reduce this error and therefore re–synchronise the robot's world representation, corrections must be made to the robot's estimated positional state. This estimated state can be described in terms of three component coordinates $\hat{x}$, $\hat{y}$ and the relative angular position of the robot's base $\hat{\theta}_b$. In all the following experiments these were taken with respect to the initial location where the robot started its actions.

For the robot's information to remain synchronised with the environment, the drift in all three of these components had to be corrected. This correction could only be made through comparison of the recorded state with current sensor information. To determine the $\hat{x}$ and $\hat{y}$ corrections the ultrasonic range measures were used and for the angle $\hat{\theta}_b$ the magnetic sensor was used.

Figure 8.1 schematically shows the processes by which the correction in the robot's state was determined. $\hat{S}_r$ is the vector which is composed of the estimates of the robot's $\hat{x}$, $\hat{y}$ and $\hat{\theta}_b$ states. This data was used in conjunction with the mapping information to determine the nearest node to the robot from which the node's range and magnetic field information V_n were ascertained. The positional error estimator uses this vector with the sensor information V_r and the estimated distance from the node $(S_n - \hat{S}_r)$ to produce an estimated positional error $\hat{S}_e$. This vector was then multiplied by a gain term K composed of an odometric correction factor $K_{x,y}$ and an angular correction factor K_θ. The x and y terms were multiplied by $K_{x,y}$ and the θ term was multiplied by K_θ. This gave rise to the correction vector $\Delta\hat{S}_r$ which was added to the estimated robot's state $\hat{S}_r$. Therefore the gain factor K determined the rate by which the robot could react to changes in its environment. This process of re–correction was performed each time the robot's sensors were fired.

In order to validate the robot's ability to re–synchronise its knowledge to an environment experiments were performed to observe correction of translational and rotational dis-

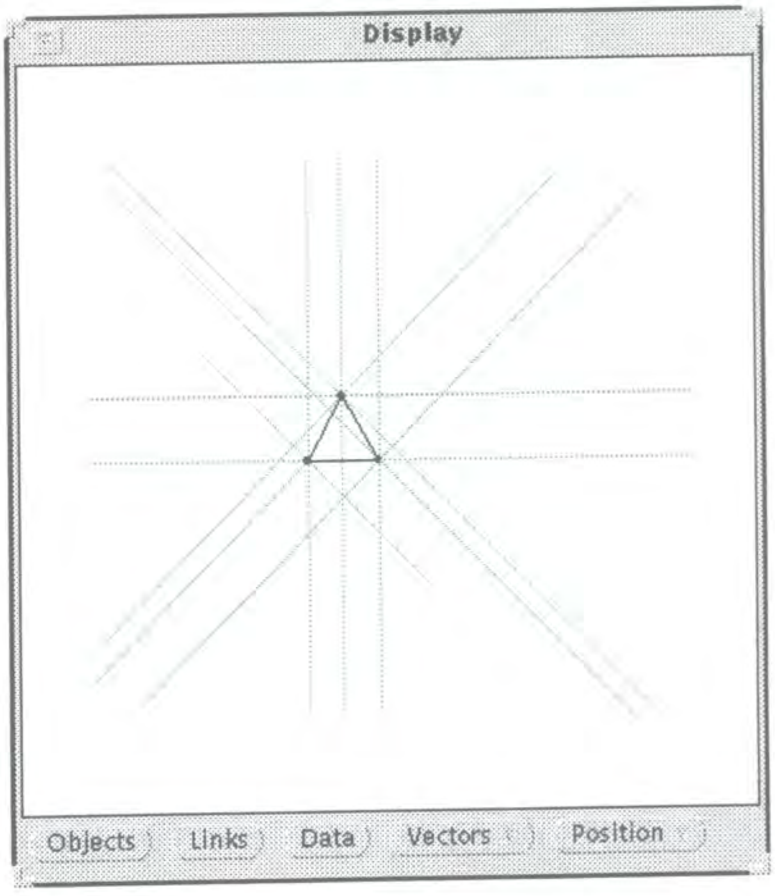


Figure 8.2: The triangular map of the environment showing range data

turbances. A simple rectangular environment of 2.46 by 2.71 metres, with dimensions primarily determined by the availability of laboratory space was used. In all these tests the robot initially set up a triangular map consisting of one triangle centred on the start position of the robot with a node at each vertex. This was created by the robot moving forward to the first vertex, or node, firing its sensors at this location and storing the information at the node. This process was then repeated for the other two nodes before the robot returned to its original starting position. The actual information taken by the mobile robot can be seen in figure 8.2. The triangular map is shown in black with the nodes as dots at the corners. This diagram depicts the range information as eight lines emanating from each node (the magnetic field data is not shown). From this information the size and shape of the environment can be inferred.

The first test consisted of a translation of the robot after the initial map had been set up to observe its ability to correct for this form of error. Both the odometric correction factor $K_{x,y}$ and the angular correction factor K_θ were set to a value of 0.5. The robot was placed in the environment and directed to set up a triangular map. Then the robot was moved, with respect to the previous figure, upward by 50 cm and to the right by 50 cm without the wheel encoder monitoring this. The correction procedure was then started. As the robot can only correct each time the sensors are fired, the robot was set to navigate to each node in turn and fire its sensors when it had arrived. After each nodal visit the positional estimate was corrected by the value $\Delta\hat{S}_r$. Throughout the test the total corrections required to the robot's positional and angular state were recorded.

Figure 8.3 shows the angular correction of the robot and figure 8.4 shows the translational correction in x and y. All values are plotted over 45 nodal visits with the translational disturbance being introduced after the 15th visit. Over the first 15 visits, where no disturbance has been introduced, corrections in both the angle and position of the robot have been required to keep the robot synchronised, offsetting its natural odometric drift. The angular correction was that angle by which the robot's head assembly had to be rotated to correct for the angular drift in the robot's base. After the robot was moved, a small disturbance was registered on the angular correction which was probably caused by a combination of effects; the non–uniformity of the magnetic field over the test region, plus positional error in the placement of the robot. The positional correction in x and y rose asymptotically towards an expected correction factor of 50 cm. This response clearly shows the proportional nature of this corrective control system. This test has therefore verified that, in certain circumstances, the robot can recover from a sudden translational disturbance.

Following the translation test, the robot's response to a rotational disturbance was examined. The robot was placed within the same environment and directed to create a triangular map. On completion of the map, and after having returned to its start location, the robot was lifted up and rotated through roughly 45 degrees anticlockwise. This direction corresponds to a positive angle for the robot. During the test both the correction factors required by the angular and positional system were recorded. Figure 8.5 shows the angular offset that was required by the robot to synchronise the angle of the head assembly. This clearly shows the magnetic sensory system creating a fast response to the angular change and within a few nodal visits correcting for the disturbance. Any angular shift in the robot's head assembly would create false range readings and therefore incorrect correction estimates. These can be seen from figure 8.6 which shows the correction required in x and y. The effects of the disturbance were transient and allowed the robot to remain synchronised with respect to positional drift. Therefore it can be seen that the robot can quickly respond to angular shifts.

Both the above tests looked at the effects of correction for translational and angular disturbances on the robot in isolation. If both a translation and a rotation were applied to the robot, it could be possible that the individual responses of the x, y and θ_b would compete, forcing the robot to synchronise to a false location. It is therefore important to test for this behaviour. The same conditions and environment were used as in the previous two tests. However in this test both the translational and the rotation disturbances were applied together. The robot was therefore moved 50 cm upwards and 50 cm right with a rotation of

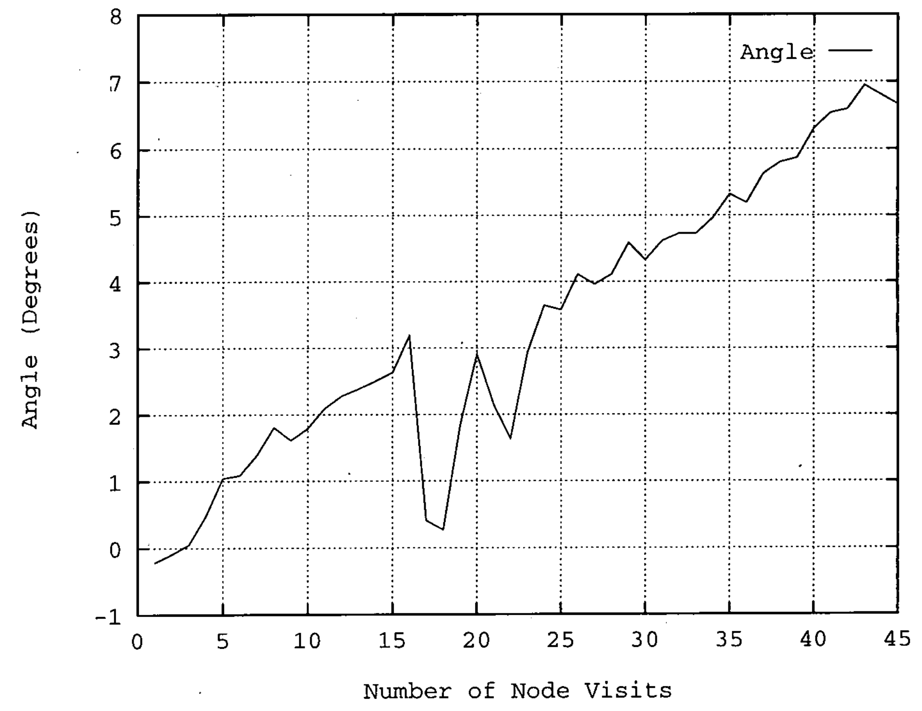


Figure 8.3: The angular correction required with a translational disturbance

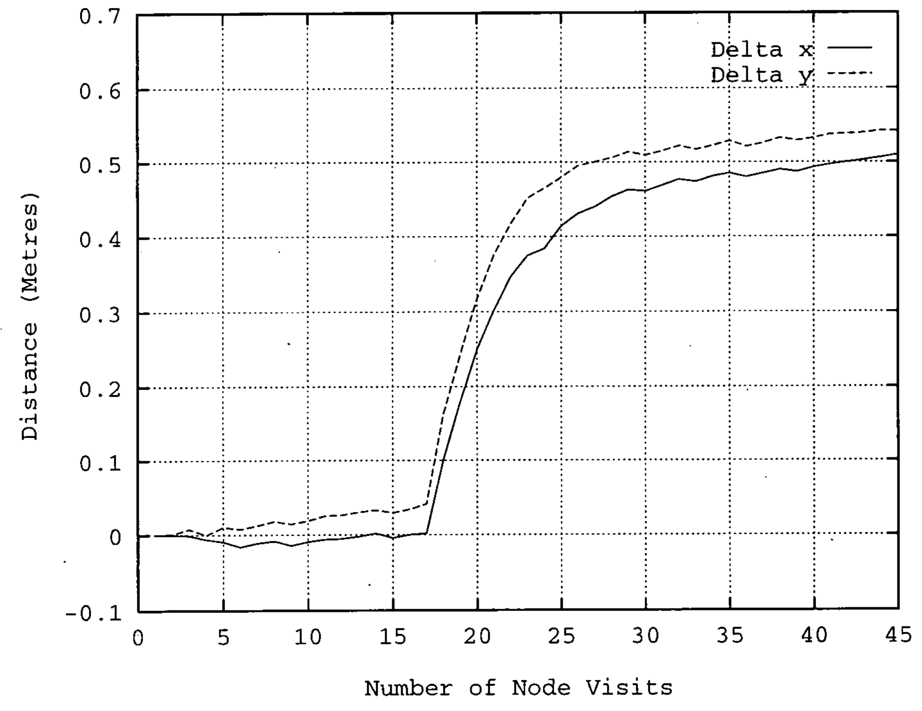


Figure 8.4: The position correction required with a translational disturbance

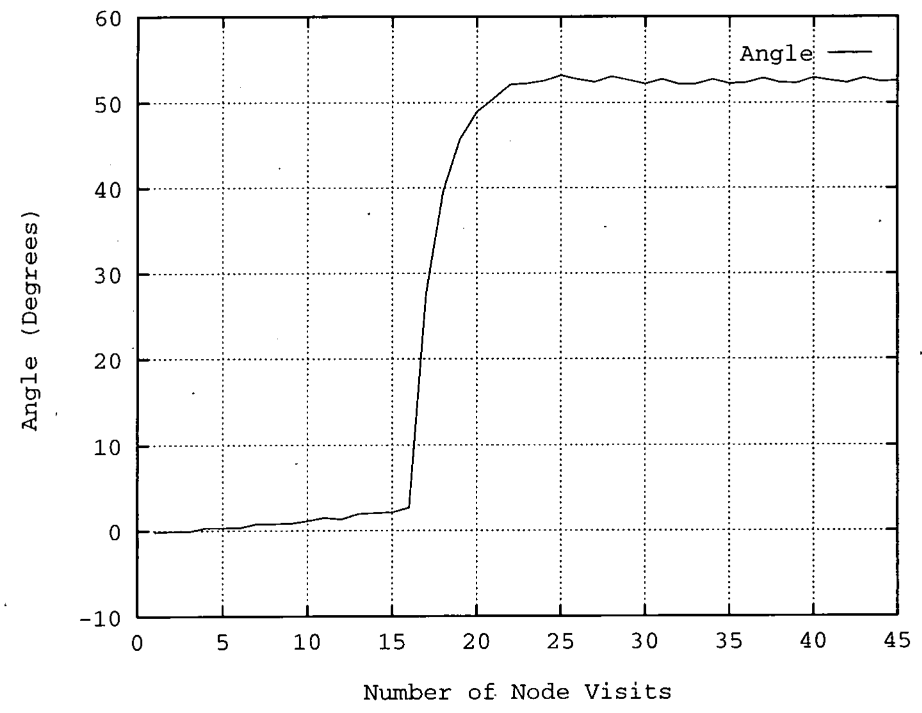


Figure 8.5: The angular correction required with a rotational disturbance

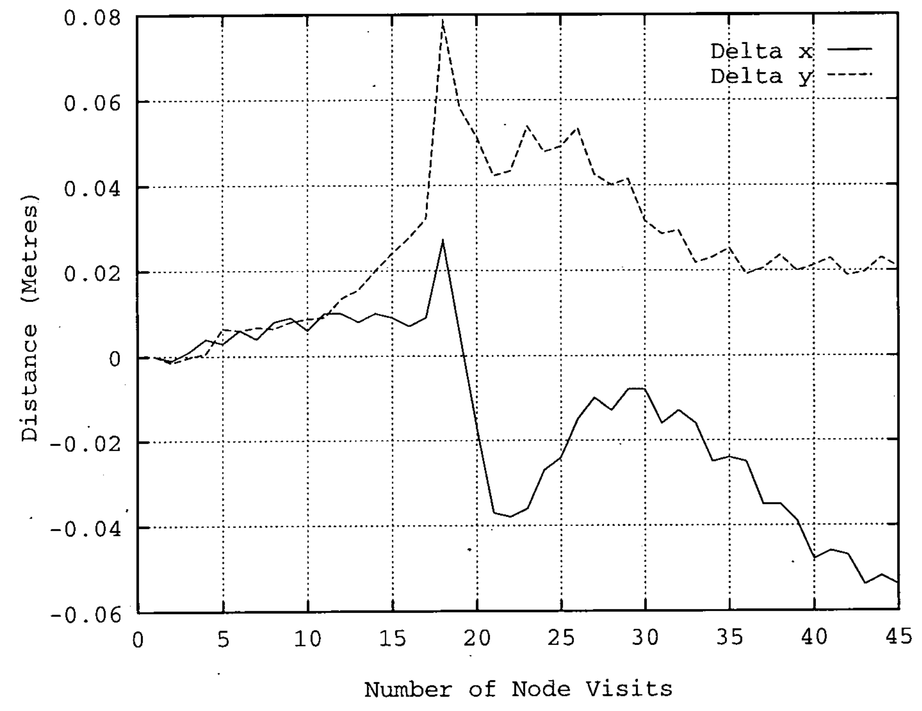


Figure 8.6: The positional correction required with a rotational disturbance

45 degrees.

Figure 8.7 shows the angular correction against node visits which shows the typical proportional response. This is explained by the magnetic field over the test area being reasonably uniform, and therefore the correction required at one location would be little different to that required at another. The positional drift is shown in figure 8.8. The correction is less smooth than in the purely translation test. However the robot does successfully re–synchronise with its environment.

The above experiments show that under certain restricted conditions the robot has been shown successfully to re–synchronise after experiencing translational and rotational disturbances. This process is fundamental to the success of a robot in building and maintaining the use of stored maps. Although the predominant use of the correction mechanism is expected to be for small changes, it will be seen later that sudden larger scale corrections may be required as part of the navigation process.

8.3 Robot Mapping using Behaviour Based Exploration

In Chapter 5 a method for building maps was described. This method involved the robot operating under based control. It was controlled by competing behaviours based on random wander, avoid obstacles and an explorative behaviour, called coverage. Under the direction of these competing behaviours the robot was shown in simulation to be able to explore and map out environments. This method of exploration and mapping will now be tested on the mobile robot to assess its performance and to validate the simulation work.

In the original simulations the random wander behaviour was included to direct the robot in the absence of any other useful control actions, and also to force the robot off any control minima where the robot might get stuck. In effect this random action can smooth over the deficiencies encountered in a behaviour based control system. Unlike the simulation experiments where the actions of a controller can be observed over long periods of simulation time, on the mobile robot the observation time was more limited. Therefore to be able quickly to determine the faults and true abilities of the control system the actions of the random wander behaviour was suppressed. Hence in these tests the robot was only controlled by an avoid behaviour, in combination with the directed exploration of coverage.

For practical considerations the coverage behaviour was slightly modified. One of the main considerations for this depended on the more limited processor power available to the robot than was available in simulation. This is especially important for the coverage behaviour, as it is based upon integrating up the free area 'seen' by the range sensors with respect to the locations of the other nodes. Therefore as the number of nodes increases so the computational burden increases non–linearly. One method of reducing the length of time it takes to compute the coverage vectors for all the nodes is by increasing the coarseness of the integration steps. It

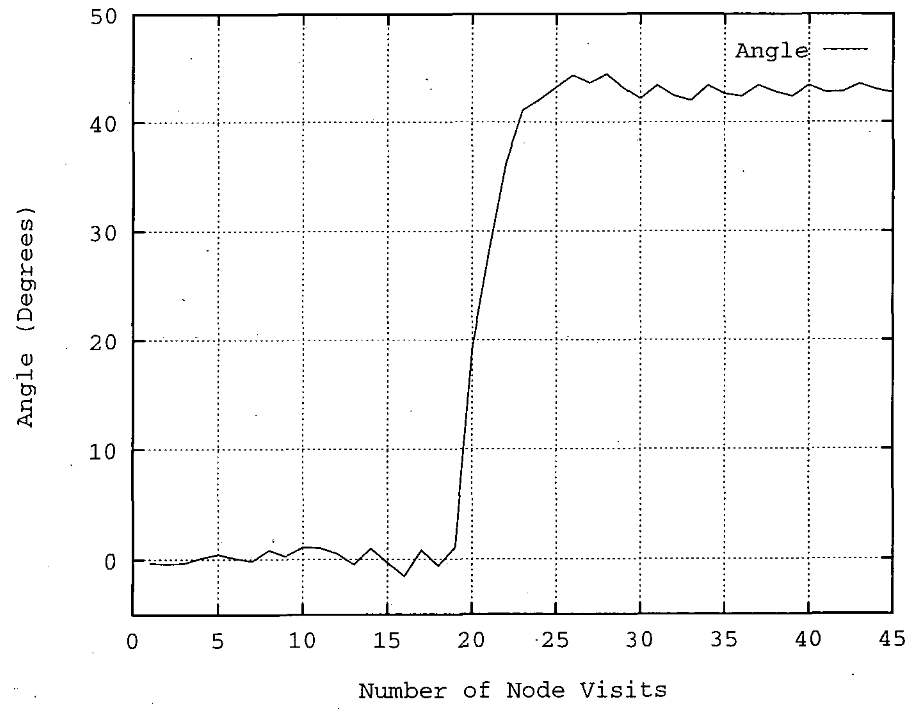


Figure 8.7: The angular correction required for both a translational and a rotational disturbance

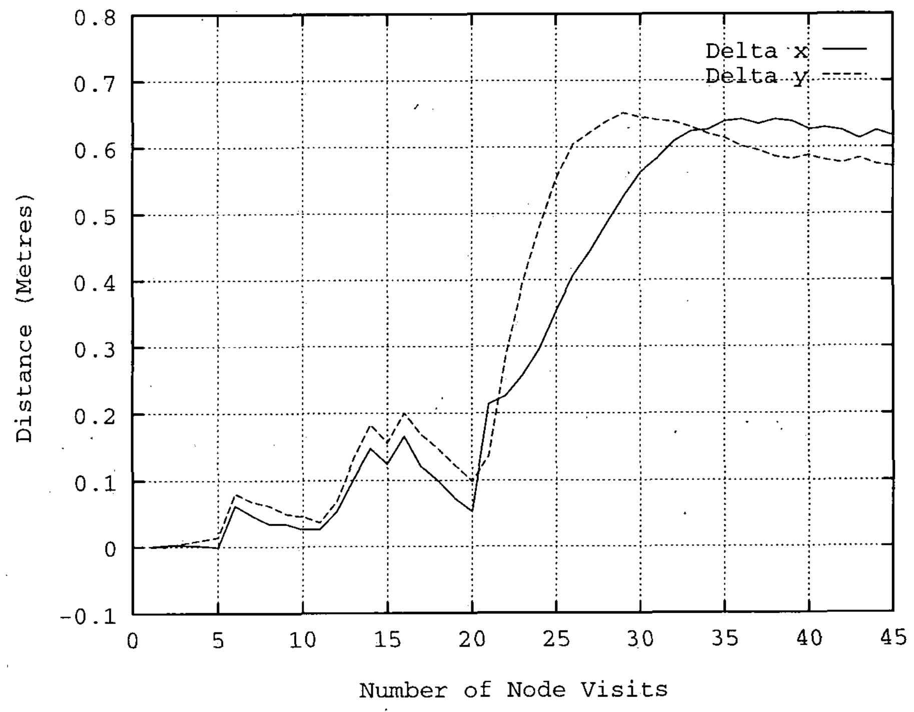


Figure 8.8: The positional correction required for both a translational and a positional disturbance

was found that the original setting for this, as based on the simulation results, took the robot's processor too long to compute. Although the exact computation time is highly dependent on the network's information, it would typically take about a minute for a network of over twenty nodes. To compensate for this, the coarseness of the coverage integration procedure was increased by eight times.

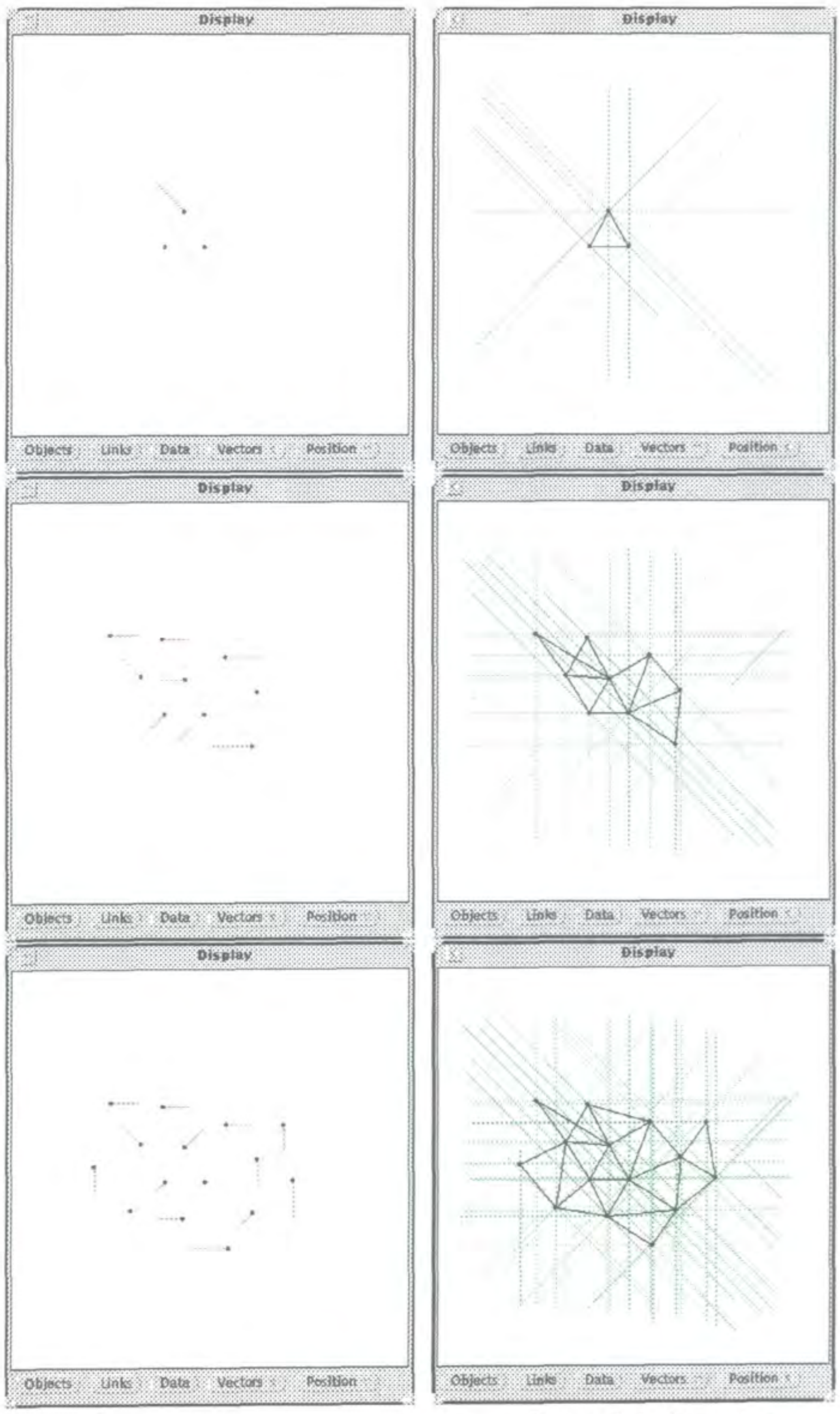
Experimentally it was found to be better for robot guidance when the angle of the coverage vector pointed to the greatest segment of free area, rather than to the mean free area. This would force a response where two nearly equal lobes of free space existed either side of node. It is important to note that the coverage vector is quantised into eight discrete angular values. However, in conjunction with the action behaviour, the angular direction of the robot remains continuous.

The initial experiment consisted of mapping the same environment as was previously used in the re-synchronising experiments. The robot was placed within the environment and made to create the primary three nodes that make up its triangular map. The mode of control was the same as that used in simulation in Chapter 5, with the robot computing the vectors for the nodes after each new node was created. Figure 8.9 shows the development of the map in three stages over time. The figures on the left show the coverage vectors at these time instants, and to the right are the corresponding network maps with included range measurements

The reduction in the processing power on board the robot causes some degradation in the quality of the vector information. However, its action is sufficient to direct the robot to search out its environment. One of the limitations of a behaviour based search stems from the robot moving whilst taking range information. The range system which employs a method of error correction called EERUF produces poorer performance whilst moving. This is because the system is based around monitoring changes in the range measurement. If the measurement values are changing as a consequence of motion this must be corrected for. However predicting the change in a range measure due to small robot movement is not always linear, and the number of rejections of correct ranges will increase. Therefore the above test was carried out with the maximum desired robot velocity set at 8.5 cm/s.

After the map had been constructed the robot was switched into navigation mode. In this mode the robot sought the node which had been visited least and then computed a path to it using the map's topological connections. This was then repeated, forcing the robot to visit the nodes in a reasonably evenly distributed manner. Initially the odometric correction factors $K_{x,y}$ and K_θ were set to zero so that typical errors could be monitored. After one hundred nodal visits, the correction factors were then each set to 0.5, thus forcing the robot to correct for any detected errors. For comparison this correction phase was continued for a further one hundred node visits.

Figure 8.10 shows the magnitude of the positional error estimate $\hat{S}_e$ as taken in the x



Coverage Vectors Network Information

Figure 8.9: Robot mapping using behaviour based exploration

and y values. It can be seen from the data that there is a considerably larger error in the estimates whilst the robot was without correction for visits 0–100. This error is erratic but relatively consistent, suggesting that sometimes the error introduced by the robot's curvature can cancel itself out, as this is dependent on direction travelled. When odometric correction is used between visits 100–200 the error is reduced and the robot remains more synchronised with its environment. Figure 8.11 shows the estimated error in the robot angle as determined by the robot's magnetic field comparisons. Here too the error is far more erratic whilst the robot is not under corrective control. Similarly the value of the error is reduced with correction.

From the above test it has been seen that the errors which are detected between the robot and its environment can be successfully reduced through the use of correction of the robot odometric state. However this correction is not to an averaged observation of the environment at these nodes, but to the first visitation to the node during construction. Any errors in this first visit are frozen into the network for its duration of usage. An important aspect which has not been explored is the adaptation of the node's information based on repeated visits. Thus those initial errors become reduced. However the rate of adaptation must be less than correction otherwise the system becomes unstable.

The method was then tested out on an environment which presents greater difficulties for a mapping robot. One of the major problems in mapping is its ability to link up disparate arms of the mapping structure. The linking of these arms where significant error has built up can potentially miss–map non adjacent areas together thereby creating discontinuities in the stored information. The simplest instance of this is a circular object placed within an environment. Therefore the dimensions of the environment were kept the same except that a circular object of radius 60 cm was introduced into the centre.

After this environment had been set up an experiment was conducted to test the robot's ability to map this using the behaviour based exploration method. The robot was initially placed in the bottom right hand corner of the environment with the its inter–nodal distance set at 30 cms. Figures 8.12 and 8.13 show six moments in the exploration of this environment. In the following discussion these will be referenced as times 1 to 6. At each instant the left hand figure reveals the coverage vector of each node whereas the right hand figure shows the network topology combined with the range information.

At time instant 1 the robot has created the three node network and dependent on its position the mapping route will be either upward or to the left. On this occasion the robot was nearest the lowest left node and therefore its direction was towards the left. As soon as the robot recognises it has moved a length delta from the network, a new node is created with associated range information. This can be seen in the second time frame where the network comprises four nodes. The coverage direction still points to the left and this directs the robot into new free space. At the third time frame the network has 7 nodes with the lower region

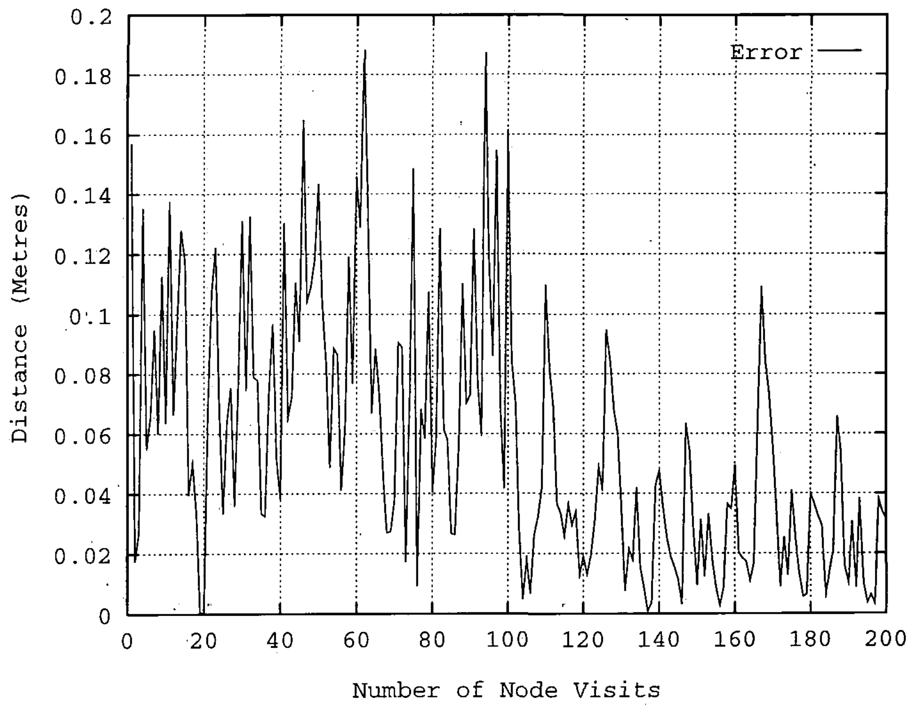


Figure 8.10: Estimated positional error during navigation of a rectangular environment

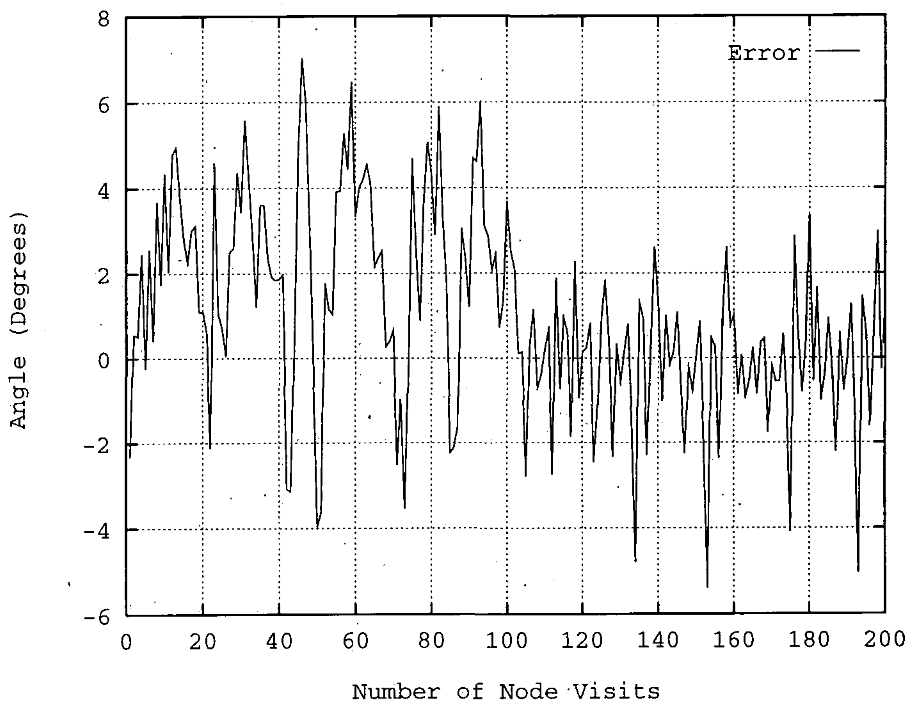
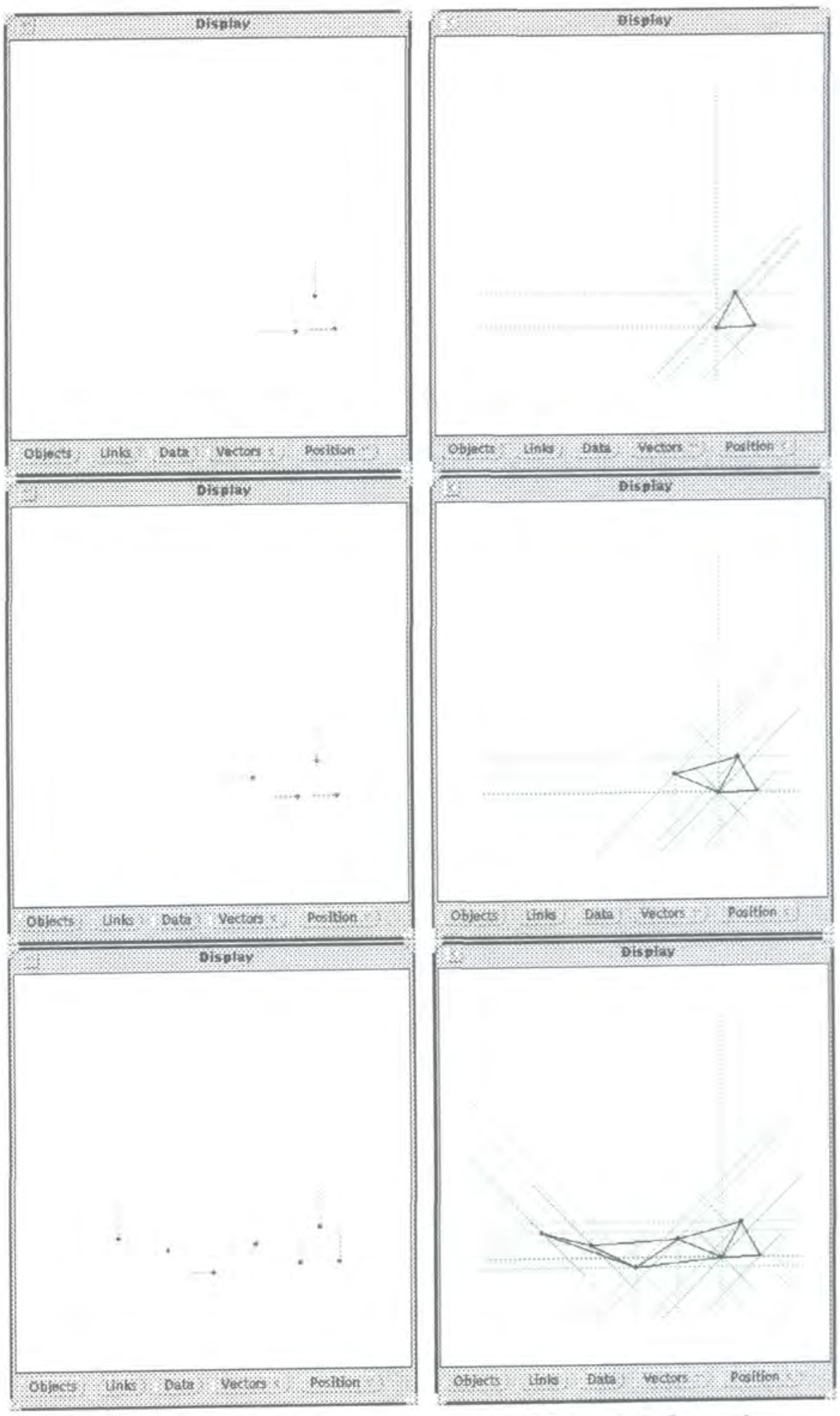
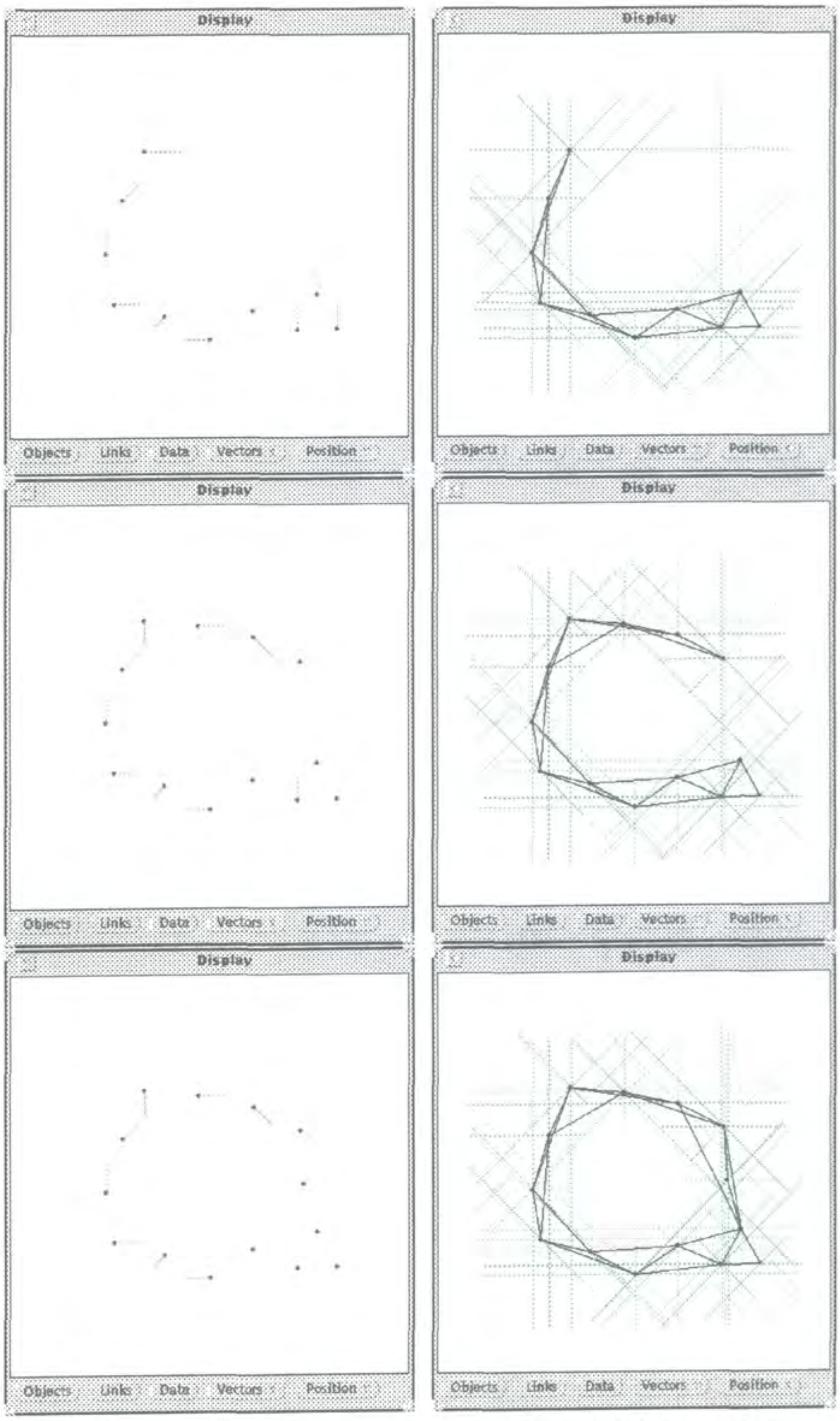


Figure 8.11: Estimated angular error during navigation of a rectangular environment



Coverage Vectors Network Information

Figure 8.12: Robot mapping using behaviour based exploration



Coverage Vectors　　　　Network Information

Figure 8.13: Robot mapping using behaviour based exploration

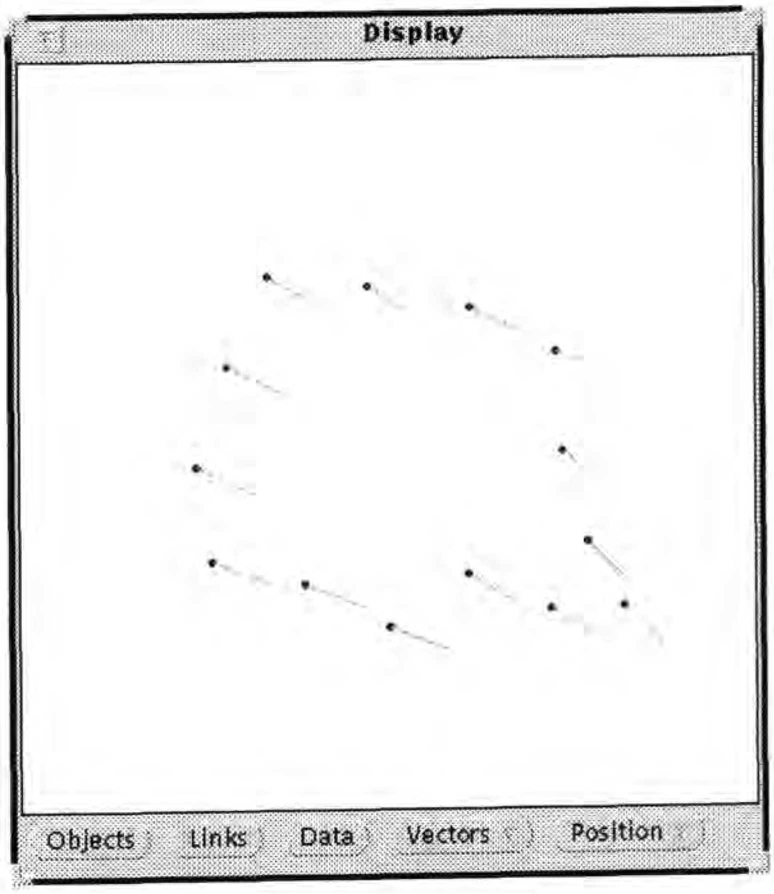


Figure 8.14: The magnetic field map of the environment

almost mapped, with the shape and the position of the circular object becoming apparent. It can be seen from the coverage vectors that they are dependent on the position of the newer nodes and some of their directions have changed reflecting the new state of free space.

At the fourth time instant shown in figure 8.13 the robot has mapped over half of the environment and this continues until, by the final frame, the robot has fully covered the region using just 14 nodes. From the range measurements the shape of the rectangular border can be seen with the circular centre section missing. However, this is not all the information that the robot possess about its environment as figure 8.14 shows the magnetic field vectors stored in the robot's network. Here the magnitude of the field is represented by the length of the line and its direction points to magnetic north. Following the exploration of the environment all the information can be saved to a data file which can be recalled by the robot for use at a future time.

So far we have shown that the mobile robot can map out an environment using the control strategies first developed in simulation. What was not explored in simulation was the effects of odometric error and the subsequent synchronising of the data using correction. If the robot cannot synchronise itself to this information then the learned information will inevitably be lost. Therefore the next tests use this map which has been formed to examine synchronisation after the development of a topological map.

Two tests were conducted, one with the odometric correction and the other without. Both tests involved orienting the robot at the start location of the previous test. Then the network that was created in the previous experiment was loaded back into the robot memory. For the environment and the robot's information to be synchronised the odometric position of the robot was reset to its starting datum of zero. The robot was then directed to navigate around the environment visiting each node in turn that it had created. This was selected as it is a repeatable cycle of 14 visits which can be monitored to examine if the sensor's error is increasing or staying within fixed bounds.

In the test without odometric correction figure 8.15 shows the magnitude of the positional discrepancy against node visits. It can be seen that over the number of visits the error is slowly rising. in fact this test was terminated by the robot colliding with an object at 39th visit. Figure 8.16 shows the error monitored by the magnetic sensor and this can clearly be seen rising away from the datum position. Therefore it is not sufficient for the robot to build maps without some form of correction. This leads to loss of synchronisation and in this instance collision.

In the second test both correction factors $K_{x,y}$ and K_θ were set to 0.5. Figure 8.17 shows the magnitude of the error in the estimated position. Unlike the previous test the graph shows a marked cyclic repetition every 14 visits. Also the magnitude of this cycle remains constant, indicating that the robot does not lose synchronisation with its environment. The measured error of the robot from its mapped data is at maximum 8 cm. Therefore as long as there is this level of clearance of the nodes from any object, then the robot will not collide. Observing the angular error of the robot as shown in figure 8.18 this is markedly smaller than without the corrections with a maximum error drift of $\pm 2^o$. It can therefore be clearly seen that the robot has successfully explored this environment and has kept this information synchronised with its observations. Hence the robot has achieved collision free navigation.

There are however problems associated with this system of exploration and mapping. Although the coverage parameter drives the robot to distribute nodes around its environment this does not necessarily lead to the creation of a connected topology. This problem was first observed in the simulations and a method by which this was resolved used active linking of the separate arms of the network. Figure 8.19 shows the results of an exploration after which the nodes were distributed around the environment but with unconnected branches. The coverage vectors indicate no propulsive force on the robot to move over the gap and therefore this method of active linking was employed. The corrected map after using the active linking is shown in figure 8.20. Following this the network can be used effectively for navigation.

The main drawbacks of this form of exploration and mapping is that its performance

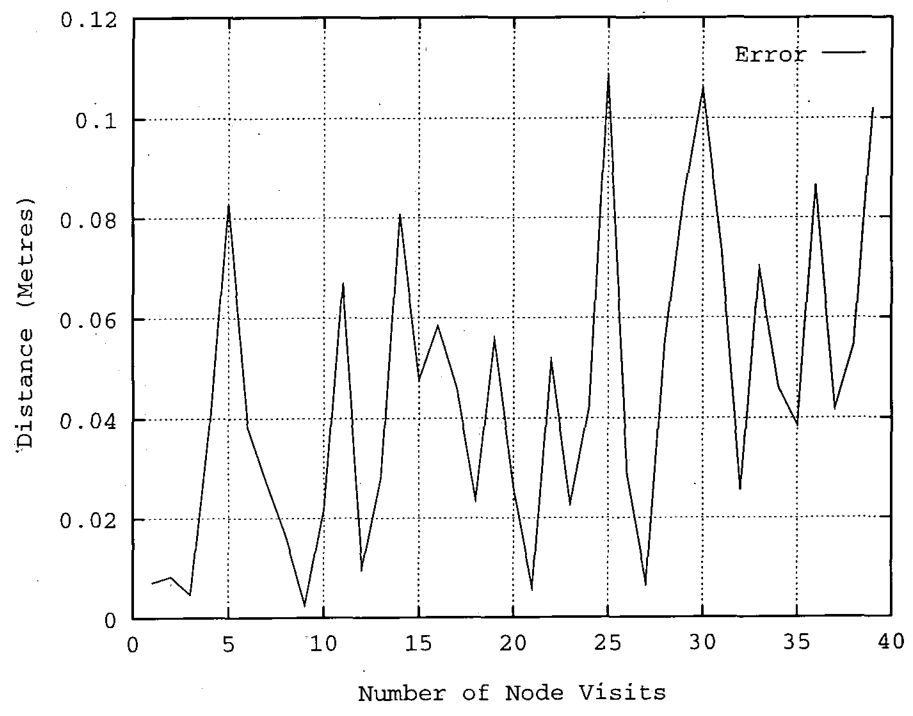


Figure 8.15: Estimated positional error during navigation without correction

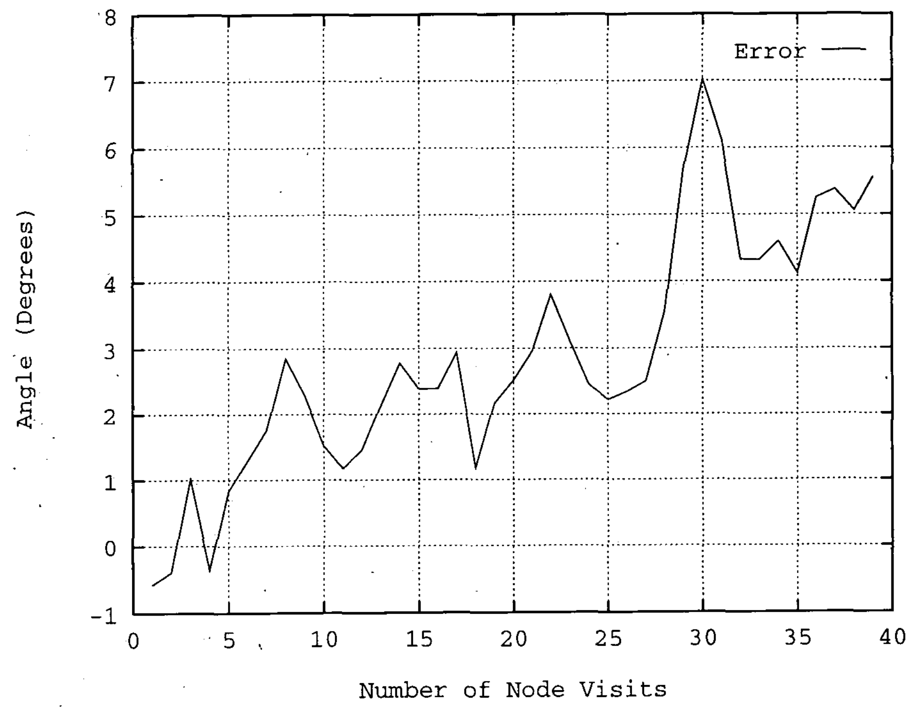


Figure 8.16: Estimated angular error during navigation without correction

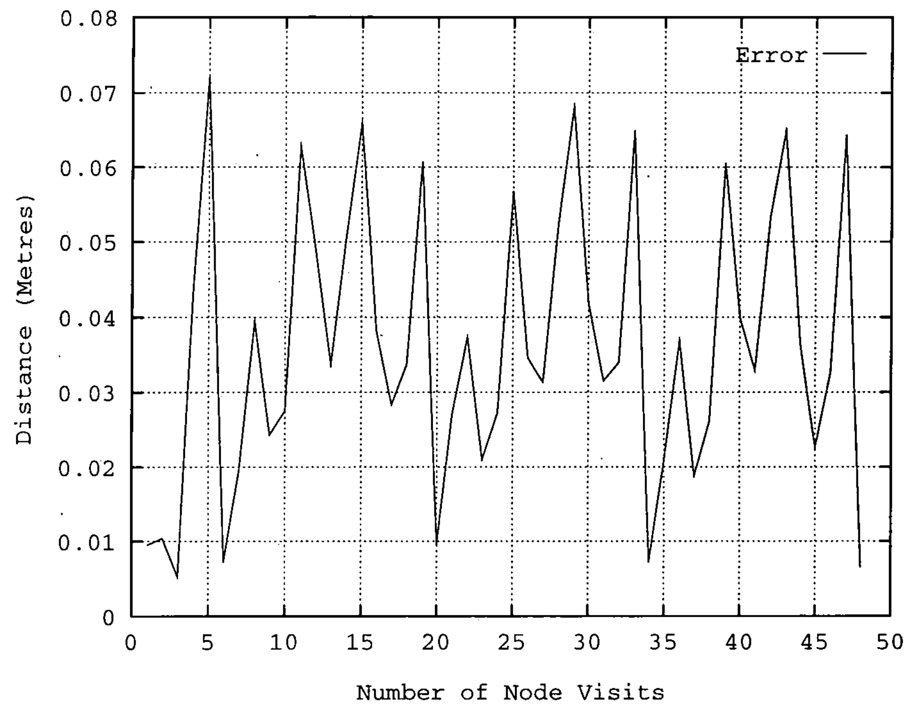


Figure 8.17: Estimated positional error during navigation with correction

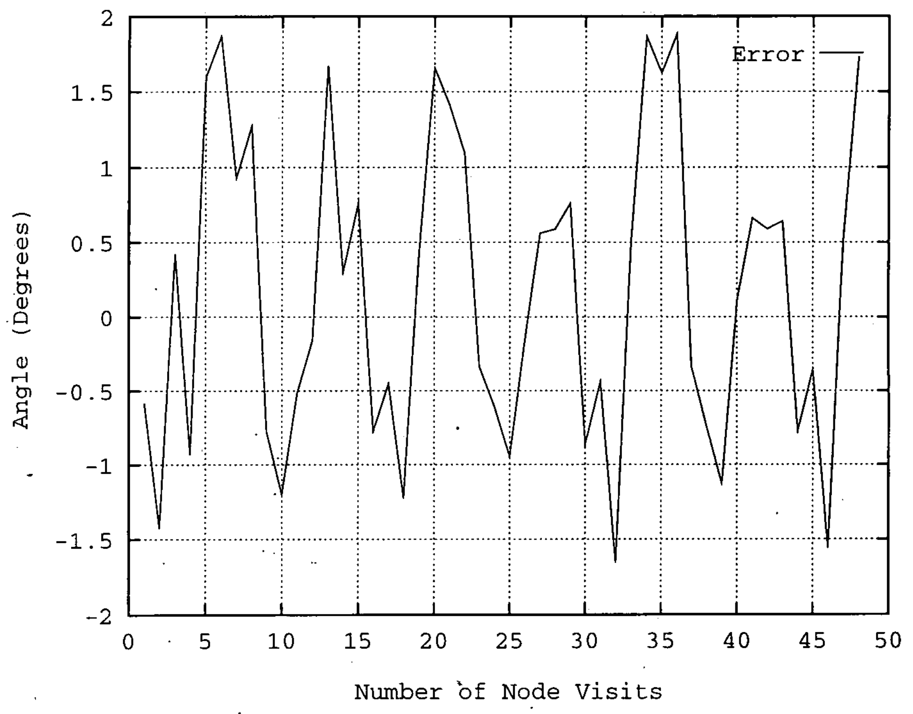
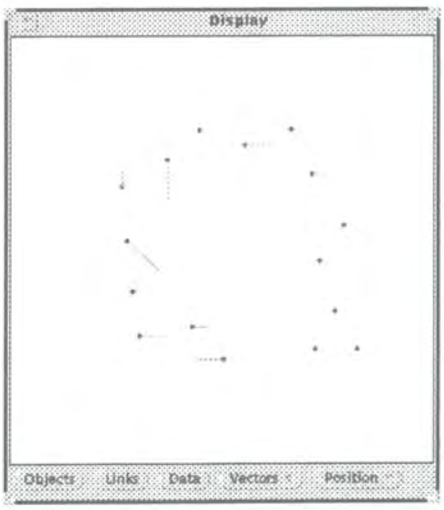
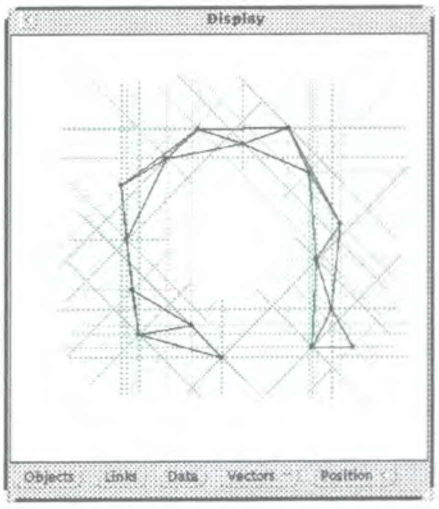


Figure 8.18: Estimated angular error during navigation with correction



Coverage Vectors



Network Information

Figure 8.19: Mapping problems using the coverage behaviour

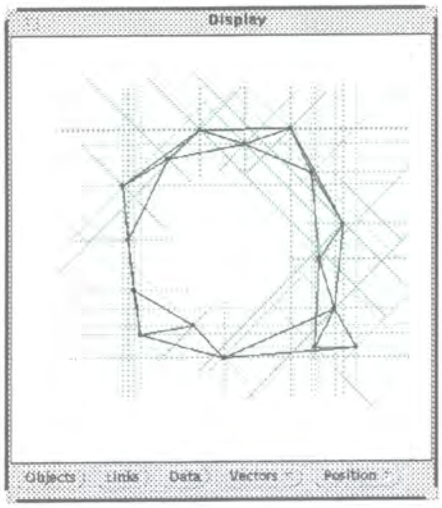


Figure 8.20: Map correction using active linking

is parameter dependent and still without a reliable method of monitoring the completeness of the search. Even with more powerful processing capability the angular quantisation of the range measurement system brings a limit to the accuracy of this technique. For this system to be able to operate effectively, greater processing capability must be available so that more accurate determination of the coverage parameters can be performed. The system does however allow for relativity coarse mapping of an environment which can be successfully used for navigation without collision.

8.4 Robot Exploration using an Exhaustive Mesh Search Algorithm

Previously the methods used for mapping have ranged from being purely passive to aiding in the exploration process. However none of these methods takes full control of the robot when it is in its search phase. A new method of mapping was developed whereby the algorithm took complete control of the robot and at the end of this search phase relinquished control back to the robot. This new mapping method is based on a triangular mapping mesh and attempts to place new triangles where sensed free space can be detected. Therefore it will be referred to as the exhaustive mesh search algorithm.

The basic principal of this algorithm involves two steps. The first step is to find, given the present network structure, the possible new locations around the robot where new triangles could be placed. Of the three nodes of this new triangle two must be already connected to the network, and therefore the problem is to find if there is enough free space to accommodate this new vertex. This can then be determined based on the range measurements of the nearby nodes. Following the determination of the locations of these potential new vertices, the robot can then navigate to these locations. If the robot encounters no collisions a new node and new triangle are formed. This technique will be described in greater detail below.

The network must be able to keep track of the locations and positional associations to other triangles of these potential new vertices. This is achieved by associating with every topological triangular link a status flag and positional location. Hence each triangle possesses, apart from its own three vertices, three locations and status flags of new vertices which if it were realised would create three new triangles surrounding the original. To keep track of the growth and state of all these new vertices the assigned status flag can have one of three states: *Triangle Present*, *Object in the Way* or *Potential New Vertex*. If there already exists another triangle comprised of this topological link whose other node is at the vertex, then the status flag is set to *Triangle Present*. If the range measurements from the nodes at each end of the link indicate there is no free space for the new vertex, then the status flag is automatically set to *Object in the Way*. If however these range measurements indicate that there is free space for this vertex then the flag is set to *Potential New Vertex*. Hence for any network the potential new vertices can be computed using this method.

The determination by the adjacent nodes of whether there is sufficient free space is dependent on the inter–nodal spacing of the triangles and a factor which will be termed clearance. Each link has an associated vertex location and also two nodes at each end. Unless both of these nodes can predict sufficient free space to the vertex the status of the link will be set to *Object in the Way.* From the vertex location the angular range measurement for each node which overlays this position can be calculated. The distance required can be then compared to the distance recorded by the range measurement at the node. This minimum distance will be the inter–nodal spacing but in practice this allows no room for error in either the range measurement or the positioning of the robot. Therefore the recorded distance must be greater than the inter–nodal distance plus the clearance value. If this is true for both the nodes then the status of the link is set to *Potential New Vertex.*

The second step involves the robot searching the network to check to see if the potential new vertices are realisable or not. The algorithm selects a new vertex with the status *Potential New Vertex* for the robot to visit. In all the following tests this choice was based on selecting the closest vertex to the robot's present location. A navigational path is then selected from the robot's location to one of the nodes connected to the associated link. After navigating to this point the robot then proceeds to the specified location. If the robot collides and therefore encounters an object the status flag is set to *Object in the Way* and the robot then re–attaches to the previous node in the network. If however the robot reaches the location successfully then a new node and hence a new triangle is created. This in turn can have new vertices and which means the network has to be re–calculated for potential new vertices. This search process is continued until all the status flags in the network are assigned *Triangle Present* or *Object in the Way* and therefore the search of the environment is complete.

For comparative purposes the first test of this form of mapping was conducted in the same rectangular environment as used in previous experiments. The inter–nodal distance was set to be 50 cm and the value of clearance was set to 10 cms. The robot was placed in the environment with its search phase activated. Figure 8.21 shows the development of the network in six stages with the vertices of status *Potential New Vertex* shown as black dots. Initially the robot set up a triangular map as before. Following this the potential new vertices were computed and this can be seen in the first figure. The robot computed that it was closest to the top left vertex, and therefore navigated to one of the adjacent nodes and then moved towards the potential vertex. The manoeuvre was successful and a new node and triangle were added to the network topology. Again the potential new vertices were calculated and the result is shown in the next figure. This search phase continued until there were no more *Potential New Vertices* as shown in the last diagram. The environment has been completely mapped and the robot can now use this information to navigate around.

As in the previous tests it was important to examine the effects of odometric synchroni-

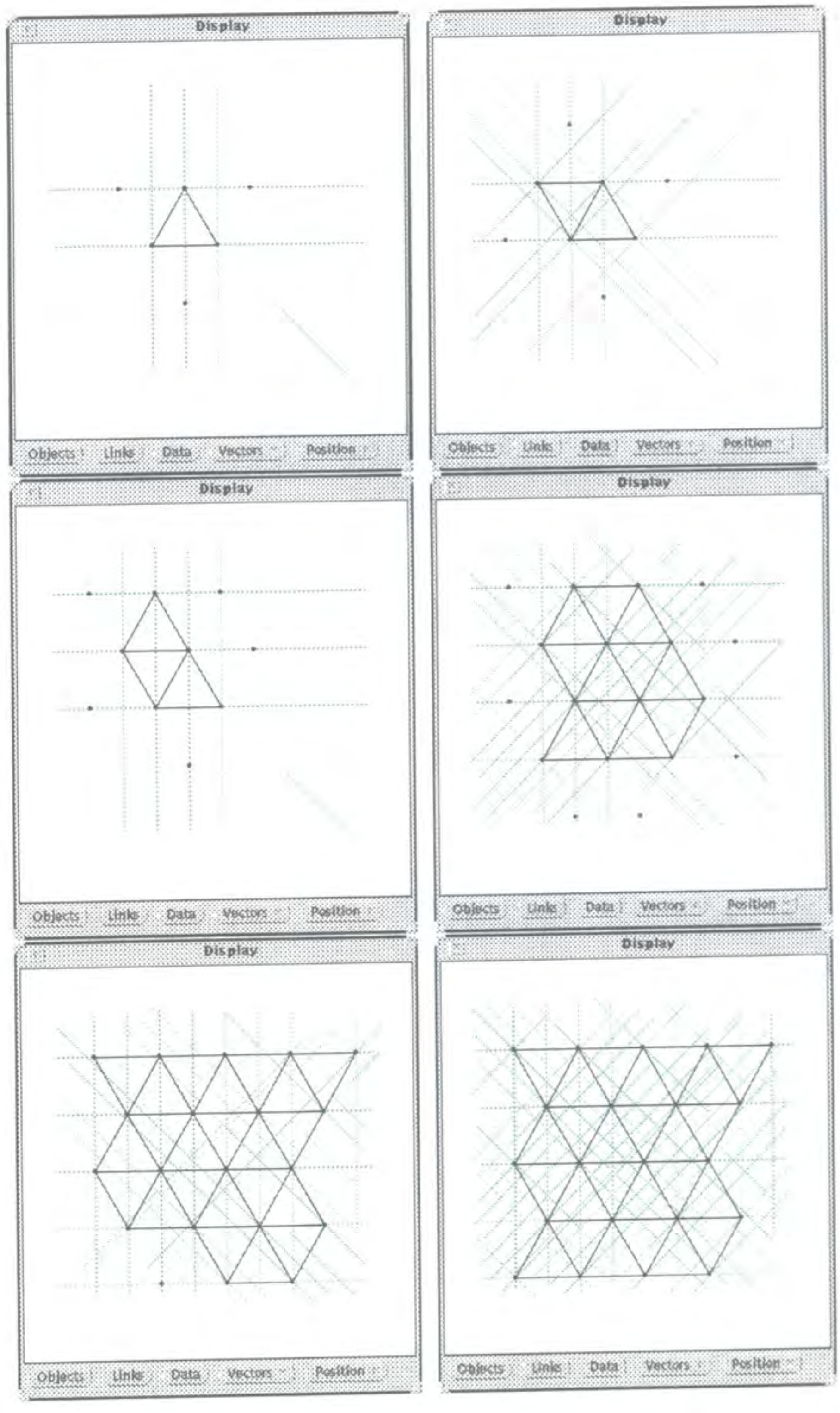


Figure 8.21: Robot mapping using the exhaustive mesh search algorithm

sation to the mapping structure. To assess the potential for odometric error to build up the robot was directed to navigate sequentially to the nodes as it had built them in turn. As far as possible, from this constructional sequencing, the robot would then retrace its steps. Before the test was conducted the robot was placed back to its original position allowing observation of the complete process of odometric drift to be observed. Figure 8.22 shows the magnitude of the positional drift. It can be seen that as the robot retraced its steps, the error remained below 7 cm. Only after the robot restarted this journey around the nodes for the second time did the error suddenly ramp up. The robot then continued with a constant error until it started its journey around the network for a third time. However the error which had built up was too large and the robot collided with a wall and the test was discontinued. In this test the angular error did not show the same drift properties observed in other experiments. This shows that drift effects are not always observed and that the collusion of fortuitous circumstances can cancel the build up of such errors.

The odometric correction system was then activated with $K_{x,y}$ and K_θ set to their usual values of 0.5. As usual the robot was positioned at the starting datum of the experiment and then directed, as in the above experiment, to retrace its steps around the environment. Figure 8.24 shows the magnitude of the positional estimated error, whereas figure 8.25 shows the angular correction. It can be seen from these results that the magnitude of the error remained below 8 cm and that the errors were cyclic suggesting that they were fixed and would not subsequently increase. Although further monitoring of the angular data would be required to establish whether or not it would follow a cyclic pattern, it remained well within acceptable bounds. The robot was therefore synchronised with its environment and could navigate without collision.

The final experiment consisted of testing out the exhaustive mesh search algorithm in the rectangular environment with the circular object in it. The inter–nodal separation was selected as 25 cm and the clearance was set to 5 cms. The robot was then placed in the bottom right hand corner of the environment. Figure 8.26 shows the following development of the robot's network over six time intervals. Initially the *Potential New Vertices* surrounded the network. In this closed environment their number did not dramatically increase as the growth continued but they become distributed at the boundaries of the search process. This process continued with the robot eventually returning full circle to link up the two arms of the network growth. The mapping of the environment was even and the quality of the range measures can be seen with not one erroneous value extending beyond the boundaries of this containment.

Two important factors can be seen from the network information. The first is that when a range measure strikes the circular object at too shallow an angle it is not reflected back. This can be seen where the central object has a less well defined boundary. Secondly and more importantly the range measurements of the later nodes do not quite match with the original

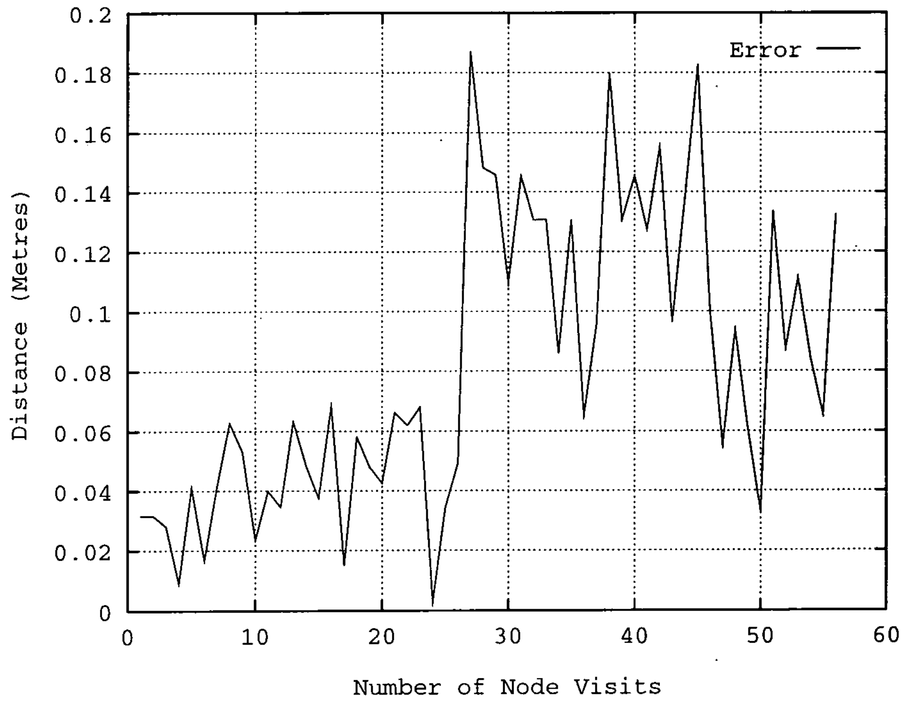


Figure 8.22: Estimated positional error during navigation without correction

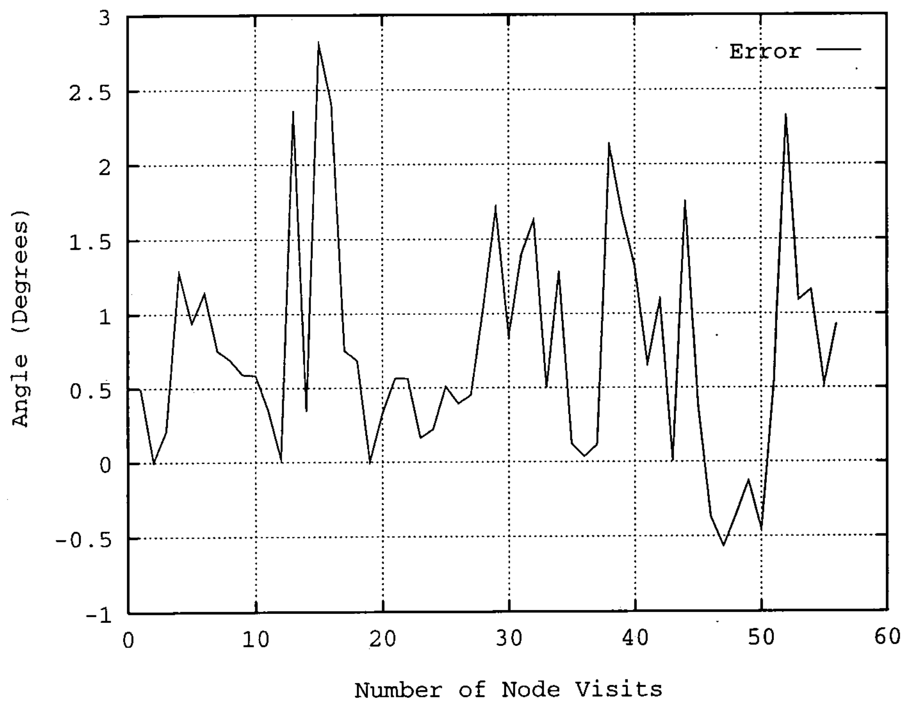


Figure 8.23: Estimated angular error during navigation without correction

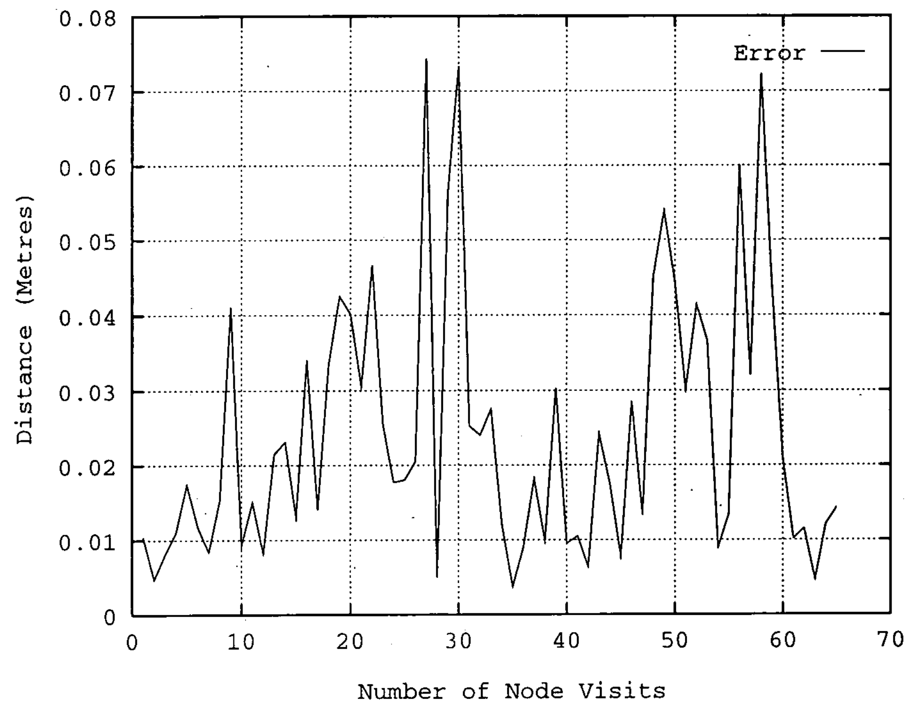


Figure 8.24: Estimated positional error during navigation with correction

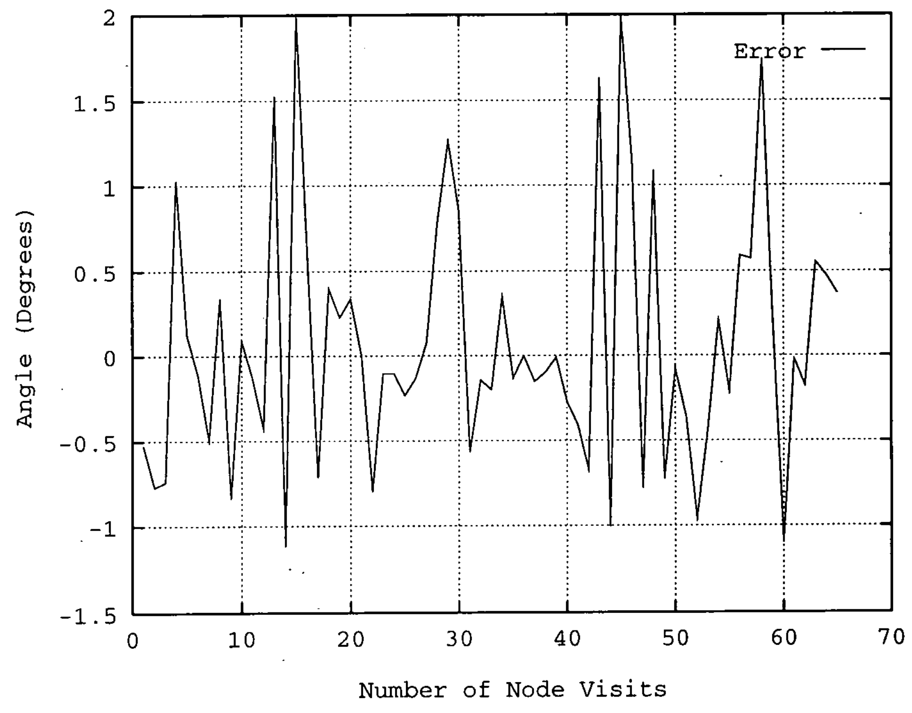


Figure 8.25: Estimated angular error during navigation with correction

nodal values. Although this is not very pronounced it means that the network contains a discontinuity. Although this has been present in the results of the other experiments it can be more clearly seen here with respect to the regularity of all the other range measurements. Its effect on the following tests will also be seen.

The network also stores values of the magnetic field direction and strength. This is shown in figure 8.27, with the length of the lines indicating the field strength and the angle representing the direction. With respect to the magnetic field vectors the exhaustive search algorithm has the advantage, that it takes samples evenly over the environment. The vectors show that although the magnetic field is not entirely uniform the rate of change is low over the test area. These test values correlate well to the direction of those vectors taken using a standard magnetic compass.

Following the search phase the robot was returned to its original starting position. As in previous tests it was then directed to navigate sequentially, in the order of creation, around the nodes whilst observing the estimated error without odometric correction. Figure 8.28 shows the magnitude of the positional estimated error against nodal visits. The error during the robot's first circumnavigation of its network did not increase above 7 cm. However as soon as it returned to the original nodes the error rose up to around 13 cm. This error remained constant as the robot drove around again, but before it could complete a second trip the test was brought to a conclusion by a collision with an object. It therefore appears that those modes of error which the robot underwent as it first mapped the environment were similarly repeated as it retraced its steps. Only when the robot returned back to its starting datum could the actual errors be detected. Hence each time the robot followed this particular path around the environment an extra positional error of around 13 cm was being introduced. As shown in figure 8.29 the error in the angle of the robot, although less affected, could also be seen to have registered a sudden change of 6^{o}.

This experiment was then repeated but with the robot using odometric correction of $K_{x,y}$ and K_{θ} equal to 0.5. Figure 8.30 shows the magnitude of the error in the positional error estimate. Over the first navigation around the map the errors stayed within the same values as those seen in the previous experiment. Similarly when the robot returned to the start nodes the error jumped up to a value of about 13 cm. However, unlike the previous experiment this was then corrected, and the errors returned back to those values experienced in the first circumnavigation. This error was shown to have a cyclic nature and therefore it is constrained so that the robot would not collide with obstacles. The results shown in figure 8.31 which are the values of the angular error estimate mirror the trend above. The error normally only varies about $\pm 1^{o}$ with a spike of 6^{o} when the robot returns to its origin. Consequently this experiment has shown that the robot can keep itself synchronised to the map within 5 cm normally with an error of 13 cm at the discontinuity in the map.

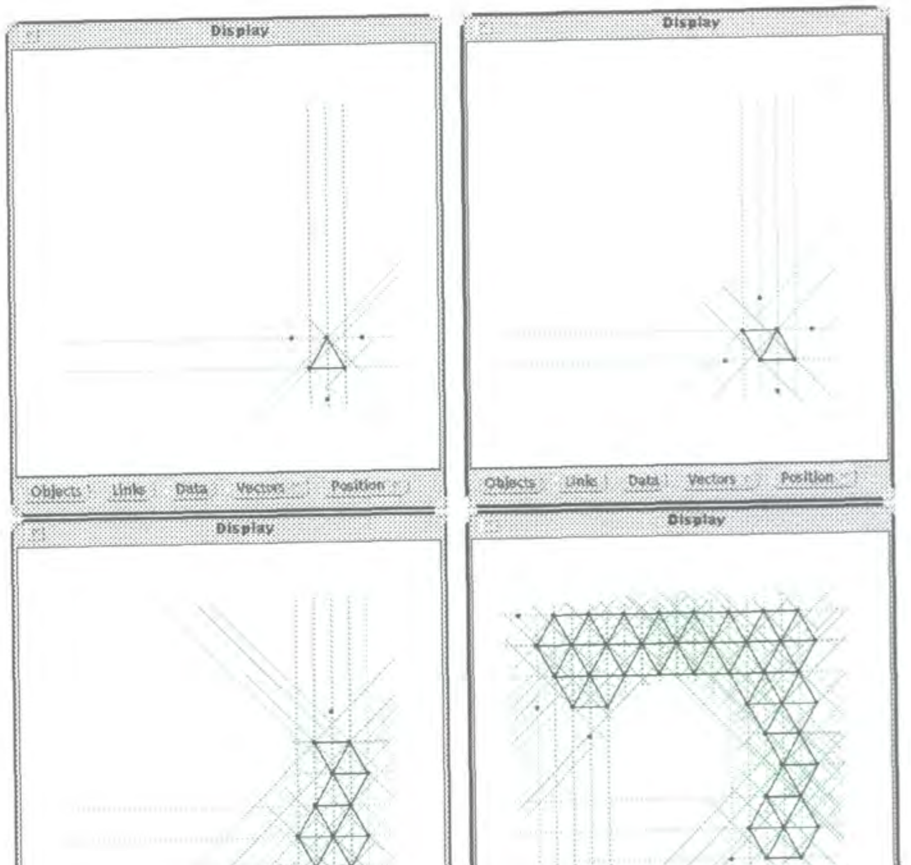


Figure 8.26: Robot mapping using the exhaustive mesh search algorithm

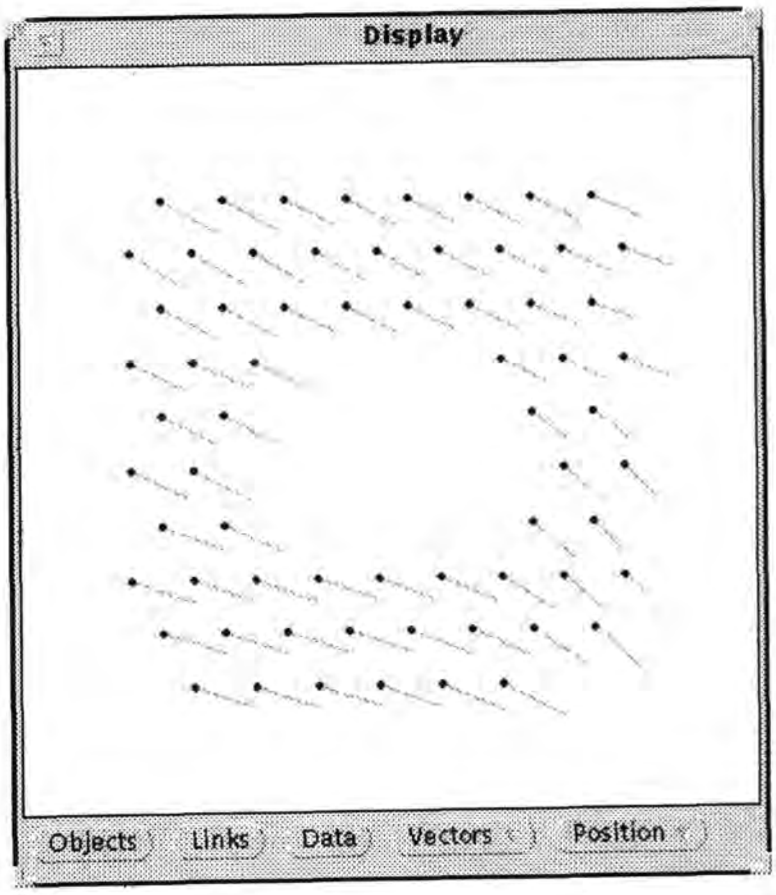


Figure 8.27: The magnetic field map of the environment

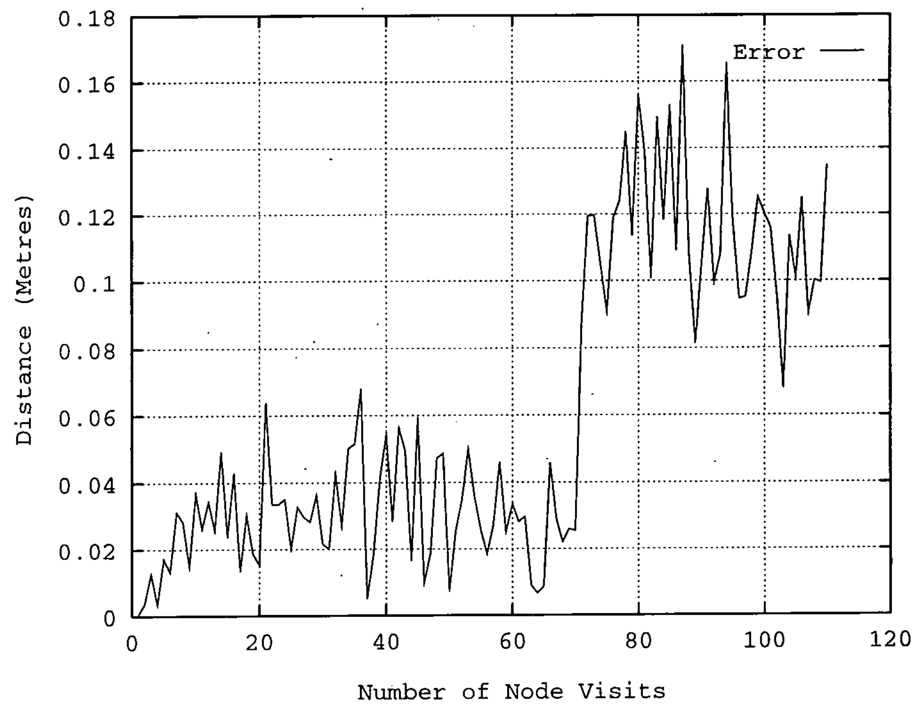


Figure 8.28: Estimated positional error during navigation without correction

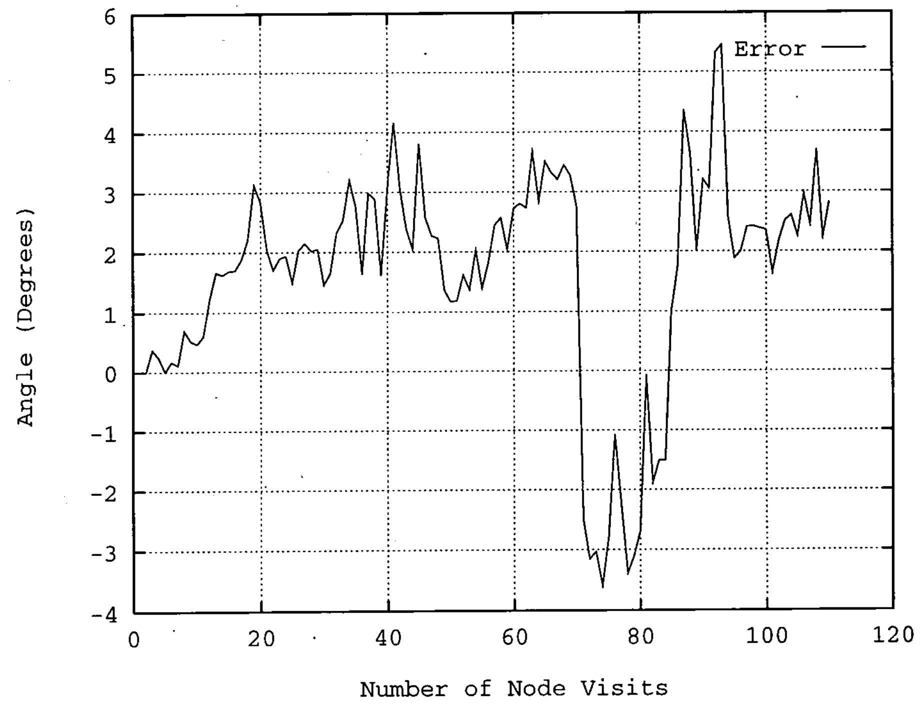


Figure 8.29: Estimated angular error during navigation without correction

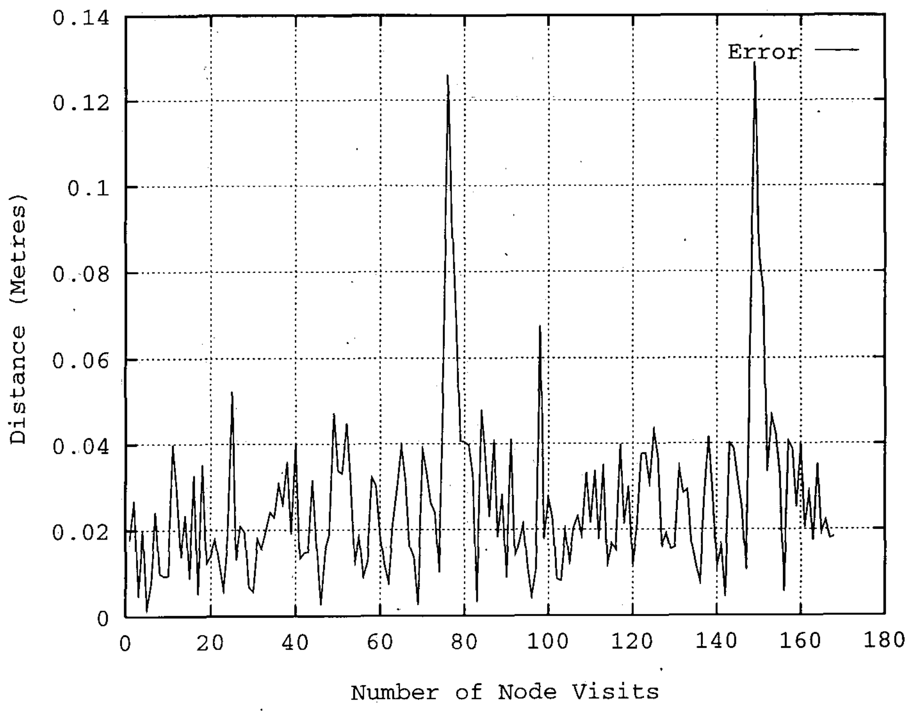


Figure 8.30: Estimated positional error during navigation with correction

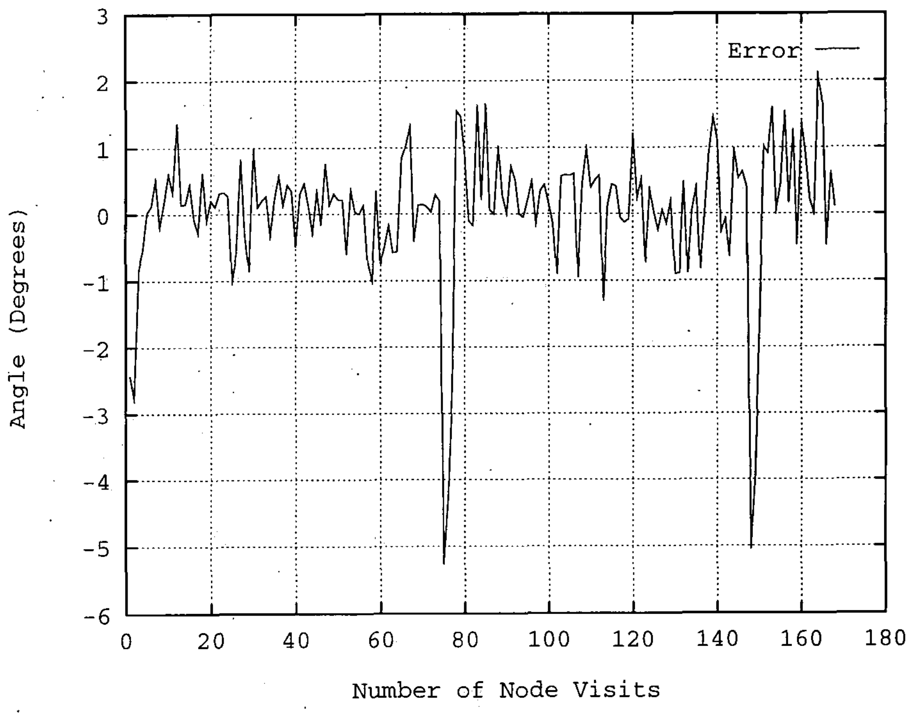


Figure 8.31: Estimated angular error during navigation with correction

8.5 Summary of the experimental work

Initially tests were conducted into the ability of the robot to be able to synchronise its mapped knowledge with the environment as detected by its sensors. Without this aptitude the robot would not be able to use its gathered information. It was shown that the robot could recover from translational disturbances by comparing its range measures with the information stored in the map. It could also recover from angular disturbances using the comparison of the local magnetic field with that stored at each node. Therefore it was shown that the robot could synchronised a three node map with its environment.

Under behaviour based control the use of coverage was shown to be able to drive a robot in search of new areas of free space. Using this method the robot could successfully map out new environments. However, the use of coverage alone cannot guarantee that the topological links of the map will be most beneficially connected. Therefore coverage must be used in conjunction with some form of method which will ensure that the connectivity of the network is analysed. It was shown that the use of the generated maps without odometric correction is not sufficient for collision free navigation. In all the tests this approach led to a collision with a wall or object. Through employing odometric correction collision free navigation was shown to be viable with the estimated positional error staying within fixed bounds.

Although this system has shown itself to be successful there are some drawbacks. The successful exploration is dependent on a number of parameters being chosen correctly. These include the weightings of the various behaviours as well as the coarseness of the coverage search algorithm. With limited computational power the coarseness of the coverage becomes a trade off between speed of action verses quality of decision. It is possible to determine from the assessment of the coverage vectors a measure of the completeness of the search phase, however it has been found not to be reliable. Therefore although this control system can control the robot to explore and subsequently navigate an environment, there are marked deficiencies.

A potential solution to the problem of explorative mapping and navigation, the exhaustive mesh search was developed. This is a computationally inexpensive algorithm which has a deterministic and definite search phase in the environment. This has been shown to explore, whilst mapping, two different environments with a regularised and even mesh. As with all the tested systems the use of odometric correction has been a fundamental requirement. It has also been shown that the regularised exploration means that when the robot links up separate regions detectable discontinuities are created in the map. However the very fact that they are detectable means that it should in the future be possible to correct for them. There are only two parameters to be selected which have a direct relationship to the robot's mapping and navigation. The inter–nodal separation will dictate the density of network mapping whereas the clearance parameter gives an error safety margin which can be determined by observations of the estimated positional error. This system has therefore shown its ability to search out new environments and then switch into a navigational phase which with odometric correction fixes the robot within positional error bounds. If these error bounds can be set below the clearance value, then this system can ultimately guarantee collision free navigation.

Chapter 9

Conclusions and Suggestions for Further Work

9.1 Conclusions

From the original literature survey it was observed that the field of mobile robot research spanned a diverse range of applications. Correspondingly this diversity has led to the development of wide set of robot experiments and solutions. However it was apparent that there were no elegant solutions to the problem of robot exploration and navigation. Those partial solutions that had been reported pointed to a need for greater investigation, and hence this work has concentrated on the problems of exploration and collision free navigation by an autonomous mobile robot.

All the presented control solutions were initially developed in simulation, and its use has been advocated as a preliminary tool for the design of both prototype robots and their control systems. Simulation allows provisional design considerations of sensor type and drive mechanics to be iteratively matched to the required tasks. With simulation, the level of difficulty experienced by the robot, in for example sensor and actuator noise, can be limited during the early development of the controller. This allows the environmental conditions to be suited to the competence of the robot controllers which therefore speeds up the early stages of development.

Attracted by the innate propensity for exploration and navigation shown by many animals the analogous use of artificial neural networks instead of real neural systems was examined for use by a robot. Although the promise of uniform topological growth was shown in idealised training conditions this could not be repeated with direct input produced by a simulated robot. It was shown that a method of random sampling from the data set could be used significantly to de-correlate the information, thus leading to smoother topological growth. This method was successfully used to train the network with data from a simulated robot exploration. However the development of this approach into a generic solution would require considerably more algorithmic complexity. In conclusion, the solutions gained through this approach, although biologically more feasible, are computationally inefficient in comparison to the methods described below.

The development of a novel solution based on a simple deterministic approach to the creation of nodal information was shown to be able to create maps that could be used for collision free navigation. In conjunction with a behaviour based search method called coverage, the time taken for a robot to perform an explorative search could be reduced. To guarantee correct topological development, a system for observation of the connectivity of the topology was also required.

The preliminary design of a robot and its control strategy capable of exploring and building navigable maps has been developed in simulation. However, none of the simulations included modelling of the accumulative errors that develop with odometric positioning systems. The problems of accurately mimicking the interaction of a robot with the physical world increase in difficulty as the level of modelling becomes more exact. Hence it has been argued that the true performance of these controllers can only be assessed by using a prototype mobile robot.

After the mobile robot was designed and the basic tests on the sub–systems were concluded the most important observation, as to performance, regarded the odometric positioning system. The implementation of a syncro–drive configuration using belt drive and having the wheels offset lead to path curvature dependent on the direction of travel. This curvature was found to introduce systematic errors into the odometric measurement system which could only be effectively reduced with reference to some other sensory system. The ultrasonic sensor system using the method of EERUF was shown to be able to eliminate the problems of crosstalk even whilst the robot was moving. The magnetic sensor was demonstrated to be accurate to within $\pm 2^o$, sufficient for angular head positioning.

The use of odometric correction was shown to enable the robot to re–synchronise itself with its environment after being subjected to translation and rotational disturbances. It is possible, in extreme circumstances, that using this method alone would fail to attach the robot to its correct location. However with the combined effects of the magnetic and range correction systems it is sufficient for localised repositioning of the robot where the spatial rate of change of these parameters is not great.

The novel mapping method, that utilised the behaviour based exploration strategy, was shown to be able to map out environments and subsequently navigate them without collision. This was demonstrated in the rectangular environment both without and with a central object. This method creates sparsely distributed mapping topologies which are dependent on a number of parameters being chosen correctly. The most important being the coarseness of the assessment of the coverage vectors, which results in a trade off between speed of action by the robot versus quality of decision. In regard to the robot switching out of its search phase there is no accurate and therefore reliable method by which to determine the completeness of the search process.

A deterministic solution to the problem of exploration and navigation was finally proposed in the exhaustive mesh search algorithm. This computationally inexpensive algorithm, which has a definite search phase, was shown to control the robot to explore and then navigate without collision. This method has the advantage that it efficiently directs the robot's position to produce a repeatable even topology. The robot could then switch out of the search phase and navigate as desired. Effectively it requires little computational power, exploiting the explored physics of the environment to determine where and when exploration is possible.

9.2 Further Work

An important area for further investigation relates to the optimisation of some of the basic sub-systems of the prototype mobile robot. One particular factor which could produce improved results is that of corrective compensation for the robot's curved path. A more detailed examination could lead to suitable ways of predicting the drift which could then be compensated. It might be possible dynamically to create a model for drift using the drive control commands in conjunction with the corrections from the sensory re-positioning system. This could potentially be achievable through the use of a filter technique. It is also possible to further optimise the performance of ultrasonic sensor system in relation to speed of operation. In all the experiments the rate at which the sensors fired was fixed but it would be possible by monitoring the rejection rate to control this rate of firing through adaptation. Such a system would match the firing rate of the sensors to the localised crosstalk conditions. This would be especially important if more that one robot was operating in the same environment, as the rates of firing of these two would back down until both could get correct range information. This was originally suggested by the authors of the EERUF algorithm but represents the best solution for two robots being able to use the same frequency ultrasonic range measurement systems within the same environment without complex synchronisation methods.

In the experiments of the prototype mobile robot it was not possible to correct the position of the robot during the explorative phase. Therefore any positional errors were frozen into the network and could not be subsequently reduced later. It is therefore suggested that investigation into re-correcting either the position or information possessed by each node could prove effective in reducing this error. It might also be possible from observations of the correction values to quantify the average inter-nodal error. If possible this value would provide a parameter reflecting the quality of the topological map and a useful index to predict error effects when linking disparate parts of a network. This kind of error has been shown to occur in the explorations using the exhaustive mesh search algorithm with a central object.

From this it can be seen that an investigation into methods of propagating this positional error back throughout the network is also required.

The present prototype mobile robot can be considered as a test bed for development of intelligent control algorithms for mobile robots. It is equipped with a range of sensors which have been shown adequate for the task of exploration and navigation within static environments. One important line of research in this direction is being undertaken as a final year undergraduate project. This work will attempt to create two operating systems one on the robot and one on a PC based simulator so that any robot controller program can be directly transferred between the two without modification. This would reduce problems in the developmental cycle and could be used as an educational tool for demonstrating robot control systems.

The prototype robot has been designed with a view to having more than one robot cooperating over a task. An obvious benefit of this approach is that the time taken to explore and map an environment could be reduced considerably. In order to be able to achieve this the robots would have to develop some form of communication strategy. Each robot would have to be able to position itself with respect to the other robot. This would require investigations into a robot's ability to relocate itself within another map.

The above description of cooperating mobile robots presumes that subsequent robots will be identical replicas of the original robot. A possibility to be considered might be that of a robot with large number of sensors which could map out an environment, and then subsequently reduce the errors between nodes whilst storing other seemingly less relevant data about local light intensities, colours, magnetic fields etc. Then it could be possible for a much simpler robot with limited sensors to be able to navigate using selected information taken from the highly descriptive map. Hence cheaper more simple robots might be able to perform navigational tasks which would be impossible to perform on their own.

The simulations and experiments have only been conducted in static environments. Dynamic environments, such as those in an office with moving people, present a more challenging problem. To be able to cope with such systems whilst still being able to explore and map, the robot must have a greater bandwidth of sensory information and a correspondingly higher computational processing power. Potential solutions might involve the use of passive visual flow fields which give rapid low level motion information combined with parallel processing formed from programmable logic. With the advances in the miniaturisation of microcontrollers it might be possible quickly to build small inexpensive robots which could cooperate and divide their combined resources between the problems of map building and coping with dynamic environments.

Appendix A

Derivation of the Motor Model

In this appendix a mathematical model for the forward propulsive force provided by the motor is derived. This mathematical model was used for simulating the dynamics of the mobile robot's motion.

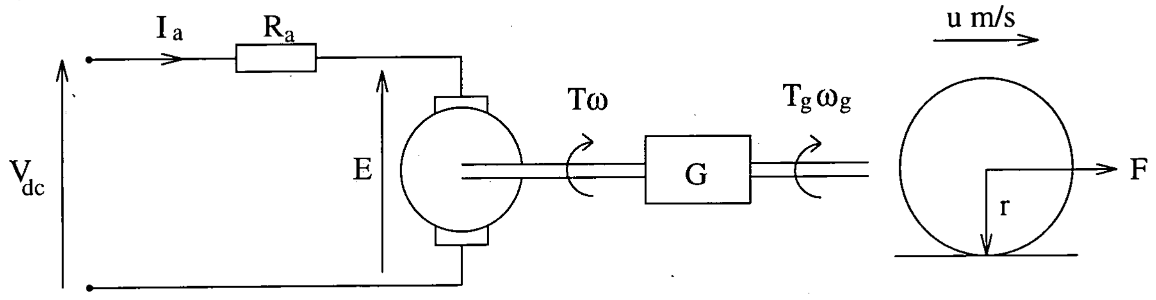


Figure A.1: The circuit diagram for the motor and load

Figure A.1 shows a circuit diagram of a motor and its interaction with a physical system. The terminal voltage V_{dc} is applied to the motor producing an armature current I_a in the rotor. A back e.m.f. E is produced in the armature, which rotates at ω rads per second, with a torque of T Nm. This is transferred to the wheels by a gearing down ratio G. The three wheels which evenly distribute the torque to the floor without slippage are modelled as one drive wheel of radius r. The wheel transfers the shaft's geared down torque T_g at speed ω_g to the floor to produce a forward propulsive force on the robot of F newtons. This produces a forward velocity of u m/s.

Applying Kirchoff voltage law around the circuit we get:

$$E = V_{dc} - I_a R_a \tag{A.1}$$

and Faraday's law states :

$$E = K\Phi\omega \tag{A.2}$$

Where K is a constant and Φ is the flux per pole. The developed torque in the motor is then given by:

$$\mathrm{T} = \frac{\mathrm{EI_a}}{\omega} \tag{A.3}$$

Substituting A.1 into A.2 we get,

$$\mathrm{K\Phi}\omega = \mathrm{V_{dc}} - \mathrm{I_aR_a} \tag{A.4}$$

and substituting A.2 into A.3 we get:

$$\mathrm{T} = \mathrm{K\Phi I_a} \tag{A.5}$$

The gearing down of $\mathrm{T}\omega$ to $\mathrm{T_g}\omega_\mathrm{g}$ gives,

$$\omega_g = \frac{\omega}{G} \tag{A.6}$$

$$\mathrm{T_g} = \mathrm{TG} \tag{A.7}$$

and the forward velocity u of the robot relates to ω_g by:

$$\mathrm{u} = \omega_\mathrm{g}\mathrm{r} \tag{A.8}$$

Combining A.4,A.6 and A.8 we get:

$$\frac{\mathrm{GK\Phi u}}{\mathrm{r}} = \mathrm{V_{dc}} - \mathrm{I_aR_a} \tag{A.9}$$

Substituting A.5 into A.7 gives:

$$\mathrm{T_g} = \mathrm{GK\Phi I_a} \tag{A.10}$$

Then substitute A.9 into A.10 to get:

$$\mathrm{T_g} = \frac{\mathrm{GK\Phi}}{\mathrm{R_a}}\left(\mathrm{V_{dc}} - \frac{\mathrm{GK\Phi}}{\mathrm{r}}\mathrm{u}\right) \tag{A.11}$$

The propulsive force F is related as:

$$\mathrm{F} = \frac{\mathrm{T_g}}{\mathrm{r}} \tag{A.12}$$

By substituting A.12 into A.11 we get the mathematical mode for the robot's propulsion:

$$\mathrm{F} = \frac{\mathrm{GK\Phi}}{\mathrm{rR_a}}\left(\mathrm{V_{dc}} - \frac{\mathrm{GK\Phi}}{\mathrm{r}}\mathrm{u}\right) \tag{A.13}$$

Appendix B

Description of the A* graph search algorithm

The A* graph search algorithm presents a method for computing the shortest paths through the network from a start to a goal destination. Given that the robot knows where it is, where it wants to move and a connected topology, this algorithm can be used to direct the robot on a collision free path.

This A* algorithm is an artificial intelligence search technique and is based on a branch and bound search . The start node on the graph is taken and its connections are expanded to form a tree of depth two. The lengths of these paths are calculated and the lower one is selected to be expanded again. The length of the path must include the remaining distance to the destination and this is normally calculated by direct distance. The above process is repeated until the goal has been reached or no further paths exist. In the latter case this means that there is no topological link to the destination node. In the A* search process multiple paths to a node are also deleted thus reducing the search space. The following shows the algorithmic steps required to implement the A* search process as described in [50].

1. Use the queue to store all the partially expanded paths.

2. Initialise the queue by adding to the queue a zero length path from the route node.

```
3.    Repeat
          Examine the first path in the queue.
          If it reaches the goal node then success.
          Else { continue search }
              Remove the first path from the queue.
              Expand the last node in this step by one.
              Calculate the cost of these new paths.
              Add the new paths to the queue.
```

```
        Sort out the queue in ascending order according to the sum of the cost of the
        expanded path and the estimated cost of the remaining path, for each path
        **If** more than one path reaches a subnode
        **Then** delete all but the minimum cost path
    **Until** the goal has been found or the queue is empty.
```

4. If the goal has been found return success and the path, otherwise return failure.

Appendix C

The Circuit Board Layouts

There are two custom built electronic circuit boards used on the robot. The power and drive circuit board houses the power supplies and the drive circuits for the motors. The control circuit board has the computer card mounted on top, and contains all the control hardware. This appendix documents the layouts of both boards and the connectors that are used.

All of the connectors, except C0 and C27, are composed of connector pins which can be selectively removed. This allows polarisation of the connectors so that they cannot be accidentally inserted the wrong way round. In the figures all the connectors are marked as rows of boxes, one for each pin. Where a pin has been selectively removed this is indicated by a cross.

The control circuit board layout is shown in figure C.1, with the descriptions of the connectors C0 to C26 listed in figure C.2. The power and drive circuit board is shown in figure C.3. The power transistors that require heat sinks are mounted flush with the upper fixing plate and are screwed down. The board is shown in sections indicating the location of the drive circuits which are then shown in the following figures. Figure C.4 is the DC motor driver circuit. Figure C.5 shows the stepper motor driver circuit of which there are two, one for the head stepper motor, and one for the steering stepper motor. Figure C.6 shows the regulated power supply, and figure C.7 is the circuit layout for the ultrasonic power supply. Finally the connections are shown for the main 'power in' connector from the battery packs in figure C.8.

On the control circuit board, C1 is connected to C31, providing the control signals for the stepper motor. C2 is connected to C35 which are the drive signals for the DC motor. C3 and C6 are inputs from the rotary encoders. C4 and C5 are inputs from the magnetic sensors. C7 to C10 are the connectors that correspond to the ultrasonic sensors 0 to 3, and C19 to C22 relate to the ultrasonic sensors 4 to 7. C11 to c14 are collision sensors 0 to 3 and C15 to C18 are sensors 4 to 7. All the ultrasonic, magnetic, and collision sensor connections are to sensors mounted on the head assembly. C24 is the connector for the head limit IR detector, whereas

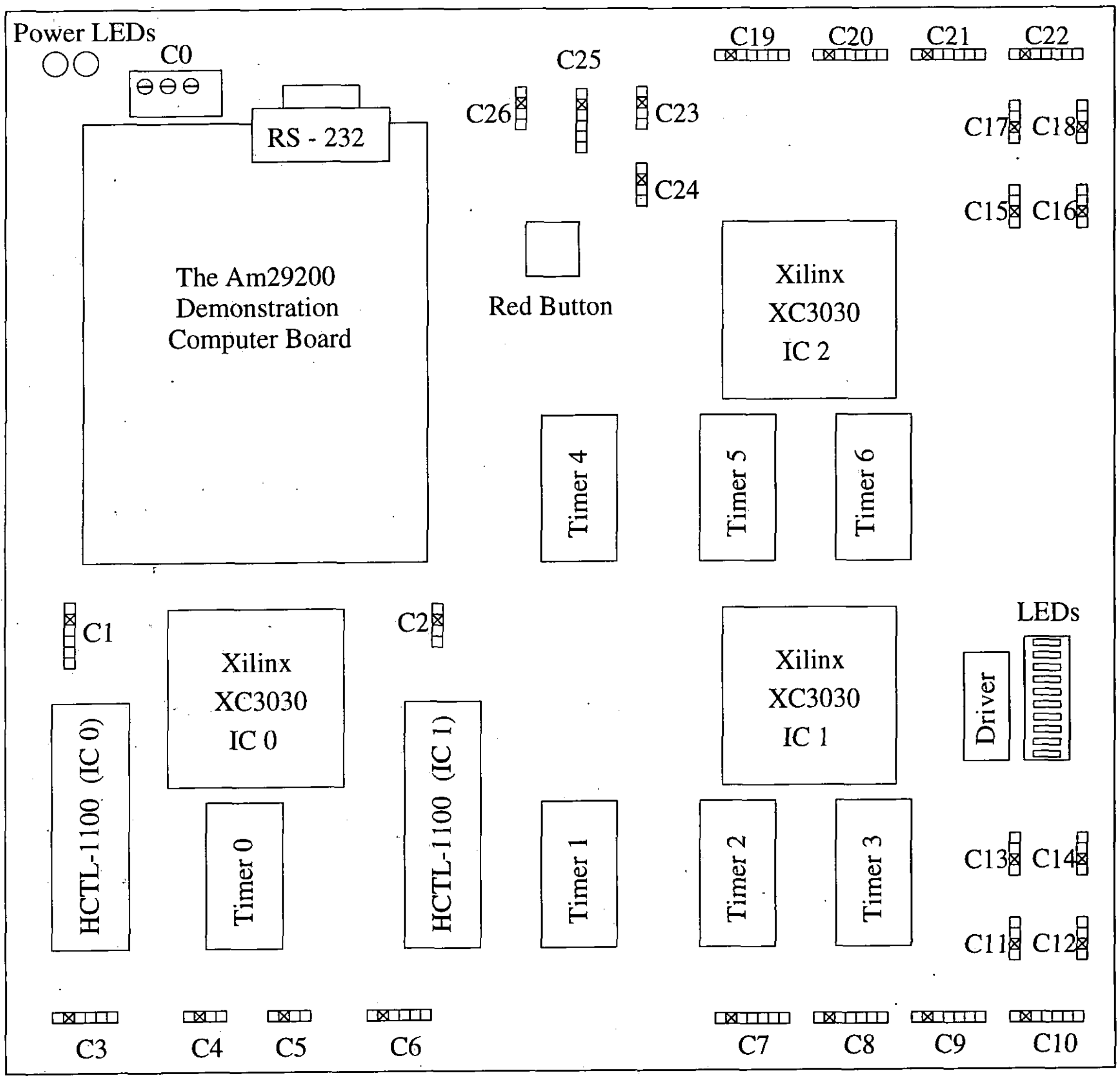


Figure C.1: The control circuit board layout

C23 links to the wheels angle revolution index IR detector. The head stepper motor drive output C25 is connected to C29. The ultrasonic sensor power control signal C26 connects to C32. On the power and drive circuit board, C28 connects to the head stepper motor and C30 connects to the steering stepper motor. C34 and C33 are connected to C0 to provide the power for the control board, and C36 is connected to the DC motor terminals. Finally the battery pack, with its on/off switch is connected to C27 to provide the power to the robot systems.

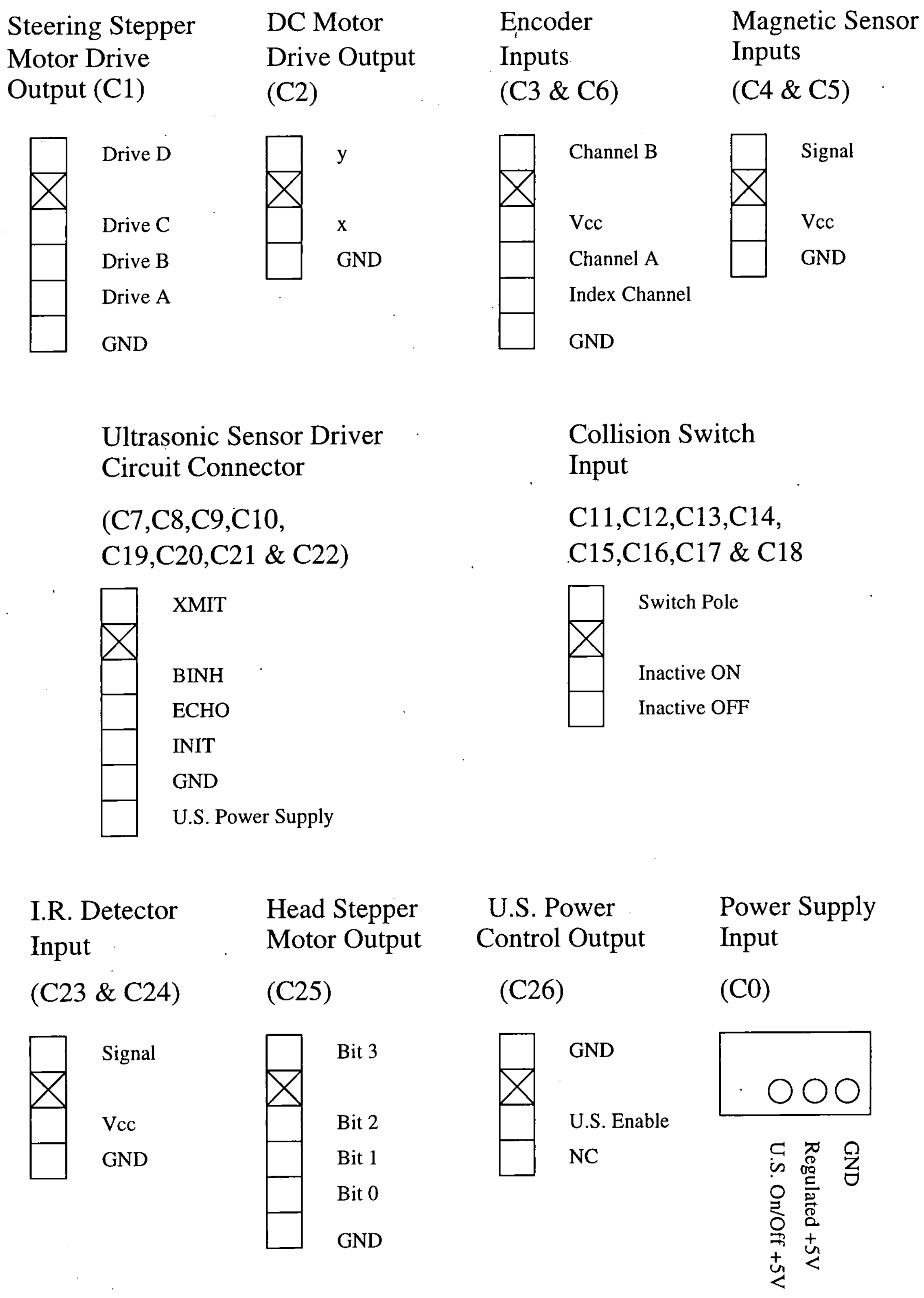


Figure C.2: The control circuit connectors diagram

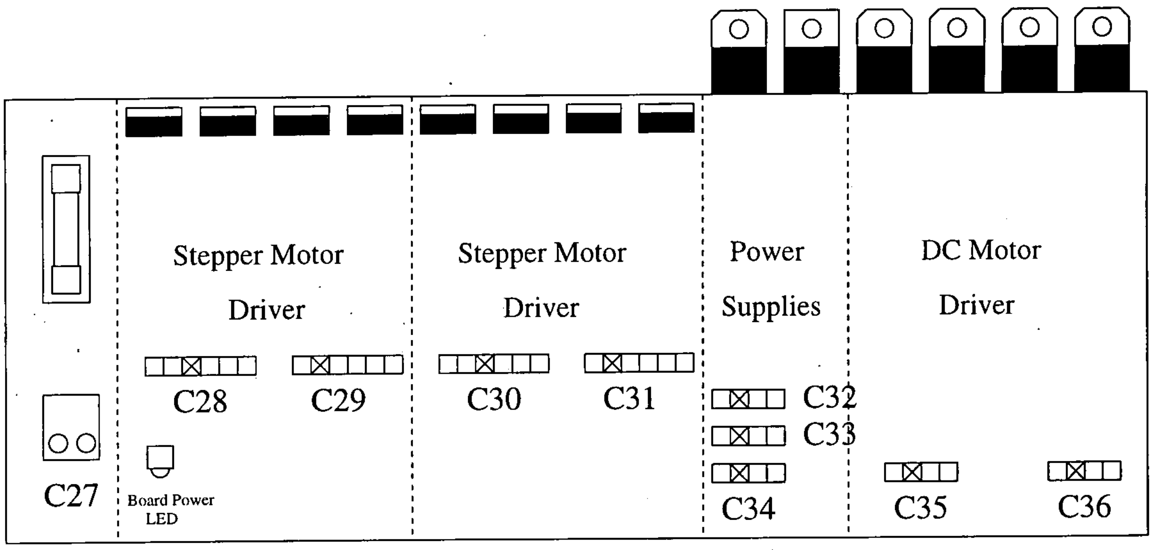


Figure C.3: The power and drive circuit board

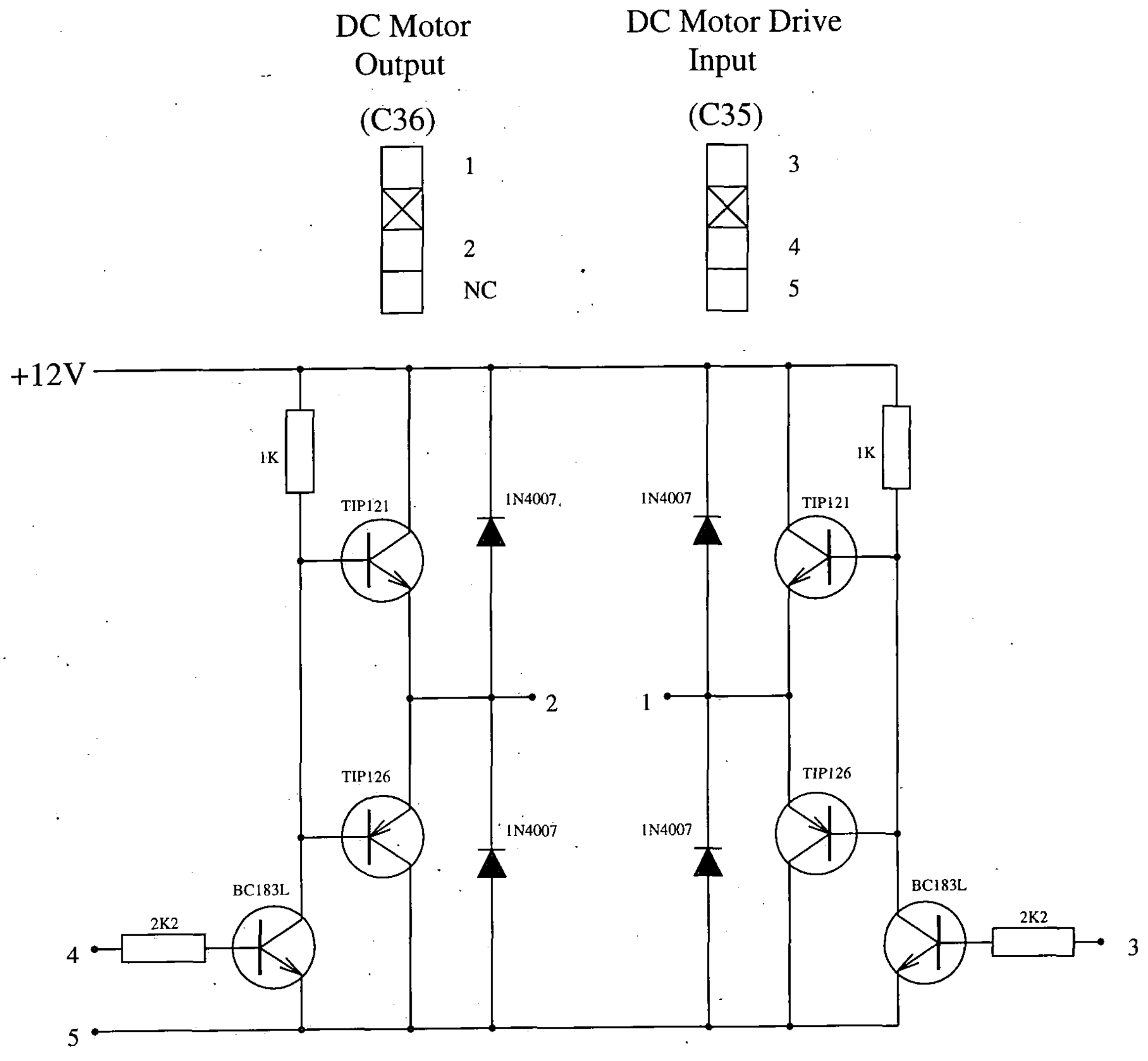


Figure C.4: The connectors for the DC motor driver circuit

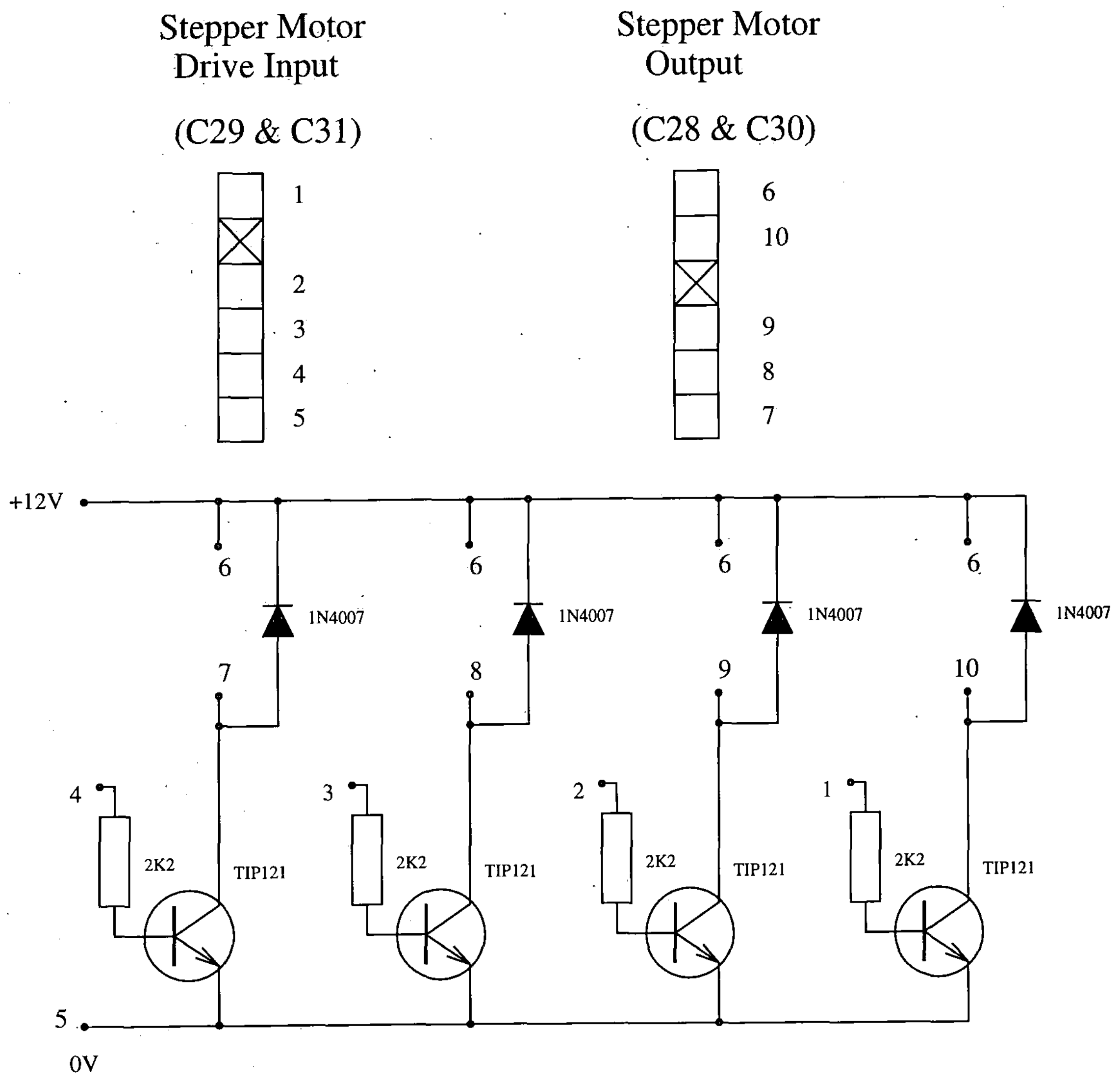


Figure C.5: The connectors for the stepper motor driver circuit

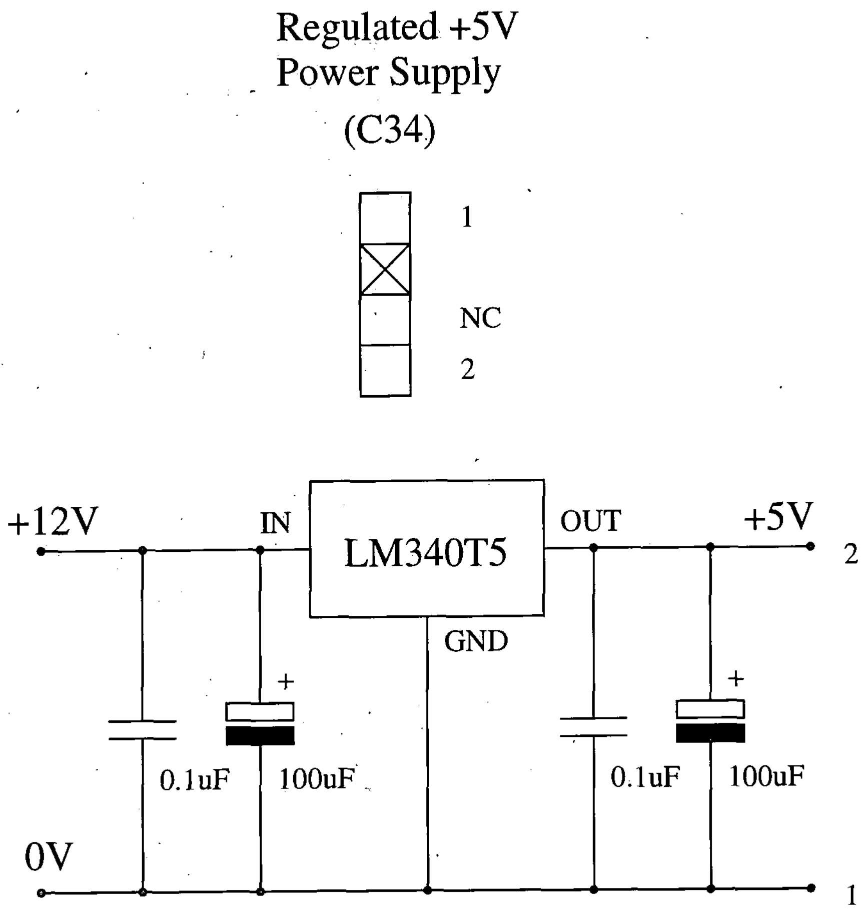
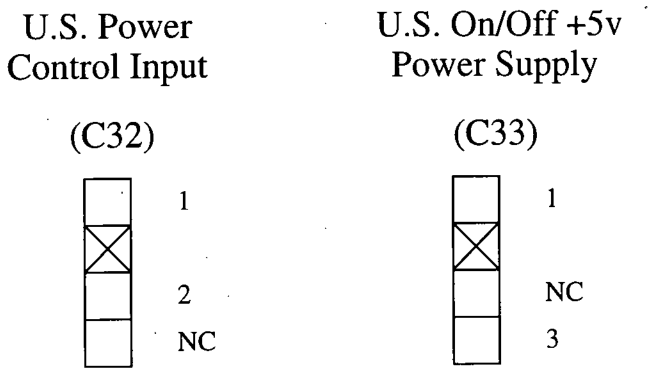


Figure C.6: The connectors for the regulated power supply



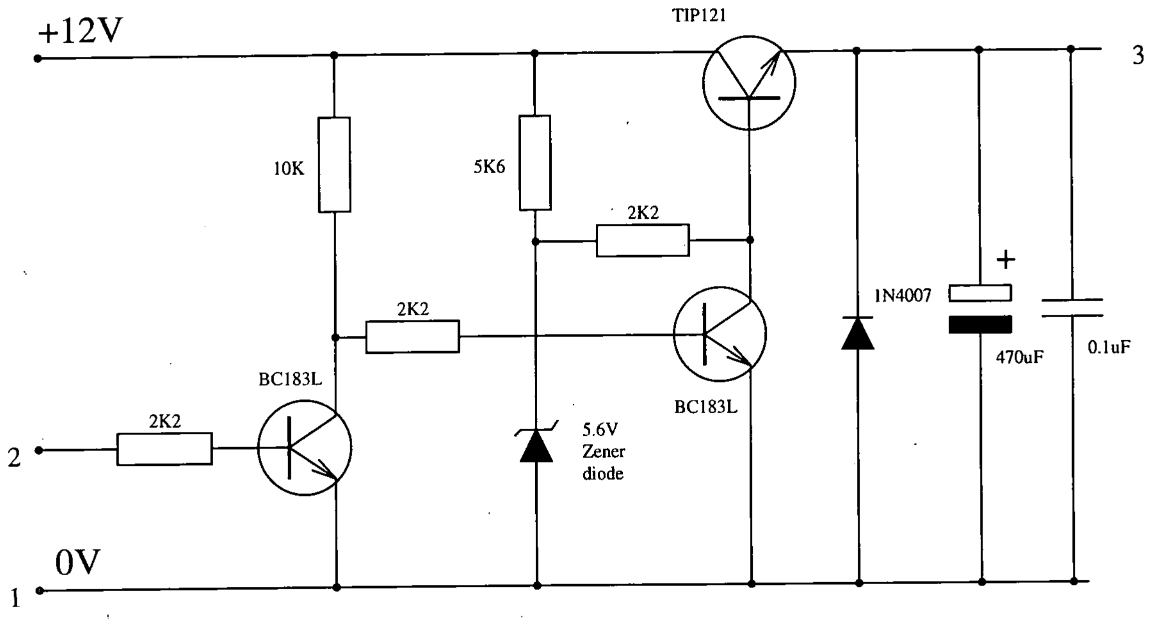


Figure C.7: The connectors for the ultrasonic sensor power supply

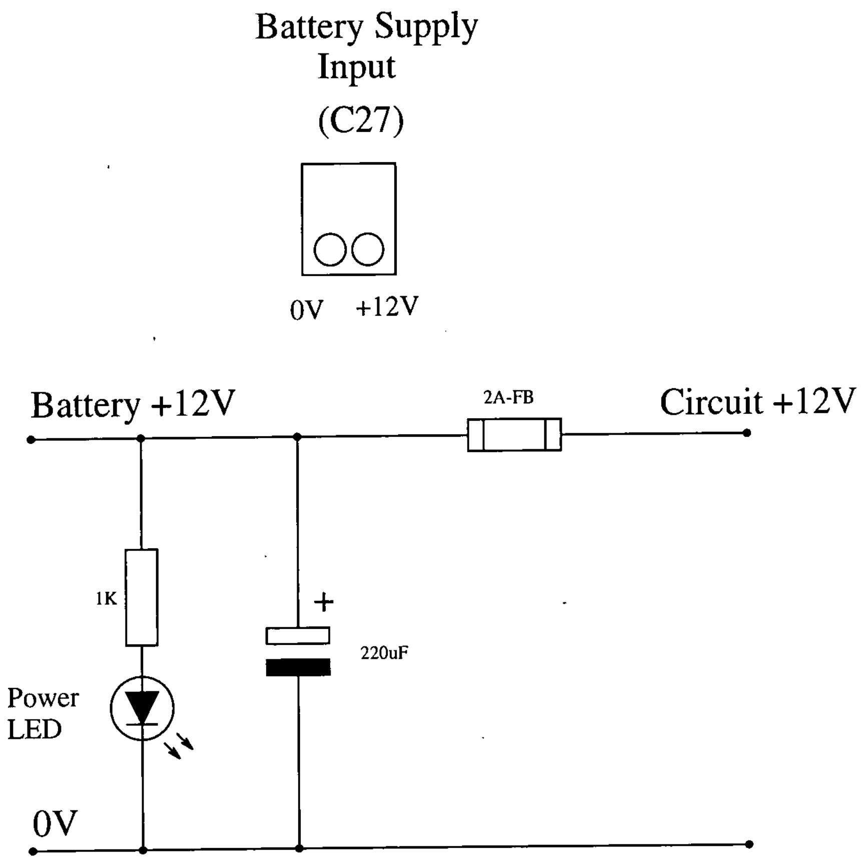


Figure C.8: The connector for the battery supply and fuse protection circuit

Appendix D

The FPGAs Configuration Downloading Description

The FPGAs are programmable logic devices which must be programmed each time they are switched on. The advantage of using them is that they can be 'daisy chained' so that only one IC needs to be directly interfaced, this will then control the flow of configuration data to the other devices. Figure D.1 shows the three devices with the first IC set–up for direct loading of its configuration, and the other two in the daisy chained slave mode. The modes are set by means of changing M0, M1 and M2.

The peripheral mode allows the device to be loaded as a memory location by the computer board. The configuration data is loaded one byte at a time in sequence. After the first IC has been fully programmed, it pipes the configurations to the following ICs, and this is repeated down the chain until all three ICs are fully programmed. To initialise the programming, a high to low transition is written to the Done/Program (D/$\overline{\text{P}}$) line by pio 7 (output pin 7 from a set of 16). After a 4μ second delay this pin is checked to see if it is low, which indicates it is in the program state. After $\overline{\text{INIT}}$ (pio 2) has gone high the device can be written to, as it is out of the initialised state. For each byte, the RDY/$\overline{\text{BUSY}}$ line must be high indicating that it is ready for more data, before a new byte can be written. The address of the IC is the peripheral interface adapter region 0 which is one of five areas of memory set aside for external devices, and hence the Peripheral Interface Adapter Chip Select 0 (PIACS0) is linked to a Chip Select line (CS0). Finally the D/$\overline{\text{P}}$ line goes high indicating that the devices are configured. The data piping is performed automatically, the information being in a serial form from DOUT to DIN under the control of a Configuration Clock (CCLK). When this process is complete, the configured logic becomes operational.

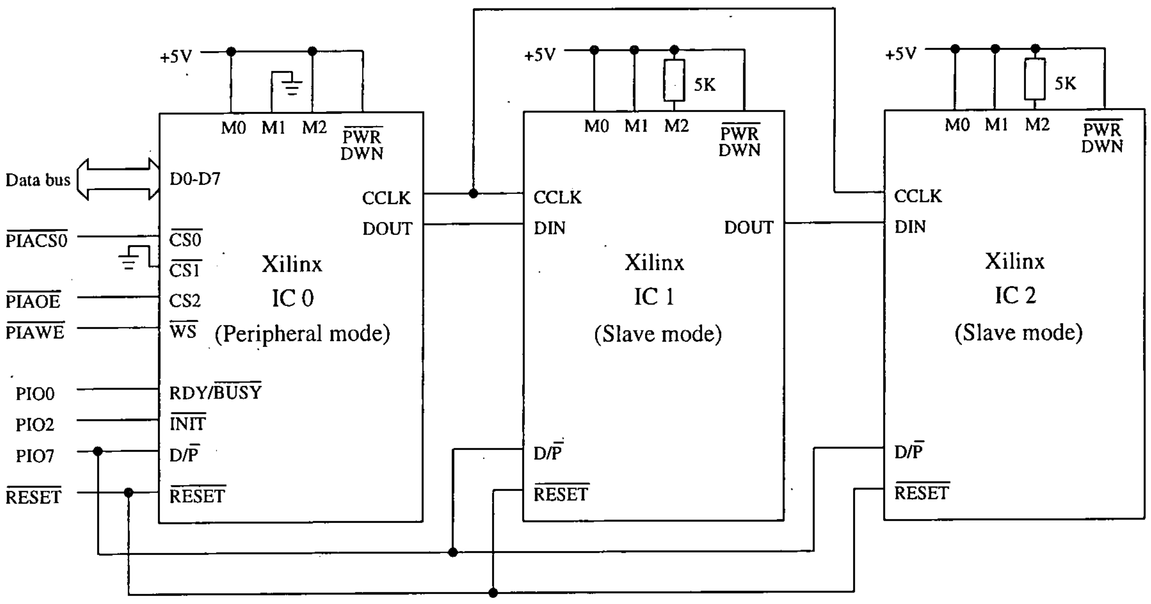


Figure D.1: Programming the FPGAs

References

[1] J. Cohen. *Human Robots in Myth and Science.* George Allen and Unwin Ltd, 1966.

[2] J. Pearsall and B. Trimble, editors. *The Oxford English Reference Dictionary*, page 731. Oxford University Press, 1995.

[3] P. J. McKerrow. *Introduction to robotics*, chapter 8, page 45. Addison Wesley, 1991.

[4] D. A. White and D. A. Sofge. Foreword. In D. A. White and D. A. Sofge, editors, *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, pages xi–xv. Van Nostrand Reinhold, 1992.

[5] L. P. Kaelbling. *Learning in Embedded Systems*, chapter 2, page 16. Cambridge,MA:MIT Press, 1993.

[6] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE J. Rob. Autom.*, pages 14–23, 1986.

[7] E. Gat. On the role of stored internal state in the control of autonomous mobile robots. *AI Magazine*, 14:64–73, 1993.

[8] P. J. Werbos. Neurocontrol and supervised learning: An overview and evaluation. In D. A. White and D. A. Sofge, editors, *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, pages 65–89. Van Nostrand Reinhold, 1992.

[9] P. K. Simpson. *Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations*, chapter 3, pages 7–22. Pergamon Press, 1990.

[10] S. Nagata, M. Sekiguchi, and K. Asakawa. Mobile robot control by a structured hierarchical neural network. *IEEE Control Systems Magazine*, 2:69–76, 1990.

[11] L. Meeden, G. McGraw, and D. Blank. Emergent control and planning in an autonomous vehicle. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 735–740, 1993.

[12] S. B. Thrun. Exploration and model building in mobile robot domains. In *Proceeding of the IEEE International Conference on Neural Networks*, volume 1-3, pages 175–180, 1993.

[13] D. Gachet, M. A. Salichs, L. Moreno, and J. R. Pimentel. Learning emergent tasks for an autonomous mobile robot. In *IROS '94 - Intelligent robots and systems: Advanced Robotic Systems*, volume 1-3, pages 290–297, 1994.

[14] P. K. Simpson. *Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations*, chapter 3, page 15. Pergamon Press, 1990.

[15] R. Pfeifer and P. Verschure. Distributed adaptive control: A paradigm for designing autonomous agents. In *Towards a practice of autonomous systems. Proc. of the first international conference on artificial life*, pages 21–30, Paris France, Dec 1991. Cambridge,MA,USA:MIT Press.

[16] J. N. H. Heemskerk and F. A. Keijzer. A real-time implementation of a schema driven toy-car. In *Proceedings of the Workshop on Neural Architectures and Distributed AI:From Schema Assemblages to Neural Networks*, 1993.

[17] M. M. Kokar and S. A. Reveliotis. Reinforcement learning: Architectures and algorithms. *International Journal of Intelligent Systems*, 8:875–894, 1993.

[18] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[19] A. Ram, R. Arkin, G. Boone, and M. Pearce. Using genetic algorithms to learn reactive control parameters for autonomous robotic navigation. *Adaptive Behaviour*, 2(3), 1994.

[20] K. Moorman and A. Ram. A case-based approach to reactive control for autonomous robots. In *AAAI fall symposium on "AI for real-world autonomous mobile robots"*. Cambridge,MA, October 1992.

[21] N. J. Nilsson. Teleo–reactive programs for agent control. *Journal of artificial intelligence research*, 1:139–158, 1994.

[22] N. Francechini, J. M. Pichon, and C. Blanes. From insect vision to robot vision. In *Phil. Trans. R. Soc. Lond. B*, pages 283–294, 1992.

[23] L. Tarassenko, M. Brownlow, G. Marshall, and J. Tombs. Real-time autonomous robot navagation using VLSI neural networks. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing*, volume 3, pages 422–428, 1991.

[24] C. Peacock and H. Bolouri. A neural network controlled mobile robot. In *Neural Networks and Fuzzy Logic*, 1993.

[25] S. B. Thrun and K. Möller. Active exploration in dynamic environments. In *Advances in Neural Information Processing Systems 4*, pages 531–538, 1992.

[26] T. Mitchell and S. Thrun. Explanation-based neural network learning for robot control. In *Advances in Neural Information Processing Systems 5*, 1992.

[27] B. Yamauchi and R. Beer. Integrating reactive behaviour, sequential behaviour, and learning using dynamical neural networks. In *Third International Conference on Simulation of Adaptive Behaviour*, pages 382–391, 1994.

[28] R. J. Mitchell, D. A. Keating, and C. Kambhampati. Learning strategy for a simple robot insect. In *IEE Control 94*, volume 1, pages 492–497, 1994.

[29] A. Ram and J. C. Santamaria. A multistrategy case-based and reinforcement learning approach to self-improving reactive control systems for autonomous robotic navigation. In *Proc. of the second international workshop on multistrategy learning*, May 1993.

[30] L. P. Kaelbling. An adaptable mobile robot. In *Towards a practice of autonomous systems. Proc. of the first international conference on artificial life*, pages 41–47, Paris France, Dec 1991. Cambridge,MA,USA:MIT Press.

[31] S. Madadevan and J. Connell. Automatic programming of behaviour-based robots using reinforcement learning. *Artificial Intelligence*, 55:311–365, 1991.

[32] M. Dorigo and M. Colombetti. Robot shaping: Development situated agents through robot learning. Technical Report 92-040, International computer science institute, 1992.

[33] M. Dorigo and M. Colombetti. Training agents to perform sequential behaviour. Technical Report 92-023, International computer science institute, 1993.

[34] J. R. Koza. Evolution of subsumption using genetic programming. In *Towards a practice of autonomous systems. Proc. of the first international conference on artificial life*, pages 110–119, Paris France, Dec 1991. Cambridge,MA,USA:MIT Press.

[35] J. R. Koza. *Genetic Programming: on the programming of computers by means of natural selection.* A Bradford Book. Cambridge,MA:The MIT Press, 1992.

[36] A. Saffiotti, E. H. Ruspini, and K. Konolige. Blending reactivity and goal-directedness in a fuzzy controller. In *Procs. of the second IEEE conf. on fuzzy systems*, pages 134–139. San Francisco,CA,March, 1993.

[37] D. Eustace, D. P. Barnes, and J. O. Gray. A behaviour synthesis architecture for co-operant mobile robot control. In *IEE Control 94*, volume 1, pages 549–554, 1994.

[38] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE J. Rob. Autom.*, pages 14–23, 1986.

[39] J. Borenstein, H. R. Everett, and L. Feng. *Navigating Mobile Robots: Sensors and Techniques.* A. K. Peters, Ltd., Wellesley, MA, 1996.

[40] N. Burgess, M. Recce, and J. O'Keefe. A model of hippocampal function. *Neural Networks*, 7:1065–1081, 1994.

[41] U. R. Zimmer. Self localisation in dynamic environments. *IEEE/SOFT international workshop BIES*, May 1995.

[42] B. Fritzke. Growing cell structures - a self organising network for unsupervised learning. *Neural Networks*, 7(9):1441–1460, 1994.

[43] P. P. Acarnley. *Stepping Motors: a guide to modern theory and practice*, chapter 4, page 48. IEE Control Engineering Series 19, 1985.

[44] J. Borenstein and Y. Koren. Error eliminating rapid ultrasonic firing for mobile robot obstacle avoidance. *IEEE Transactions on Robotics and Automation*, 11:132–138, 1995.

[45] R. Noble. *FGM-3: Magnetic Field Sensor.* Speake & Co Limited, Elvicta Estate, Crickhowell, POWYS, NP8 1DF, 1996.

[46] *The SA-29200 Demonstration Board User's Manual.* Advanced Micro Devices,Inc, 5204 E.Ben White Blvd. Austin, Texas 78741-7399., 1992.

[47] Hewlett Packard. *General Purpose Motion Control IC*, 1990. Technical Data Sheet: Component HCTL-1100.

[48] Hewlett Packard. *Design of the HCTL-1000's Digital Filter Parameters by the Combination Method.* Application Note 1032.

[49] Beckman and Spizzichino. *The Scattering of Electromagnetic Waves from Rough Surfaces*, chapter 2, page 10. Pergamon Press Ltd. Headington Hill Hall, Oxford 4 and 5 Fitzroy Square, London. W1, 1963.

[50] P. J. McKerrow. *Introduction to robotics*, chapter 8, page 454. Addison Wesley, 1991.