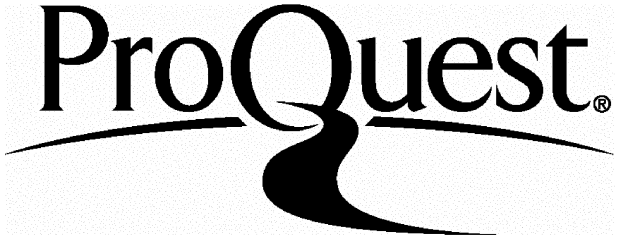
Autonomy in the Real-World:
A Behaviour Based View of Autonomous Systems Control in an Industrial Product Inspection System

Miles Pebody

Sira / University College London Postgraduate Training Partnership

A thesis submitted for the degree of Doctor of Philosophy
The University of London

13th June 1996



ABSTRACT

The thesis presented in this dissertation appears in two sequential parts that arose from an exploration of the use of Behaviour Based Artificial Intelligence (BBAI) techniques in a domain outside that of robotics, where BBAI is most frequently used. The work details a real-world physical implementation of the control and interactions of an industrial product inspection system from a BBAI perspective. It concentrates particularly on the control of a number of active laser scanning sensor systems (each a subsystem of a larger main inspection system), using a subsumption architecture. This industrial implementation is in itself a new direction for BBAI control and an important aspect of this thesis. However, the work has also led on to the development of a number of key ideas which contribute to the field of BBAI in general. The second part of the thesis concerns the nature of physical and temporal constraints on a distributed control system and the desirability of utilising mechanisms to provide continuous, low-level learning and adaptation of domain knowledge on a sub-behavioural basis. Techniques used include artificial neural networks and hill-climbing state-space search algorithms. Discussion is supported with examples from experiments with the laser scanning inspection system. Encouraging results suggest that concerted design effort at this low level of activity will benefit the whole system in terms of behavioural robustness and reliability. Relevant aspects of the design process that should be of value in similar real-world projects are identified and emphasised. These issues are particularly important in providing a firm foundation for artificial intelligence based control systems.

Contents

To
Mum, Dad, Erica and Leonora

Acknowledgements

My supervisors for the duration of this work have been Professor John Campbell of University College London and Dr. John Gilby of Image Automation Ltd. and Sira Ltd. I would like to thank them both first and foremost for their extensive, expert and invaluable help over the three years that I have been in London and secondly for the relaxed and friendly nature of our meetings and discussions.

Much of the practical work that is reported here was undertaken at Image Automation Ltd and I am indebted to the staff there for their interest and help in many practical aspects of the experimental work. In particular I would like to thank Val Holding, Pat Williams and Simon Hattersley, all of whom contributed in many ways.

Thanks also to the staff and students of the Sira Technology Centre with whom I have shared the last three years. In particular Tony Allnutt and Bill Simmonds, and also Sean Dodd, Chris Seal, Sonia Ruby, Rebecca Simpson, Ian Ho, Laurence Court, Isobel Bond, Robin Parker and last but not least Grant Bedford who also put-up with me as a flatmate for two years.

At UCL I was happy to share an office and consequently many valuable discussions with Tim Norman. Thanks to him for his continuous willingness to listen and make suggestions. Others that I must also thank from the UCL Computer Science department include Raghbir Sandhu, James Kadirire and Stephanie Warrick. Others at UCL who I would like to thank include Bill Glencros and David Miller.

Thanks also to Tim Smithers, Maja Mataric, Rebecca Gay and Jo Illsley for advice, discussion and for proof reading of earlier drafts of this thesis.

This research was undertaken within the Postgraduate Training Partnership established between Sira Ltd. And University College London. Postgraduate Training Partnerships are a joint initiative of the Department of Trade and Industry and the Engineering and Physical Sciences Research Council. They are aimed at providing research training relevant to a career in industry, and fostering closer links between the science base, industrial research, and industry.

Finally. I came to be in the fortunate position of being able to study for a PhD. as a result of obtaining an MSc. from the University of Edinburgh. This was in no small part due to the considerable financial help provided by my grandparents. This thesis is also dedicated to their memory.

1. Introduction

This thesis presents a series of experiments and discusses the issues, relating to the application of a rapidly expanding, but still young, paradigm in the domain of Artificial Intelligence (AI) and autonomous systems control. Popularly known amongst its practitioners as Behaviour Based Artificial Intelligence (BBAI), it is the result of an increasing acceptance that traditional AI techniques have proved largely unsuccessful when applied to the control of real world situated systems and robotic systems in particular.

In 1986 Rodney Brooks of the Massachusetts Institute of Technology published a paper entitled "A robust layered control system for a mobile robot." [Brooks86] in which a significantly different approach to the construction of mobile robot control structures was presented in the form of the Subsumption Architecture and its associated bottom-up development methodology. This definitive work laid the ground for the subsequent opening of a new direction for research into the practice and nature of intelligent control. Since that time many variations of the subsumption architecture have been proposed as well as many alternatives that follow the same basic premises. The work has matured and there are many impressive demonstrations of control techniques - for example [Brooks89], [Ferrell93], [Horswill93], [Arkin87] [Mahadevan & Connell 92] [Pfeifer & Verschure 91] and [Gat *et al* 94]. However, there is little evidence of use outside of the robotics domain. This thesis addresses the techniques and philosophies behind BBAI and looks at the potential usage in areas other than that of mobile robotics. In particular, an industrial product inspection application is used for illustration. The nature of BBAI system design and functionality is characterised, and a direction of further development is suggested, with particular emphasis on the need for increased automation of control structure generation with concentration on issues of scaleability. In response to this, potential solutions to some of these problems in the form of learning and adaptive mechanisms are presented as an extension to a basic subsumption-like control architecture. Experiments demonstrating the successful use of the extended subsumption architecture in a real-world situated and distributed test-bed are reported.

The work presented in this dissertation deals with many aspects of system development. For example, the systems discussed are distributed, or at least decentralised, in that there is no single and dominant centre of computation or control. The systems involve parallel architectures and use a number of techniques as and where appropriate, for example artificial neural networks and heuristic search strategies. This work also covers aspects of asynchronous system interactions, as well as bringing in discussion on learning and adaptivity using models from psychology and

ethology such as reinforcement learning [Kaebling93] and classical conditioning [Pfeifer & Verschure 91]. Finally, in keeping with the practices of BBAI other aspects of real-world biological systems are used for both illustration and inspiration, for example the neuroscience work of [Robinson90].

The rest of this chapter introduces aspects of an industrial domain with an emphasis on research into the nature of systems control. The main points of the thesis are then presented along with a guide to the contents.

1.1. An Industrial Viewpoint: The Real-World?

There is a significant difference between pure research and actual application-oriented engineering development. Research is a scientific exploration of a phenomenon that uses both empirical and theoretical techniques. It is often quite a polarised endeavour with researchers fighting to promote their ideas, work and results. When an idea, mechanism or other result of research is adapted for commercial use, the basis of development changes from one concerning the furthering of knowledge to a concerted effort to build a practical system that is (amongst other factors) reliable, robust and easily maintained. The result is a merging of ideas by applications engineers who are not inhibited by the philosophical flavours or principles of a particular paradigm. Their task is to provide solutions to a justifiably conservative and sceptical industry. The constraints on a system that has to operate in the real-world (of the manufacturing industry, for example) are dictated by issues of cost and competitiveness.

Until recently by far the most used, developed, and accepted method of control has been that of classical control theory (an introduction to which can be found in references such as [Dorf92] and [Bissell94]). Mechanisation and mass-production have incorporated control systems and stimulated control research in parallel since the Victorian industrial revolution. Smithers and Brooks discuss the effect of history on the development of both control theory and AI (in [Smithers94] and [Brooks91]) and highlight a continuing requirement for increasingly sophisticated automatic systems that are able to function in increasingly diverse domains with a decreasing degree of supervision. The techniques of classical control are proving to be limited, especially in non-linear and extended real-world systems which require greater variety in system behaviour, for examples and discussion see [Jones *et al* 91], [Harris94] and [Franklin & Selfridge 90]. As a result the techniques of AI-based control are being used to augment the classical controllers. Perhaps the best example of this is in the development of "expert controllers" or "adaptive controllers". These are a combination of AI-developed expert systems and the fundamental building block of the classical control domain: the Proportional Integrating Differential (PID) controller. These hybrid systems typically utilise a database of rules (predefined by an expert) to update the critical parameters of a feedback control system to match changing circumstances in the controlled process [Rodd & Verbruggen 92]. Despite this

development there are still many problems facing the control community as systems requirements become more complex. One example is the increasing use of distributed computing and localised embedded control in large (particularly industrial) systems. These issues are discussed in [Rodd & Deravi 89] and [Musliner *et al* 94].

This thesis deals with the real-world application of a new methodology in AI control: that of Behaviour Based AI. To date most BBAI work has dealt with the control of relatively simple mobile robots. Although this is in keeping with one of BBAI's fundamental principles concerning the necessity of testing systems and experimenting with them in their intended domain (which in the case of mobile robots is the real-world) much evidence is still needed to demonstrate that the ideas are sound in a broader range of control applications. One of the major contributions of the work reported in this dissertation is that the focus has been on the implementation of a BBAI control architecture for an industrially situated system. Consequently, whilst the issues addressed are generally relevant to BBAI, they are also of interest and relevance to potential industrial users.

The requirements of an industrial system are similar in general to the requirements of the mobile robots used in BBAI:

Reliable and robust behaviour: This is particularly important in an industrial setting since the systems must be able to run for extensive time periods (if not continuously) with little or no operator interaction. The industrial environment is typically noisy, dirty and unforgiving of mistakes. Each of these factors is a typical target for BBAI researchers in the design and construction of their robots.

Cost effectiveness: An overly expensive solution to a problem both in terms of material resources and operator time will be ignored by industry which necessarily operates on a commercial footing. In a similar way BBAI research is typically conducted under the constraints of limited funding which, when materials and components are needed for construction of real-world test-beds, provides a similar setting to that of industry.

Ease of operation: Industry requires that machinery be maximally self-sufficient, but when operator interaction is required this should be straightforward and easily achieved. The direction of BBAI research towards autonomy and self-sufficiency complements this. However, ease of use is an issue that is implementation-specific and is not necessarily best demonstrated on one-off prototype systems.

Ease of maintenance: A system that requires extensive maintenance is not attractive to a commercial concern. Although BBAI systems have not yet been developed for an industrial situation, it would seem that the behaviour based modular construction that is inherent in the BBAI methodology is extremely relevant in this respect.

Flexible integration with other systems: A piece of equipment is more desirable if it can be incorporated easily into existing sets of machinery and at the same time have a potential for simple and straightforward expansion or upgrading. The modular approach of BBAI in conjunction with issues of multi-agent interactions are important in this respect.

1.2. The Thesis

The results presented in this thesis are arranged in two sequential parts that have arisen from an exploration of the use of Behaviour Based Artificial Intelligence (BBAI) techniques in a domain outside that of robotics (where BBAI has been used most frequently). The work details a real-world physical implementation of the control and interactions of an industrial product inspection system from a BBAI perspective. It concentrates particularly on the control of a number of active laser scanning sensor systems (each a subsystem of a larger main inspection system), using the Subsumption Architecture [Brooks86]. This industrial implementation is in itself a new direction for BBAI control and an important aspect of the thesis. However, the work has also led on to the development of a number of key ideas which contribute to the field of BBAI in general.

1. Synthetic autonomous systems (of which the mobile robots of BBAI research are a small subset) have been constructed and developed successfully on a bottom-up basis using the methodologies of BBAI. However, there has in general been little further breakdown or recognition of low-level behavioural phenomena that are particularly evident in truly distributed control systems. This first point of the thesis has been to build a case for recognising a lower bottom level of interaction in autonomous systems development than has been customary so far. In other words, a sub-behavioural layer of activity is highlighted that consists of its own agent-like processes interacting at a subsystem or component level, each with its own local set of physical and temporal constraints, below those of the main physical agent. Additionally the work reported here suggests that truly distributed behaviour based systems may be viewed more extensively as a hierarchy of levels of communities of semi-autonomous and autonomous agent entities, the granularity of which depends on the level of abstraction used by the observer of the system. It is claimed that viewing distributed systems in this way provides a useful framework for design. This outlook adds to the more usual approach of BBAI control that, whether by design or side-effect, tends to treat an agent as a singular physical entity situated in an environment that is controlled by some singular architecture of parallel-behaviour-achieving processes.

2. Existing control architectures from BBAI are successful due to the incorporation of designer's specialised domain knowledge into all levels of the target agent's control structure. As systems become more complex, the provision of this domain specific knowledge is becoming more difficult. The number of variables and process configurations

that have to be pre-specified is increasing and the design process inevitably takes place with incomplete information about future system interactions. One solution to this problem is to provide systems with an ability to acquire the missing knowledge for themselves. This is currently being implemented by researchers in the field using techniques of adaptation and learning, with the level of study and application being generally at that of the physical agent system's behaviour. The second part of this thesis builds on the first part by addressing the problem of acquisition of domain knowledge at the lowest levels of a behaviour based system, below those of the physical agent's behaviour. It suggests that agent entities at each abstraction of system level (identified in part 1 above) must be given their own means to learn and to adapt to ongoing changes in their situation on a continuous basis.

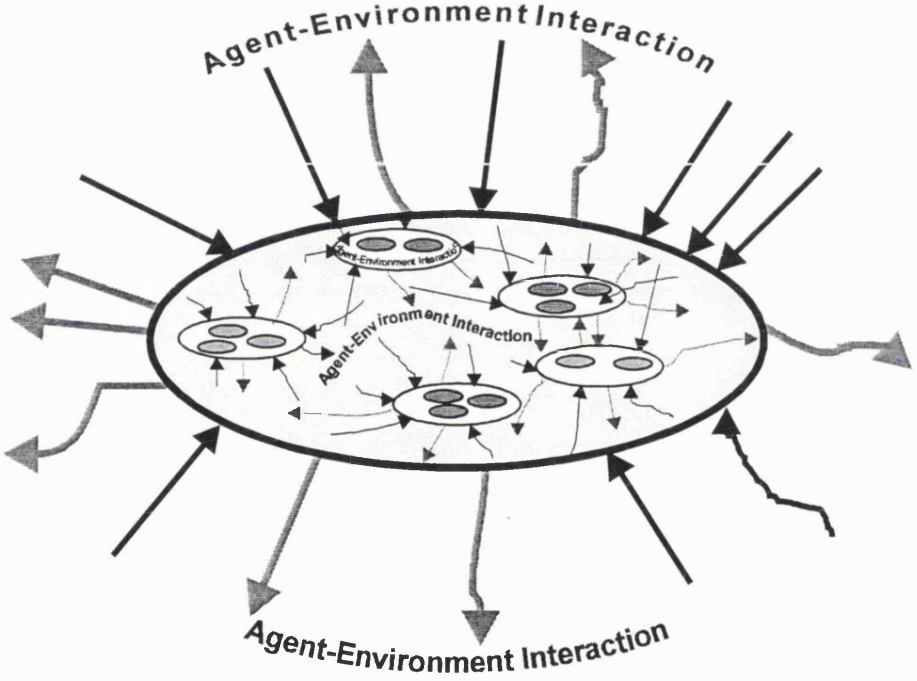


Figure 1.1. An agent as a hierarchy of lower-level agent processes.

The first part of the thesis addresses the increasingly relevant question of distributed and local control of component parts of a system. For example, many mobile robots typically utilise more than one programmable microcomputer within their systems. Often there is a central processor, responsible for supporting the primary algorithmic control structure, which has access to one or more "slave" processors which, in turn, deal with specific aspects of sensor and actuator functionality. It is suggested that, rather than viewing their association as a master-slave relationship, it will become increasingly useful to regard each of these systems as independent and interacting components of a system, or in other words as a group of asynchronous interacting agents at a lower level than that of the physical real-world agent. The world for this level of agents consists of sensory and internal signal transfers and possibly a lower level of interacting

processes. Figure 1.1 shows the idea of a hierarchy of agency levels. At each level agents interact and influence their local and global situation.

The second part of the thesis examines the problem of providing component subsystems of a real-world agent with an ability to tune or adapt automatically to changing situations. This direction was taken as a result of observing that much of the global performance of a system is highly dependent on low-level mechanisms. Higher-level behaviour cannot compensate for unreliable and poorly-performing processes at the sub-behavioural, subsystem level. In [Brooks86] and the later work [Brooks & Stein 93], it is emphasised that the subsuming higher layers of a subsumption architecture add levels of overall competence above thoroughly tested and debuged levels. They do not and cannot correct for bad design or defective lower control mechanism and can only influence its behaviour. In response to this, a building-block approach to embedding learning and adaptation mechanisms into behaviour based control programs is presented. The idea here is that complex input-output mappings associated with basic behaviour rule specifications are learned on a localised basis and that critical parameters are tuned continuously. This building-block approach leads on to and reinforces the view of a system as being constructed from a hierarchy of interacting agent processes. This view means that learning and adaptivity may be usefully built into a system at each level of agency with the dynamics that control such activity being controlled by component parts at whatever level is being addressed.

At the end of the day this thesis is about providing systems with a suitable set of interaction dynamics that allows them to perform their required tasks in a maximally reliable and robust way. Ultimately this is an engineering task. Many parts are required to function together and many "off the shelf" components and techniques are available for use. It would be foolish not to make use of these wherever possible in the design and construction of an autonomous agent system. However, many of these techniques have not been used together previously so experimentation is required to find and characterise the nature of the new combinations. In doing this it is probable that new aspects and factors will be encountered that require new solutions and techniques in order to make the system behave in the right way. The former exercise is the engineering aspect of this work while the latter is the scientific contribution. The thesis reports on work that covers both these aspects: Engineering and Science. The engineering has been in the application of BBAI in a new domain: that of an industrial situation. The science has arisen out of perceived needs and requirements of the BBAI techniques used.

1.3. Contents of the Thesis

The work of this thesis deals to a large extent with the low-level control of a sensory subsystem using techniques and methodologies from BBAI. At this level it may be claimed that there is considerable overlap with techniques of classical control theory, although it may also be argued conversely that there are significant differences. Consequently the next two chapters are of an

introductory nature and present further detail on this debate. Chapter 2 discusses issues of general relevance to AI based control and in particular brings in aspects of control theory in order to add some perspective to the connections, similarities and overlaps between the fields. It is intended that chapter 2 serves to introduce some of the background behind the relatively young field of behaviour based AI which is presented in more detail in chapter 3.

Chapter 3 provides an overview and case study of aspects of BBAI, an already much-referenced topic but one that has, until this point, received less attention than it deserves. This then leads into three chapters that present the first part of the thesis. Chapter 4 covers the implementation of an industrial laser inspection test-bed that is used throughout this dissertation to illustrate the points raised. This experimental test-bed is also used as a source of empirical results, the first of which are presented in chapter 5. Chapter 6 discusses the nature of behaviour based control in the light of the work of chapter 5 and details ideas for a distributed and multi-agent approach to system construction. The first part of the thesis is concluded in chapter 6 with emphasis on the existence of a sub-behavioural level of agent activity that is evident in distributed control systems at the low levels of sensor and actuator subsystem interaction. Chapter 6 ends by suggesting that the way forward for complex systems development requires the automation of many of the design and development processes that are currently hand-specified and constructed (or programmed).

The second part of this thesis is introduced by chapter 7 which outlines a perceived need for mechanisms of low-level adaptivity and learning in BBAI control systems. Several novel solutions to the problem of tuning critical parameters automatically and associating complex input state representations with basic rule declarations are presented. Ideas for this work are taken from the requirements generated as a result of the first part of the thesis and from the perceived current state of BBAI. These solutions are then the subject of experimentation in the laser scanning test-bed and are reported in chapter 8. The work there builds on the behaviour based control structures implemented in chapter 5 and uses the performance as a basis for comparison of the new enhanced and adaptive systems. Chapter 9 discusses the results and finally chapter 10 concludes the thesis by bringing the two parts together.

2. Background Issues

This chapter covers the following:

Issues concerning intelligence and control.

Issues concerning the nature of autonomy and self-sufficiency.

Different approaches to control systems.

The focus of this thesis, in both the first and second parts, is on the distributed control of autonomous systems. Illustration and experiments are provided in later chapters on an industrial laser scanning inspection system test-bed. As such, they have an emphasis on relatively low levels of control (in comparison to the task planning activities that are often regarded as being typical of artificial intelligence) which may be seen by some as the domain of classical control theory. This, however, is not necessarily the case, as will be shown throughout the course of this dissertation. In order to introduce the notion of artificial intelligence based control and how it relates to some of the characteristics of classical control, this chapter deals with a number of issues that are popularly regarded by either or both the AI and classical control communities as being definitive aspects of their fields and that at the same time have led to significant overlap between the two.

This chapter is presented in three main sections. The first discusses the respective views of each community, AI and classical control, on aspects of "intelligence". It is not intended to be definitive but just to acquaint the reader with the wide range of continuous and ongoing debate on the subject. Section 2 then continues to discuss an only slightly less contentious topic: the nature of automatic, autonomous and self-sufficient systems. This is relevant in the light of the first part of the thesis as described in the previous chapter, concerning an agent based view of distributed control systems. Finally section 3 details some aspects of classical control and AI control that are in need of characterising prior to chapter 3 and the introduction to the field of behaviour based artificial intelligence. This chapter aims at providing discussion of an introductory and general nature as background for the main body of the thesis.

2.1. Intelligence and Intelligent Control

This thesis is primarily a text describing work from an AI perspective and consequently, it is necessary to set out briefly what is meant or, perhaps more to the point, not meant by "intelligence" as used in the context of this dissertation. Depending on the research community in

question intelligence has different connotations. For example, in the control engineering field "intelligent control" is often taken to refer to any form of control technique that is not of the basic feedback servo type control mechanism or its derivatives. These are most often the result of applying AI research in the control domain ([Bissell94] and [Dorf92]). The name "Intelligent Control" is perhaps useful in itself as a label of a class of control techniques but it does not satisfy any considered or philosophical view of the nature of intelligence (such as discussed in [Brooks90], [Brooks91], [Beer90] and the introduction of [Luger & Stubblefield 89], for example). "Intelligence" is the central theme of AI research, so here it is perhaps surprising to find that there are probably as many definitions as there are researchers ([Smithers91] lists a selection of definitions from the AI community). Other fields too, including cognitive science and psychology, profess to study or deal with aspects of intelligent behaviour, but again there is very little evidence of any single underlying phenomenon that is being studied that can be specifically labelled as intelligence.

There is an ever-continuing requirement for increasingly complex and sophisticated control solutions to an increasingly wide range of task and application domains. As systems become more complex, so too does the control problem. Applications such as, for example, the automatic control of aircraft, trains and automobiles are becoming more common as well as the use of extensive automation in the area of industrial manufacturing. These systems are required to be both maximally reliable and maximally robust for as wide a range of situations as possible; indeed, in many applications this is the first priority. This advance in complexity has occurred hand in hand with technologies such as electronics, computers and software, where the range of potential solutions is now also feeding back to generate new ideas for applications. In recent years the notion of autonomous systems (as opposed to automatic) has come to the fore as enabling technologies have matured and made possible what at one time were only ideas in science fiction. One active area of research is in the development of free-swimming underwater exploration vehicles that are being designed to work unaided for several days at a time performing tasks such as mapping ocean beds and inspecting pipelines at depths far beyond the reach of any human or human-containing vessel (for example, see [Zorpette94]).

As systems become more complex and capable of wider-ranging task competence, it will become more difficult for them to be seen as simple automatic machines. They will manifest many types of task oriented behaviour and be able to respond to many different situations and problems. It will become difficult to avoid the conclusion that, although these systems are built from known sets of components and programs, they might be intelligent in some way simply because of the observed nature of their abilities and interactions with the world around them. It is not the place of this thesis to dwell on the philosophical issues of intelligence, but the point that a system may "appear" to be intelligent rather than explicitly built or programmed to be is considered to be of central importance. The idea that intelligence is in the eye of the beholder is

certainly advocated. After all, in the final analysis we are interested in systems that can perform some useful task in a reliable and robust way, so that using this as a measure of success rather than some abstract notion of intelligence would appear to be more useful. The idea of intelligence resulting from an observer's interpretation of a system's behaviour is discussed and illustrated beautifully in [Braitenberg84]. Moreover, the idea that intelligence is not built in but rather emerges from the interactions of a system and its domain is a standard viewpoint of BBAI. Other references to this can be found in [Moravek88], [Dreyfus93] and [Brooks86]. In order to avoid confusion, this thesis will not utilise "intelligence" as a defining property of a system.

2.2. Agents: Automata, Autonomy and Self-Sufficiency

"Agent" is a term used with great variety of meaning in the AI, computer science and engineering literature. In the last few years "agent" has been increasingly and popularly used in connection with software processes and in particular with mobile software processes that are intended to navigate and perform tasks on a system of networked computers. [Riecken *et al* 94] provides an extensive discussion and series of papers on a number of aspects of agenthood. In general it seems that an agent is taken to be some discrete entity that has some specific task and some specific domain in which it attempts to realise that task. For example, an estate agent is a person given the task of buying and selling property, where the domain is the world of mortgages, house-buying, surveyors and contract solicitors. An electronic mail handling agent might be a continuously-running software process that is able to sort and prioritise a user's daily messages, the domain being that of the Internet and electronic mail and the user's preferences in types of messages.

The Distributed Artificial Intelligence (DAI) community tends to use "agent" in reference to a specific process within a system. This is particularly so in [Minsky85] in which an AI system is seen to consist of a number of agencies each made up of a number of agents that contribute to the global behaviour of the system. Although Minsky's work here is not really in the same vein as that of more recent DAI, it does suggest some of the main features of such systems. One of these is the notion that agents might enter into communication and interact in some abstract way in order to achieve a task. Another popular vision of an agent based system is that of the blackboard architecture [Hayes-Roth88]. In this scheme a number of concurrently-executing agent processes interact via a global repository of information called a blackboard. Messages consisting of tasks, data and results are written and removed as the overall task of the system progresses.

In BBAI an agent is most often taken to be a physical individual that interacts with the physical world in some organised and structured way, with "interact" being the key aspect of the description. Smithers in [Smithers92] suggests that a useful categorisation is that of "...a coherent system of processes that reliably and robustly effects specific changes on its environment while

receiving any effects from its environment which may be (direct or indirect) consequences of its actions or may simply be the effects of [other] unrelated events...". This provides a notion that both agent and its world are of fundamental significance and that the processes of an agent (or more specifically a synthetic agent) are not only those that exist in its programming. The size and traction of a robot's wheels have as much effect on the robot's behaviour as any software control strategy: a robot with small wheels will be more susceptible to changes in direction due to small bumps, a robot with larger wheels may not be able to navigate through a gap between two close objects, a robot with one wheel smaller than the other will have a tendency to steer continuously to one side, and so on. It can be seen that these physical aspects of a synthetic agent have an important influence on its interaction with its world. In BBAI natural agents are also recognised as being more than a collection of controlling processes. As a final illustration of the nature of an agent as seen by the BBAI community it is interesting to refer to the work of [Beer94], [Cliff & Miller 95], [Cliff *et al* 93] and [Yamauchi and Beer 94] as well as that reported in [Smithers94]. These take a dynamical systems view of both the agent and its environment.

After these illustrations of some component characteristics of an agent, questions concerning the nature of its behaviour, or environment interaction, must be addressed. There is much discussion in the literature and increasingly in the press of "intelligent agents". For the purposes of this dissertation and for the reasons outlined above in section 1.1 I shall not be using this as a descriptive term or label. However, this leads to a potential problem which is interesting in passing but probably does not have any major significance in this thesis. A commonly-used characterisation of an agent is to further label it as an autonomous agent, with autonomy being used to mark the agent as being self-governing (to use the Chambers dictionary definition [Schwarz93]) in some way. In fact Smithers' characterisation of an agent given above usefully encompasses an autonomous agent. This typifies an assumption that the systems being studied in AI are different in some way from ordinary machines or automata (for example, a production line robot arm that repeats a set of motions continuously and blindly, often regardless of the fact that there may be a problem). An autonomous agent is taken to have some internal task achieving potential and an ability to work towards the completion of that task allowing for external influences that may either be beneficial or detrimental.

Assigning a label of autonomy to an agent presents a familiar problem. It is always possible to argue pedantically that a machine like a factory robot does indeed have its own task achieving potential, the task being to wave its arm around in a particular way, and that it is just a matter of scale as to the degree of autonomy such a system exhibits. Issues along these lines are discussed in [Rosen87], [McFarland94] and [McFarland95] (and in many other more philosophical works) from an ethological viewpoint. In McFarland's work an autonomous agent is differentiated from an automaton by the fact that we may be able to analyse the latter and consequently identify its component and functional parts. It is claimed that for a truly autonomous agent this is not

possible. However, this is a subjective view and does not make any gesture towards the possibility that at some time in the future we may have the techniques to analyse all manner of agents (natural or synthetic) in the same way that we now analyse automata. The issue regarding the contribution of the history of the agent's behaviour to its current state is interesting but does not change the fundamental conclusion: that autonomy is also (as well as intelligence) a characteristic dependent on the views of the observer. For the purposes of this dissertation it is perhaps useful to consider the characteristics of autonomy of an agent as falling in a space bounded by axes of complexity of agent system, agent task and agent environment (see figure 2.1). Within this space a complete automaton (for example, a clockwork mechanoid) may exist in the domain of near-zero complexity in all 3 dimensions while a car assembly robot may score highly on system complexity but low on task and environment complexity. The various areas of artificial intelligence (classical and BBAI) and control systems are shown in boxes in order to provoke some thought as to the nature of the systems dealt with in the respective fields.

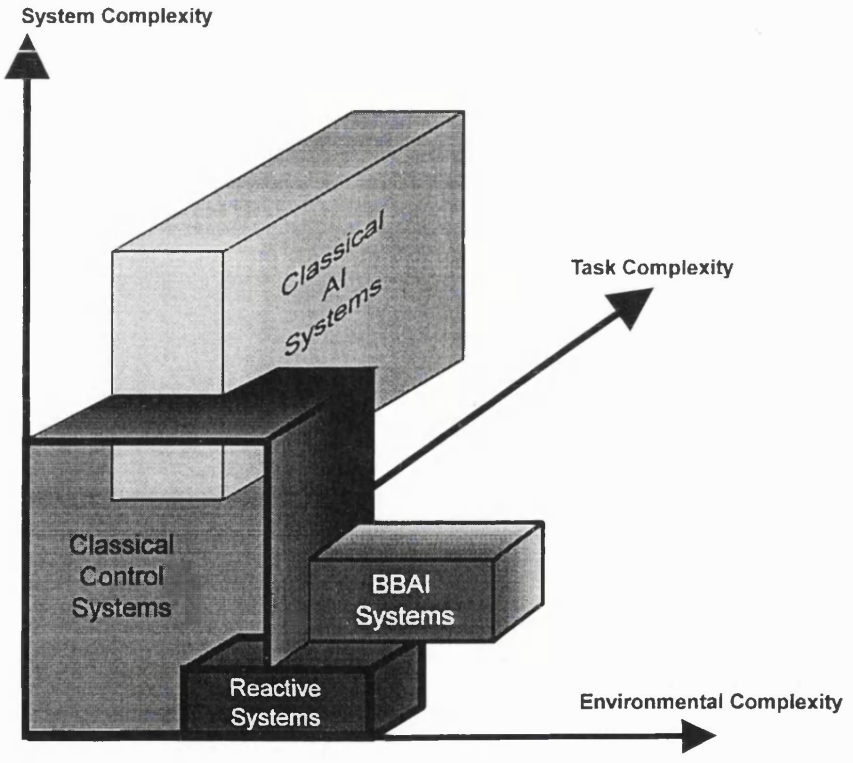


Figure 2.1. A three-dimensional space presenting a characterisation of different degrees of autonomy.

The final characteristic of being an agent that I wish to discuss here is that of self-sufficiency. At first glance this may appear to be similar (if not the same) to autonomy but, as discussed again by McFarland, self-sufficiency is generally taken to have energy or power supply connotations. A self-sufficient agent has an ability to care for itself, either through the process of metabolism as happens in natural agents or an ability to recharge batteries as might be found in the behaviour of a mobile robot such as those reported in [Steels94]. The significance of self-sufficiency to the

control of many complex synthetic systems is perhaps small. A system of software processes (software agents) does not necessarily encompass a need for aspects of self-sufficiency and so here again we are reduced to a subjective assessment of the state of play by the observer, especially when it could be claimed that a cow is not self-sufficient but dependent on the activities of the grass in the meadow for its survival (as well as the bacteria in its intestine).

For the purposes of this thesis then, an agent is seen as a specific collection of processes that are closely linked and that interact with a broader set of loosely-linked processes that constitute the agent's environment. Although in BBAI an agent is commonly seen as being physically embodied rather than simply another term for a concurrent software process, it will be argued in the course of this dissertation that an "agent" view of systems with complex interactions can be useful on a number of different levels. As the task and domain requirements for systems become more complex, and I refer here particularly to systems that interact with the real-world such as robots, the component parts of those systems also become more complicated. It is already the case that even the simple laboratory examples of mobile robots have multiple centres of computation, some specialised and some as a parallel means of increasing computational power ([Brooks89], [Smithers94], [Steels94] and [Vertommen95]). The view that a system might be constructed from numbers of specialised component parts that are capable of some form of self-regulation and self-adaptation is becoming fact. Issues surrounding the integration of these parts and of achieving system requirements through their interactions therefore make up a widening and very open research topic. It is possible that these systems will be viewed best as a community of heterogeneous asynchronously-interacting autonomous or semi-autonomous agents. This is taken further in chapters 5 and 6.

2.3. Different Approaches to Control

The work reported in this thesis necessitates some discussion on the nature of control, and in particular some of the assumptions of both classical control and classical AI methodologies. The perceived need for this has come about largely as a result of the increasing use of AI based techniques in systems control. Although these ideas are not a novel part of this thesis in themselves (much of the work in the AI community addresses control problems in very different ways from those of classical control engineers), it is an important aspect of the reported work, and an awareness of the state of play will set the scene for the rest of this document and the discussion on BBAI.

As a brief aside from this part of the introduction it is interesting to note that there is an increasing tendency within BBAI to move towards the use of an alternative methodology in the design of autonomous agent systems. Empirical results reported in many instances suggest that the whole notion of control in terms of both a separate controlling agent and an information based

design paradigm are too limited in their ability to provide a sufficient means of analysing, understanding and designing complex autonomous systems. Here again I refer to the work in [Beer94], [Cliff & Miller 95], [Cliff *et al* 93] and [Yamauchi and Beer 94] and the use of the techniques of dynamical systems theory. These recent ideas offer an approach to examining systems not in terms of information flow between component parts of a system but in terms of interactions between processes of a system. While at first this may seem to be saying the same thing in a different way, it is the case that the latter view allows a more complete picture of a system and its interactions with its environment to be built up. For example the factors affecting the behaviour of a mobile robot are not limited only to the signals emanating from sensors and the commands being sent to actuators. Other factors - for example, high-friction floors slowing down motors, hard walls causing colliding robots to bounce off violently in (for all intents and purposes) random directions, as well as more subtle effects such as the partial saturation of light-sensitive sensors by sunlight - can all make significant contributions to a robot's behaviour and are difficult, if not impossible, to classify in terms of specific and quantifiable information.

I continue with some standard views and definitions of control techniques.

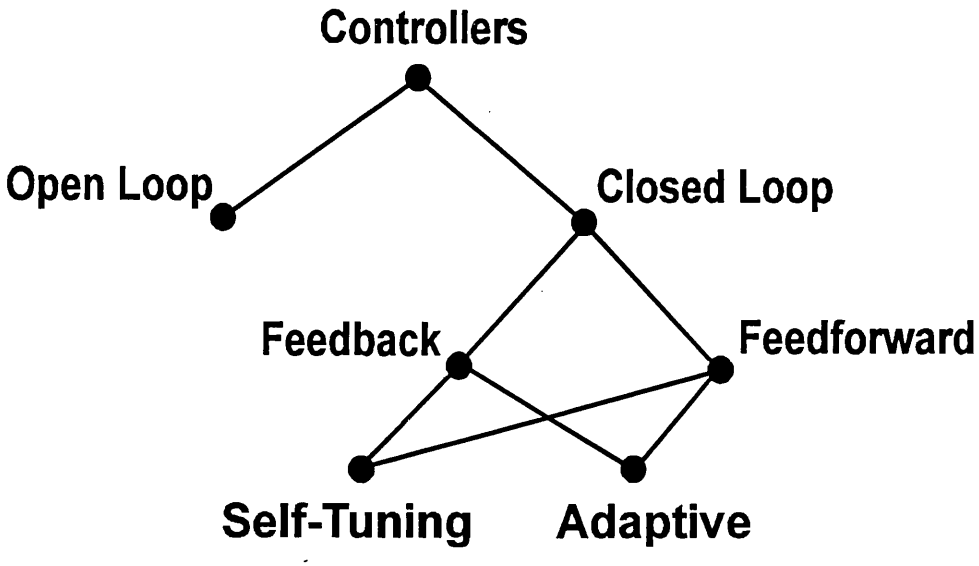


Figure 2.2. Categories of controller

A controller may be seen as a device that takes a number of inputs and generates an action output based on some form of internal model. These controllers fall into two main categories: i) Open-loop controllers and ii) Closed-loop controllers [Bissell94]. Open-loop controllers (figure 2.3a) are basically a single-shot command affair with the controller output being based on at least one input command or signal. It is generally the case that such controllers are unable to allow for varying environmental effects in the generation of their output. Such controllers are also referred to as reactive controllers. An example of such a reactive control action can be seen in the reflex of a person who unknowingly picks up a hot pan from a stove, expecting it to be cool enough to

hold (a robot controller for this behaviour is described in [El-Gamal *et al* 92]). The reflex motion of dropping the pan in response to the burning pain can be seen as an open-loop or reactive form of control activity. Closed-loop control on the other hand differs in that the action output is generated from a combination of current system state and a command or demand input signal.

Closed-loop controllers can be categorised further into two groups: i) feedback controllers and ii) feedforward controllers. Feedback control (Figure 2.3b) is the mainstay of the classical control community and is in fact the only type of control dealt with in any detail in typical texts such as [Dorf92] and [Bissell94]. In feedback control the system attempts to maintain a status-quo of input and output values by measuring directly and trying to minimise a difference between the desired and actual outputs. It is true to say that the feedback controller is currently the most widely used category of controller, being found in an almost endless list of applications from amplifiers to servo motors to chemical process controllers.

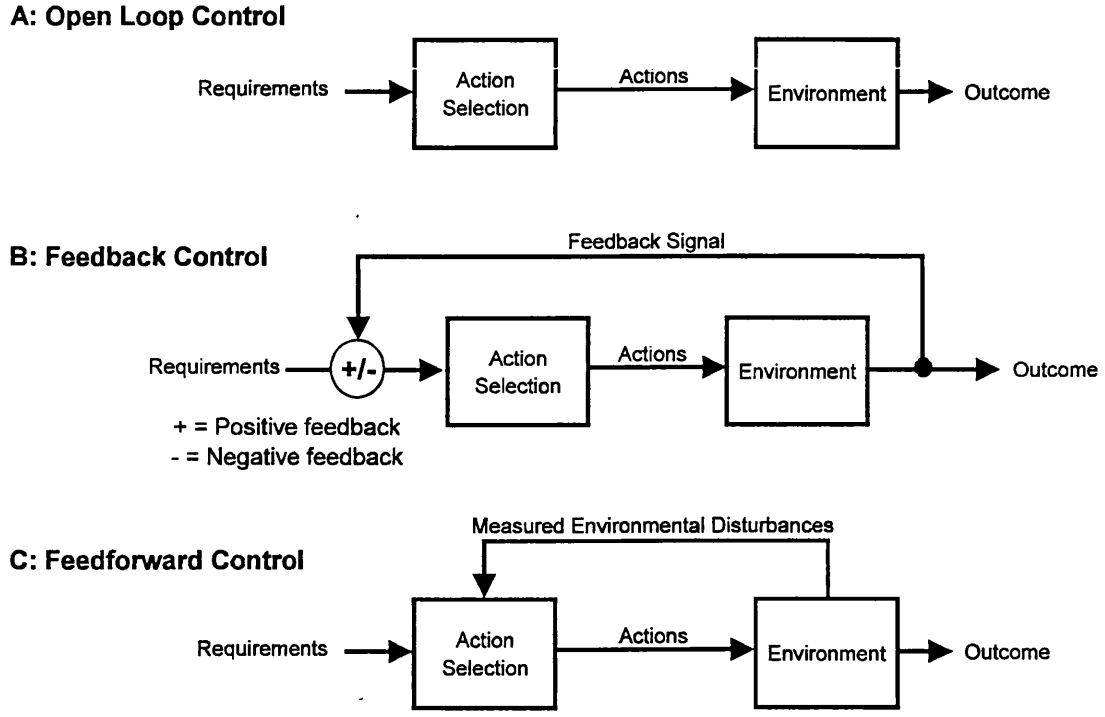


Figure 2.3. Block Diagrams of the three main categories of classical control systems.

Feedforward control (figure 2.3c) can perhaps be associated more readily with the control systems developed within the AI communities. In fact [Dorf92] hardly touches the subject and seems to suggest that feedforward control is little more than a form of open-loop control with output being generated without direct reference to the current situation. However, feedforward control (even in AI terms) can be seen to be closed-loop with the output of such controllers being based on a desired action signal and a number of other inputs that reflect the state of the system in some known way. In fact this form of control is often confusingly referred to within the AI

community as feedback due to the fact that the effect of the agent's actions can be fed back through the environment to sensory inputs which are in turn used to form the basis of the next action. However, this is not the same as the direct measurement of error used in classical feedback control [Smithers94b] and [Smithers95]. The differences between classical control and AI based control can perhaps be seen as little more than a difference in level of abstraction (in terms of the form and route of the closed-loop information flow). This is explored further in the following sections.

2.3.1. Models and Symbols

Models are used and regarded in different ways [Pebody93]. The term "model" may be applied to the mathematical function that is used to provide a classical control based system with its ability to generate output values from its inputs. "Model" may also be applied in a similar way in reference to the logic based functions of AI control systems, again utilised for the generation of output values from a number of input variables. The models in these instances may be applied to all types of control system mentioned above: open-loop, feedforward and feedback. The model is the transformation function used in the "black box" of the control system and is named as such due to the fact that it accommodates an input-output mapping that is able to match certain factors of the system's external world. It is usually the result of much design and development effort. The relevant general idea is that the controller is provided with an ability to predict the course of events in the world, thereby enabling a suitable output to be generated. This type of modelling is also used in the Psychology and Ethology communities to recreate and experiment with human and other animal behaviours (for example see [Rutkowska94] and [DeSchutter & Nuyts 93] for the respective fields).

In the AI field the term "model" is also (perhaps confusingly) used in a different way in order to refer to abstract knowledge held by a system. Traditional AI techniques often utilise a symbolic information structure referred to as a "symbolic world-model". In this case the model is a collection of tokens or symbols with an abstract meaning and is accessed by symbolic planning and reasoning systems which in turn generate a system-level output. This classical AI view of a model has been advocated since the earliest days of AI research and was firmly built into the methodologies of AI by Newell and Simon in the paper [Newell & Simon 76] that presented the physical symbol system hypothesis as an explanation and methodology of intelligent systems.

Here I have briefly identified a number of views on the nature of models. This discussion is taken further in chapter 3 where Behaviour Based AI is introduced. Issues concerning models and knowledge representation are revisited in chapter 7 during discussion on learning and adaptivity.

2.3.2. Continuous Time, Discrete Time and Linearity

Another important factor that differentiates the classical control approach and that of AI is the way in which the respective methodologies deal with time. The techniques of classical control utilise quantitative mathematical models and as such generally operate in a continuous time domain. In other words, these systems are able to provide a control response at all instants. Conversely AI, with its rule based models, treats time in discrete steps. At each time step a logical decision-making process is executed that takes into account the current state of the system and any other available information. Such systems have a strong relationship to the computational nature of digital computers which are inherently discrete in terms of time. This is in contrast to the older techniques of classical control which originated in systems that utilised analogue electronics (although now more complex control solutions are implemented on digital systems, bringing in some element of discrete timing).

The real-world is the everyday physical world in which people exist and interact along with all manner of other agents both natural and synthetic. Time in the real-world is continuous and the nature of the interactions within the world is often unpredictable and chaotic. These interactions are generally referred to as being non-linear [Bissell94]. A system is linear (and hence a suitable domain for classical control techniques) if and only if all the control coefficients for the system are constant values. The linearity of a system is defined in terms of the system excitation $x(t)$ and response $y(t)$. When a linear system at rest is subjected to an excitation $x_1(t)$, it provides a response $y_1(t)$. Further, when the system is subjected to an excitation $x_2(t)$, it must provide a corresponding response $y_2(t)$. For a linear system it is necessary that the excitation $x_1(t)+x_2(t)$ result in a response $y_1(t)+y_2(t)$. Additionally it is necessary that the magnitude scale factor is preserved in a linear system. For example, a system with an input x which results in an output y must have the relationship such that the response to a constant multiple factor K of the input x results in an output equal to Ky.

There is only a small subset of events and activities that are in any way linearly organised. This has a significant effect on the extent to which the continuous-time techniques of classical control theory can be used directly. For complex systems in the real-world the task of creating an accurate model that is able to provide a suitable output at all instants of time taking all factors into account is obviously impossible. Hence it is general practice to make so-called linear approximations when developing classical control models [Dorf92]. These approximations exploit generalities in system behaviour in order to reduce the number of variables required for a mathematical description. Thus classical control is also referred to as linear control, the main characterisation here being that the output response is linearly related to the input.

The set of logical rules that make up an AI based control system act in discrete time steps, as already mentioned above. Each rule is compared with the current system state and if a match is

found then the rule is used. This is a powerful way of dealing with non-linearity (and it can also deal with linearity). Rather than utilising a continuous-time model that has to provide an output for an infinite number of connected time divisions, the discretely accessed rules of an AI system can each be distinct and matched to different world situations. In this way the output of the controller can follow non-linearity in the world because it does not have to match world activities on a continuous-time basis. However, there are problems here as well. The discrete nature of the AI system leads to potential problems in resolution of the controller. The number of rules required to deal with any real-world situation is potentially infinite and so again approximations or compromises have to be made during the design of the rule based control model.

2.3.3. The Real-World, Real-Time and Event Synchronisation

Finally in this chapter on control I wish to dwell briefly on the issues relating to the real-world response time of an agent. In general we may say that a more successful agent is one that can respond more quickly (in an appropriate way) to changing situations in its world, so long as the response is a suitable one. However this is an abstract AI view which has its roots in behavioural psychology (discussed in [Boden94] for example). It is a different notion from that of response time used in classical control and which is also applicable and relevant to AI control. It is not the place of this introduction to provide extensive detail on the basics of classical control theory; for this, the reader is referred to any modern control theory text book, of which the aforementioned [Dorf92] and [Bissell94] are two of many. However the following provides an outline of some relevant factors.

As we have seen, a classical control system utilises a feedback loop in which the actual output of the controller is fed back to the input and compared with the desired output. This comparison is used to generate an error signal which in turn provides the input for the controller and so on ad infinitum. However there is a time delay involved in this process that is present in all systems. This time delay is the time interval between the start of an event at one point in a system and its resulting action at another point in the system. It may result from, among other sources, friction in any mechanical parts and propagation delays in electronic circuits. This delay can lead to problems with the stability of the system. If the delay is too long then the response of the controller will always arrive too late and a succession of increasing errors can lead to instability of the system. If the response of the target system is at the right speed, then the system will function properly, and in order to achieve this a damping factor is often applied to the system in order to slow it down.

Damping the response speed of a system so that the controller has time to react is achieved using various methods. For example, a physical device to provide friction to a moving mechanism may be used, such as the butterfly spinner that slows down the movement of a clockwork

mechanism. Alternatively a part of the controller's mathematical model may include a term to add damping artificially to the system. If this damping factor is set correctly the system will settle to its new state in an optimum and stable way. This is known as *critical damping* (figure 2.4a). If, however, the damping is too severe, then the system's response will be slowed (perhaps due to excess friction) resulting in at best a delayed settling time and at worst a complete failure for the desired output to be achieved. Imagine a lift full of people. In a normally-loaded situation it transports its passengers efficiently from floor to floor. However, should too many people get on board (assuming that the space in the lift is large enough for the extreme number of people), the electric motors in the roof will become overloaded and perhaps be unable to raise the lift, or only raise it slowly. This is an extreme example of an over-damped system (figure 2.4b) and the associated problem of undershoot.

The other extreme to an over-damped system is an under-damped one. In this case the output of the controller moves quickly and overshoots the desired target before the controller is able to make corrections. The eventual correction from the controller results in reversal of the original output which again moves quickly, overshooting the target, but perhaps to a lesser degree. The system may eventually settle, but only if no other disturbances influence the output. (figure 2.4c). A system that never settles suffers from what is known as the *hunting problem*. The effect of this on a lift full of people can be left to the reader's imagination (guided by figure 2.4).

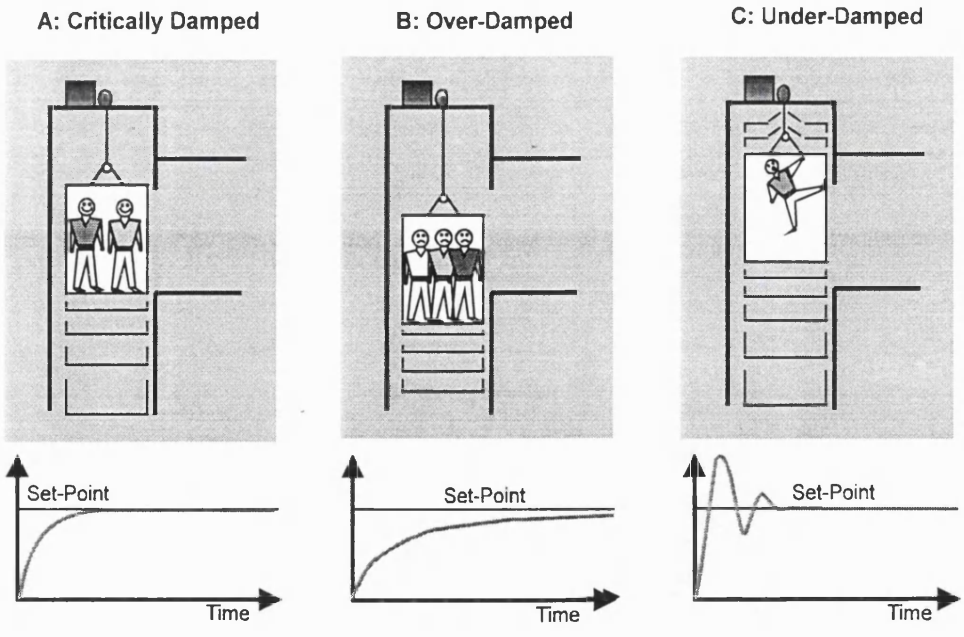


Figure2.4. Examples of a: a critically damped response, b: an over-damped response, and c) an under-damped response.

The problems associated with the timing of classical control systems are also found in AI control systems although not in quite the same way. Because AI controllers are working in

discrete time steps and at each step the output is not necessarily a linear response to the current world state, the nature of oscillation is different. In general the timing of AI based real-world agents is loosely determined by external events, for example a mobile robot travelling in a straight line detecting an obstacle and steering to avoid it. It is the event of encountering the obstacle that triggers the avoidance behaviour. As long as the robot is not travelling too fast, it has time to avoid the object. If it is too fast, or computationally too slow, then the chances are that it will collide before selecting an appropriate action. These timing issues are often referred to in the AI community as real-time issues and as such are not directly related to factors concerning internal computer clocks or other time sources internal to the robot. They concern world events that are outside the control of the agent but to which the agent must be able to respond within some critical time constraint. Thus the agent's timing is dictated largely by external events, and it may be said that its controller must be able to synchronise its activities with those events. In computer science and electronic engineering the notion of "real-time" is similar but most often the critical event is generated artificially for system synchronisation purposes. Nevertheless, such systems still have to complete the relevant computation prior to the event in question.

2.3.4. Adaptive and Non-Linear Control

The techniques of classical control theory, although based predominantly on that of the abovementioned feedback control, may be found in many more advanced forms. In parallel with the development of digital computers, advanced digital control techniques have been developed to deal with non-linearities in system behaviour. Both [Dorf92] and [Bissell94] deal with these aspects in their respective concluding chapters.

In its simplest form a digital controller converts continuous-valued sensor inputs to digital format through analogue to digital converter devices. Classical feedback loops implemented in software then generate a digital response that is converted to analogue form via digital to analogue converters. If the time interval between successive digital signals is small compared to the time constants of the plant or actuator, then the whole control system essentially acts as a continuous system. More advanced examples of digital controllers demonstrate a form of adaptive control. In order to deal with the problems of non-linearity in the real world, software has been developed to tune the parameters of the feedback control system. In this way the controller may adapt to different environmental situations. An example is presented in [Bissell94] in the from of the adaptive control of an aircraft. Here the control parameters are changed in real-time as the aircraft changes height, thereby adapting to the dynamics of flying at different altitudes. Similarly self-tuning process controllers are designed to adjust their own settings in response to changing plant conditions. This is commonly done by the self-tuning process injecting a small change into the system and monitoring the response. [Berghuis *et al* 93] presents an example of these techniques in the form of an adaptive robot controller while [Chen & Ni 93] use

dynamic calibration in a laser radar scanning system. These ideas are discussed further in part 2 of this thesis.

2.4. Summary

This chapter started by discussing aspects of intelligence and how it is virtually impossible to provide a suitable all-encompassing definition. The conclusion here was that it was generally not an important issue in any case since the real heart of the matter boils down to the measured performance of a system in its designated domain. The next background topic in section 2 dealt with aspects of autonomy and self-sufficiency. Again ideas were aired, but it was concluded that the assignment of such attributes to a system were largely conjectural and generally of little practical use. These topics were discussed in order to introduce these concepts for the BBAI introduction in the next chapter.

Section 3 of this chapter presented a number of aspects of classical control and contrasted them within the conceptuality of AI control. Differences between the basic assumptions of AI control and classical control were highlighted in order to prepare the reader for the general theme of this thesis: the low-level use of AI control techniques from a behaviour based perspective. Chapter 3 now proceeds to introduce the main characteristics and working assumptions of behaviour based artificial intelligence.

3. Behaviour Based Artificial Intelligence

This chapter covers the following:

BBAI terminology.

Illustrations of BBAI control architectures.

Designing and building systems that interact with the real-world.

In the last chapter some fundamental characteristics differentiating between classical control and AI control were identified and discussed. The potential power of AI based control techniques in dealing with non-linear problem domains has stimulated extensive research in many areas of system development and in many applications. However, the task of building a system of rules in the manner of classical AI systems has not furnished demonstrable success in the provision of control structures for real-world autonomous agents ([Brooks91], [Simmons94] and [Miller93]). As a response to this a body of work has emerged that focuses on a different approach to building and researching AI systems. This chapter introduces the relevant paradigm: Behaviour Based Control, a methodology resulting from the field of Behaviour Based Artificial Intelligence (BBAI).

Behaviour Based Artificial Intelligence is not a specific or well defined field of research. Rather it is more of a collection of diverse topics loosely focused on an interest in autonomous and self-sufficient systems that are embedded in a particular environment. While the name "BBAI" is gaining strength and recognition for a field, it is the case that the community shows a considerable diversity of interests ranging from the modelling and synthesis of real biological systems to more engineering-oriented robotic work. A common aspect of the work is in the use of one or more synthetic agents as a basic tool for experimentation. This approach has come to the fore recently in a number of international conferences such as the those of Artificial Life (both Artificial Life in North America and the European Conference on Artificial Life) and the "Animals to Animats: Simulation of Adaptive Behaviour" conference series. The work is also becoming of import in a number of other conferences such as the IEEE International Conference on Intelligent Robots and Systems. On another front, psychologists and ethologists are interested in demonstrating and verifying models of natural behaviour in synthetic systems, and neuroscientists have likewise used the medium to experiment with models of nervous systems. For example, the work reported in [Arreguit & Vittoz 94] deals with the construction of an artificial retina and [Osorio *et al* 94] look at a combination of sensory mechanisms in terms of their

neurological function. Robotics engineers and AI researchers are also interested in using the methodology to experiment with new ideas and techniques.

The first part of this chapter introduces the reader to some general terminology that is popularly used within the general community of BBAI. This is followed by a number of examples of behaviour based control frameworks that have been developed. Finally the chapter discusses the issues surrounding the "real system" versus "simulated domain" debate which is of particular significance to the BBAI field in general and on the work presented later in this thesis.

3.1. Some Working Terminology

This section overviews some of the common terminology found in the BBAI literature. It discusses the issues in preparation for the rest of the thesis: behaviour based, bottom-up development, distributed architectures, non-symbolic activation patterns, interaction dynamics, situatedness, embodiment, groundedness and emergence.

3.1.1. Behaviour Based - as opposed to function based

A fundamental aspect of BBAI is the behavioural decomposition of a system. This can be witnessed in both the research inspired by natural systems, for example [Mataric91], [McFarland94b], [Cruse91], [Webb95] and [Arkin90], and in the design and development of synthetic systems of which [Gat *et al* 94], [Ferrell93] and [Arkin90b] are examples. [Webb & Smithers 91] discusses the connection between the two. Other similar work is evidenced in [Edelman *et al* 92]. Whilst a behavioural decomposition of animal behaviour may seem to be an obvious way of analysing a natural phenomenon, it has not been evident in the case of autonomous systems design. The past history of AI, with complete fields dedicated to the isolated research of different functional aspects such as vision [Marr82], speech [Chomsky65] and actuation [Critchlow85] all in isolation, illustrates the point.

The approach of BBAI systems design is to break development down to the implementation of task achieving behavioural modules, for example a homing behaviour of a mobile robot towards a source of light. These behavioural modules are built to include all necessary mechanisms, in terms of both physical hardware and algorithmic computation, that are required to realise the particular task in hand. The result is a fast and dedicated process (or collection of processes) capable of complete control of the agent in a particular behavioural situation. Problems of design turn on the provision of a mechanism that can provide for the selection of the right behaviour at the right time, or in other words identify a particular agent-environment situation that can trigger a change in behavioural pattern. This problem, known as action selection, has been dealt with in a variety of ways. Examples are the explicit but embedded action-selection dynamics in the spreading activation networks reported in [Maes89] and [Pebody91] and the more implicit and implementation-specific techniques reported in, amongst others, [Steels93] and [Arkin87]. The

issue of implicit or explicit action-selection mechanisms is an ongoing topic or research and debate in the BBAI community.

3.1.2. Bottom-Up

Figure 3.1 compares a classical AI system decomposition with that of a behaviour based system. It is taken from the Brooks 1986 paper. The behavioural modules can be seen to be assembled in a bottom-up fashion in contrast to the classical AI structure which is typically the result of a top-down development. A behaviour based system results from a requirement to implement low-level tasks in order to provide a framework for higher modules that necessarily build on lower-level capabilities. Each layer is implemented and tested independently of any higher level of control. Although higher layers may interact and overrule the activities of lower levels, in the manner of the overriding of reflexes, the lower levels are completely self-sufficient within their own behavioural domain. This introduces a powerful redundancy into these systems in the form of a low-level reflex that may continue to function despite the absence of higher-level commands or instructions.

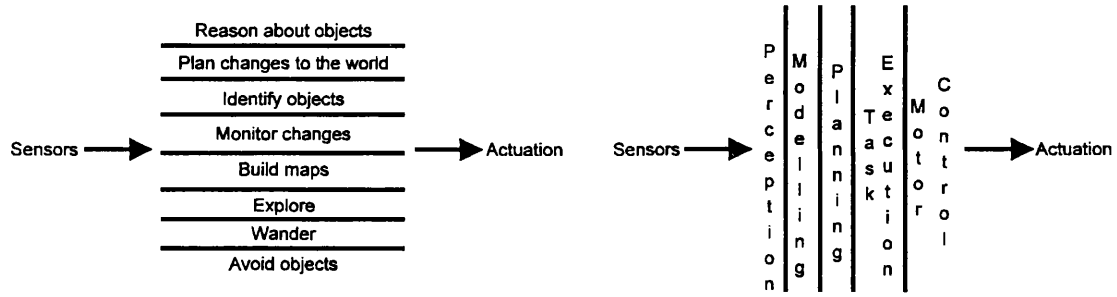


Figure 3.1. Comparing a classical AI control structure (right) with one that is behaviour based (left). Taken from [Brooks 86].

3.1.3. Distributed - Decentralised

It should be apparent by now, given both the bottom-up development and behavioural module oriented construction of behaviour based systems, that these systems have an inherently distributed structure. The development of classical AI systems has necessarily led to the pipeline architecture that is shown in figure 3.1a which separates the sensory and classification functionality of the input stages and the planning and reasoning "higher-level" functions of the central modules. The actuation and output control functions are then dependent on all that has gone before. Each of these modules with its general-purpose functionality requires central storage of information. In contrast the behaviour based system localises all aspects relevant to a particular behaviour, sensing and acting, into collections of simple, fast, tightly-running parallel-control loop processes. These concurrent (or more often pseudoconcurrent) processes operate independently, responding to local conditions which may or may not include sensory information

from other parts of the system. The result is a network of interacting control processes that simply act on a given set of inputs which may be sourced either from physical sensors or from other internal processes.

This approach to distribution of control is typified in the subsumption control work reported in [Ferrell93] (or [Ferrell94]) and the control work of [Pfeifer & Verschure 91], and extends to physical distribution of systems in the form of cooperating robot groups, for example as in [Parker93], [Mein91], [Steels94], [Beckers *et al* 94] and [Mataric95]. This is markedly different from the work of, for example [Hayes-Roth88] and [Yuta & Premvuti 92] and the industrial views of distribution reported in [Jones *et al* 91]. Work of a distributed nature in an industrial environment is reported in [Freund & Buxbaum 93] and [Oliveira *et al* 91].

3.1.4. Non-Symbolic - Activation Patterns

Classical AI based systems rely on the formal manipulation of symbols for the realisation of their actions. As such they follow a set sense-think-act cycle using some framework of formal logic. This is a result of the extensive AI research following, amongst others, the hypothesis of the Physical Symbol system of Newell and Simon in [Newell & Simon 76]. In these systems the nature and form of a symbol are clear. It is an abstract token that is used to represent some aspect of the agent's world, and the relationship and interactions of a symbol to other symbols is supposed to match the relationship of the real-world artefacts that the symbols represent. An example of this kind of symbolic relationship can be found in [Eastman61], in which the simple relationships are expressed in terms of basic natural-language statements. Issues of symbol acquisition are discussed in [Harnad90] and [Malcolm *et al* 89].

In contrast to the hard use of symbols in classical AI, BBAI does not use any direct form of symbolic manipulation in its agent control mechanisms. A behaviour based system is assembled using a multitude of techniques most often organised around the distributed execution of simple algorithmic processes that interact so as to provide the required behaviour. [Mataric92] details an example of a distributed and non-symbolic map acquisition in a mobile robot. In any eventuality, the processes of a behaviour based agent are non-symbolic in their operation. While it may be claimed that the patterns of activation that trigger particular behaviours might be interpreted as some form of symbolic representation [VanGelder94], it is the case that these are artefacts after the event, a result of an observer's interpretation of one instance of a complex and dynamically changing system. Further discussion along these lines is likely to become deeply philosophical and it is not the place of the present thesis to dwell on this topic. Further (and inconclusive) discussion can be found in [Smithers94c] while [Steels90] and [Steels95] report on a continuing refinement of a behaviour based approach.

3.1.5. Dynamics of Interaction - No World Models

The distributed, non-symbolic characteristic of a behaviour based system has already been outlined above. However, there is another factor that differentiates between aspects of internal state and representation: the issue of world models. This requires separate attention. The issues concerning different uses of the term "model" were dealt with in the previous chapter. In this instance, "model" is taken in its AI sense as an abstract assemblage of knowledge. Such knowledgebases as found in systems reported in, for example [Simmons94] and [Knick & Schlegel 94], have been used to provide a central world model for an autonomous agent in the form shown on the right-hand side of figure 3.1. Such knowledge structures are used as the bases for supplying planning and reasoning mechanisms with symbolic information for calculating the next action of an agent. In classical AI the timing of such processes is not important. For example, in a chess-playing system the timing of any output result is not an important performance factor. However, a mobile robot that trundles over a cliff because it is still searching its world model for a suitable next action is in serious trouble.

Behaviour based systems however, due to their inherent parallel and distributed architecture and non-symbolic nature, do not require and can not realistically support such a centralised structure as a world model. The maintenance of such a body of internally stored states is both time-consuming and costly in terms of the computational resources required, resources that have been demonstrated to be utilised better in speeding up the response times of behaviour based modules [Gat93], [Miller94]. Thus another characteristic of behaviour based systems is that they do not employ any formal internal world model and in fact are designed to exploit a minimum of explicit internal domain knowledge, basing their actions instead on the wealth of information directly available from sensing and acting in the real-world [Brooks90], [Brooks91]. Examples that have addressed these issues and provided demonstrations of control in real-world robots include [Mataric92b] and [Nehmzow & Smithers 90].

3.1.6. Situatedness

The target of traditional AI has generally been to research and develop systems that demonstrate intelligence independently of their environment. This is typified by the classical test for intelligence suggested in [Turing50] in which all a system has to do to establish its intelligence is to respond to a person's questions in a similar way to another person. I say "all a system has to do" because in terms of BBAI such a successful system would not even begin to meet conditions such as the reliable and robust achievement of tasks in the real physical world. Other examples of the independence of traditional AI systems from their environments can be found in the many examples of abstract problem solving such as the Blocks World programs of Winograd reported in [Critchlow85], [Winston77] and [Luger & Stubblefield 89]. In contrast discussion of these views from a BBAI perspective can be found in [Brooks86b], [Prem95] and [Webb93].

BBAI studies and implements situated agents. In other words, the agents have behavioural characteristics that are highly dependent on their environments. In fact it is the case that the agent and environment of BBAI systems are so tightly coupled that it is difficult to examine one aspect individually. This follows closely the observations of researchers in aspects of biology, in particular ethology, that animals (or in other words natural agents) have evolved to function within particular environmental niches. For example, a spider has evolved to function in its niche which perhaps consists of a hedgerow. This spider is able to spin a web to catch food, reproduce (with the help of a friend) and generally survive in all manner of diverse conditions, wind, rain, snow, frost etc. This is certainly an impressive range of behaviour even if we are not willing to subscribe to it as being intelligent. However, if the spider is moved through some unfortunate circumstance into a bathtub, it becomes completely lost. A typical response is for it to keep walking forwards and upwards in an effort to escape, but with the inevitable consequence of falling back down the slippery sides to the bottom - this behaviour does not seem so impressive. The difference is in the spider's situation (figure 3.2); it was not "designed" for the bathtub environment [Smithers91].

In BBAI a synthetic agent is also viewed in this way, with all aspects of its behaviour and activities including its environmental situation taken into account. Recognition of the phenomenon of situated behaviour can contribute in the design of specific solutions to many control problems. Aspects of an agent's environment can be used to advantage in developing more efficient and cost-effective synthetic systems with the problems of finding general-purpose solutions being conveniently sidestepped [Horswill93]. Consequently aspects of maintenance and search of symbolic world model structures are irrelevant since the agent is specifically designed to function as a result of its environmental situation. This is in contrast to the techniques used in, for example, [Fikes & Nilsson 71] and [Simmons94] where aspects such as motion are planned from a database-like structure of the vehicle's perception of its world. In the words of Brooks in [Brooks90], "the world is its own best model".

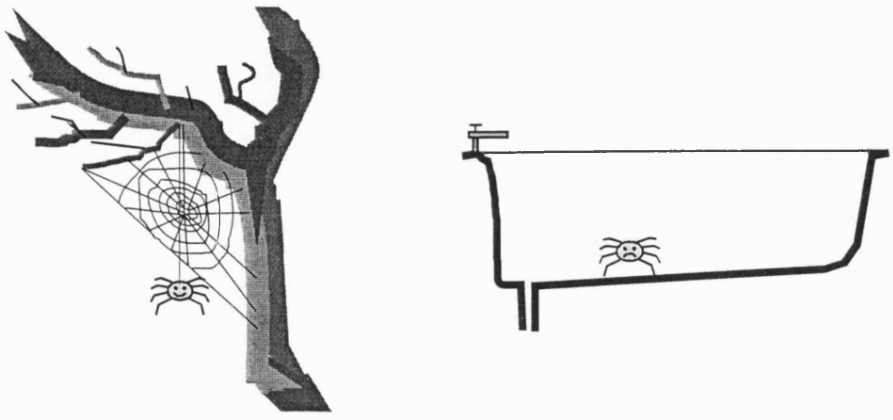


Figure 3.2. Left: A spider situated in a hedge. Right: A spider situated in a bath.

3.1.7. Embodiment

A behaviour based agent is situated in its environment. In general the environment of interest is that of the real-world. Thus an agent must be physically embodied in some way. Most behaviour based systems focus on the implementation of experiments with mobile robots. However, other agent-environment situations exist that are otherwise physically embodied, and the work reported in this thesis deals with one such.

The physical embodiment of an agent is realised in its morphology and has as significant an effect on its behaviour as any internal control structure. In the case of mobile robots the physical embodiment is obviously in the form of the shape, size, materials and construction of the vehicle. Factors such as wheel size and type characteristics can have a fundamental effect on agent performance. For example, a big robot over (say) 1.5m in width will not be able to negotiate most normal office doorways while a small robot may become entangled in ground clutter such as wires and cables. Alternatively a robot with small wheels that require less power to turn will become stuck more often on bumps and ridges in contrast to larger wheels. As an extreme example a vehicle with one drive wheel smaller than the other will have an inherent tendency to turn constantly in one direction unless the motors are driven at different speeds. There are countless other more subtle characteristics of agent morphology that are equally significant. For example, how does the vehicle respond to colliding with obstacles (which as any mobile robot designer knows will inevitably happen); is it massive and hence relatively unaffected or is it light with a tendency to bounce back? If it is heavy then the motors may stall causing a severe drain on the batteries, while if it is light the recoils will mean that any form of proprioception though wheel movement is ineffective. The use of various sensor types is also equally influential on agent ability. This is described in [Smithers95] as the details of "getting the interaction dynamics right".

3.1.8. Groundedness

One of the major problems faced by the supporters of classical AI is that of grounding an abstract symbolic process in the real-world. In other words, the problem is to make the symbols used in such systems match the aspects of the real-world that they are supposed to represent. This is evident from the massive amounts of work conducted in the centres of research in machine vision (typified by [Marr82]) that attempt to process image data to separate, recognise and label objects. For example, identifying a physical chair that is actually a chair and not a similar-looking object, which may or may not be used for sitting on, and then equating it with a symbol, for example "*h*", which may be manipulated logically in conjunction with other symbols and have the same relationships as the real-world artefact. For example, "*h* on _" where "_" is a symbol for floor.

The classical problem of symbol grounding [Harnad90] is not an issue in BBAI because: i) symbols are not used explicitly in any part of the system, and ii) BBAI systems, being situated in

a particular environmental niche [McFarland95], are intricately tied to their worlds with actions being a result of the real-world situation and not some form of internal world-model.

3.1.9. Emergence

Finally in this section the issue of emergence is touched upon. In the original work on emergence in [Brooks86b] (as well as [Pfeifer93]) it is suggested that ill-defined features such as (in particular) "intelligence" but also a host of other folk-psychological notions such as "desires", "emotions" and "goals" could not be useful in defining or constructing an autonomous agent control system and would in fact not be directly observable in any localised component part. Rather, these phenomena would manifest themselves as a result of the interactions of the agent's behavioural modules, the agent's physical body and its environment. Thus, they would emerge as the agent went about its activities in its world. This relies totally on the observer of such an agent-environment system being able to interpret such notions in the resulting behaviour. [Pfiefer88] and [Churchland95] are also relevant here.

Emergence has also attained popularity, perhaps mistakenly, as a term used to describe the manifestation of a behaviour as some form of holistic effect resulting from the interactions of the many parallel parts of a behaviour based system. In other words: getting something for free. This is illustrated by the discussions in [Maes89], [Simmons94] and [Crowley *et al* 94]. This however is a highly dubious attribution. A required behaviour does not usually emerge spontaneously as a result of the interaction between parallel processes, but rather it manifests itself after much careful design and development [Mataric92], [Brooks89] and [Steels89]. Emergent behaviour that is acquired for free more often results in a poorly and even catastrophically performing system rather than in one that is "better" than before. It is possible or even likely that, in the future, emergent behaviour will become a significant phenomenon in the area of autonomous systems, but in the real-world where systems must perform to a predetermined specification it would be undesirable to rely on such properties appearing unexpectedly out of the blue. Our working definition of emergence, then, is the manifestation of folk-psychological characteristics which we may (as observers) attribute to a system and that are not directly designed in.

3.2. Examples of Behaviour Based Control Architectures

Many design frameworks, control strategies, methodologies, architectures and other structures that come under many other collective names have been proposed as solutions to the problem of controlling an autonomous agent in a real-world environment. In the literature different researchers provide different levels and qualities of detail about their pet programming frameworks or design paradigms. It may well turn out that the systems are interchangeable and that the basic principle of bottom-up design is the most significant aspect of a system's success. It is certainly the case that several of these frameworks are interchangeable. For example, the Subsumption Architecture and its accompanying Behaviour Language may be used to implement

a Spreading Activation Network like that of [Maes90] which had previously been programmed in LISP ([Brooks90b] and [Maes & Brooks 90]). Other techniques such as neural nets and genetic algorithms are also of interest as they may provide useful solutions to some problems, for example the interpretation of large arrays of sensory data. The rest of this section outlines in greater detail three frameworks that have emerged from the field of BBAI and appear here in order to provide a flavour of the methods that are used: (i) Spreading Activation Networks; (ii) Process Description Language PDL; (iii) the Subsumption Architecture. Other long-lived and successfully-applied control architectures include: the Autonomous Robot Architecture or AuRa [Arkin87],[Arkin90] and [Arkin90b], and "A Language For Action: ALFA" [Gat *et al* 94].

3.2.1. Spreading Activation Networks

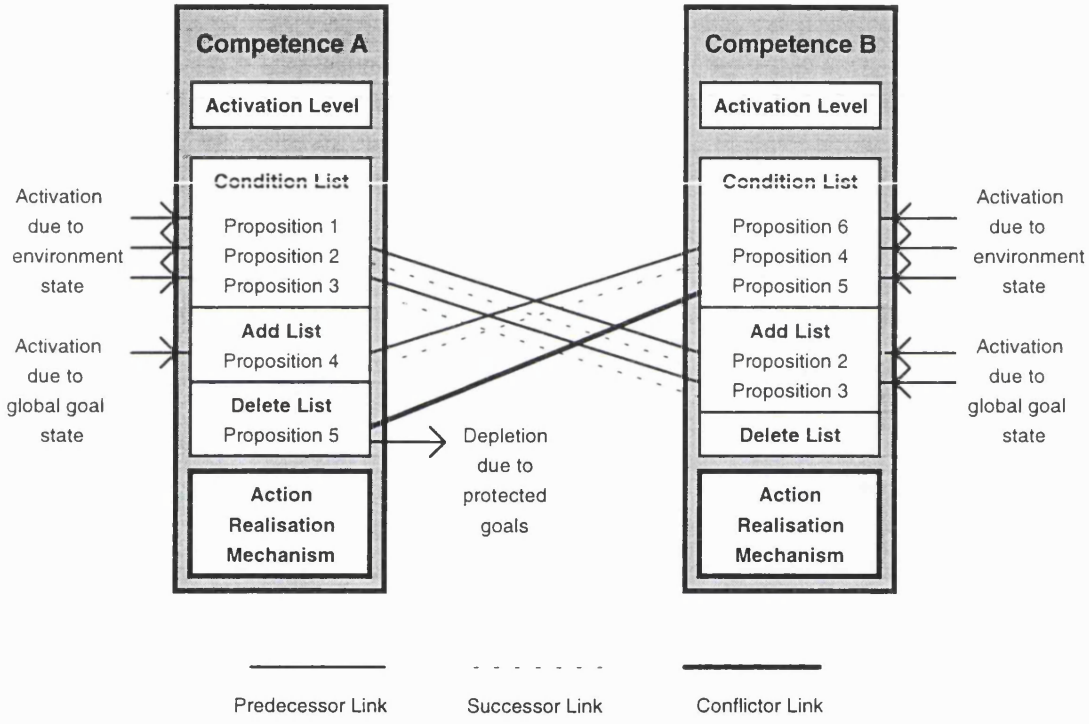


Figure 3.3. Two competence modules with example interconnections and activation energy inputs and outputs.

There are several examples of agent control frameworks that rely on a mechanism known as a spreading activation network. Perhaps the most interesting of these, as far as behaviour based control is concerned, is the action-selection mechanism that was developed and documented in [Maes89], [Maes89b], [Maes91], [Maes91b], [Maes91c], and implemented in a mobile robot [Pebody 91], in a simulated domain [Tyrrell94], and used in [Giszter94] to model the reflex behaviours of frogs. The mechanism provides for a dynamic action-selection which is based on the current state of the environment, the agent's current actions and its previous activity. Other examples of this type of mechanism have been somewhat more specialised. For example, [Mataric92] presents a spreading activation mechanism that was used to store landmarks and

provide for subsequent navigation in a mobile robot. [Parker92] implements a spreading activation network in a similar manner to that of Maes to control a number of cooperating agents but which has yet to be run on real-world robots. The main part of this section will describe the action-selection mechanism as implemented by Maes as a means of demonstrating the technique.

Maes' spreading activation action-selection network is a truly distributed control mechanism that allows for a dynamic selection and sequencing of agent actions. The system does not specifically *plan* its activity but an appearance of planned actions emerges from the interaction between the component parts. The system consists of a number of prewired/preprogrammed interconnected competence modules. The connections are links along which an activation energy flows, the quantity of which depends on the dynamical interaction among all modules. Each competence module consists of the following elements:

Competence Realisation Mechanism: The mechanism of this element is not prespecified; it may be any mechanism that executes the desired competence or completes the required task (for example, a piece of hardwired electronics or a simple piece of program code). This element is enabled when the competence is activated (see below).

Activation Level: This is a value (or register) that contains the competence module's current level of activation energy and indicates the module's applicability to the current situation in which the agent finds itself.

Precondition List: This is a list of logical (true or false) states that may refer to sensors or internal states of the agent. A competence module can only become activated if all of its precondition states are true.

Add List: This is a list of conditions that are turned true should the particular competence module succeed in its task execution.

Delete List: This list contains conditions that are falsified should the competence module successfully complete its task.

There are three types of interconnections between the competence modules: Predecessor, Successor and Conflictor. A quantity called activation energy flows along these connections, either increasing or decreasing the likelihood of a competence module becoming activated. A number of parameters determine the interaction behaviour of the modules by influencing the amount of energy transferred along the connections. These are:

Activation Threshold: This value provides the initial threshold activation level that must be surpassed by executable competence modules. It is decreased by 10% every time a competence module is not selected and reset to its original value once one is.

Environment Activation: This value effects the activation energy injected due to the state of the agent's environment.

Goal Activation: Activation injected due to the agent's goals is controlled by this value.

Protected Goal Depletion: This value affects the removal of activation energy from modules that undo completed goals.

There is a sequence of operations that each module executes in synchronisation with all other modules:

i) The module accumulates the amount of activation that it receives from the interconnections within the network.

ii) The module calculates the amount of activation that it outputs on its links for the next cycle.

iii) If all the module's preconditions are met and its activation level is greater than a global threshold and greater than all other module's activation levels, then its action realisation mechanism is enabled.

iv) A global decay value is applied to all module's activation levels.

The above has very briefly outlined the mechanism of Maes' spreading activation network, many aspects of which originated from the STRIPS planner reported in [Fikes & Nilsson 71]. The important point to note is the fact that the mechanism has no central control; each module assesses its own suitability to function within the current state of the world. The resulting behaviour is an ordered sequence of activities that give the agent an emergent behaviour, which is one that cannot be found by examining any individual competence module (although if one examined the complete network it might be a different case, seemingly less emergent).

Although Maes, [Pebody91] and [Giszter94] all report successful implementations of the spreading activation algorithm it should be noted that the extensive work reported in [Tyrrell94] suggests that there may be some problems. These particularly focus around the nature of selecting the values of the various configuration parameters (which control the inter-module flow of activation energy) and the stability of the interactions between the network's competence modules. However, in addition to this it is worth noting that several extensions to the algorithm have been developed so that network interconnections and parameter values can be learned dynamically and thus adapt continuously to changes in circumstances (e.g. a competence module failure) [Maes & Brooks 90], [Maes91] and [Giszter94].

3.2.2. Process Networks And The Process Description Language: PDL

The ideas of a dynamic network of interacting processes and of the PDL have been developed by Steels and others at the AI Laboratory in the Flemish Free University of Brussels ([Steels92], [Steels93] and [Steels95]). It has been used to program a number of mobile robots to execute tasks starting with obstacle avoidance and wandering to homing behaviours in which the robot searches for light sources within a normal office environment. Work is continuing and more

sophisticated behaviours are being experimented with such as battery recharging, landmark recognition and map making. It is intended to add these to the basic obstacle avoidance and navigation skills already developed [Steels94].

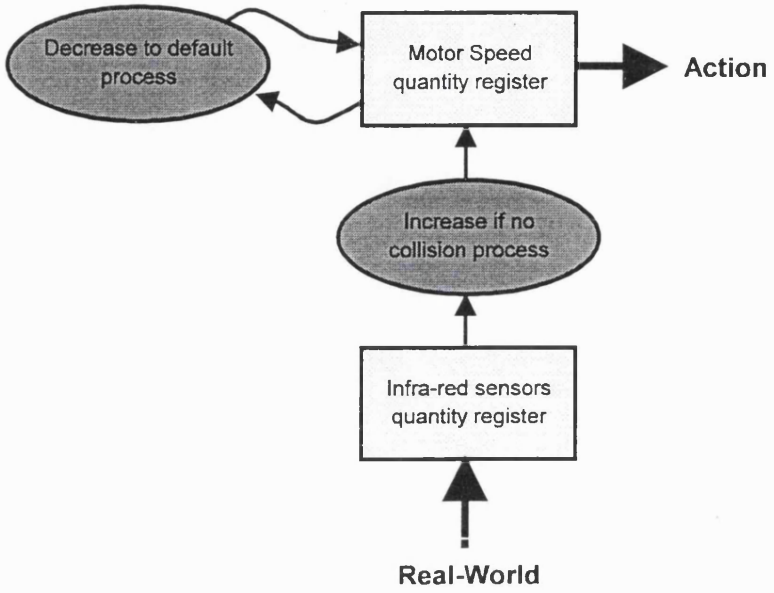


Figure 3.4. A simple process network without internal quantities.

PDL provides a framework of processes that interact by modifying values called quantities. A quantity refers to a sensor value, an actuator's value or some internal value. A PDL process receives input from a preprogrammed set of quantities and outputs a value that weights a quantity. The fact that a quantity may or may not be tied directly to a real sensor or actuator is not significant to the functionality of the process. The main point with this network is that all quantities are related, without any necessity for categorisation or discretisation of their states, although there may still be discrete thresholds which may be subject to other adaptive processes. All processes are thought of as operating in parallel, although the overall scheme may be implemented on a single microcomputer with a time slicing mechanism.

In order to reduce the complexity of the analysis and design task, further structure has been added in the form of a number of recurrent patterns of process/quantity combinations. More detail is to be found in [Steels93].

3.2.3. The Subsumption Architecture

In 1986 Brooks published a paper entitled "A Robust Layered Control System For A Mobile Robot" [Brooks86]. The paper was one of the first reports of the now prolific stream of work resulting from the new approach to AI (other publications by Brooks that expand on this work include [Brooks89] and [Brooks90b]). It presented the Subsumption Architecture and an example of a real-world demonstration using the described techniques to control the movement and navigation of a relatively simple mobile robot. The behaviour of the robot was reported to be

reliable and robust and achieved cheaply with a minimum of computational resources. This was compared to attempts at using traditional AI techniques to do similar tasks (e.g. the robot Shakey, developed at the Stanford University AI Laboratory [Nilsson84]). The comparison is still an impressive demonstration of the power and success of these newer methods.

The Subsumption Architecture is a bottom-up behaviour-oriented implementation of a layered network of semi-synchronous augmented finite state machine processes. It provides a mechanism that enables the real-time control structures of mobile robots to be built up in layers starting with very basic reactive response and reflexes. Further layers of the system can be added without the need to modify already-operational lower ones. One of the main characteristics of the system, one that typifies that of all BBAI research, is the nature of the interaction of the layers with sensors and actuators and thus with the real-world. Rather than using a functional approach which leads to a sequential pipe-line mechanism of sensors → sensor-processing → map-building → action-planning → action-realisation → actuators (or something similar) the Subsumption Architecture results in a massively parallel mechanism with modules that have access to virtually any sensory or internal information. The redundancy that results enhances the system's performance by making it more robust and more able to respond quickly (as a reflex) to dangerous situations.

The mechanism consists of a number of finite state machines that have been augmented with real-time clock information, hence: Augmented Finite State Machines or AFSMs. The AFSMs are interconnected via single-element buffers and "wires". Information is not stored in the buffer, but only the value of the most recently-received wire signal. Consequently it is up to the AFSM to respond in some suitable way as messages can be lost if new ones arrive before the old ones have been used. The inputs and outputs of a particular AFSM may be inhibited or suppressed by a wire output of another AFSM. Inhibition is applied to a module's output and prevents this output from reaching its destination. Suppression affects the input to an AFSM. Its effect is similar to that of an inhibition but additionally the suppressing AFSM forces a new value onto the wire. Each AFSM also has a reset default input wire. Figure 3.5 below presents a single AFSM.

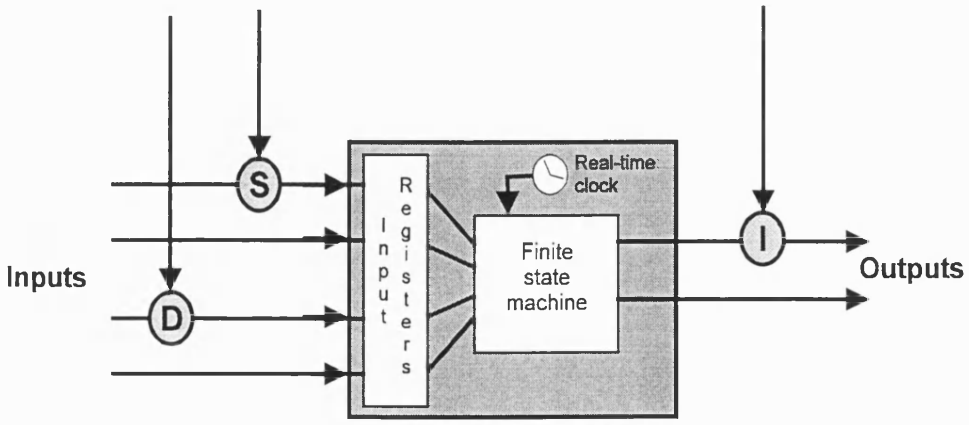


Figure 3.5. An augmented finite state machine and its interconnections: S: Suppressor; D: Default; I: Inhibitor.

Each AFSM acts on an asynchronous basis; there is no form of built-in handshaking operation (although this mechanism may be programmed in by the designer should this be desired). The system is, however, loosely synchronised by a characteristic time constant which determines the real-time processing rate of the AFSM. In other words the clock ticks of the AFSM real-time clock dictate the frequency at which the input registers are read and processed. This has an important effect. In normal software (software that does not have a similar real-time construct) the system slows down as the program becomes larger and more complex. The speed of execution is effectively determined by the size of the program. In stand-alone utility-type software (e.g. data-bases, spreadsheets, word processors etc.) the only effect that this has is to raise the user annoyance factor. However, in systems that interact directly with the real-world and that must respond to events as a matter of survival, this slowing down can lead to mistiming of actions. The characteristic time factor avoids this by ensuring that every AFSM's processing rate remains the same despite any later additions to the software loading on the processor - although inevitably the processor will eventually become overloaded and thus begin to slow everything down.

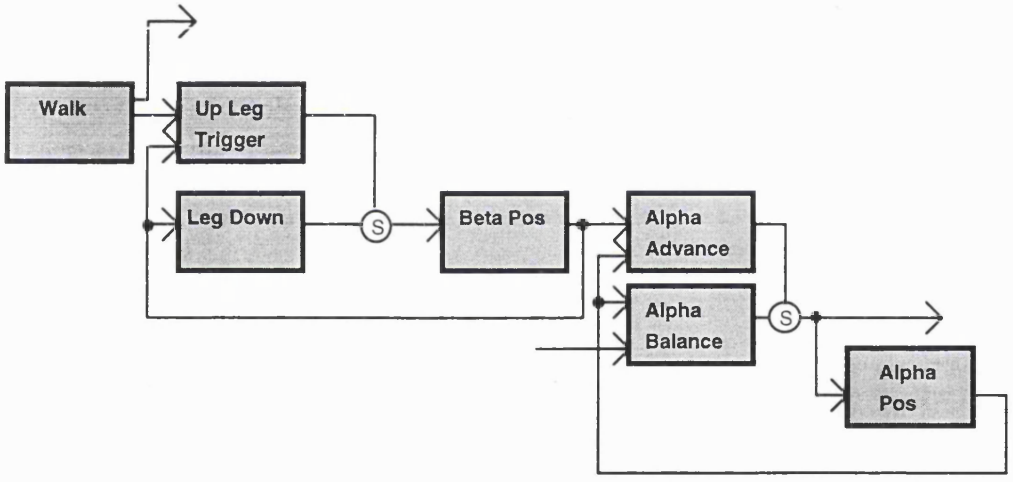


Figure 3.6. An example of a subsumption control structure, used for the low-level control of a walking robot's legs (taken from [Brooks89]).

A single layer of a system will consists of several AFSMs and, as higher layers are added the lower layers become totally embedded in the system. The higher layers influence the lower layers by inhibition and suppression and also by their effect on the actuators and hence the vehicle's situation in the world. Figure 3.6 shows part of the control structure for the low-level control of the 12 degrees of freedom of the 6-legged walking robot Genghis in [Brooks89]. This work was later expanded to deal robustly with the 18 degrees of freedom of the Hannibal robot ([Ferrel93], [Ferrel94]). These control solutions provided robots that were able to navigate in complex environments and respond to various stimuli with changes in behaviour. The resulting behaviour of the robots was reliable and robust, with Hannibal being able to compensate for missing and damaged limbs. Another significant factor contributing to the success of these systems was the relatively small computational expense (typically a few 16-bit microprocessors) of the solutions

offered. More detail on the programming of agents using the Subsumption Architecture and details of robots that have been built can be found in [Brooks89b], [Brooks90] and [Brooks91] as well as other robotic work emanating from the MIT AI laboratory ([Parker93] and [Mataric91]). Brooks' work is continuing with the development of the Cog robot as outlined in [Brooks & Stein 93]. Examples of other work using the subsumption architecture can be found in [Porter92], [Webber & Bisset94], [Mahadevan & Connell 92], [Gat *et al* 94], [Pinhanez95], [Cornell *et al* 94] and [Zorpette94] to name but a few.

3.3. Designing and Building Real-World Interaction Systems: Real-World Control or Interaction?

The real-world does not consist of discrete events, it does not contain repeatable situations, but just similar ones, and it is stochastic and chaotic. The methodology of BBAI illustrated by the agent design architectures outlined above attempts to deal with the problem of providing an agent with a sufficient behavioural repertoire to allow it to survive and carry out tasks in a maximally reliable and robust way. In order to achieve this, the agent and environment are considered together as a single agent-environment system. Development and adaptation to any part of the system may be considered should it be of benefit to the overall performance. Three factors result from this approach: i) that the agent can not be said to control its activities in the classical control sense of the word; ii) classical AI techniques do not usefully fit a continuous time agent-environment situation, and; iii) the importance of the environment is reflected in an increasing awareness of the necessity for experimentation with real-world embodied systems.

Control and Interaction

From a BBAI perspective the control of an agent is performed by a part of the system that responds to influences from both internal and external effects. It is often possible to identify this as being mostly made up of the software part of a behaviour based mobile robot, and "controller" is in fact the popular name for this part of an agent. However, the functionality of this "controller" is not really separable from the rest of the agent in the same way as is the more classical view of a controller. While in classical control terms the controller is seen as being a discrete part of a system, in the BBAI paradigm this differentiation is not possible since the boundary of any "controller" is blurred and all parts of an agent-environment system are recognised as being of more or less equal significance. For example, is the controller bounded by that which is software? Is it bounded by the sensors and actuators? and, if so, at what point? At analogue to digital signal conversion? At the physical boundaries of the sensor? what if the sensor is reliant on some form of actuation by the agent in order for it to work (for example, active sensors such as sonar)? Finally the notion that an agent controls its environment is, in the case of a mobile robot, patently untrue. If anything the reverse is more often true, with the agent doing its utmost to maintain some semblance of status quo. When classical control is used, it is normally in

extremely restricted domains, and in particular the domain has to have some semblance of linearity. An example of a classical control approach to autonomous systems development may be found in the Cybernetics literature, much of which dates from the 1960s and before. For example, see [Wiener48] and [Pratt87]. A more recent approach along these lines can be found in [Albus81].

Classical AI and Real-World Time

The mechanisms of classical AI tend not to allow for the temporal constraints on a real-world situated system, and provide generally centralised solutions to a problem. Although many formal systems have been developed with logics specially designed to cope with temporal reasoning [Long89], these deal with time in an abstract sense rather than being grounded in the real-world of sensory information flow. An agent that experiences a continuous sensory input is immersed in the sequencing of environmental events. It is the rate of change in the environment that dictates the passage of time for the agent. An agent must be able to respond to these events in a timely manner if it is to have any chance of survival. Consequently the distributed and localised fast control loops of a behaviour based system are much better suited to generating a timely response. This is in contrast to the homunculus approach of a classical AI system which, with its pipeline architecture and formal symbolic manipulation, requires either vast computational resources to support rapid decision-making or vast amounts of time to come to a decision. This trade off between computation and time is often a no-win solution to the problem of real-world, real-time interaction.

Simulation Versus the Real-World

The importance of an agent's environment and morphology as factors contributing equally to an agent's behaviour is regarded as a fundamental aspect of BBAI. The agent is uniquely embodied and situated. Consequently experimentation must take environmental concerns as seriously as those of the agent in question. This has resulted in the use of two types of research tool: simulated worlds and real-world robots (most often small mobile robots). These each have a number of good and bad points. Real robots are complex, require significant resources to develop and maintain, and are often claimed to be too difficult to set up to do "interesting" experiments. However, these problems are all associated with the real task in hand: that of developing and researching real-world autonomous agents. Such problems should ideally be dealt with first in the typical bottom-up development path of BBAI. Lessons learned from this route are all valuable contributions to autonomous systems research. In contrast computational simulations, which are often run with fancy graphical representations of the worlds and agents that they are intended to represent, are relatively easy to set up and maintain in the first instance, being relatively cheap in terms of cost and time resources. Prototyping of agent control tactics is comparatively fast and trouble-free. For example, one does not have to remember to make sure that the batteries are charged before conducting an experiment. However, the problem with the simulated world is

fundamental; it is not the real-world. In short, it is very difficult to provide a simulated model of the real-world that includes more than the most rudimentary of physical interactions, and any model that is provided is only as detailed as the designer has allowed. In fact it may be claimed that simulated solutions are not valid as solutions to anything other than the simplistic dynamical problem that is simulated. Issues concerning simulations versus the real-world are discussed in [Webb90], [Brooks90], [Jakobi *et al* 95][Smithers94] and are generally tied to the embodiment and situatedness factors of BBAI.

Work that has involved the experimentation of agent control mechanisms in simulation and the subsequent transfer to the real-world includes that reported in [Harvey *et al* 94] and [Nolfi *et al* 94]. Both of these examples involve the use of genetic algorithms ([Goldberg89]) and require many iterations of a build and test cycle that can only be realistically attempted in the fast prototyping domain offered by that of computer simulation. In the case of Harvey's work the evolved mechanisms are periodically transferred to real-world robotic systems and tested before further evolution takes place, while Nolfi's group's approach is to evolve control solutions that are able to adapt to their environments and hence deal with the transfer from simulation to real-world.

3.4. Conclusions and Connections with the Project Work

From the discussion in the first part of this chapter it is clear that the ideas and methods of classical AI do not fit well into the behaviour based paradigm. This has already been observed to be the case in practice when it comes to physical real-world implementations (for example, [Simmons94] and [Nilsson84]). However, there are a number of domains in which the application of classical AI techniques may prove to be of significant benefit. For example and perhaps most notably, in providing a high-level designer and operator environment for communications with a lower-level behaviour based system. This concept has been expanded on in the form of a "hybrid" approach to autonomous systems design, the thought being that higher-level symbol based systems may be grounded in the activities of the lower behaviour based layers. In [Jaeger95] it is suggested that the strengths and weaknesses of the symbolic AI and dynamic BBAI approaches are more or less complementary. Examples in this area of integration include the robotic assembly work of [Malcolm95] in which an AI planning system is situated on top of behavioural modules that implement the various actions specified by the top level planner. Other work with this system level of hybridisation includes that of [Cooper *et al* 95] and [Downs *et al* 95] while [Ma *et al* 95] reports on the integration of artificial neural networks and knowledge based systems.

Another aspect of hybrid systems is one that utilises the various AI techniques on a subsystem level with the granularity of the mix being finer that that mentioned above. For example [Holgate & Clarke 95] integrate artificial neural networks and state machines in a distributed learning

system and [Gaussier & Zrehen 94] utilise a similar arrangement for perception and action flows in mobile robots. This form of hybrid system is the main focus of the second part of this thesis starting in chapter 7.

The desirability of such hybrid solutions depends on the nature of the target system with each situation having to be accounted for on an individual basis, in keeping with the current state of the art in the pure behaviour based approach. A BBAI purist might argue that bolting together two fundamentally different approaches to systems design is in fact bad engineering practice. Interfaces, it might be claimed, would be ill-defined and arbitrary and it is in any case not clear that the phenomena of "reasoning" and "planning" are any more than emergent characteristics of a complex agent-environment system (albeit of a special nature) and arrived at through an observer's introspection. However on a systems engineering level there are clear advantages in utilising proven components that have been developed tried and tested elsewhere.

To date, much of the significant behaviour based work has dealt with the implementation of relatively simple mobile robots that most often utilise binary sensor and actuator control, although more complex analogue sensor and actuator systems are now increasingly being utilised [Ferrell93]. For example, compare material in [Meyer *et al* 92] and [Cliff *et al* 94]. However, after 10 years of concerted effort it would seem that some move towards larger and more complex systems is needed to demonstrate that the techniques are useful and can be scaled up to larger problem domains in the form of more complex environments and task requirements. This, however, necessitates input from several diverse fields ranging from mechanical and materials engineering to computer science, as well as useful input from areas such as neuroscience and ethology. This is difficult to achieve in all but the largest of multi-person projects, one such being the Cog project that is currently in the engineering phase at the MIT AI laboratory [Brooks94].

The two ideas presented in this thesis regarding the distributed and localised autonomy of agent architectures and the necessity for embedded adaptivity and learning were outlined in the first chapter. These follow on from this perceived need for research into the implementation of larger and more complex behaviour based systems that are situated in the real-world (as opposed to simulations). While the work reported in this thesis has been undertaken as an individual project, it has been approached with a broader view. Existing systems have been taken from industry that provide a "pre-engineered" test-bed for the implementation of experiments in this expansive vein. Because of time constraints the issues relating to the development of the behavioural control of a subsystem of the main equipment set was undertaken but with (at all times) an eye for the larger scheme of a truly distributed real-world control system. A real-world test-bed framework for experimentation is the topic of the next chapter.

4. An Industrially Situated Behaviour Based Control Test-Bed

This chapter covers the following:

An industrial laser-scanning inspection system.

An experimental test-bed.

The behaviour language - a framework for constructing subsumption control structures.

Industry is continually searching for solutions to the problem of automatic control of increasingly complex and diverse applications. The reasoning for such expansion ranges from cost reduction through greater efficiency and an increasing profusion of low-cost sensor devices (leading to large availability of information) to providing actuation in domains too dangerous for manual activities [Rokey & Grenander 90]. This increase in automation has led to an increase in the number of interacting self-regulating machine systems or work cells, which may be regarded as distributed systems or colonies of heterogeneous agents [Jones *et al* 91]. It is also the case that the individual systems are increasingly utilising distributed control techniques with localised processing of, for example, sensory information. These systems may also be seen in the light of a colony of heterogeneous agents. In both cases the agents are at least to some extent physically embodied. They are also situated in particular environments with specific task requirements. But, being physically distributed, global data structures such as those reported in [Hayes-Roth88] and/or global control (for example, [Fraichard & Laugier 93] and [Simmons94]) are not efficiently implemented. The domain would appear to be most suited to the utilisation of behaviour based control techniques.

Currently the vast majority of interesting work in behaviour based artificial intelligence (BBAI) has focused on the control of small mobile robots within a laboratory setting. It is also the case that while many of these mobile robots utilise more than one processing resource and one can claim them to be distributed systems, few if any actually implement any form of behaviour based control in a truly distributed manner. A more common scenario is the use of a central processing unit running a number of pseudo-concurrent processes which send commands to one or more slave processors that provide a built-in and fixed actuator and sensor control functionality. The work reported in this chapter addresses two aspects of behaviour based control: i) an unusual agent-environment task domain is used for the implementation of ii) a truly distributed behaviour based control architecture. The behaviour based framework used is that of

the Subsumption architecture and the task domain is the control of an active laser-scanning sensor used for industrial product inspection.

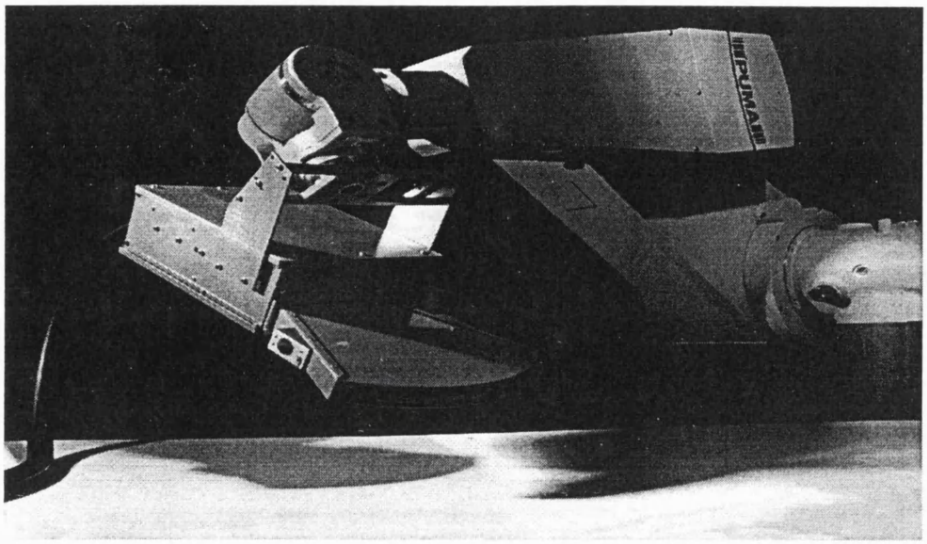


Figure 4.1. A laser-scanning inspection head mounted on a robot arm.

This chapter details a test-bed, oriented towards industrial applications, that was constructed to support experimentation with aspects of behaviour based control. The application of a technique in a new situation allows the strengths and weaknesses to be tested and highlighted within a different framework. The work reported here evaluates the use of behaviour based control in just such a situation and in the course of this thesis a number of consequent issues will be addressed. This chapter introduces the problem domain of an industrial laser-scanning surface inspection system and details a subsumption architecture programming framework that was used for all the experimental work presented in this dissertation. This test problem is, for the current state of the BBAI field, a novel domain and as such provides new perspective in which to evaluate behaviour based control.

4.1. Industrial Inspection: An Active Laser Scanner System

The laser-scanning system that forms the focus of the experimental work in this dissertation is used to inspect materials ranging from continuous lines of float glass and plastics to the finish on painted surfaces. Figure 4.1 shows a robot-arm-mounted device that has been used for inspecting the paint quality on car bodies while figure 4.2 shows a static mounted system above a continuous and moving product. This equipment is typically used in an environment that is noisy, often unstructured, dirty and demanding on hardware. In these respects it presents a set of requirements similar to those addressed by the real-world systems of BBAI. The system presented is known as FastScan and is one of a series of product inspection systems marketed by Image Automation Ltd, a member of the Sira group of companies. Image Automation are currently directing their efforts towards providing the glass and plastics industries with real-time, on-line product inspection and automatic grading. The FastScan inspection systems are under a

continued program of development, with a continued evaluation of other potential product inspection applications. The work of this thesis is directed towards a search for increasingly robust and adaptive control strategies that will enable the laser scanner system to operate in more exotic and unstructured industrial situations.

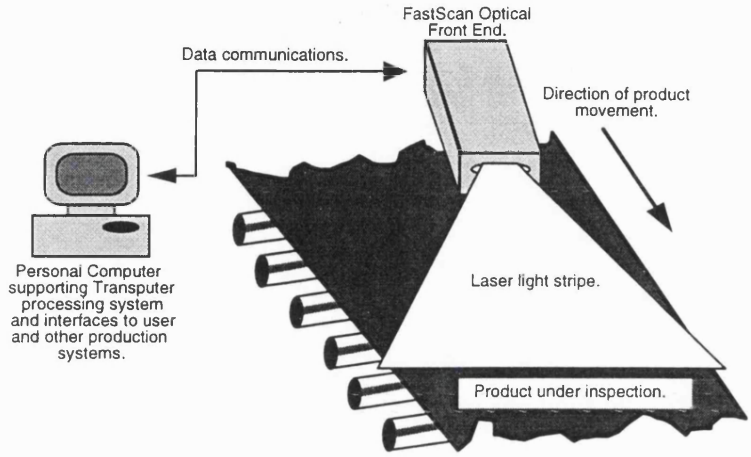


Figure 4.2. A single headed FastScan laser-scanning inspection system.

The engineering and construction of an inspection system is, in many respects, a process of developing a tailor-made system designed for a specific and unique industrial situation. It is in effect a highly specialised and complex sensor. Potential customers for product inspection systems come from different industries with different inspection requirements (even for a similar product such as, for example, sheet glass). Obviously, producing a new inspection system for every new customer would be prohibitively expensive. Image Automation has therefore taken a modular approach. The FastScan system is arranged around a number of basic component parts which can be configured for different industrial production scenarios. The test-bed used for the experimental work of this thesis is a subset of the main system. The subset was chosen as a target for experimentation because it provided a compact and well-defined problem that was both relevant to the development of the FastScan system as a whole and also served to capture a number of key aspects relevant to the field of BBAI research. Moreover it was also of a size and complexity that suited the time and cost resources of the project that is reported in this thesis.

4.1.1. The FastScan Product Inspection System

The FastScan system inspects a product in real time, at the same speed as the inspected product is produced and travels past the FastScan sensors. During production, information concerning any defects or faults is used to control, sort and grade the inspected product automatically. In this respect the laser scanner is an integrated part of a number of plant production systems, the first level of agency mentioned in chapter 1. In turn the FastScan system consists of a number of parts: (i) a host computer, usually an IBM compatible personal computer; (ii) scan analysis and defect classification electronics; (iii) laser and light collector control electronics and; (iv) the laser and light collection assembly. Figure 4.3 shows a block diagram of these parts and their

interconnections. The test-bed system detailed here has been constructed around the third and fourth elements which together make up the FastScan front-end system.

FastScan operates by striping the laser rapidly across the target product surface at a rate of approximately 2000 Hertz (a period of 500μS). This is done by shining a red helium-neon laser onto a 12-sided reflective polygon rotating at approximately 10,000 revolutions per minute. Laser light is then reflected back from the product surface, captured by specialised optical mechanisms and focused onto a number of photomultiplier tubes which convert the light into an analogue electrical signal. This is the task of the laser and light collection assembly in (iv) above.

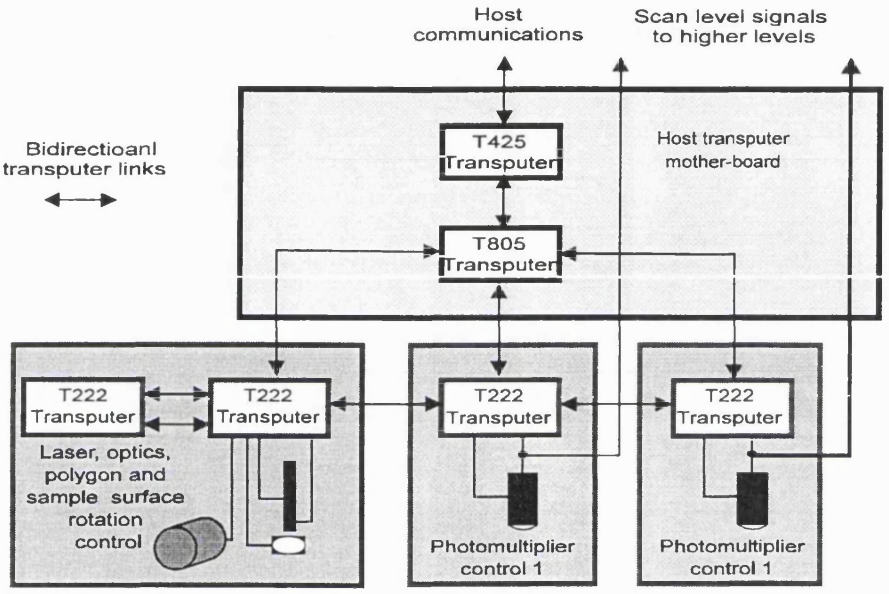


Figure 4.3. Functional units of the laser-scanning inspection system.

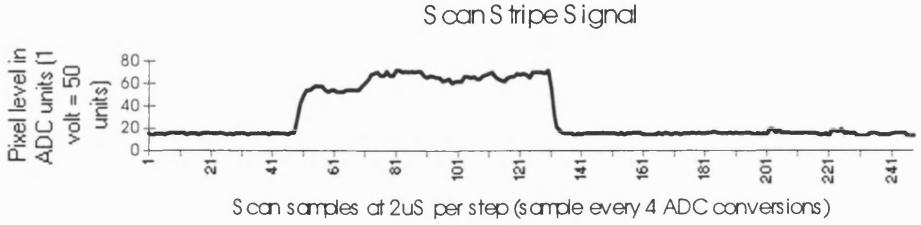


Figure 4.4. A single laser scan line signal showing an ideal signal pedestal.

As the laser stripes across the product surface, faults and defects manifest themselves as positive and negative spikes in the detected light level signal. Some defects (such as holes) prevent any return of light and others focus it directly back into the system's optics (a characteristic of certain scratch defects). Several highly sensitive photomultiplier channels are used to capture different types of defect through differing optical systems which are particularly designed for certain characteristics and situations. Features such as dust, dirt, scratches and holes with size scales down to around 500μm are typically the object of attention. Figure 4.4 shows a

representative scan across a product surface which correlates with the raised central part of the scan. This is known as the scan pedestal. It is the task of the laser and collector electronics to control the various systems including the photomultiplier devices so that this signal format is maintained in as many situations as possible (given the generally inhospitable industrial environment mentioned above) in order for the scan signal to be analysed and any detected defects classified. As the product passes under the laser scan, a defect map of consecutive scan stripe data is built up. Different defects can be recognised and used to trigger required actions in other work cells and machinery that constitutes the overall production line.

4.1.2. An Experimental Test-Bed

As mentioned above, the experimental test-bed that has been the focus of the experimental work in this thesis was built around the laser and light collection components and their respective control requirements. This section details the hardware functionality and operation of the test-bed system. The actual control problem is dealt with in section 4.1.4. In particular, the reasons for treating this part of the system in isolation are highlighted. The experimental test-bed consisted of five main modules: (i) a host computer; (ii) optics and laser control electronics; (iii) channel 1 photomultiplier and control electronics; (iv) channel 2 photomultiplier and control electronics; (v) a laser and light collector "optical front-end". Additionally, and not a part in the same sense as the first five there was (vi) a unit for presentation of test product surface. All of these are shown in Figure 4.5.

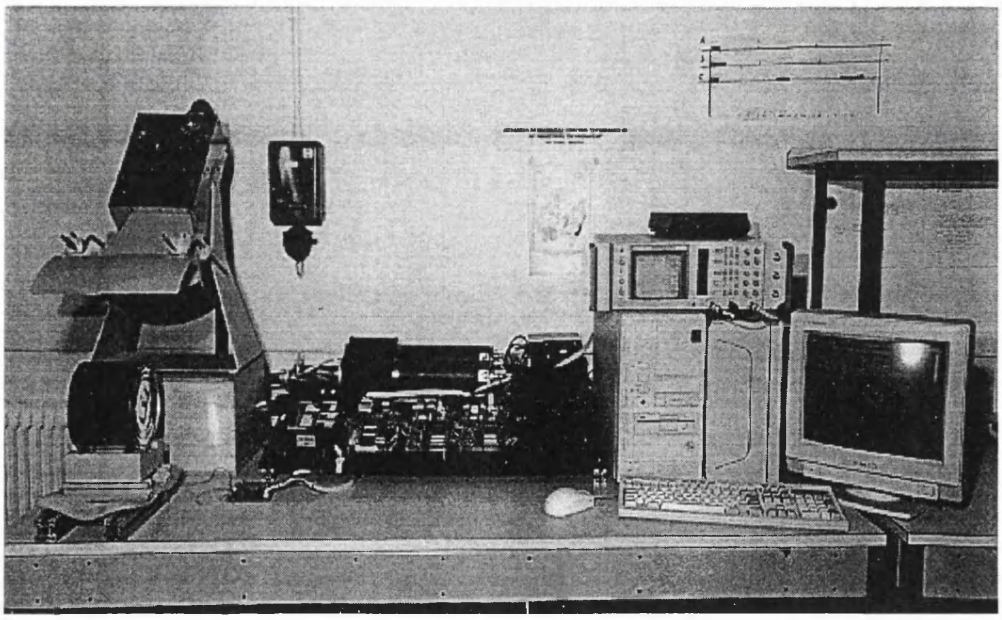


Figure 4.5. The two-channel laser scanner test-bed equipment. On the right is the host computer, in the centre the control electronics and on the left the system for the presentation of the test product surface.

Host Computer

An IBM compatible personal computer was used to support a number of top-level transputer processors on an internal expansion card. The PC also provided for program development, run-time status monitoring, data sampling and subsequent off-line analysis work.

Photomultiplier control hardware

Each of the two photomultiplier channels was controlled by software running on a dedicated transputer microprocessor. This aspect of the system is somewhat complex and provides the main experimental interest. It is discussed further in section 4.1.4.

Optics and laser control hardware

A single transputer based controller is used to monitor laser power and rotating polygon performance. This module also provides for the detection of a binary *start_of_scan* signal used throughout the system. For the purposes of the test-bed system, this controller also provides output to a variable-speed motor that drives the product sample past the laser.

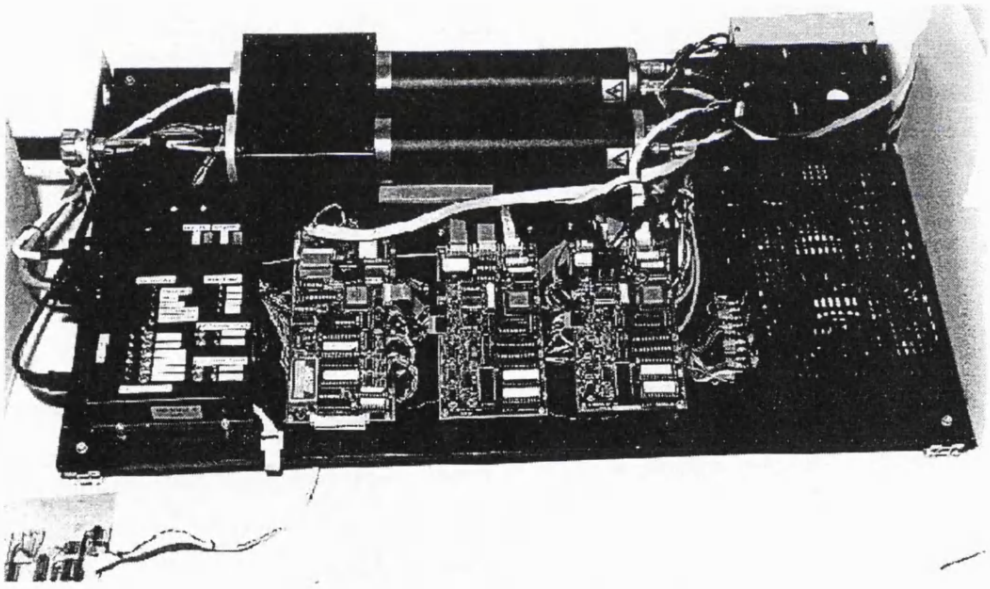


Figure 4.6. Test-bed control electronics.

4.1.3. Test Product

In an industrial situation the FastScan system is mounted so as to have a suitable field of view of the product that is to be inspected. In most cases the product passes under the optical front-end which illuminates the surface with the laser and collects the returned light into the photomultipliers. In the case of the test-bed system there was no continuous-flow production line, and an alternative had to be provided in the form of a continuously rotating drum and a sample inspection surface, as in figure 4.7. The rotation mechanism was in part belt-driven which effectively added a degree of irregularity to the timing of the drum's rotation. This useful feature

meant that any control structure under test would not be able to rely on specific timing of any repeating test-surface features.

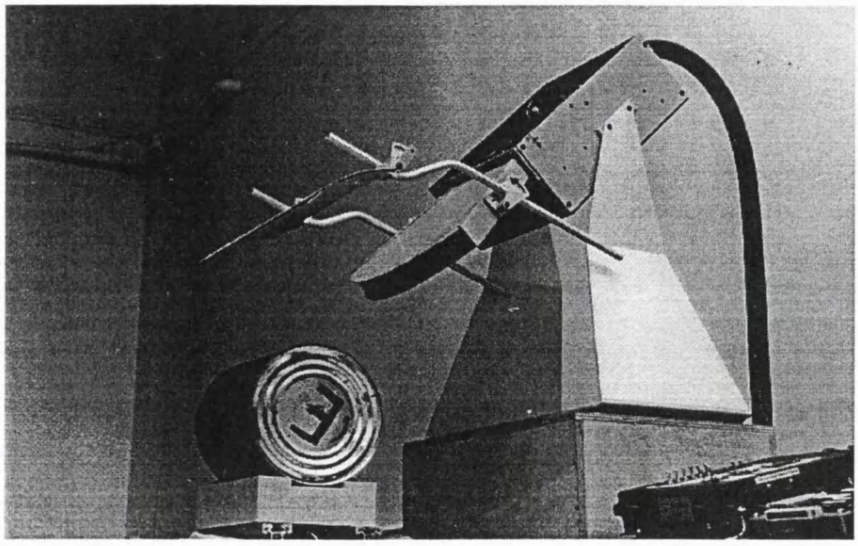


Figure 4.7. The test surface rotation unit.

Controlled by software running on the optics and laser electronics module (which is detailed in chapter 5), the test surface may be set at a number of different rotation speeds, thereby effectively changing the coarseness and rate of repetition of the sample inspection surface. Additionally, different product types and scenarios were provided by assembling a collection of drums with different surface characteristics. These are shown in figure 4.8.

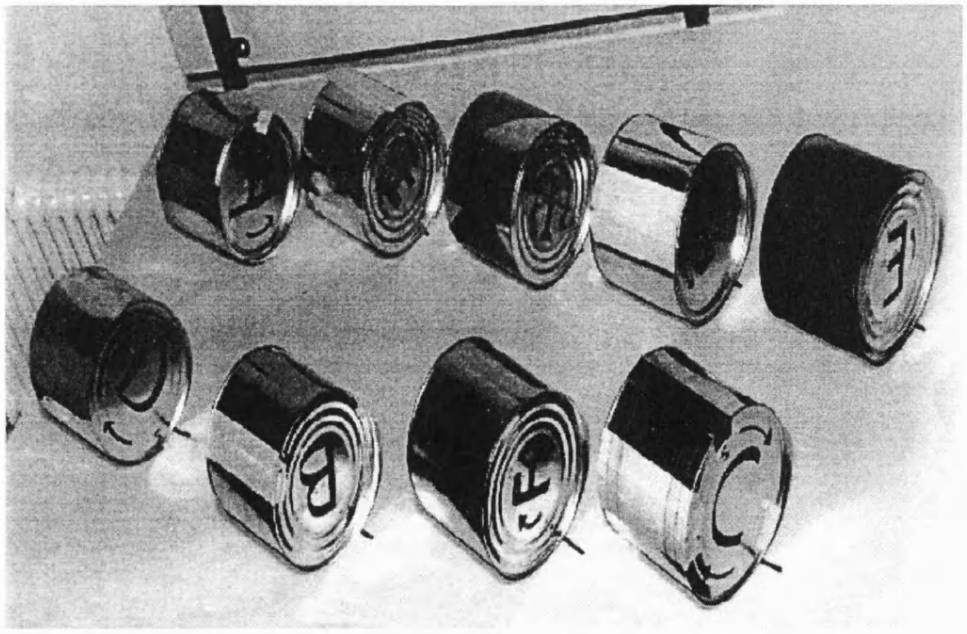


Figure 4.8. A collection of test surfaces.

4.1.4. The Photomultiplier Control Task and Environment

The task domain used in the majority of the experimental work reported in this thesis concentrates on the low-level control and maintenance of the inspection system's optical head. By far the most significant and resource-consuming aspect of this activity is the control of the photomultiplier output signals.

Photomultiplier tubes are light-detecting devices that have a huge dynamic range, and are used in this application for that very reason. Light falling on the tube's cathode end causes electrons to be emitted from a phosphorous layer on the inside surface. The electrons are accelerated down the tube by a number of extra-high-tension (EHT) voltage dynodes towards the anode end of the tube. The electrons striking the anode generate an output signal with a voltage level that is indicative of the intensity of the detected light. Figure 4.4 shows a typical signal as the laser scans across an object in ideal conditions. The control problem is one of controlling a photomultiplier sensor channel so that it maintains an optimal signal output with a mean pedestal amplitude of 1 Volt peak to peak. The photomultiplier controllers each have a low-resolution 8-bit analogue to digital converter which provides up to 1000 pixels per scan line of light intensity data in the range 0 - 255 units (referred to as ADC units). 0 indicates an absence of light and 255 a maximum. The plot in figure 4.4 was made by sampling these pixels.

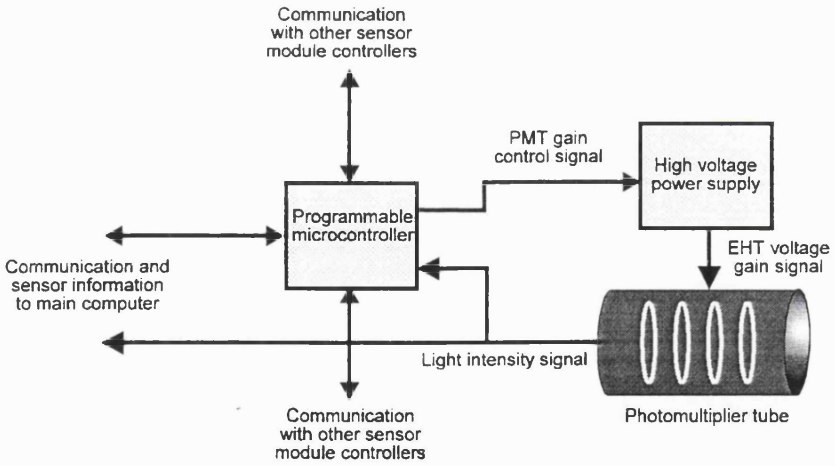


Figure 4.9. Photomultiplier control electronics.

The dynamic range of the output signal of a photomultiplier may be controlled by varying the voltage on the EHT dynodes of the tube. The graph in figure 4.10 shows a typical photomultiplier response for a constant light level and three different sensor-surface distances while 4.11 shows the response for differing ambient light levels. The Y axis shows the mean level of the signal pedestal, as in figure 4.4 (where this is the value used in current solutions to the control of the photomultipliers). In both cases the gain on the X axis (EHT voltage level) was increased from

zero until the output signal from the photomultiplier became saturated and no longer accurately reflected the detected light intensity. It can be seen that this response characteristic is highly non-linear and can be categorised into four main states that are indicated on the two graphs: (i) Quiescent; (ii) Near-Linear; (iii) Fold-back-A, and; (iv) Fold-back-B (the shape of the corresponding scan signals is shown in figure 4.12). As well as surface-sensor distance and ambient light, the regions also change as a result of other environmental effects such as changing product surface characteristics. The sensor controller must maintain a set-point operation in the near-linear region in order to provide a stable and usable light-intensity signal. Should the system be in any other state the sensor signal would be unusable. Different control tactics are required to return to the near-linear state from the others. Consequently, fast assessment of signal state is a fundamental aspect of the control process.

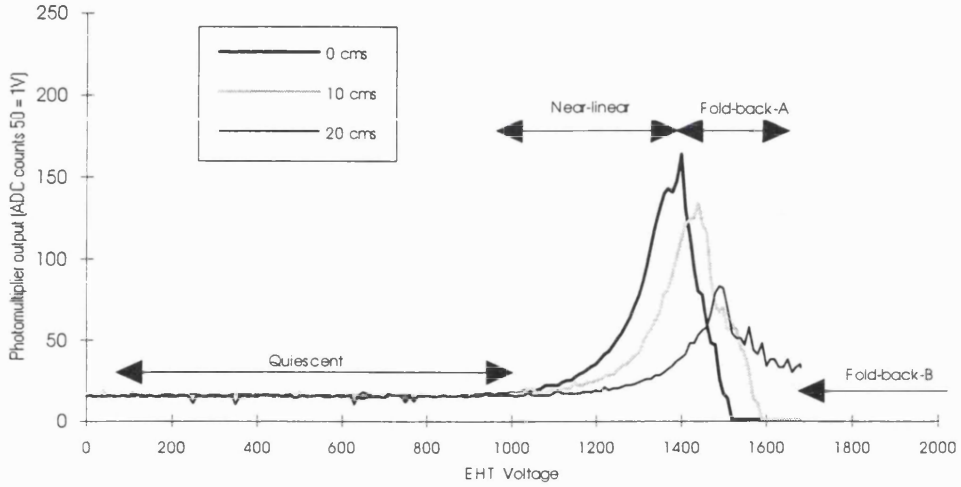


Figure 4.10. Response of the test-bed photomultiplier for constant light level and changing sensor-surface distance. Plot shows mean amplitude of scan pedestal against photomultiplier EHT voltage. State 1:Quiescent; State 2:Near-linear response; State 3:Unstable fold-back-A; State 4:Zero level fold-back-B.

Of particular significance in the control of the photomultiplier is the fact that a set-point of 1 volt may be achieved in either the near-linear or fold-back-A signal states. Clearly the latter is undesirable since it provides a highly distorted signal (see figure 4.12). It also is problematical in that control solutions that use only the mean level of the scan pedestal as reference can not simply distinguish between the two characteristic states. Highly unstable oscillation can result and this restricts the nature and extent of situations in which the FastScan system may operate. For example, while the management of control for a continuous product such as glass or plastic is relatively straightforward, a product that is sectioned and broken provides significant challenges to the existing control techniques. The search for a suitable control solution that can cope with this increased environmental complexity is the basis of the practical work reported in this thesis.

The control task, then, is to attempt to maintain the nice signal in figure 4.4 at a level of 50 ADC units (1 volt) irrespective of what happens in the world, e.g. changing ambient light levels,

changing sensor-surface distance, changing surface reflectivity. All of these have significant effects. In an extreme case the surface may suddenly disappear (the product on the line breaks or the product is in sections) resulting in a complete absence of reflected laser light and thus laser light being returned to the sensor at all. These effects may or may not be part of the standard behaviour of the production line. Consequently one of the problems faced by the controller is to be able to respond appropriately. In the case of a momentary break in product surface, an intuitive reaction of a controller might be to ramp up the sensitivity of the photomultiplier to attempt to recover a vanished signal. However, this will be to no avail and the maximum EHT voltage will be reached. The problem is aggravated when the surface reappears, as the photomultiplier's sensitivity, being at a maximum level in conjunction with the sudden arrival of high levels of light, will saturate the output signal (resulting in a fold-back state). This can be likened to people trying to see in a darkened room and then suddenly opening the window shutters to let in bright sunshine; momentarily they are blinded while the irises in their eyes adjust to the new light levels. Thus the photomultiplier system is also momentarily blind and the EHT voltage must be adjusted to adapt to new light levels (the reverse is true for the case of moving from bright light into darkness) and obviously the quicker this is done the more effective and useful is the system.

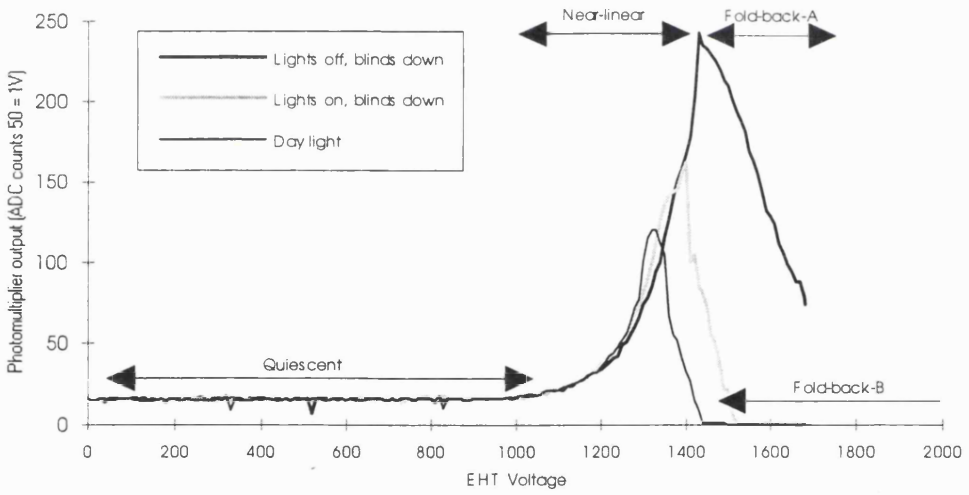


Figure 4.11. Response of the test-bed photomultiplier for constant sensor-surface distance and changing light levels. Plot shows mean amplitude of scan pedestal against photomultiplier EHT voltage. State 1: Quiescent; State 2: Near-linear response; State 3: Unstable fold-back-A; State 4: Zero level fold-back-B.

In order to control the photomultiplier, the generally accepted solution is to use an average level of the signal between the left and right surface edges. However, there is a potentially much richer source of information in the form of the 1000 scan signal pixels from the analogue to digital converter. From the plots in figure 4.12 it can be seen the four characteristic signal states are most distinctive. One aspect of the present work deals with the utilisation of this rich source of information. Also there are other sources which do not originate locally in an individual

photomultiplier controller, e.g. other photomultiplier channels in the system. Further sensor channels may provide useful secondary information on the state of the environment and a controller would be able to compare its own performance with that of others. Further, the possibility of self diagnosis is raised. In addition, it is often the case that other systems in the production line can provide information that may be useful in helping the photomultiplier controller to make suitable control actions. BBAI techniques are particularly suited to dealing with problems of this nature, where large quantities of diverse (in terms of type and quality) information are available for use from both local and remote sources.

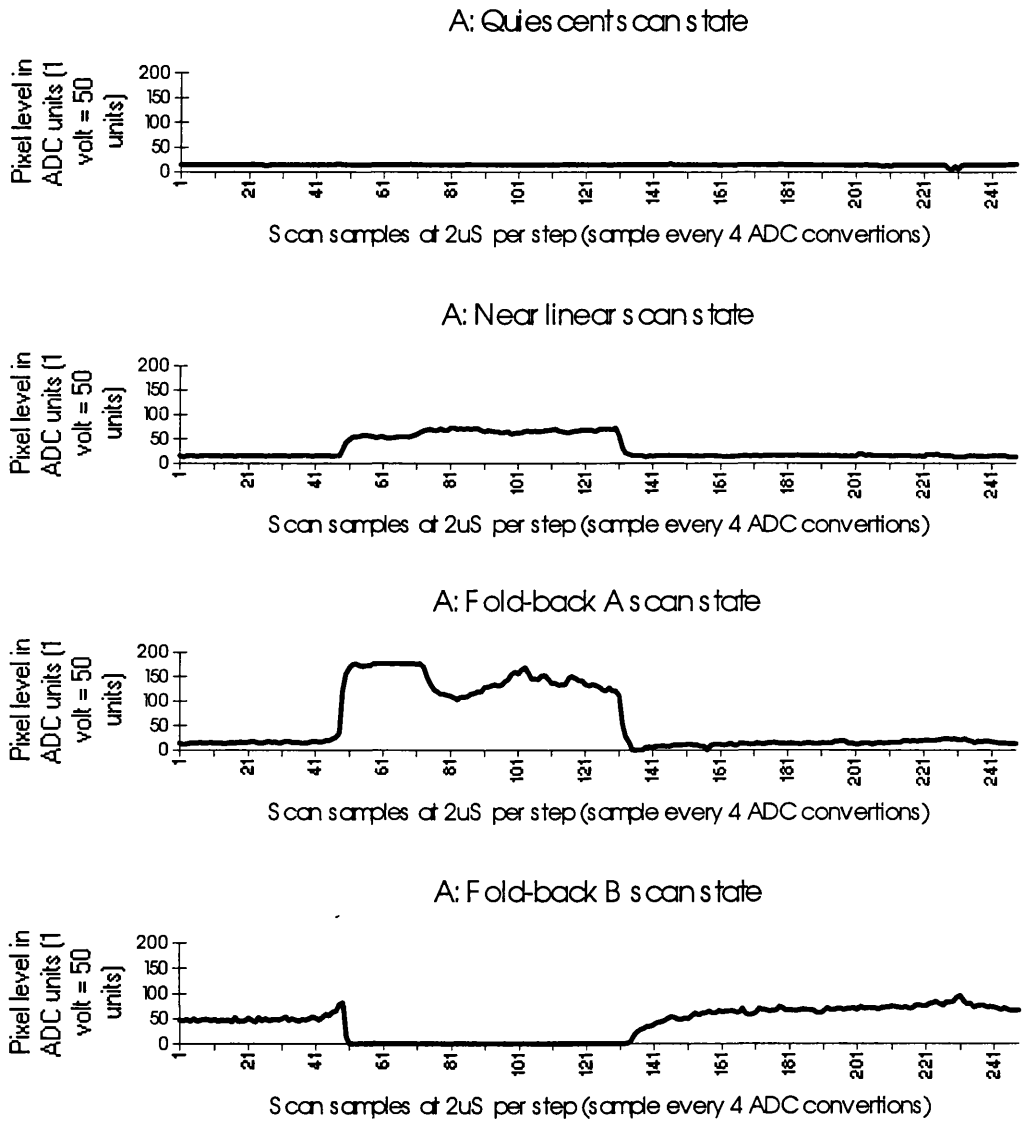


Figure 4.12. Typical scan line shapes for each photomultiplier state: A: Quiescent; B: Near-Linear; C: Fold-back-A; D: Fold-back-B.

Finally, other issues that are important include those of timing. The sensor's laser stripes across the field of view with a period of 500mS. It is clearly not possible to control the sensitivity of the photomultiplier within the scope of a single scan stripe and neither is it possible for each

separate scan stripe to be controlled individually. The stability of the sensor's output is in any case a result of the rate of change in the world, e.g. the speed at which the surface characteristics change. A surface passing at 1m/s will be scanned 2000 times per metre, each scan covering 500 μm. The real-time control rate at which the sensor should be able to adapt to changes is realistically not going to have to be so fast. In fact a real-time processing rate of a comparatively slow 20Hz would enable the system to react to changes of 10mm and larger, and this is generally an acceptable and realistic performance. The current maximum speed of the test-rig sample surface is of the order of 0.35 m/s, allowing a scan rate of 5700 scans per metre.

4.1.5. Computational Resources

The experimental test-bed is hosted by a personal computer which supports a number of transputer microprocessors on an internal expansion circuit card. The internal transputers are connected via transputer links to the individual control modules of the laser scanner optical front-end as detailed above. This configuration and utilisation of transputer processors provides a powerful and flexible framework for experimentation allowing reconfiguration of network topology in both hardware and software should it be desired. The nature of the transputer communication links is such that the addition of extra computational power through new processors is simply a matter of adding the device. This is analogous to the way additional behavioural layers are built up in the Subsumption Architecture.

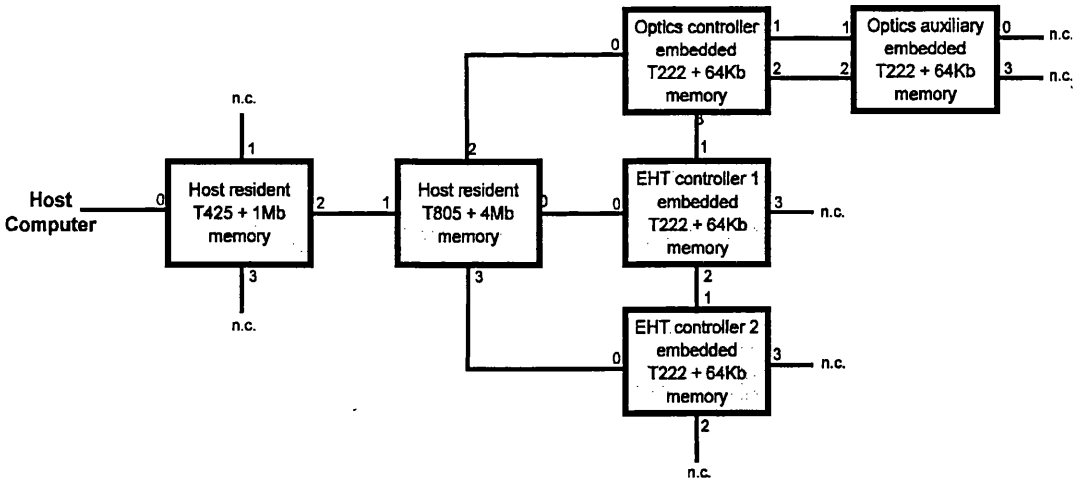


Figure 4.13. Test-bed transputer computing network and resources.

Figure 4.13 shows the configuration of the transputer network. The transputers used are a mix of types. T222s are 16-bit processors, the T405 is a 32-bit device and the T805 a 32-bit device with an onboard floating-point arithmetic unit. Programs are downloaded and then distributed to target processors via the T405 which is connected via interface circuitry to the host computer. All basic software is written in parallel 'C' which is a specialised form of ANSI standard 'C' that includes extra functional capacity to deal with the parallel process model and serial communications link protocols of transputers.

The T222 transputers are standard components of the hardware, photomultiplier and optics control electronics. The T405 and T805 however are an addition to take the place of the parts of the FastScan system not included in the test-bed. The T405 was utilised primarily to provide a user interface via the host computer while the T805, with its floating-point arithmetic unit, was used to run more complex control algorithms. This arrangement arose as a result of the fixed configuration of the test-bed system. The low level T222 devices were pre-designed components of the hardware. The requirements for an interface with the PC host computer were not sufficient to merit the resources of the T805 device which was best utilised with balanced communications links to all lower-level transputers.

4.2. A Distributed Subsumption Architecture

The subsumption architecture was overviewed as the best-known of the behaviour based control methodologies in section 3.2.3 of chapter 3. It has had a foundational role in many subsequent developments of behaviour based control and is still in use to this date as a research tool in the form of the Behaviour Language [Brooks90b]. This control architecture is in fact also beginning to be used in the engineering community for the development of mobile autonomous vehicles such as unmanned submarines [Zorpette94]. For these reasons it was decided to develop and use a subsumption framework for the distributed control of the laser scanner test-bed by constructing a tool to translate the augmented finite state machine specifications of the behaviour language into the parallel 'C' used to program the laser scanner transputers. This is referred to and detailed below as the behaviour language translator program.

The main feature of the behaviour language is the construction of a network of small and simple asynchronous processes known as augmented finite state machines (augmented in that they have real-time clock information built in to control timing) known as AFSMs. A number of AFSMs that have related functionality are built into "behaviour modules" which provide the layers of a subsumption architecture. Each AFSM has a number of input registers and a rule that defines the state transition of the AFSM. Each AFSM executes according to a built-in time constant known as the characteristic time (an error is flagged should the processor become overloaded and time overruns occur). Further details on the behaviour language can be found in Appendix A: "A Transputer Environment For Building Subsumption Control Architectures" and in [Brooks89] and [Brooks90].

The 'C' programming toolset used for the transputer network allows a number of individually compiled programs to be targeted onto a network of transputers. Each program with its "main()" 'C' function (and its collection of lower-level transputer processes) is configured to run as an individual transputer process set which may be loaded as many times as necessary onto one or many transputers for execution as one or many concurrent processes. Interprocess communication using the transputer channel protocol is taken care of by a virtual-link utility which is a background task automatically configured by the programming toolset. A subsumption

network program consists of at least one such "main()" program core which is referred to here as a "behaviour-node".

An important aspect of the transputer implementation of the subsumption architecture involves the nature of the software load and processor bootstrap procedure that initialises a network of transputers. One "top-level" transputer is connected to the host computer via one of its links. This transputer is known as the "server" transputer and is treated differently from the other "slave transputers" in the network. The server transputer must handle all communications with the host computer including loading software onto the network and access to files, VDU and keyboard. It does this by communicating with a server program running on the host computer. The user effectively communicates with the transputer network through a pipeline consisting of the host server program, the server transputer and any subsequently-programmed protocol on the slave processors. A network of transputer processes must include at least one top-level process that has been linked with special libraries that deal with the host communications. Consequently every subsumption network requires a special behaviour-node which is compiled and linked to deal with these extra tasks and provide the system services to the rest of the network. This node is referred to as the server-node. The server-node must reside on the server transputer while normal behaviour-nodes may be installed on any transputer including the server so long as a server-node process is also there.

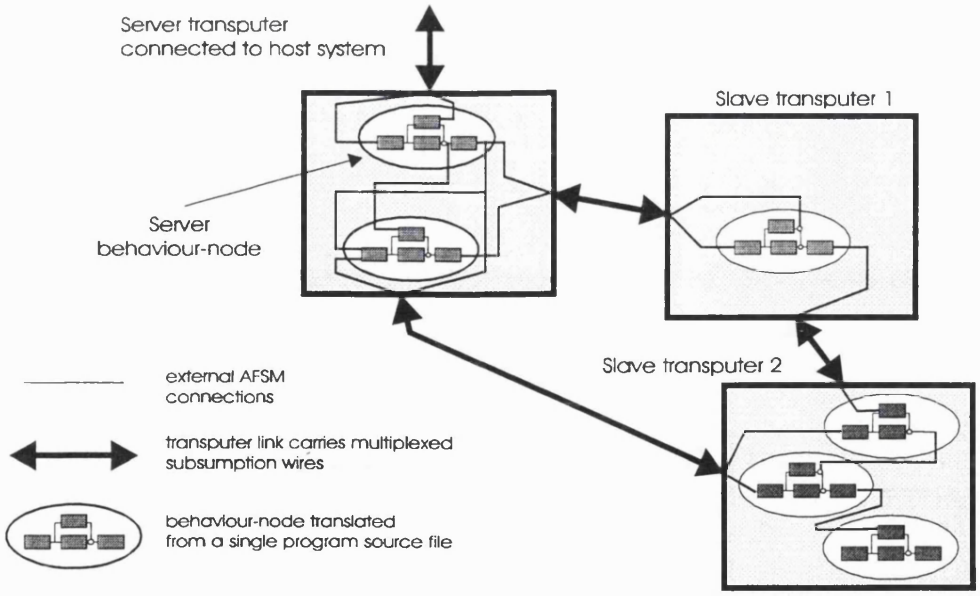


Figure 4.14. Six behaviour-nodes on a network of three transputers.

Transputer processes are inherently synchronous, conforming to a dataflow model of computation. Whilst the subsumption architecture is also of a dataflow format, the AFSM processes are basically asynchronous in operation. This difference initially posed a problem but was solved with the inclusion of extra (background) transputer processes acting as desynchronising buffers located between each pair of AFSM processes.

Another problem was how to implement the functionalities of the wires and the different connections which, in the subsumption architecture, arise as a property of the wires themselves. Transputer links have no function except to transfer data.

After experimenting with a number of process combinations, including single, twin (input buffer and AFSM driver) and triplet (input buffer, AFSM driver and output buffer) sets it was found that the most effective and computationally efficient solution was to implement each AFSM as a pair of transputer processes: an input buffer process and an AFSM driver process. These appear in figure 4.15. The subsumption wires and any connection functionality (suppress, default and inhibit) are implemented by the input buffer process. Hence each wire appears simply as a transputer channel. The subsumption connection node is not obviously distinguishable by viewing the transputer network channels alone. The following describes each transputer process in more detail.

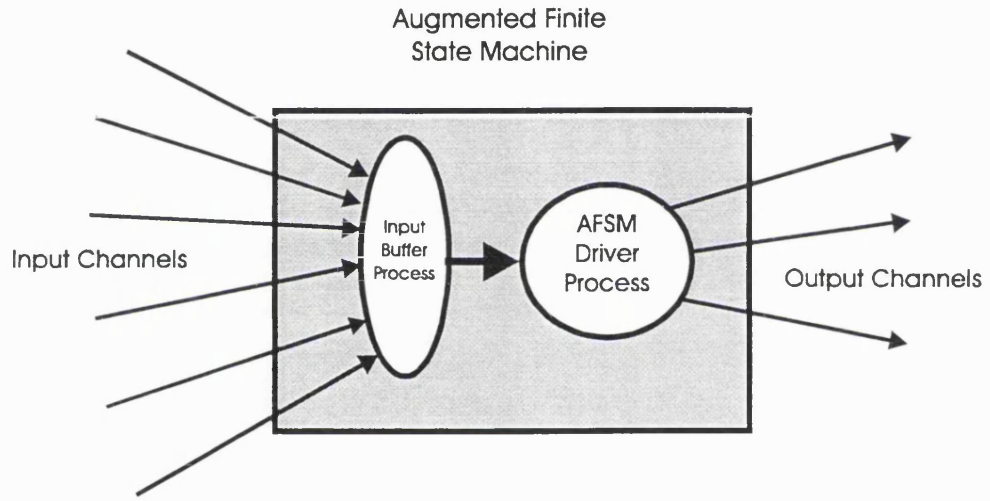


Figure 4.15. Transputer processes making up a single augmented finite state machine.

4.2.1. The Input buffer Process

The input buffer process has two functions. The first is to provide the desynchronisation between the transputer AFSM processes and the second is to implement the functionality of the subsumption wire connection nodes (inhibit suppression and default). Figure 4.16 shows how a number of AFSM outputs each suppress those of lower AFSMs and eventually affect an output AFSM. This is implemented as three separate channels, all connected to the input buffer of the final destination AFSM. It is this input buffer that performs the arbitration between the various suppressor nodes.

Channels into the input buffer are grouped into sets of wires as shown in the example in figure 4.16. The feature that brings these wires together is that they all have a common destination, as they are all connected either directly or by wire nodes to the destination AFSM D. The wires of each block are grouped and processed together because the value at the destination input buffer depends on the activity of all these wires and this must be taken into account when

AFSM registers are updated. A simple order of priority is used for the suppress and default nodes with the highest (in the case of suppress) active AFSM taking priority over lower ones. This priority is set when the behaviour program is translated into 'C' and the priorities are sorted taking into account the characteristics of the wire set. For example, default nodes are effectively an inverse of 'suppress', resulting in a reduced rather than increased priority for the wire concerned. This scheme has been implemented following careful study of the subsumption architecture implementations reported in [Brooks89], [Brooks90] [Brooks90b] and [Porter92], and has been demonstrably adequate for the subsumption based work reported in this thesis. However, it may prove difficult to implement networks that are more complex than simple ordered priority lists and for this further development of both the behaviour language and transputer parallel 'C' translator would be required.

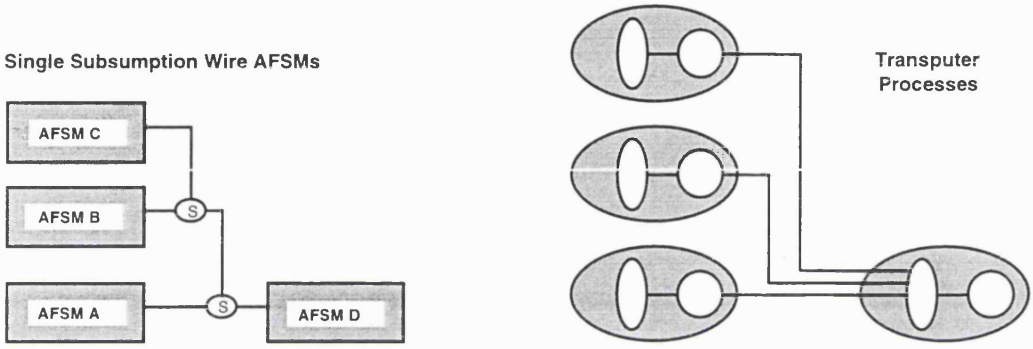


Figure 4.16. A set of wires with suppression nodes (left) as found in a Subsumption network and (right) as implemented in Transputer channels and processes.

Within the input buffer process is a main loop that polls the input wire set by set, testing continuously to see if any have data and servicing them if they do. A channel must be read in order that the sending process can continue operating. It is this continuous polling that desynchronises the system and prevents deadlock between processes, thus effectively breaking the dataflow model of computation used by transputers. At the end of each iteration the internal channel that connects to the AFSM driver process is tested to see if the AFSM driver is requesting an update of its internal registers. Finally, before repeating the loop the process is put on hold in order to give the other processes CPU time. This is presented in the pseudo-code listing below.

```
Input buffer process pseudo-code:
        Loop forever:
                Test each input wire set for active channels
                        if active then read in data
                        if not suppressed
                                update register
                Test AFSM driver update request
                        if active send all register data and status to AFSM driver
                Reschedule the process
        Loop End
```

4.2.2. The AFSM Process

This process executes the rule part of the AFSM as designed by the programmer, which takes the form of a block of standard 'C' code. This process also initialises and sets running the input buffer process during system start-up. A loop executes the relevant program instructions repeatedly with a characteristic time period that is determined when the program is compiled. This ensures that all the AFSMs in the network are loosely synchronised. The pseudo-code below outlines the operations. During the execution of the AFSM process the input registers are updated automatically. However, outputs must be commanded explicitly by the AFSM function as specified by the designer.

```
AFSM process pseudo-code:
        Initialise and run input buffer process
        Loop forever:
                Store real-time clock reading for use in characteristic time calculation
                Complete handshake with input buffer process to update registers
                Execute AFSM rule block functionality
                Test time and suspend process until system characteristic time is up
                        If time overruns then flag error to time monitor running in main()
        Loop end
```

4.2.3. 'C' Main() Function and Program Initialisation

Each behaviour-node is built around a main() 'C' function. This function initialises external and internal inter-AFSM channels and the AFSM driver processes (while buffer processes are subsequently initialised by the driver processes). It then sets all the AFSMs running and goes into a continuous loop that monitors the real-time performance of all the AFSMs in the local behaviour-node. The time monitor function actually differs depending on whether the program is to be a server-node or a normal node member of the network. The time monitor on the server transputer must monitor local AFSM performance and also poll inputs from other network nodes. The server-node maintains a record of time status reports from itself and all other behaviour-nodes. The time monitor on a normal behaviour-node simply polls local AFSMs and sends information to the server-node should any timing overruns occur.

```
Behaviour node main() function pseudo code:
        Initialise external channels from other transputers
        Initialise internal channels
        Initialise AFSM driver processes
        Set AFSMs running
        Loop forever:
                Perform time monitor function
        Loop end
```

4.2.4. An Example AFSM

The behaviour language translator program, mentioned at the beginning of this section, has been used to build the subsumption architectures that are outlined in the experiments in the rest of this

report. This section briefly presents the format used for specifying a single AFSM and its interconnections. The defmachine(){ } construct below is translated into the AFSM process pair as outlined above.

```
defmachine( some_name){
decls:   type register_name_1 [ = initialisation value],
         type register_name_n [ = initialisation value];
rule:
         whenever( condition){
                  rule block of 'C' code;
         }endwhenever;
}endmachine;
```

Each AFSM declared must have a unique name. AFSM input and output registers and any normal 'C' variables must be specified with a normal 'C' data type in the *decls:* field. An optional initialisation assignment can be made at this point to pre-set the variable at some value. If no initialisation value is supplied, a default of zero is used. Next the *rule:* field contains the block of 'C' code that realises the action of the AFSM. 'C' code is usually placed within a whenever() construct since this provides the looping and characteristic time functionality of the AFSM. The 'C' code may contain normal 'C' function calls and output() statements which cause a register's contents to be sent to its destination AFSM via a subsumption wire. A subsumption wire between two AFSM registers may be of any 'C' data type (although arrays must be treated specially; see appendix A) and is specified:

connect(*source, destination_1* [, *destination_n*]);

where *source* and *destination* are any valid AFSM registers. If the AFSM is located on another behaviour-node then it must be declared as *external(source)* or *external(destination)*. If the connection is to be a suppressor, default or inhibition node onto another wire, then destination must be: *suppress(destination), default(destination), inhibit(destination),* where destination is the destination AFSM register of the original wire. Suppression, default and inhibit connections to AFSMs in other *behavioural-nodes* are specified by *external(suppress(destination))*.

The declaration of AFSM processes in this way hides the actual transputer dual-process construction of the AFSM and subsumption wire functionality. The programmer can effectively build systems in the same way as those detailed in the behaviour language [Brooks90b] by declaring AFSMs and wire connections. Other programming, including low level input-output to hardware devices, is easily coded in 'C'. The behaviour language implemented here is in fact only a subset of that used by Brooks. Other features such as monostables and behaviour activation that were included by Brooks have not been implemented so far. The framework presented here was found to be sufficient for the experiments run with the laser-scanning application, and has been used throughout this thesis. It was the case that these programs did not include large agent behavioural repertoires so that activation and inhibition at this level was not deemed necessary.

4.3. Discussion and Conclusions

This chapter has presented a test-bed system that provides an industrially-situated and distributed problem domain for a control system. The chapter has also outlined an implementation of a subsumption architecture programming framework which in conjunction with the test-bed has been used for the experiments detailed throughout this thesis.

The photomultiplier control problem is an interesting one in its own right, especially when issues regarding varying and disappearing surfaces are concerned. This added unreliability of the environmental situation increases the non-linearity of the control problem significantly from that usually dealt with by classical control techniques. A significant problem is the nature of the changing surface. The controller of the photomultiplier must be able to differentiate between changes in the light intensity signal that have resulted from some internal actuation and those caused by some environmental influence. It is this aspect of the system that will be the focus of experimentation in later chapters.

The transputer processing architecture of the system also provides some interesting implementational aspects for a system based on a subsumption architecture. Although distributed networks using other microprocessors may provide similar results, transputers have proved particularly suitable in the present work. Primarily this has allowed us to ensure a truly distributed target system. It is the case that many implementations of a subsumption-controlled agent have not been truly distributed but instead have utilised a central processing unit and perhaps a number of slave processors and software device drivers that control input-output functionality independently of the main behaviour control software, as in [Brooks89], [Jones & Flynn 93], [Ferrell94] and [Steels94]. An important exception to this is the work in [Connell89] which describes a can-collecting mobile robot that utilises a subsumption control structure resident on 24 processors. Other distributed implementations of subsumption are reported in [Cornell *et al* 94], [Porter92] and [Webber & Bisset94]. The total distributed nature of the transputer subsumption implementation reported here, along with the modular architecture of the laser scanner test-bed, provides an interesting framework for experimentation with behaviour based systems. In particular, the study of issues concerning the interaction and integration of several virtually independent parts of a system is possible.

Finally the nature of the test-bed allows extensive sampling and storage of run-time program and system status which can then be analysed off-line in greater detail. This facility makes the test-bed a powerful tool for experimentation with different control tactics and even strategies.

The next chapter details a complete behaviour based implementation of the laser scanner optical head and its associated operating environment, using the subsumption architecture.

5. The Behaviour Based Control of an Active Laser-Scanning Sensory System

This chapter covers the following:

The definition and integration of a subsumption control structure for the laser scanner test-bed.

A series of experiments that illustrate the behaviour based control of the test-bed system and the effects of altering behaviour-level priorities.

Some initial conclusions on the experiments from an applications viewpoint.

This chapter illustrates the use of a behaviour based design methodology on the laser-scanning inspection system test-bed that was presented in the previous chapter. The modular nature of the system hardware necessitated a correspondingly modular and distributed approach to the software element of the control. The control solutions described in this chapter can be seen globally as realising a three-layer subsumption control structure although at the local level of the separate controllers a two-layer system was more evident. The complete system was developed in the bottom-up manner typical of BBAI techniques. The latter part of this chapter reports on experiments with the tactics used for the control of the photomultiplier modules. These experiments serve both to illustrate the development and implementation of a behaviour based system and as a means of demonstrating the effectiveness of the methodology. The distributed nature of the system necessitated the development of a number of the control modules in parallel, which resulted in an almost stand-alone development for each subsystem: (i) motor and laser control; (ii) photomultiplier control channels; (iii) user interface. Figure 5.1 shows these modules.

This chapter is presented in 7 sections. The first three deal with the separate parts of the subsumption control structures that were assembled to control the system. The emphasis here is on the control of the behaviour of three separate component subsystems of the experimental test-bed. This approach has been taken as a result of the natural divisions in the test-bed architecture. The motor and laser control subsystem is dealt with first, followed by the photomultiplier subsystems and then a top-level user interface. Section 4 details the mapping of the software subsumption control structure onto the transputer network of the test-bed, and this is followed naturally by section 5 which outlines a series of experiments which tested the

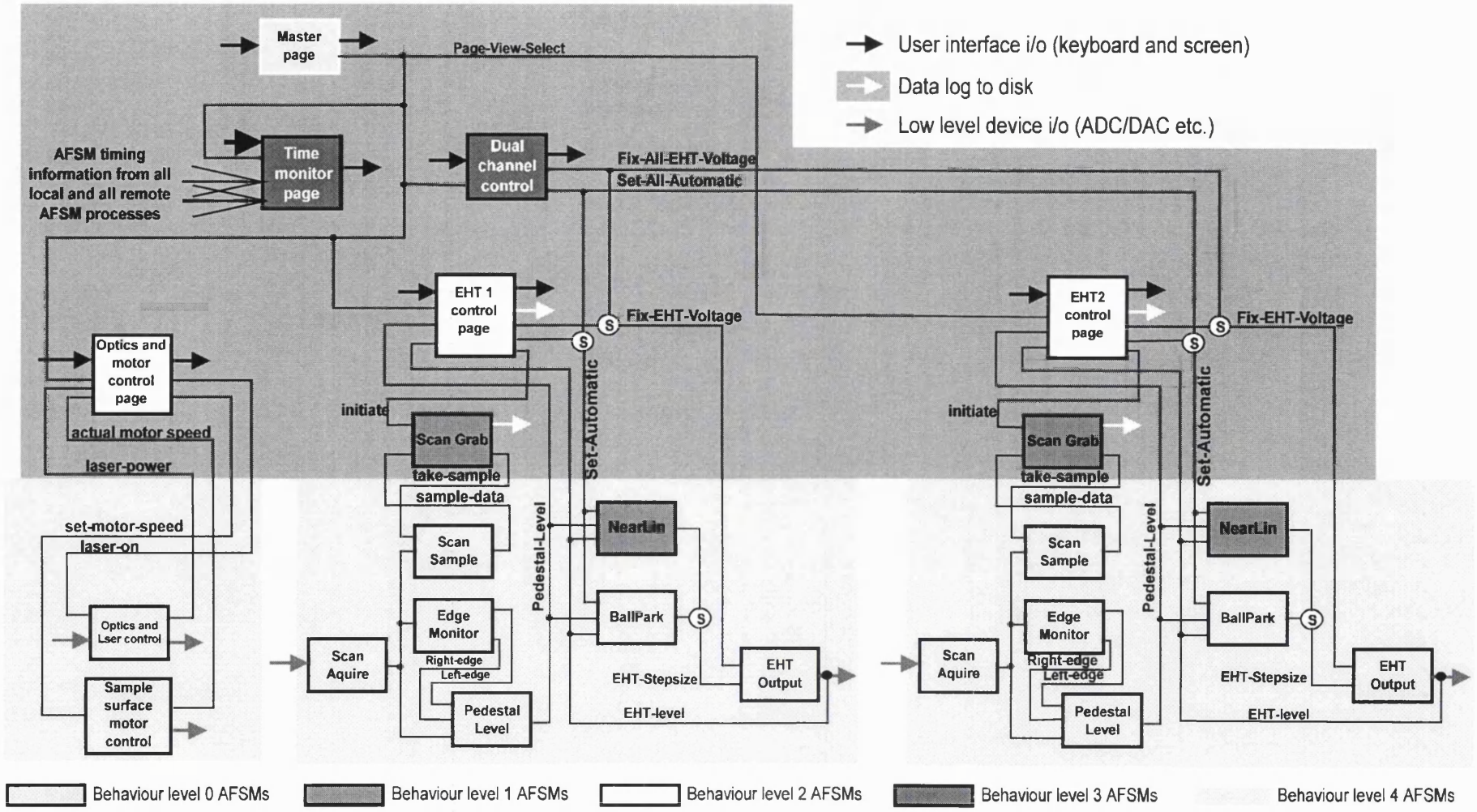


Figure 5.1. The behaviour-nodes of the laser scanner test-bed subsumption architecture.

structure. Section 6 presents the results and section 7 discusses some aspects of this particular application of the subsumption architecture in preparation for the more general discussion and conclusions that are presented in chapter 6.

5.1. Motor and Laser Control Subsystem

This part of the laser scanner system was located on the optics controller module of the test-bed. It was required to monitor laser power and deal with switching of the laser depending on the setting of a number of safety interlocks and commands from the operator. Much of the functionality of this subsystem is dependent on the industrial situation in which the FastScan system is employed. In a full system the Optics controller is used to provide a number of interfaces with other systems, but this aspect was not required for the test-bed system. However, a particular and unique function relevant to the test-bed system only was the task of controlling and maintaining the test-surface rotation speed. This was performed by a subsumption control structure as shown in figure 5.2 below. Pseudocode listings for the functionality of each of the AFSMs of this system appear in appendix B.

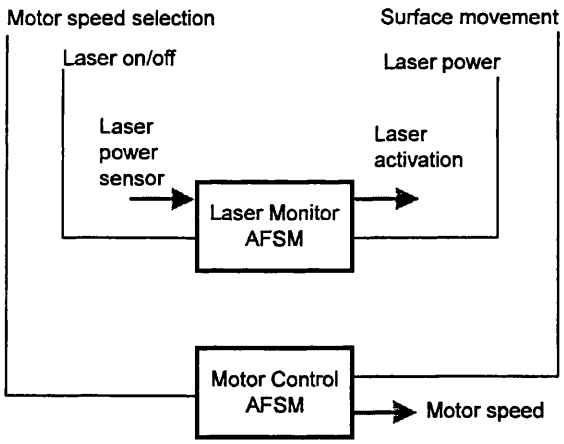


Figure 5.2. The subsumption control structure for the Motor and Laser control subsystem.

Laser Monitor AFSM:

This AFSM provided the hardware control and signal distribution of the laser power sensor. The main function in the work reported here was for use as an indication that the laser was on and thus illuminating the surface under inspection. This information was particularly significant if a surface had disappeared. The fact that the laser was still on was a useful cue indicating that the physical system was still operating normally in this respect.

Motor Control AFSM:

This AFSM controlled the speed of the motor for rotation of the test-bed surface. In our experiments the speed was pre-set by the operator and maintained by this AFSM. It is possible to

foresee further use of this functionality in providing an automatic variation of speed depending on the level of detail and stability of the laser scan signal as different surfaces are inspected.

The speed-maintenance functionality was required on the test-bed due to some irregular surface sample shapes tending to cause occasional stalling of the motor when running at low speeds. The response of the AFSM was to increase the power setting of the motor briefly until the drum restarted its rotation, whereupon the power was reset to its original value. Detection of a motor stalled state was provided by monitoring the scan signal for continuous small changes in level which was a normal condition as the product moved past the scanner. A stalled condition was validated by a continuous run of a number of static scan signals from all active photomultiplier channels. A pseudocode listing appears in appendix B.

5.2. Photomultiplier Control Modules

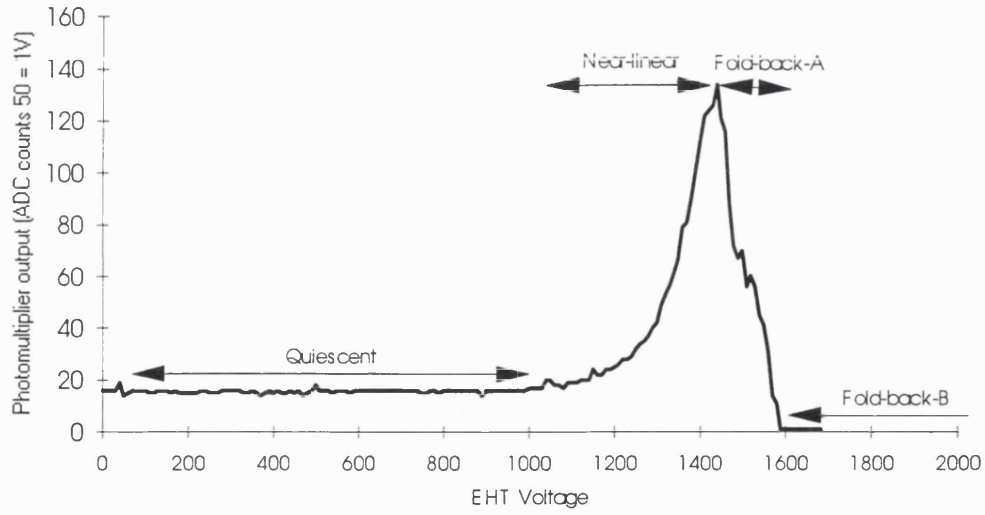


Figure 5.3. The photomultiplier output response, given constant ambient light levels, constant surface-sensor distance, constant surface characteristic and increasing EHT voltage.

The nature of the photomultiplier control problem was discussed extensively in chapter 4, section 1. Figure 5.3 below shows again a typical response of the photomultiplier scan pedestal to an increasing EHT voltage and the four characteristic signal states: quiescent, near-linear, fold-back-A and fold-back-B. The task of these control modules (one for each channel on the test-bed) is to maintain the set-point scan pedestal level of 1 volt (50 ADC units) in the near-linear signal state. One particular catch here, as highlighted in the previous chapter, is that the fold-back-A state may also be controlled to a similar set-point signal level. By examining the scan level alone it is not easy to distinguish between the correct set-point and the fold-back set-point. The fold-back set-point state must be avoided since the signal is extremely distorted and will generate errors in the defect detection parts of the FastScan system (which is not a part of the test-bed). The following gives details of a subsumption control structure designed to deal with these problems.

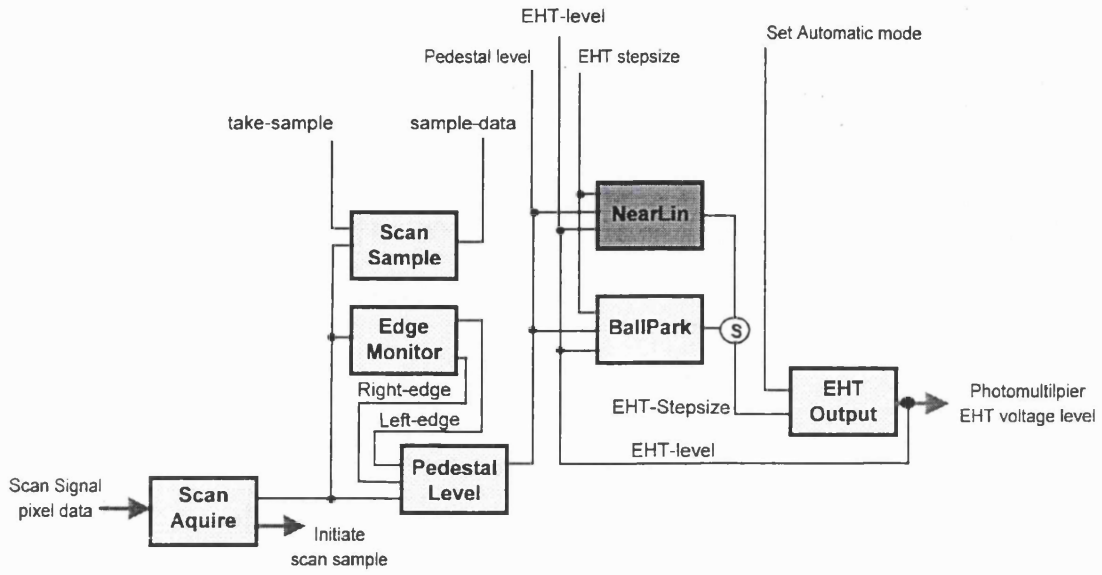


Figure 5.4. An example of a subsumption control structure that was used during some of the laser scanner photomultiplier control experiments reported in section 5.5.

From the signal characteristic shown in figure 5.3, it is apparent that the near-linear part of the response would be suitably controlled with a relatively straightforward proportional error control scheme. In contrast the other regions, quiescent, fold-back-A and fold-back-B require specific strategies which must return the photomultipliers to the near-linear region as quickly as possible. The behavioural requirements here can be seen in the need for a smooth, precise and accurate control process to take care of the near-linear situations and a fast, coarse control of the other quiescent and fold-back situations that can return the system to the near-linear state as quickly as possible. Various combinations of a two-layer subsumption structure were used to deal with each of the aforementioned characteristic states of the photomultiplier. These are the focus of section 5 which details the configuration of the subsumption control structure and the experimental set-up. Subsequent sections of the chapter describe a set of AFSMs that were used in various combinations in the experimental runs but which generally followed the format shown in figure 5.4. Firstly the interfacing with hardware input-output devices is detailed. This necessary "housekeeping", performed generally on the lowest layer 0, concerns the operational protocols of analogue to digital and digital to analogue conversion devices used for reading in scan-line signal data and writing out EHT voltage level settings. The next section details two AFSMs: *NearLin* and *BallPark*. These two, dealing with the fine-tune near-linear control and coarse large stepping control respectively were used in various combinations to experiment with the layered nature of behaviour based control. Figure 5.4 shows an example of the subsumption structure used. Further details and other structure examples are given in section 5.5.

5.2.1. Level 0: Hardware Interfacing

These AFSMs consisted of two groups clustered around the functionality of reading input devices and writing output to actuation devices in the form of the photomultiplier EHT amplifier. As such they formed the main part of the bottom layer of the photomultiplier subsumption control structure. Given the singular nature of the behavioural requirements at this level of system control the assignment and specification of AFSM tasks was relatively straight forward.

Scan Acquisition: *ScanAquire, PedestalLevel, ScanSample* and *EdgeMonitor*

These four AFSMs formed a small subsystem that processed the scan-level input data. The function of the *ScanSample* AFSM was to provide a scan stripe capture function in support of the user interface layer; it was not an active player in the control of the photomultiplier. The *ScanAquire* AFSM controlled the analogue to digital conversion of scan stripes and loaded the data into a vector of up to 1000 pixels. This vector was then accessed by the other AFSMs in this group. Each of these AFSMs was asynchronous; there was no handshaking or other form of synchronisation. The *EdgeMonitor* processed the scan stripe vector by searching for large gradients in pixel value (each pixel contains a value representing the intensity of the light detected). The first positive gradient was taken as the left edge and the last negative gradient the right edge. The *EdgeMonitor* was set up to execute only every four characteristic time cycles. This was because we judged that the surface edges would not be changing fast enough to warrant such a fast update, and in any case the *PedestalLevel* AFSM was able to provide a sufficient output so long as the left and right edge signals were somewhere near the right place.

The *PedestalLevel* AFSM calculated the mean level of the scan line between the left and right edges of the signal's pedestal and then used a rolling average of the last 10 scan lines in order to filter out noise. We decided on this aspect of the control after noticing that if the operator selected the EHT voltage output and left it fixed at one value, the scan-level fluctuated by a small number of ADC units either side of the desired set-point. The rolling average effectively filtered out this noise.

EHT Output

This AFSM formatted and output an EHT voltage level that was a function of the *EHTVgradient* input connection. The input provided a simple signed rate of change value which the *EHTOutput* AFSM used to step the EHT voltage level. This AFSM also received two inputs from the user interface level: (i) *AutomaticEHT* which effectively switched the lower level control on and off and (ii) *FixedEHT* which was a user-selected constant EHT output level.

5.2.2. Actuation Control AFSMs

Two AFSM processes were developed to control the photomultiplier. The first was the *BallPark* AFSM, built to act in the quiescent and fold-back states to return the photomultiplier to the near-

linear state, i.e. to get the scan signal into the right general range so that a fine-tune process could be applied by the *NearLin* AFSM.

Ball Park AFSM: Coarse Control

This AFSM attempted to find the EHT voltage such that the scan-level pedestal was driven into the near-linear region (figure 5.3). This was achieved by outputting a value to the subsumption level 0 *EHTOutput* AFSM in the form of a particular size of step adjustment to the output voltage. Output of this AFSM consisted of a single signal - *EHTVgradient* - which was set according to the identified state of the scan level signal. This choice was intended to minimise the transition time to the near-linear characteristic region. Inputs used a combination of the current scan level as provided by the *PedestalLevel* AFSM and the current EHT voltage level as output by the *EHTOutput* AFSM. See the table in figure 5.5 for an outline of the five programmed states of this AFSM.

AFSM state	Identifying characteristic	*BallPark* output
0. Inactive	No identifiable input condition	Inactive
1. Quiescent	9 < Scan level < 18	Step size = 200 Volts
2. Near-linear	Scan level gradient polarity = EHT output gradient polarity	No output, desired state
3. Fold-back-A	Scan level gradient polarity ≠ EHT output gradient polarity	Step size = -20 Volts
4. Fold-back-B	Scan level < 9	Step Size = -100 Volts

Figure 5.5. Table of *BallPark* AFSM input-output state mapping. Signal levels are in ADC units where 1 unit = 0.02 Volt.

NearLin AFSM: Linear Set-point Control

AFSM state	Identifying characteristic	*NearLin* output
0. Inactive	No identifiable input condition	Inactive
1. Negative set-point error	Near-linear state and scan level < 47	*step_size* = ADJUST_FACTOR × (set_point - scan_level)
2. Positive set-point error	Near-linear state and scan level > 53	*step_size* = ADJUST_FACTOR × (set_point - scan_level)
3. Signal in range	Near-linear state and 47 < Scan level signal < 53	No output, desired set-point achieved

Figure 5.6. Table of *NearLin* AFSM input-output state mapping. Signal levels are in ADC units where 1 unit = 0.02 Volt.

If the scan level was detected to be within the limits that characterised the linear response region (the same constants were used as in the *BallPark* AFSM above) of the photomultiplier, this AFSM acted by calculating an *EHTStepSize* output as a function of the size of the difference between the current scan pedestal level and the fixed set point of 1 Volt. The output of the *NearLin* AFSM was combined with that of the *BallPark* using various subsumption wire nodes (suppress and default). This is detailed below in section 5.5 of this chapter as a focus of the

reported control structure experimentation. The table in figure 5.6 details the input-output mapping of this AFSM which operated with 4 basic states. The output caused the *EHTOutput* AFSM to alter the current EHT voltage accordingly. The constant parameter ADJUST_FACTOR was selected as a best compromise to approximating the scan-level gradient increase in response to EHT voltage increase. The compromise results from the fact that, as mentioned previously, this gradient is continually changing as an outcome of varying environmental conditions.

5.3. User Interface

This series of layers consisted of a number of AFSMs that were located as a top-level control structure designed to act on a global basis above whatever arrangement of layers existed on local low-level controller modules. The AFSMs were arranged to present the operator with a keyboard and screen interface to the activities of the laser scanner control systems and to provide data sampling and file storage facilities. In this respect it is possible to envisage the operator as a higher level of control, acting within and as part of the subsumption structure by subsuming the behaviour of the lower levels. Figure 5.7 shows this module's subsumption control structure.

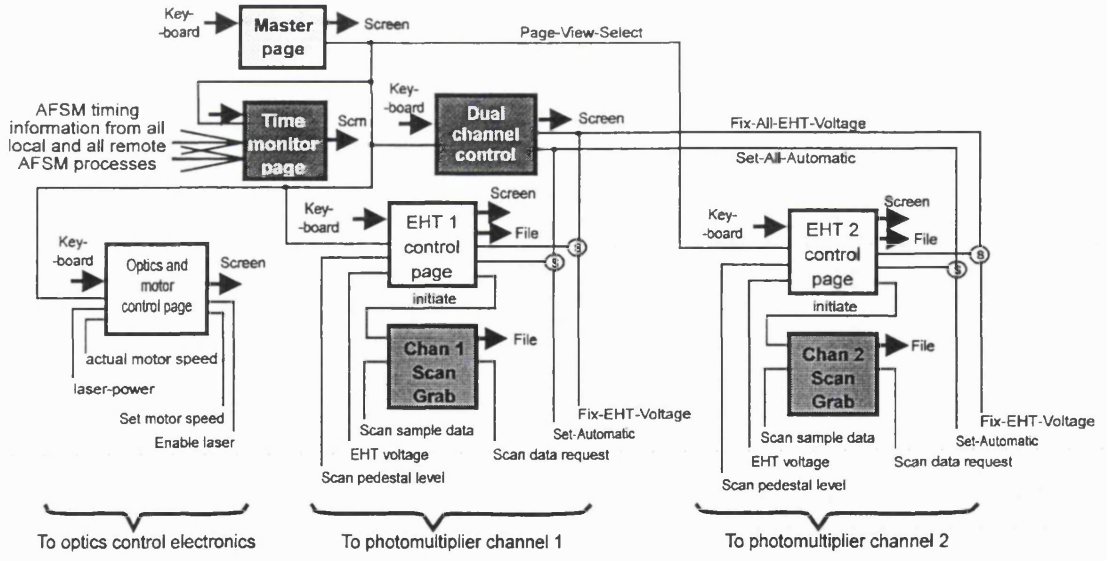


Figure 5.7. The subsumption control levels of the user interface. Bottom, layer 0: channel data sampling AFSMs, layer 1: sensor channel user/control interface, layer 2: Multi-channel control, and the top layer 3: Main keyboard and screen input output.

The input for this level of control comes from the user via keyboard entries and from lower levels of control via subsumption connecting wires. Input from files may also be regarded as input to the system at this level. Output or actuation at this level is via the host computer display, to host computer files and downwards to affect the behaviour of the lower subsumption control layers. A number of "pages" were created to present the user with different information. These are listed below. Each page AFSM was arranged in an order of priority so that at any one time

only one is attempting to write messages to the screen. This is achieved using a hierarchy of suppressor nodes on the AFSM outputs. Figure 5.8 shows a typical screen for a display page.

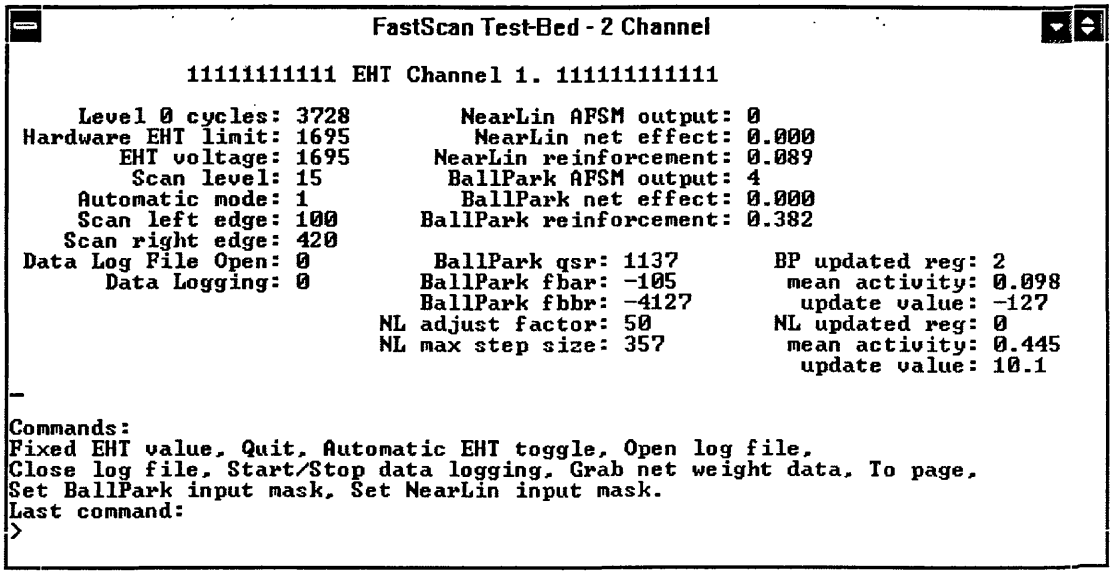


Figure 5.8. The page screen display for the user interface *EHT1Control* AFSM.

Master Page: This page provides the overall coordination of other interface pages and presents the user with a top-level opening screen.

Time Monitor Page: This page provides a display of all system AFSM timing. If any AFSMs are computationally overloaded then it is possible for them to cause characteristic time overruns. This would result in indeterminate behaviour interactions. Despite the fact that the AFSMs are asynchronous, it is the case that, if transputer processes overrun their characteristic time allowance, actuation timing and external event timing can be corrupted. This page allows the operator to ascertain if any such events are imminent or occurring.

Photomultiplier Control Pages: These pages provided user switching to enable the system to run on either automatic or fixed EHT modes. In fixed mode the operator was able to set the output EHT voltage level manually. Other channel-dependent control functions are provided to set up run-time data sampling of certain aspects of the channel's behaviour. Each page actually utilised two AFSM processes: one to deal with the main user interface functions and the second to deal with the sampling of scan line arrays. The second process was used because of the necessity to transfer a large amount of data. It enabled a single scan to be transferred over several characteristic time cycles thereby reducing the real-time loading on transputer communication links.

Optics Control Page: This page provides for an operator interaction with the laser control and surface rotation control electronics. Laser power (as sampled by the optics control module) is displayed as well as motor control information. Provision is made for manual selection of the speed of rotation of the test surface drum.

5.4. Software-Hardware Mapping

The laser scanner test-bed provided a target network of 6 transputers. Of these there were two located in the host computer and the rest embedded amongst the various controller modules. As is always the case in real-life (and, in particular, in reflection of the general-purpose experimental nature of this implementation), the target hardware did not ideally match the breakdown of software functionality that was required to implement the proposed subsumption control structure most effectively. This section outlines a possible mapping of the subsumption control structure described above onto the test-bed that was detailed in chapter 4. Figure 5.9 shows the mapping of behaviour-nodes onto transputer hardware.

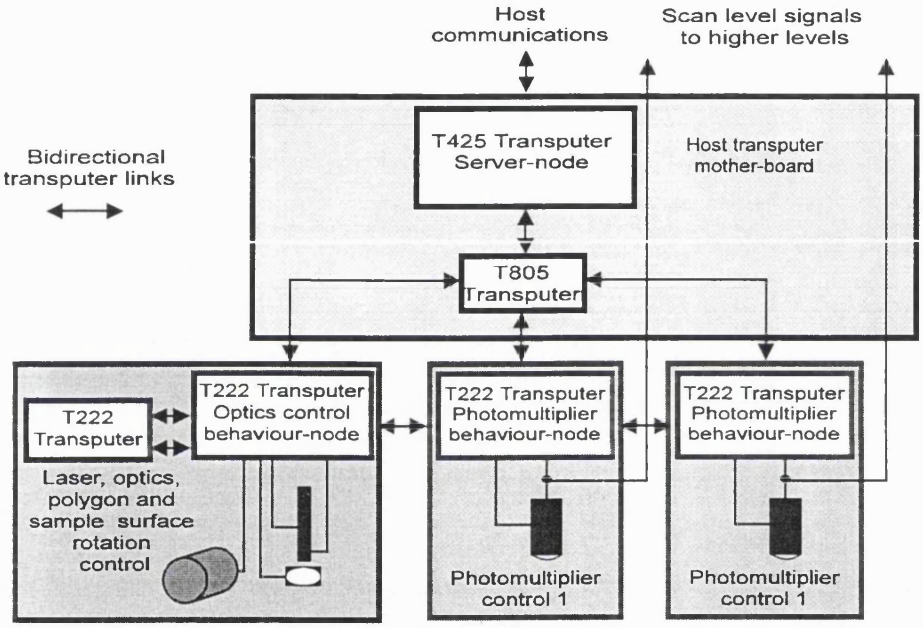


Figure 5.9. Functional units of the laser-scanning inspection system.

The transputer subsumption framework required at least one server-node process set to provide a top-level interface with the host computer. This was the function of the user interface layer which was built into a single parallel 'C' program of processes. This server node was located as the only behaviour-node resident on the top-level T405 transputer.

The relative simplicity of the optics control part of the subsumption control structure enabled an implementation of a single behaviour-node resident on the optics control module's main T222 transputer. Connections to other processes utilised the auto-routing of the transputer programming environment with connections to the other behaviour-nodes located on the most direct transputer links.

The photomultiplier subsumption control structure was more complex but in this example we were nevertheless able to implement it as a single behaviour-node that was loaded once onto each of the photomultiplier control modules.

5.5. Experiments

The experiments described here illustrate the performance and operational characteristics of the subsumption controlled laser scanner. The object of the exercise was to illustrate the use of this multilayer control strategy and compare its performance with that of a more traditional single layer. These experiments, in conjunction with the control implementation details of section 5.3, will serve as the basis for extended discussion in chapter 6. The rest of this chapter deals directly with the experiments and the results as far as they affect the laser scanner test-bed system.

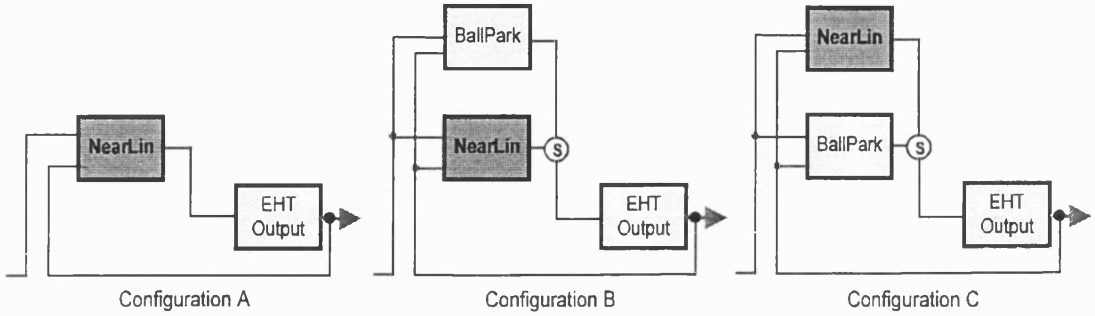


Figure 5.10. Basic behaviour-defining elements of the three subsumption control structures used for the laser scanner photomultiplier control experiments: A: Near-linear control only; B: level 0 near linear control and ball-park level 1; C: ball-park level 0 and level 1 near-linear.

One of the main and most obvious benefits of the behaviour based approach to control appears to be the bottom-up development of competence. This series of experiments started with a simple single-layer-only test run that applied control only in the linear region of the photomultiplier's response. The system was tailored and set up to respond to the particular surface and environmental conditions in much the same way as a straightforward classical controller would be. This was implemented by using the *NearLin* AFSM only, without the actions of the *BallPark* behaviour. The second and third runs of the experiments used combinations of the complete subsumption control structure outlined above. In the first instance the level 0 *NearLin* AFSM was supplemented with a subsuming level 1 BallPark AFSM, the idea being that when the NearLin AFSM lost control of the situation the BallPark would be able to take over to recover the near-linear signal region. In the final set of experiments the order of the layers was reversed. *BallPark* was situated in level 0 with *NearLin* suppressing from level 1. This third configuration is perhaps the most intuitive behaviour based control structure, given the bottom-up design constraints. This point is discussed further in the concluding section. Figure 5.10 shows sketches of the three structures.

In all cases of the experiment, four test runs were made, each with different surface types (figure 5.11):

1. Continuous single-tone surface.
2. Two-tone surface consisting of two repeating patches of differing reflectivity.

3. Segmented surface consisting of a single level of reflectivity separated by a repeating pattern of non-reflective patches.

4. Multi-tone broken-surface.

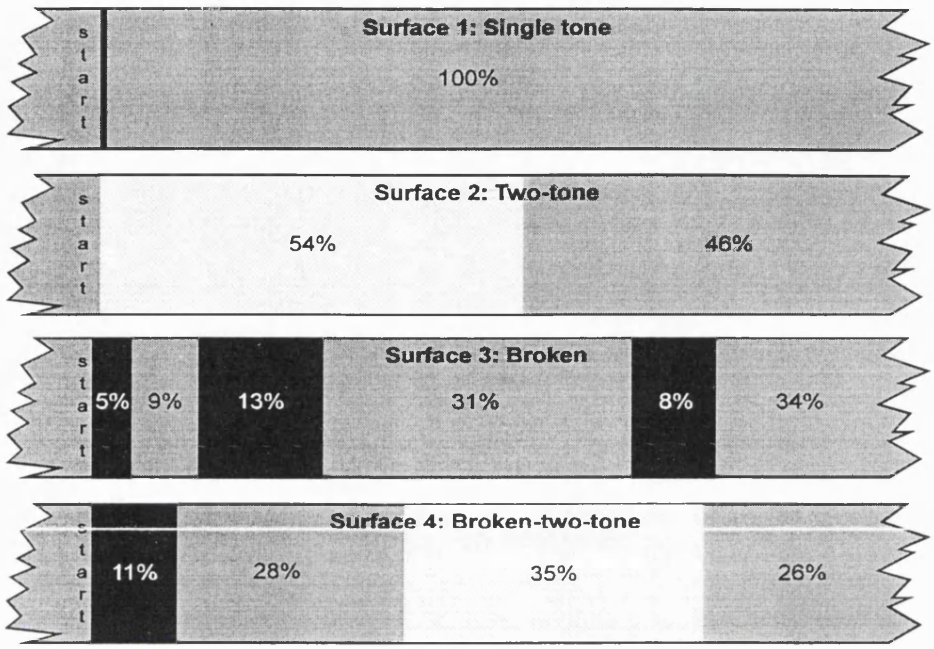


Figure 5.11. Surface configurations used on the test-bed's rotating surface for the three experimental control architectures.

Each test run was conducted in a controlled lighting environment to remove any influence of differing daylight conditions (direct/shaded sun) on the test-bed photomultipliers. Each test configuration was run for a period of 60 minutes.

Each surface gave a different set of stimuli to the controller taken from possible industrial situations. Surface 1 was a continuous reflective surface and as such provided a basic example of the sort of situation currently encountered by the FastScan system. The other three surfaces each introduced additional complications including stimuli designed specifically to test the recovery from scan signal fold-back. For example, surface 3 provided a broken or jointed surface, similar to a production line that presented individual discrete items, such as television screens or car body panels, for inspection. In addition to these large-scale features the surfaces were by no means perfect, as they contained numerous scratches, dents and other blemishes that increased the variability of the signal significantly. Perhaps the most significant in these was the join in the surface that provided every sample with a small break or uninspectable region in the order of 2 mm in length.

Each surface was 490 millimetres in circumference and was rotated continuously past the laser scanner at a rate of 2 revolutions per minute (effectively a surface speed of 16 mm/second). Being in part belt-driven the rotation mechanism (as detailed in chapter 4) ensured that the actual

passing of the surface was of an irregular nature with the exact timing of each rotation varying to some indeterminate degree.

5.6. Performance

The performance of the laser scanner system was in principle relatively easy to quantify; it was simply the amount of time, out of the total time of operation under suitable conditions, when the sensor was in a state able to provide a stable scan signal as output to higher-level defect identification subsystems (which are not part of the test-bed). In practice however, there was not a clear-cut boundary between a "good" and a "bad" signal. Although the system was attempting to maintain a set point of a 1 Volt peak-to-peak signal on the photomultiplier, there was a region around this level which still provided a suitable signal. Indeed it will be evident from the description of the *NearLin* AFSM above that it only attempted to control the signal to maintain an output signal region of 0.94V - 1.06V. When the signal was in this state it was possible to flag this particular characteristic time instant as being a "good" state in which the scan signal pedestal provided an optimum level to make possible the differentiation of surface defects in the form of positive and negative spikes. While the signal was still usable outside of this region, it was increasingly less so with defect spikes becoming gradually more indistinguishable from signal background noise. Consequently near-linear time spent outside of this region was not counted.

A measure of performance was then made by calculating the percentage of time that the system was in a "good state". It should be noted that this gave a means of comparing relative performance of different control strategies on the sample surface sets illustrated in figure 5.11 and not an absolute measure of system performance. Also, allowance had to be made for time periods in which there was no surface for the scanner to inspect (i.e. a situation in which a valid signal was impossible to achieve), such as during a break between surfaces.

A second performance measure available involved comparison of the mean scan signal levels. The nearer the mean level to the set-point, the better the system is doing (although, again, there is the problem that different surfaces had different areas that were inspectable, so that results in this format are still only relative).

Finally, a third measure is available in the form of the controller's actuation: the EHT voltage output. While specific levels of voltage serve only as an indication of the amount of light being detected by the photomultiplier, the standard deviation of the EHT voltage can be used as an indication of the spread of EHT voltage activity. A greater standard deviation was indicative of a controller that was spending more time either oscillating around the set-point or constantly hunting for the correct setting. A stable controller on the other hand would have a smaller standard deviation since it would have a better ability to maintain the set-point.

A more substantial comparison of the control experiments was not practical within the resources and time available for the project. This would have necessitated experimentation with

other FastScan systems. The experimental test-bed was constructed because there were no real industrial implementations of the FastScan system in the modality and situation of interest. In any case, the use of a fully operational system was not an available option for the project, even in the basic scenario of inspection of a continuous surface under realistic industrial conditions (e.g. inspection of float glass production).

The results of the experiments detailed in the previous section are summarised using the above three measures in the graphs of figure 5.12. The data represent a period of 60 minutes run time for each experimental configuration. It can be seen from the comparatively large success rate values that in two of the four tests controller configuration C was the most effective. The instances in which it was bettered were those in which a continuous surface (surfaces 1 and 2) was present. There, the constant use of the linear controller in the form of the dedicated *NearLin* AFSM was by far the most stable. In all cases the C configuration had a more stable performance than that of configuration B; indeed, system B can be seen to be significantly worse off in the case of the continuous single-surface test.

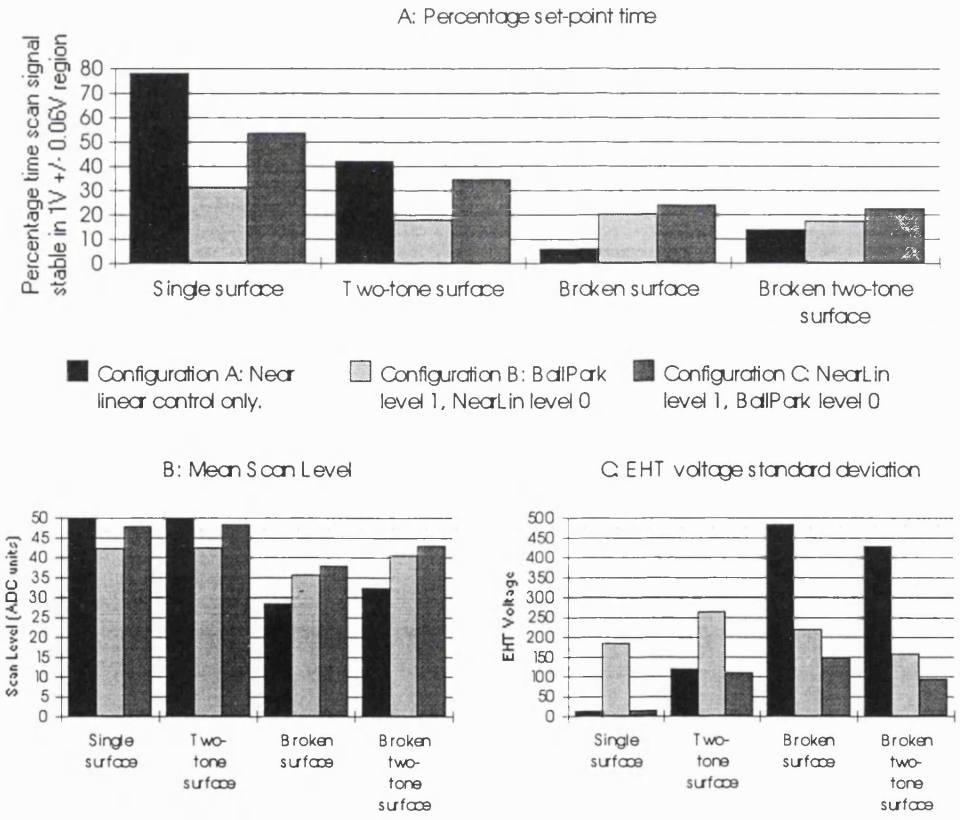


Figure 5.12. Bar charts showing the results of the four experimental runs of the varying controller configurations. In each case the data cover the full 60 minute period of each test run. A: Percentage of time during which the set-point was achieved +/- 0.06V; B: Mean scan level; C: Standard deviation of EHT voltage output.

Because the test surfaces were not matched in any way other than by the desire to provide an intuitively sufficient set of test situations, any comparison between performance is largely subjective. The number of factors contributing to the system's performance is in this case too large for any concise quantitative analysis. What can be said results from examination of the statistical information given in the charts in figure 5.12 and from observations made during the test runs. Plots of run-time system state are used in the following text to provide indications of scan level and EHT voltage output in the form of a state space for the EHT voltage-scan levels. Where useful, the *BallPark* AFSM state and *NearLin* AFSM state are given as time series plots. For these it is necessary that the reader interpret the pictorial representation of the characteristic activity of a system. Unfortunately space is sufficient here to show only plots that illustrate specific points made rather than a complete record of the events. The following sections discuss the results of each set of experiments in more detail.

5.6.1. Controller Configuration A: Single Layer Linear Control

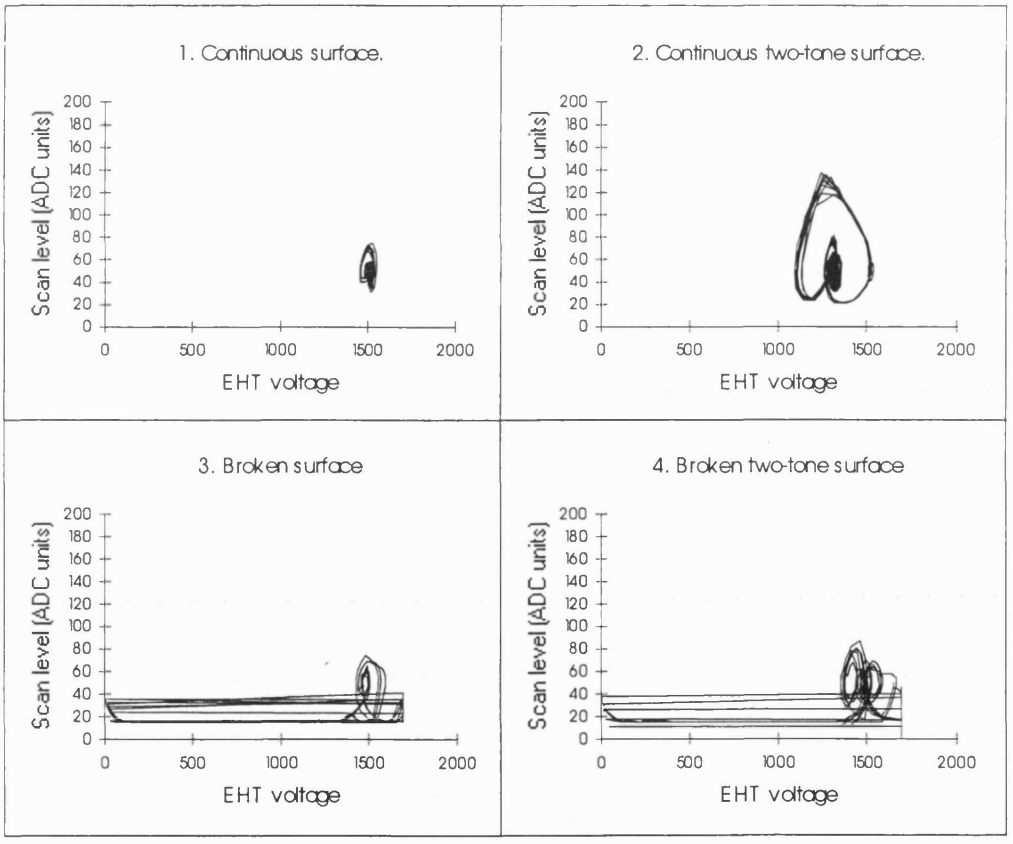


Figure 5.13. Scan-level / EHT-voltage state space plots over the full test period for control architecture configuration A (*NearLin* AFSM only).

This control solution was included in these experiments to provide a reference for the performance of the multilayer subsumption control structures. Although this photomultiplier controller was implemented using the subsumption architecture, it is only a single-layer strategy. As such, it had to be tailored to function correctly in the given environment. From figure 5.12 it

can be seen that in the continuous-surface test this controller provided by far the most stable photomultiplier output signal. This controller also provided the most stable control of the multi-surface sample although it was only more effective than controller C by a factor of about 7 percent. This multi-surface success may be attributed to the relative similarity of the two surfaces used and to the fact that the near-linear region EHT voltage level range for the two overlapped somewhat. For both the broken-surface tests this controller provided the lowest performance solution with a particularly poor result in the case of test 3.

A particularly interesting result was the apparent improvement in performance for the broken multi-tone surface test. It is however possible, on examination of the surface map in figure 5.11, to attribute this to the small size of the broken non-reflecting surface. In the case of this single-layer control there was not a rapid reaction to ramp up the EHT voltage in response to the lack of signal. The result was that by the time the surface had reappeared the signal was still in the near-linear region for that surface type and so control could continue in a relatively stable way. Had this break been larger, in the manner of those in the broken-surface test 3, then it is quite probable that the performance would have been reduced significantly. This is indicative of the critical aspects of system timing in real-world control. It is the case that if this controller had been even slower, then the scan level signal would have become even more stable for the continuous surface.

Run-time scan-level / EHT-voltage state space plots are shown in figure 5.13. It can be seen for the continuous surface in test 1 that the controller copes well. The set-point of 50 ADC units is maintained with an EHT voltage of around 1500. What little variation there is, is due to dirt and distortion of the surface along with probable environmental effects (such as the presence/absence of sunlight). The second plot shows the activity of the controller for the two-tone surface. It can be seen here that the controller was considerably more active. The main part of the surface was of the same material as surface 1, but the second section was of a highly reflective material that had a tendency to scatter the light. The left curve, corresponding to a drop in EHT voltage, is correlated with this surface. As the scanner moves from one to the other the controller has to re-adapt. This is seen in the curve to the right where the EHT voltage increases. The behaviour is effectively an overshoot of the required set-point.

The lower two plots in figure 5.13 show situations that include broken-surfaces. In plot 3 the characteristic movement around the set-point is distorted by the controller's unstable response to the disappearing surface. What is happening is this. When the surface disappears the controller ramps up the EHT voltage until either the surface returns and the signal goes into fold-back or a hardware EHT voltage limit is reached (hence the rightmost vertical straight line in the plot at around 1700 volts). If the maximum EHT voltage is reached, the action of the EHTOutput AFSM resets the voltage level to 0 volts in order to protect the photomultiplier. This can be seen as the rapid transition of the trace to 0 volts and a scan signal level of approximately 20 ADC

units. The plot for the broken two-tone surface in test 4 is similar but the characteristic two-tone transition loop from test 2 is also observable.

5.6.2. Controller Configuration B: Two Layers, NearLin 0, BallPark 1

This controller presented an intuitive extension to the single subsumption layer of controller A. The idea was that the *BallPark* layer would add competence to the system in its ability to deal with disappearing surfaces by subsuming the lower-level *NearLin* AFSM when the photomultiplier signal characteristic became unstable. This control solution proved to be, perhaps surprisingly, unstable on the unbroken surfaces. It would seem to be reasonable to expect this control structure simply to add the corrective behaviour of the *BallPark* AFSM as and when necessary. However, this was obviously not the case. It seemed that sporadic coarse adjustments from the *BallPark* AFSM tended to disturb the smooth control of *NearLin* at inopportune moments. This can be attributed to the fact that identifying the onset of a real change in signal characteristic was not a trivial problem to solve. Distinguishing between changes in signal quality resulting from the system's internal actuation and external environmental perturbations is not easy when no direct reference is available.

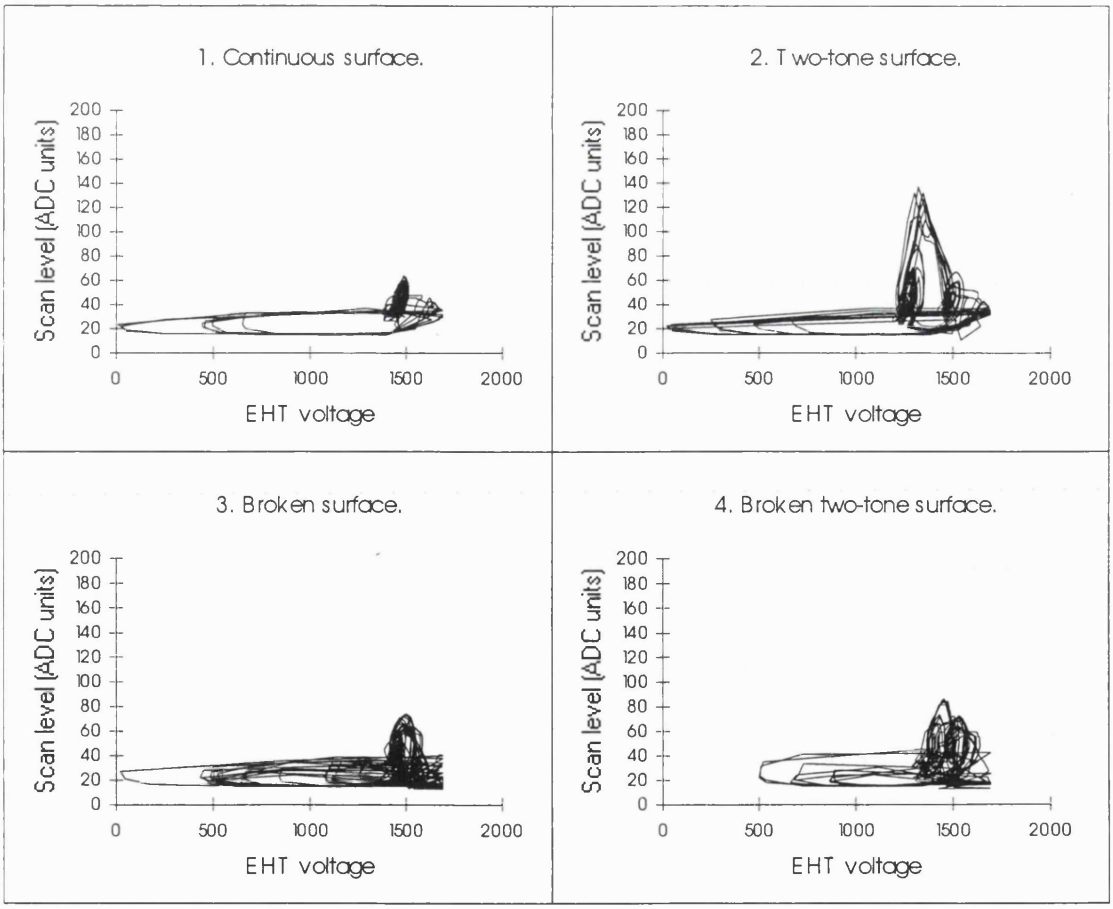


Figure 5.14. Scan-level / EHT-voltage state space plots over the full test period for control architecture configuration B, *NearLin* AFSM level 0 and *BallPark* AFSM level 1.

It can be seen from the AFSM state space plots in figure 5.14 that for the continuous-surface tests 1 and 2 this control structure was considerably more unstable than that control configuration A (*NearLin* only). The characteristic state-space plot typical of the broken-surface specimens in the first test appeared in all results here. This can be attributed to two things. Firstly, the join in the surface that appeared once in every rotation and other transitive environmental effects were sufficient to trigger the *BallPark* AFSM and secondly, once triggered, the coarse control output of the *BallPark* AFSM suppressed the smooth actuation of the *NearLin* AFSM which then tended to upset the overall stability of the system by driving the photomultiplier into one of the fold-back states. The fold-back-B state resulted in a large reduction in EHT voltage which in turn necessitated a reacquisition of the near-linear state from the quiescent.

5.6.3. Controller Configuration C: Two Layers, BallPark 0, NearLin 1

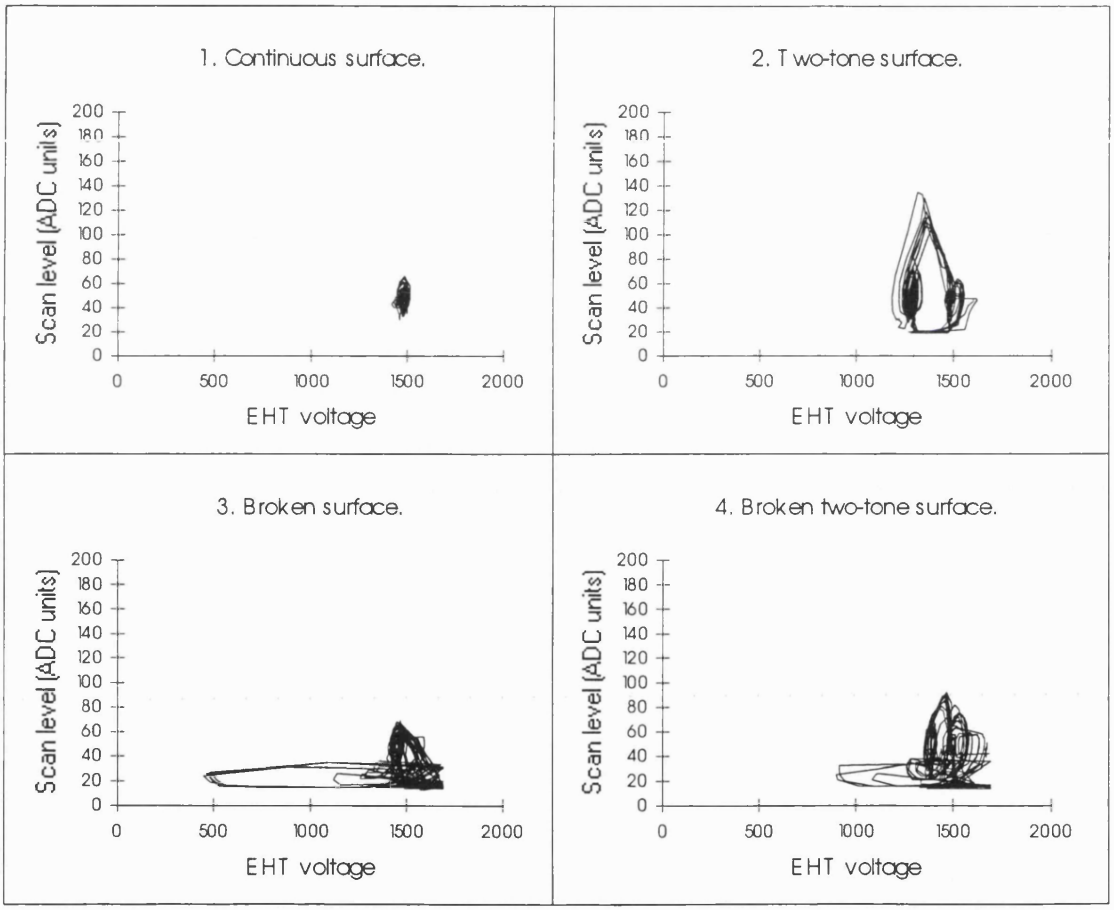


Figure 5.15. Scan-level / EHT-voltage state space plots over the full test period for control architecture configuration C, *BallPark* AFSM level 0 and *NearLin* AFSM level 1.

This controller configuration was perhaps the best compromise solution to the general photomultiplier control problem as explored in these experiments. This is especially the case when it is remembered that the percentage performances shown in figure 5.12 chart A were only approximations of a "good signal" and that in fact at times the signal was often of adequate

stability, being sufficiently close to the declared "good region". This was evident from observation of the system during run-time as can be seen from the plots for the corresponding test run in figure 5.15. This subsumption configuration performed at least adequately in the continuous-surface tests and provided the most stable response for the broken-surface. It is interesting to note the improvement in performance from the relatively simple reversal of controller behaviour levels from that of the configuration B controller (this being achieved easily within the subsumption design framework).

It would appear from examination of the AFSM state transitions that the low-level use of the *BallPark* AFSM was more appropriate than that of the *NearLin* AFSM. This is intuitively correct since as already mentioned above, the nature of the two control outputs is best suited for this configuration. Allowing the smooth control behaviour of the *NearLin* AFSM to subsume the coarse action behaviour of *BallPark* means that if the photomultiplier state is in doubt (often the near-linear state is difficult to differentiate from the fold-back-A state) the *NearLin* action will prevail, effectively damping out by suppression the large-scale actions that lead to signal loss and instability.

5.7. Discussion and Conclusions

This chapter has provided an illustration of the design of a distributed control structure for a multi-component laser-scanning sensor system. It has detailed the aspects of the low-level control of a complex set of loosely-coupled subsystems that have, in their own right, a potential for a limited degree of autonomy. The chapter has not dealt with issues concerning the higher-level functionality of feature detection and response except in the localised and restricted scope of individual processes where this has been relevant. The control structure was used to support a number of experiments that looked at the issues of a bottom-up, layered approach to the development of a behaviour based control strategy for the photomultiplier subsystems of the laser scanner test-bed.

There are two areas of conclusion that can be drawn from the above experiment and its associated work. The first concerns development at an application-specific level of an industrial laser-scanning inspection system while the other leads to a commentary on the use of the behaviour based control methodology. This section will deal mostly with the former while the next chapter will go into greater depth on the more general issues that have arisen as a result of this work and are illustrated by these experiments.

5.7.1. Laser-Scanning Test-Bed Application

The characteristics of behaviour based design can be seen in this system in the form of the layered approach to its development. They can also be seen in the use of AI-type rules in the definitions of the control processes as given in tables 5.5 and 5.6. These rules which are each in

their own right relatively simple, acting only on a small part of the main problem, were used in combination within a particular subsumption layer and within multiple layers in order to achieve a global behaviour of the system. It is this relatively easy process of defining rules for a particular situation covering a small part of the system's behaviour that made the construction and subsequent testing of the test-bed control example relatively straightforward. From this work a number of points present themselves for discussion:

- System design and development
- System structure
- System definition - agent-environment boundaries
- Action selection

System Design and Development

The design and construction of a behaviour based control structure for the laser scanner test-bed posed few (if any) major problems. The development framework offered by the subsumption architecture on the transputer network of processors supported an easy implementation of a multi-process software program. It was particularly the case that the intricate structure of the many communications links typical of such systems was achieved simply, which must be attributed largely to the asynchronous nature of the subsumption AFSM processes. This allowed processes to be coupled without the necessity for specific concern over rendezvous of process states or handshaking-type protocols. It turned out that the natural form of the behaviour based control structure meant that there was little need for concern regarding guarantees of data or information transfer. This is in keeping with the reports in [Brooks86] and [Brooks89]. Interestingly other implementations of subsumption architectures hardly mention this aspect of process timing, for example [Mahadevan & Connell 92], [Jones & Flynn 93] and [Gat *et al* 94], and it is not clear in these reports whether these timing issues were hidden from the design process or simply not addressed. Aspects of system timing are discussed in more detail in the next chapter.

An aspect of the control program's development that did require considerable work was that of setting up the AFSMs to interact in the right way, either with signals from external sensors or from other internal AFSMs. Each AFSM typically utilised a number of critical constant values in the form of state identification thresholds, maximum and minimum ranges of input values and specification of signal gradients as well as other sets of values used to determine an output. It was necessary to tune these parameters incrementally by running the systems and observing the resulting behaviour. Thus the effectiveness of the system was significantly dependent on the knowledge and ability of the designer to specify and develop both the internal structure and rules of an AFSM as well as the network interconnection and interaction of multiples of AFSMs. Some combinations proved to be more effective than others. For example, control configuration C in the

experiments detailed here generally proved to be the best performing. The control structures outlined in this chapter are generally the result of several iterations of program development, each differing as a result of testing the complete system in its intended domain. One advantage of the bottom-up development approach, characteristic of BBAI and which is also evident here, is that the lower layers may be tested first, providing the designer with an opportunity to learn more about the intended interactions and behaviours of the target system. Perhaps the best example of this is in the development of the two photomultiplier control channels. While these each utilised the same basic AFSM structure, the setting of internal constants that enabled a reliable and robust performance differed considerably between the channels. This was a result of variations in hardware components, in this case the photomultiplier tubes themselves. Here, building in the lowest levels of control or device drivers in the same framework as that of the higher more abstract layers (such as in this case the user interface layers) proved to be a useful way of integrating different parts of the system.

In the final respect of system structure there were three distinct aspects of design and construction that seemed particularly complementary when used in combination: (i) the physical transputers and their suitability for relatively simple network interconnection and expansion; (ii) the software process configuration and the transputer process execution environment, and (iii) the use of subsumption AFSM processes as a system-wide design framework.

System Structure: Communal or Unitary System

The resulting subsumption control structure for the laser scanner turned out to be distributed or modular on more than one level of abstraction. For example there were the discrete component modules of the photomultiplier, optics and high-level processor subsystems as well as the lower-level network of AFSM processes. If at the level of hardware module control and interfacing there is one scale of distributed system, there is another at the level of controller modules and then perhaps further ones at yet higher levels of the scanner system in the form of the optical head, the host computer front-end and defect detection system and finally the even higher level of the complete industrial plant. Viewed in a behavioural perspective these component parts can be seen as low-level reflexes of sensor control, higher-level intermediary control, and supervisory system-level control which in this case was provided by operator intervention via the user interface layer. This top layer could equally well have been replaced with a layer providing an interface to other systems in the industrial environment. Each level shows the same distinct characteristics of a distributed system and to some extent aspects of autonomous behaviour can be identified in the component parts. The major differentiating factor of scale is the level of abstraction from the dynamics of the real-world.

Given that the control solution presented in this chapter can be viewed as distributed on several different levels of abstraction, the question must be asked as to whether it is sensible to regard these subsystems as autonomous agents in their own right. [Brooks91] and [Smithers92]

emphasise the need for an agent to be physically embodied and situated in the real-world. This can be said of all levels from the discrete component controllers, in the form of the photomultiplier and optics controllers, upwards through to the discrete "agents" that are the different machines on a production line. Thus it is postulated that the view of a behaviour based system moves from one of a single bottom-up layered system of processes to that of a colony of interacting agent processes with each agent interacting on an autonomous or at least semi-autonomous basis. The issues concerning levels of autonomy and system composition in terms of agent colony versus a single bottom-up layered structure are central to the first part of this thesis and are the main topic of the next chapter. This chapter has served to illustrate the issues in question from the standpoint of a real-world application.

System Boundaries: Where Does The Real-World Begin?

The physical configuration in conjunction with the relatively complex requirements of the laser scanner lead to the implementation of a distributed control system that utilised physically distributed centres of computation to supervise separate subsystem components such as the optics controller and the two photomultiplier controllers. In conjunction with the issues raised in the previous section a related issue also appears in the form of discussion regarding the boundary between the agent and the environment. In many AI-controlled systems such as mobile robots this boundary is clearly identified as being marked by the sensors and actuators of a system or by the limits of software activity in the form of accessing hardware input-output devices ([Langer *et al* 94], [Herman *et al* 90] and [Balch *et al* 95], for example). These kinds of interpretations are not so clearly applicable to the laser scanner test-bed for a number of reasons listed bellow and dealt with in more detail in chapter 6. We suggested that this implementational detail had a significant impact on the view of the system as a whole and in particular its boundaries with the real world.

i) The system is genuinely distributed, so that there is no "central" control provided by software-accessing input-output. Thus the input-output is also distributed and differs from subsystem to subsystem.

ii) The processes of the system deal with input-output not only to local sensors and actuation but also as communication with processes elsewhere in the system.

iii) There is a certain degree of active hardware in the system that performs operations that receive triggering from software. Thus the difference between software and the rest of the hardware is not clearly defined.

iv) The complete test-bed system is, at the top level, a sensor in its own right. It must nevertheless act in its environment in order to function in a useful way. Thus the environment provides a critical loop in the processes of controlling the sensor which are hidden to the higher levels of the industrial inspection system.

v) In the typical way of behaviour based systems, improvements in performance were possible by making adjustments not only to the "controller" itself but in fact to any appropriate part of the system. A crude but relevant example of this can be seen in the adjustment of ambient lighting conditions so that the laser light was not completely saturated by other light sources.

From these points it is evident that an agent-environment boundary is not a hard and fast identifiable feature and in general would seem to be more often used as an arbitrary division of convenience. While it might be possible to create such an artificial division in more simple automaton-like robot experiments, in more complex systems such as this industrial system it is not. If anything, it might add a degree of perhaps confusing complexity to the control problem. The control problem in this light changes from one of attempting to control the agent or environment explicitly to that of setting up a complete agent-environment system to function in a desired way. At the low levels of photomultiplier control the environment provides the basis for action in the form of altering EHT voltage levels. If the EHT voltage were not constantly adapted, then the sensor would become unstable due to its loss of coherence with the environment. At the higher level of the inspection system it is possible to view the laser-scanning system as a sensor which generates a signal that is indicative of the amount of laser light reflected by a surface under inspection. Here again we are forced to return to the view of a system in terms of levels of abstraction and recognise that the agent-environment boundary, if this is indeed a helpful construct to use, is not a clear-cut border of activity but varies at different levels of system interaction. The ideas expressed here constitute a fundamental aspect of the first part of the thesis and as such are dealt with in greater detail in the next chapter.

Doing the Right Thing at the Right Time

From the experiments conducted with the laser scanner control structure developed in this chapter, it was evident that the use of a simple single layer structure in the form of the *NearLin* AFSM, with its linear control response, was considerably more effective in the particular case of a continuous surface with small or no characteristic changes (as in surfaces A and B). However, the performance of this configuration deteriorated massively (by a factor of up to 90%) for the broken-surfaces C and D. With the addition of the *BallPark* AFSM (either as a level 1 activity or the more successful level 0) the system's general performance over the wider range of conditions improved, although in the specific case of the continuous surface there was a performance reduction. It is clear that effort spent in better defining the periods of actuation of these AFSMs would be well directed. In particular, if the *NearLin* AFSM is so effective on continuous surfaces, why not redesign the system to identify the ongoing situation better so that the *BallPark* AFSM does not interject its actions when they are unwelcome? There are two issues here: (i) that the asynchronous nature of the subsumption control structure (especially between AFSMs on different transputers) can result in sporadic and indeterminate interaction between AFSMs and

(ii) that the differentiation between a surface that has disappeared momentarily and one that has changed radically in reflectivity is not a simple problem to solve.

The first of the two issues highlighted above concerning the asynchronous nature of the subsumption architecture is really more indicative of the nature of behaviour based system design and the implementation of larger and more complex agents. The benefits of an inherently asynchronous set of processes are many. For example, the asynchronous communications of the subsumption architecture are cheaply implemented, and there is no need for costly (in terms of time and computational resources) mechanisms of synchronisation such as clocking, handshaking or rendezvous, i.e. mechanisms that can lead to a system being overly brittle. Robustness and utilisation of redundancy is more complicated to achieve in a system in which component or subsystem failure results in deadlocked processes. A chain of events can then lead to the halting of the entire system. Even the benefits of implementing a synchronous set of processes tend to be lost when the non-software aspects of the agent-environment system are also considered. The subset of software processes that is part of the total set of agent-environment processes, must interact with an asynchronous real-world. Consequently different parts of the agent must function at different rates. If they are designed to be asynchronous in the first place, then requirements and procedures for dealing with a lack of data become embedded in the dynamics of the system. However, such systems are not completely without synchronisation; instead it is provided by the movement, transfer and interaction of data within the system as a whole, ultimately being triggered by the agent's interaction with its environment. If there is data then the process acts, if not it doesn't (and can't anyway) and the fact that it doesn't should not result in the rest if the system going into deadlock. Problems of synchronisation magnify as systems become larger and more complex. It would seem that methods and techniques that facilitate the construction of asynchronous systems would lead to a useful increase in built-in robustness. These issues are taken further in the second part of chapter 6.

The second problem suggested above, that the recognition of system state is a far from trivial problem to solve, is in fact one inherent in many AI systems and can be traced back to the frame problem [Hayes79], [Minsky81] and [Pfeifer95]. The identification of the different photomultiplier states in the control structures detailed here relied on the explicit declarations of a number of simple rules that were built into the AFSMs, but there are always exceptions to the rules that cause instability when they are encountered. This hand-crafting of details is a major bottleneck in the design of complex systems and is increasingly being recognised as an important area in need of research attention. For example, the scan-line pixels read in by the ScanAquire AFSM provide a massive amount of information but in a form that would be very costly to classify into rule sets by hand. This problem, as encountered in the behaviour based control of the present laser scanner system, is the focus of the second part of this thesis and is taken further in chapter 7.

Perhaps the solution to this problem of dealing with exceptions and out-of-the-ordinary situations should be application-specific at the level of development in individual subsystems. In other words perhaps a particular subsystem should be tuned to suit its particular operational domain. This, is the current state of affairs with this laser-scanning system and BBAI in general. One of the objectives of this research has been to examine techniques of increasing the flexibility and robustness of these systems so that they may function in more diverse environments. In the flavour of BBAI we should analyse the environmental situation of this laser-scanning agent and then decide whether it will encounter many breaks or large changes in surface. If not, then the system may be tuned explicitly for this restricted domain. Extra layers of control are not required and only serve to reduce performance and add complexity. The point that emerges from such considerations is that the complexity of the agent control strategy needs to be matched with the complexity of the environment ([Pfeifer93] and [Pfeifer96]). Obviously a mismatch in either direction is undesirable and will most probably result in an ineffective agent-environment system.

5.7.2. Conclusions

A number of aspects have emerged as a result of the work with the laser scanner test-bed that is illustrated in this chapter. They highlight some significant points regarding aspects of both behaviour based control (and BBAI) and also of the general architecture and implementation of real-world situated distributed systems.

In section 5.7.1 above, matters concerning the structure of the laser scanner test-bed control were discussed in the perspective of our view of the system as a collection of semi-autonomous processes within the body of the laser scanner. A hierarchical view of a behaviour based system was suggested that began at the lowest levels of agent "physiology" and continued upwards towards a community level of physical agents. As a result of this work which deals with a truly distributed system it has become apparent that the simple view of an agent as being a system of processes physically embodied in the real-world is only part of the story concerning the structure of a real-world agent system. The work in this respect has led to the ideas that make up the first of the two parts of the present thesis. The next chapter discusses these issues in detail.

The second part of the thesis concerns the nature of the design process for the laser-scanning test-bed system in particular and behaviour based systems in general. During the construction of the system and the implementation of the experiments reported in this chapter, it became apparent that the success of the target system was largely a result of the skill and ability of the designer to assemble a suitable agent structure. In the case of this work the structure mostly concerned the programming of transputers, but in many behaviour based systems this work would include the design of physical morphology of the agent as well. In the case of the laser scanner, standard stock component parts were mostly used. This designer input is built into all aspects of the system and as such manifests itself as inbuilt or pre-programmed designer knowledge of the domain which is most often the critical differentiating factor between system success and failure.

As system requirements become more complex in order to deal with more complex task and environment situations it will become increasingly difficult, if not impossible, for a designer to provide (let alone pre-specify) the complete set of domain knowledge required to make a system function. The second part of this thesis, beginning in chapter 7, introduces these ideas in more detail and leads on to a further set of experiments in which ways of enabling an agent system to learn and adapt to ongoing environmental changes and situations are presented in the light of the results that we have detailed here and more extensively in chapter 6.

6. Distributed Control and Real-World Multi-Agent Interaction Systems

This chapter covers the following:

Summary of BBAI.

A new lower level of behaviour based control.

A multi-agent view of each control level in a behaviour based system.

An introduction to the second part of the thesis.

Now that we have introduced the field of behaviour based control and presented an application-oriented implementation of a real-world system that uses the techniques to manage the system's interactions, this chapter turns to discussion of some of the issues raised in the course of that work. From the specific results and conclusions that arose from the implementation of the distributed laser-scanning system, this chapter attempts to provide some more general comment on how these factors might affect or contribute towards the future of autonomous systems research and the field of behaviour based AI.

A Recap of BBAI and the Work so Far

In chapter 3 some background to behaviour based artificial intelligence (BBAI) was given. A number of key aspects were presented in the form of situatedness, embodiment, groundedness, non-symbolic activation patterns, distributedness and emergence. These along with the important bottom-up approach to research and development form a field of study that crosses many disciplines from anatomy and neuroscience to mechanical and electronic engineering. This mix of research has resulted in many examples of physical realisations of agent-environment interaction systems that demonstrate the fundamental characteristics of behaviour based systems; that an agent is not taken as a discrete system separately from its environment but rather as a part of a whole system that is viewed as being an intimate coupling between two sets of agent and environmental processes [Beer94], [Smithers94b]. This pattern is continued with research and development on an upward path of abstraction from mechanisms and structural components, to processes and process structure, to theories of agent competencies and lastly to a global system view of task and environment [Smithers92] and [Steels95]. Intelligence is not regarded as a specific or quantifiable attribute of an agent but is seen to emerge as a result of the interaction of the agent and its environment together. In this way the folk psychology ([Churchland95]) of everyday concepts such as "thinking", "remembering", "planning", "wanting", "fearing" etc. is

also side-stepped. A synthetic agent must be built up with all aspects of behaviour completely embedded in its mechanism and not bolted on top as and when convenient (which is seen as typical of classical AI and control).

Chapter 4 presented a real-world test-bed that had been built to experiment with the implementation of behaviour based control techniques in a somewhat unique domain: that of industrial production line inspection. The system was shown to be a suitable target for the methodologies of behavioural control inasmuch as it provided a complex non-linear task realisation problem that was situated in a real-world environment. It was stressed that, despite the fact that it was not a mobile or self-sufficient system in the manner of a mobile robot, the test-bed still provided a valid problem domain which in fact had several advantages over the normal target of mobile robotic systems. An important feature of the test-bed was that it provided a basis for experimentation with a system that was genuinely distributed both in terms of physical functionality and computational resource. The development framework of the Subsumption Architecture [Brooks86] and the behaviour language [Brooks90b] was overviewed and its mapping onto a multiprocessor transputer network detailed. Aspects of timing and the asynchronous nature of the subsumption processes in the form of augmented finite state machines (AFSMs) was outlined.

BBAI and the Test-Bed Experiments

The last chapter detailed the development of a subsumption control structure for the complete scanner sensor system. Due to the distributed nature of the target system it was necessary to split the control structure not only in terms of behavioural layers but also in terms of local requirements for sensation and actuation. The result was a subsumption architecture that was genuinely distributed across a multiprocessor system with collections of AFSM processes making up "behavioural nodes" on the transputer processors. The characteristics of this control implementation meet the characteristics of BBAI, as outlined in chapter 3, in the following ways:

Situated: The laser scanner was situated in a real-world environment, that of an industrial production line.

Embodied: The laser scanner system was a physical system that experienced its world through its sensors and actuators, in this case the digital array of the scan stripe signal from a photomultiplier. Its actuation was in the form of adjustment to the EHT voltage and thus effectively the gain of the photomultiplier.

Distributed: The distributed nature of the laser scanner has been emphasised already.

Non-Symbolic: There was no symbolic manipulation in any of the modules of the control implementation described in chapter 5. In fact it is hard to see how or if such symbolic mechanisms could have been used with any benefit.

Emergence: The behaviour of the laser scanner front-end emerges as a result of the interactions among all the AFSM processes. Although there has been discussion in terms of the emergence of intelligence, it is clear that such a description has not been really useful in the context of the low-level embedded control of this sensory system.

Groundedness: The activities of the system are based on the close coupling of the real-world and internal system state and are thus truly grounded.

The simple experiments described in chapter 5 served really only as a means to illustrate the nature of the multilayer control architecture. A complete inspection system would necessitate the implementation of considerably more complex structures than the low levels presented here.

This Chapter

The main focus of this chapter has two main components. Firstly the issues regarding the nature of the bottom-up research and development methodology of BBAI are discussed and results from the laser scanner implementation are drawn upon to illustrate an apparent need for a lower level of starting point, one that addresses aspects of adaptive sensation and actuator control on a behavioural level below what is usually seen as the main agent-environment behaviour-generating system. The second part of this discussion then continues from the first by turning to the issues of the distribution of behaviour based systems and subsystems, bringing in consideration of the physical location of processes and the abstraction of their interactions from the domain of the physical world.

6.1. Bottom-Up Development From the Real Bottom

The development framework provided by the transputer network, the transputer run-time execution environment and the subsumption architecture implementation enabled the relatively straightforward construction and development of a distributed subsumption control network. Boundaries between actual processors were hidden by the inter-AFSM wire protocol of the subsumption architecture which was implemented using virtual transputer links (high-priority processes that multiplex and demultiplex software links onto transputer hardware). It was thus a relatively simple task to embed AFSM processes at all levels of system control from dedicated hardware controllers to general-purpose computational resources (for example, the T405 transputer on the laser scanner test-bed). Consequently it was possible to utilise the full computational resources of the system in the most efficient way with potential bottlenecks in the physical communications links being avoided through careful AFSM process distribution. This resulted in a number of critical factors that are discussed in this section. One such is the ability to speed up reflexive actuation by locating complete control loops locally on processors that were nearest to the physical location of the respective system-environment activity, thereby reducing or eliminating propagation delays between processors. From examination of the literature it would seem that BBAI research into mobile robotics has not yet fully recognised or accepted the

importance and necessity of maintaining a close mapping between program structure and robot-agent physical structure (which includes electronic hardware and robot morphology).

6.1.1. Early BBAI Systems

From the early examples of behaviour based mobile robot test-beds, which tended to consist of simple and single control processors and binary state sensors, there has been a steady increase in complexity of hardware resources towards multiprocessor multistate sensor systems. The early examples were typified by those reported in [Donnet & Smithers 90] and [Brooks86]. They consisted of easily-obtainable low-cost components, typically 8-bit microprocessors running at rates of a few megahertz with a few kilobytes of memory. These computational resources nevertheless proved completely adequate for the sensation and actuation faculties of the test-bed robots which were in general of a binary signal/command type. Examples are bump sensor switches, or infra-red sensors with a preset hardware threshold that enables the generation of a single-bit binary value whose state is dependent on the absence or presence of incoming infra-red light. This first generation of test-bed robots were and are still used in impressive demonstrations of behaviour in normal unstructured environments such as offices and workshops. It is particularly interesting to note the simplicity and performance of these systems compared to those that utilise the techniques of classical AI such as [Nilsson84], [Simmons94], and the distinctly non-behaviour based [Langer *et al* 94]. These simple mobile vehicles have demonstrated fast, reliable and robust navigation of their environments including obstacle avoidance behaviours, taxis behaviours (such as phototaxis: homing in on a light source) and in some cases simple object manipulation ([Connell89] and [Pebody91]). It must be pointed out, however, that impressive examples of fully autonomous mobile vehicles using more classical AI and control techniques have been developed, for example [Dickmanns & Muller 96] and [Jochem & Baluja 93] describe road following autonomous vans, and [Schell & Dickmanns 94] an autonomous aircraft landing system that uses visual flight queues. These examples are also extremely complex, Jochem and Baluja discuss the issues of using massively parallel computer architectures to implement machine vision. This contrasts to the low computational power of the BBAI approach which although still in its early days has stimulated a wide-ranging effort to develop a new generation of mobile robot test-beds using more up-to-date, but still (in comparison) simple technology.

BBAI research is now tending towards the utilisation of a new generation of robot test-beds which incorporate impressive ranges of sensors and sensor types that provide a rich range of views of the real-world. Analogue or multi-bit binary signals are used both as input from sensors and to drive actuators. Along with this increase in spatial resolution of the agent's world more powerful processing resources have been provided, including an increase in available memory. This increased memory has in turn opened up a new dimension in sensory resolution: that of the temporal domain in the form of stored sensor, actuator and internal states. While aspects of the

time dimension have been used in the past, the resources of mobile robots (of the untethered variety) have always been significantly limited in capacity. The limited resources of the first generation of robot test-beds provided a clear demonstration of the levels of behaviour that were achievable but it was (and still is) anticipated that the new generation of vehicles would offer significant improvements.

6.1.2. Balancing System Resources

The increased sophistication of sensor and actuator devices has led to the usage of dedicated embedded control processors employed in various combinations. Examples occur in the robots used by [Smithers94], [Ferrell94], [Nehmzov & McGonigle 94], [Nourbakhsh *et al* 95] and [Balch *et al* 95], to list a small sample. The expectation seems to be generally that the same mechanisms and structures that were used on the first generation of test vehicles can be generally expanded and scaled up to provide a basis for more complex tasks, environments and thus behaviours. However, this has not generally been the evident result, as can be seen by comparing the robotics sections of both the artificial life and simulation of adaptive behaviour conferences series over the last five years or so ([Varela *et al* 91], [Meyer *et al* 93], [Cliff *et al* 94], [Morán *et al* 95]). The demonstrations typically consist of some form and combination of obstacle avoidance, exploration, taxis and object manipulation. From observations of robot demonstrations it is not clear that there has been any significant improvement in performance despite the developments in new and "better" technology. Although however, the work reported in [Horswill93] describing the robot Polly that is able to give laboratory and office tours is a clear exception to this trend.

This problem of better robots making things worse is well illustrated in a paper by a similar name [Smithers94] and also in other work: [Miller93] and [Mason93]. All discuss issues that depend on the quantity and nature of sensor devices in mobile robots. Of course the question of what constitutes a "better" sensor or actuator is totally application-dependent. This is in part the reason for the success of behaviour based control, where robots are given a set of sensors, actuators, interaction mapping mechanism and environment task domain that are all matched in a specific balance of resources and cost to achieve a particular reliability and robustness of behaviour. Adding more complex sensors and/or actuators to this set does not necessarily contribute to an improvement in a system's performance. It can in fact make things more difficult to manage since there is a large increase in the number of critical variables, internal states and degrees of freedom of the system.

6.1.3. Ideal Sensors and Actuators

The binary sensors and actuators of the early behaviour based robots are in effect realisations of virtually ideal sensors. Their relative simplicity allows a better understanding of their behaviour in which, at any particular instant in time, they present and act on a simple two-state true-false

indication. The considerable advantage of this arrangement is that the designer is able to categorise clearly and concisely sensor-actuator mappings so that little more than a simple rule is involved in the agent's control mechanism. There is a complete dependence on the designer's ability to understand the agent-environment problem domain and provide the right sensor-actuator mapping mechanisms. Although this simplistic view is radically complicated when the changes of sensor state over time are taken into consideration, it is still the case that any sensor with more than two states will require more internal mechanism for translating the signals into some form of suitable response actuation. The BBAI approach to sensing and control utilises the skill of the designer to make use of particular dynamics of environment-sensor-actuator characteristics in defining the internal mechanism of an agent [Horswill93]. The designer's knowledge is embedded in the agent control structure in the form of "hardwired" sensor-actuator responses. In the relatively simple first generation of robot test-beds this was an almost trivial task; in the second generation it is at best manageable and at worst impossible. Clearly any third-generation development is going to require significantly different emphasis at all levels of construction.

6.1.4. Increasing Sensory Resolution and Behavioural Repertoire

Sensory resolution can be enhanced in a number of ways: individual binary sensors can be replaced with analogue sensors, an increased number of binary sensors can be used, an increased number of analogue sensors can be used or the temporal dimension may be opened up by sampling one or many sensors over time. All these "improvements" dramatically increase the number of possible states that the system's control mechanism has to deal with. This is the mechanism that has to be in place before any form of behaviour realisation can be implemented. The result is a requirement for an increased computational resource which, in many cases, effectively provides a mapping of complex sensory input into a simple binary decision about a possible action. This increased computational loading is currently taken care of by the more powerful processors used on the new-generation robots. However, even if the designer is able to provide a suitable mechanism, this only maintains the level of behaviour at that previously achieved by first-generation vehicles. Much effort is directed towards handling the now massive amounts of sensor state data. To match the increase in sensory and actuation capabilities, considerable computational resources have been required to match the behavioural levels of the first generation of robot test-beds. In order to increase the behavioural repertoire, there would seem to be a requirement for yet further increases in processing power. Clearly the addition of sensory resolution (both spatial and temporal) and the further development of behavioural competence of mobile robots is desirable in order to understand further and design more complex and sophisticated systems. However, the way in which these test-bed systems are expanded and enhanced (both physically, in terms of hardware, and in terms of control structure) would seem to be of fundamental importance. This was demonstrated by the addition of the *BallPark* behaviour

to the laser scanner control system reported in chapter 5, where even the simple reversal of AFSM suppression had significant effects on performance.

The current tendency in dealing with the hardware implementation of enhanced sensor and actuator provision is to pre-program embedded microcontrollers to handle more complex sensors and actuators at a level below and hidden from that of the "real" AI control mechanism which remains located in a central processing unit (for example see [Ferrell94], [Steels94], [Smithers94] [Vereertbrugghen93] [Kweon *et al* 92] and [Ruyssinck92]). Typically commands are sent down a communications link to a low-level slave processor which responds with a simple execution of the "master's" request. The benefit of this form of modular arrangement is that off-the-shelf components can often be used, which substantially reduces the cost and work involved in developing and building a system. This is a good enough reason for such solutions in many application-oriented projects, but in the case of the construction of test-beds for behaviour based robotic research it has resulted in what is perhaps the misguided encouragement of a view that sees sensors and actuators as nothing more than input-output interfaces with the real world. While it can be argued that this view might be satisfactory in the case of binary sensors and actuators, it would appear that this is only because it is possible to take short cuts with these limited devices and get away with much of the baggage of real-world noise and indeterminacy. The lesson to be learned is apparently that there is much more to sensing and acting than issuing requests and commands to a remote component from a higher-level actuation realisation mechanism that is generally the preferred domain of AI research interest. This resultant "enhanced behaviour based" hardware architecture does not conform to the real behaviour based approach as the component modules are based functionally rather than behaviourally.

6.1.5. Inter-Process Timing

In addition to the above argument, and perhaps in continuation, there are more practical problems that result from the master-slave(s) approach to robot hardware configuration. The relegation of sensors and actuators to low-level slave control on embedded processor subsystems has an inherent potential for introducing significant propagation delays between the sensing of a stimulus and the actuation of a response. The fast *stimulus→response* cycle characteristic of the tight control loops of behaviour based architectures is lost to a more lengthy *request data→receive data→generate response→send command* loop. It is the treatment of these devices as being separate from the AI control problem that leads to these difficulties

This argument is the same as or similar to that presented in [Brooks86]. However, the message of that paper seems to have been somewhat obscured recently by the continuing application of more complex robot control systems. Here I refer to the nature of truly distributed control on networks of physically separate microprocessor components. It is no longer practical or sufficient to restrict implementation of control to a single central processing unit that is able to "read" and "write" to the real world; rather, the implementation of AI control structures must

additionally encompass the aspects of sensor and actuator dynamics, with behaviour realisation being physically distributed across the agent. For example, fast reflex responses may be implemented directly on low-level processors while actions with more complex constraints and with less need for speed can be implemented on higher-level hardware. This was clearly demonstrated in the laser scanner test-bed work of the previous chapter, with the reflex responses of the photomultiplier controllers being built into the lowest levels of the subsumption control structure and consequently located in the lowest-level transputers which were embedded locally in the relevant electronic circuitry.

The world is not static. Much effort is invested in providing AI control systems with increased ability to cope with unexpected events in reliable and robust ways by adapting to ongoing changes in situation. However, it would seem that the scope of dynamic adaptation of sensor and actuation control at lower levels is largely ignored. Work that does deal with adaptation is typified by [Clark *et al* 92]. This lower physical level of detail is often argued to belong to a more application-specific engineering domain rather than being a relevant part of mainstream AI. While it was possible to do AI-type research with the simple systems that incorporated binary state sensors and actuators, the new state of affairs is vastly more complex. The use of discrete controller modules for sensors and actuators in itself seemingly suggests that low-level adaptability and localised action are possible and necessary. The provision of more complex sensors and actuators seems to be taken for granted in the AI field and much work concentrates on using these "powerful" new systems for more complex and interesting behaviour realisations - with often disappointing results. It was suggested in [Smithers94] that this disappointment is often being blamed on the "lack of suitable technology". The methodology of BBAI should not become entangled in this problem, which is really not relevant if the working characterisations of the field concerning distributedness, situatedness and groundedness of behaviour based systems are remembered. The solution, we suggest, is to be found in a more thorough adherence to a bottom-up development strategy.

6.1.6. Lowering the Bottom of Behaviour Based Systems

It is perhaps possible to conclude that the problem discussed here is in fact one of scale. Having increased the sensory and actuation resolution at the physical interface level of a robot it is not only a case of using these new facilities at higher behavioural levels, but it is also necessary to step down a level and provide adaptive control of subsystems. With the first generation of test-bed robots the low-level binary mechanism was relatively simply hardwired or prespecified by the designer. This is no longer possible. it is unrealistic to expect designers to be able to foresee all eventualities and preset such a complex array of devices. In the final analysis these new "better" systems have a lower-level "bottom" than their simple ancestors. It is suggested, as a result of the work reported here, that the starting point for the bottom-up research and development that is

necessary for a behaviour based approach to systems development has moved. It has gone down a level.

6.2. Distributed Systems of Layered Agent Processes

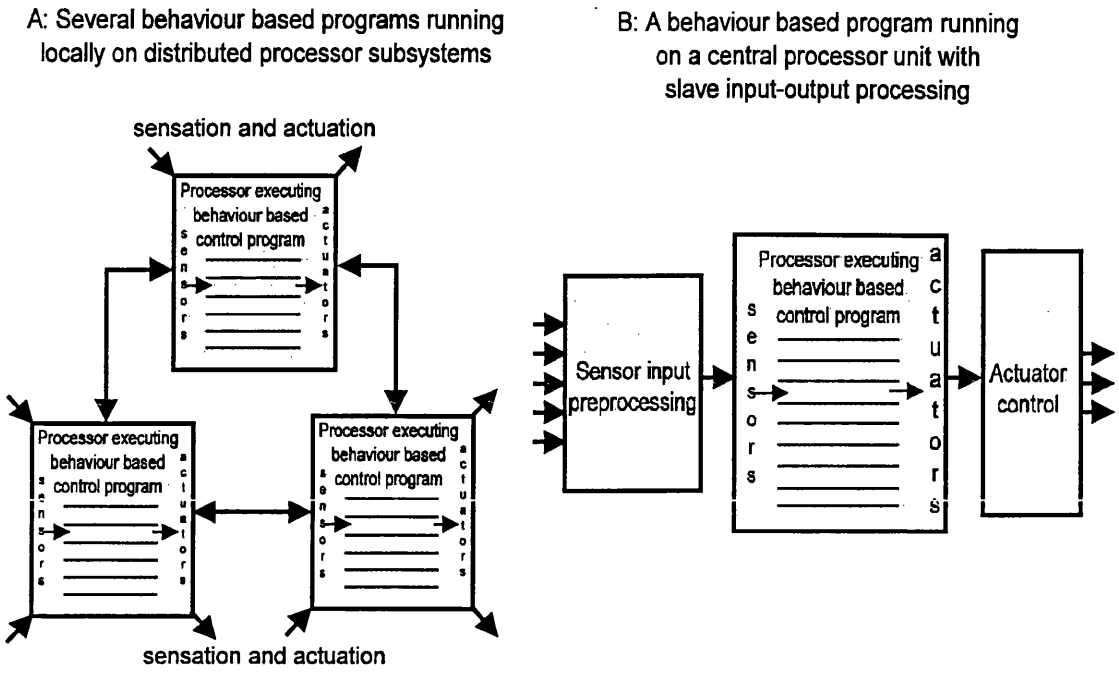


Figure 6.1. A real and a pretend behaviour based control structure using multiple processes. A: the extra controllers are incorporated into the behaviour based structure and B: a classical pipeline is formed by localised sensor and actuator processors being used as "invisible" control devices.

The previous section outlined the idea that the increased sophistication of recent development in behaviour based robot test-beds has necessitated a reassessment of the starting point in the bottom-up development and research of autonomous systems. While the early use of binary system state precluded the need for extensive sensory manipulation and adaptation, the increasingly complex analogue devices now used are often being controlled locally by dedicated and embedded processors ([Ferrell94], [Steels94], [Smithers94] [Vereertbrugghen93] [Kweon *et al* 92] and [Ruyssinck92]). This development has led to an architecture that effectively reintroduces the functional pipeline of classical systems (see figure 6.1b) as opposed to the proven parallel structures of true behaviour based systems (figure 6.1a). This difference is significant and the philosophies of the behaviour based approach that highlight the importance of the complete agent-environment system would bear this out. It is the case that even if the "central" processing block of figure 6.1b is executing a behaviour based control program (such as something with a subsumption structure) the physical realisation of this agent-environment system is not truly parallel. Moreover, the timing of such pipelines, which can have a critical effect on the behaviour of the complete agent-environment system, is generally not accounted for by the central block program except in the form of tailoring the parameters of the controller to function in the right way. This tailoring is normally an empirical process: often the reasons for a

set of parameters (such as time constants and activation thresholds) being used are not understood, they just work. This is clearly not an ideal state of affairs for the development of a system control structure. A return to the genuinely parallel and distributed structures of behaviour based control as characterised in [Brooks86], [Connell89] and [Malcolm *et al* 89b], would be likely to reinforce much of the reliability, robustness and fast response characteristics of such systems.

The last section concluded by suggesting that the bottom starting point required for system development had gone down a level to that of the control of sensor and actuator behaviour. The complete picture resulting from the laser scanner control implementation in chapter 5 was, however, less clear than this. Views of a system in terms of level are helpful in theory but seldom seem to turn out to reflect the actual true state of affairs when it comes to building real-world implementations. There are frequently constraints from system hardware and/or task-environment that preclude a purist implementation. This is perhaps indicative of a problem in the basic view of systems in purely "levelist" terms.

This section presents the case that, in the light of the work and experiments detailed in chapters 4 and 5 and the conclusions regarding the lower-level starting point of behaviour based systems, the rather simplistic view of systems in terms of layers is not sufficient in dealing with complex real-world agents. The next section discusses the shortfall in a "layers only" viewpoint and suggests a more distributed and localised view in terms of layered subsystem components. The physical constraints on the different abstractions of agents are then discussed with emphasis on the low levels of software processes. This is then followed by the idea that systems of agents are seen better in terms of agent interactions than communication of information. First background is given on the area of Multi-Agent Systems research.

6.2.1. Layers are not Enough

Although a system of layers (for example, those advocated in [Brooks86], [Albus81] and [Newell81]) provides a useful view of a system in terms of different details of task, structure and interaction as well as different levels of observer abstraction, these all tend to focus on a system as being a single entity and do not generally allow for physical variation and distribution into different sub-units within the levels. The range of levels in a real-world situated agent must always start at the physical level or some boundary (no matter how hard to define) between agent and environment. For a complex agent the upward series of structures will vary from behaviour to behaviour and across the various subsystems of the agent (for example, the number of subsumption levels between the physical world and the top user-interface layers detailed in chapter 5, figure 5.1, differs between the photomultiplier controllers and the optics controller).

Despite the fact that the levelist views referenced above are all abstractions of a system, they give little account for the inevitable variation evident in truly distributed systems. There would

seem to be a need for a view of systems that also accounts for clusters of related processes both in terms of physical location and behavioural relationships. This has been evidenced by the laser-scanner work reported in chapter 5 and also in other reports of behaviour based implementations such as [Ferrel94], [Colombetti *et al* 94], [Gaussier & Zrehen 94] and [Steels95] which all deal with modular and parallel approaches to the construction of complex agent systems. The ideas discussed here can perhaps be seen as being somewhat akin to those of Minsky's society of mind [Minsky85] which presents layers as agencies of interacting agent processes. However, for the same reasons that BBAI differs from classical AI, the ideas in this part of the thesis differ from those of Minsky.

Figure 6.2 shows three possible levels of complete subsumption control structures. Each is situated at a different level in terms of abstraction from real-world interaction. The top level is that of an industrial production line. A number of control behaviours might act to maintain the production line operation as a whole by interacting and communicating with subsystems, each controlled in turn by subsumption control structures. Then there is a lower level of component parts, often utilising embedded microprocessors. These also can be seen as behaviour based agents and are controlled by their own local subsumption structures. In all cases of these subsumption system levels, the control structures are built using the augmented finite state machines and behavioural levels of the subsumption architecture. There is no significant difference in these components, but there is a difference in the nature of the information that they are responding and acting on and the degree to which they can interact. We thus have a number of levels of distributed systems which may be seen at each level as a community of interacting agents, with an agent being the atomic unit at each level and consisting of a number of internal interaction-determining processes. Figure 6.3 shows an instance.

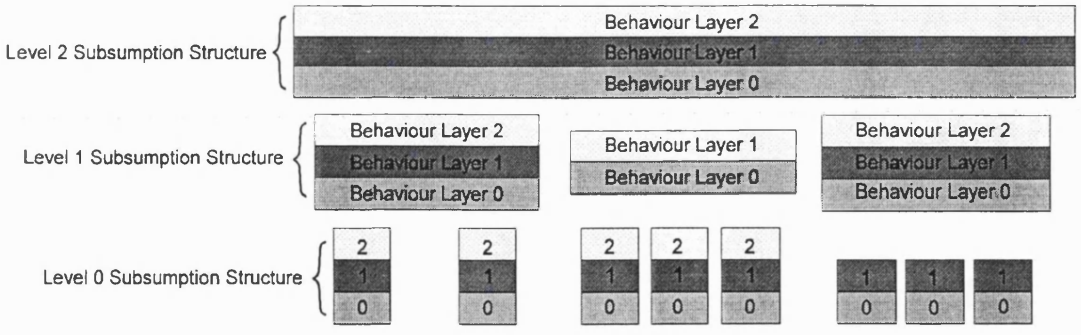


Figure 6.2. Levels of subsumption control structures in a distributed system. Notice that each level effectively consists of a number of multi-level subsumption control structures.

At the top level of the distributed system in figure 6.3 is the industrial production line consisting of a diverse and concurrently-operating set of machines, tools and work cells. Each of these can be seen in turn in behaviour based terms as agents in their own right, all situated in the production line with their own task and environmental niches, and interacting and cooperating with other agents. Typically each of these machines is a complex device like the laser-scanning

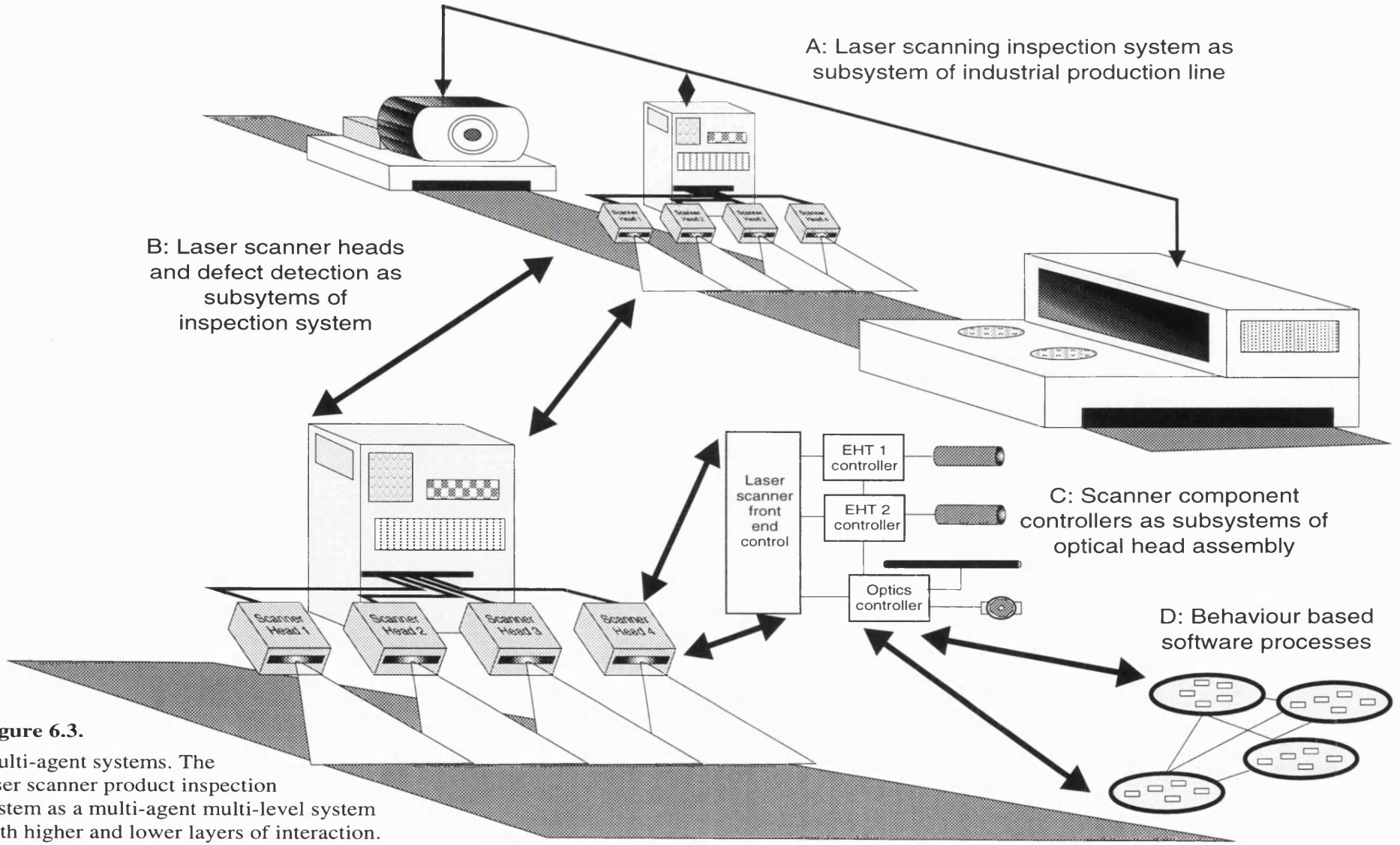


Figure 6.3.

Multi-agent systems. The laser scanner product inspection system as a multi-agent multi-level system with higher and lower layers of interaction.

inspection system used in the work of, and as illustration in, this dissertation. Figure 6.3b shows an often-used configuration of this system with a top-level defect identification and classification unit and a set of four laser scanners acting in parallel to cover greater surface areas. At this level again, the constituent parts can be regarded as situated agents in terms of the BBAI working characterisations. The lower levels of the individual laser-scanning channels (which are the focus of the experimental work of this thesis and as such have already received much attention as regards their individual construction consisting of a parallel network of asynchronous subsumption AFSM processes) can be seen as yet another level of distributed system (figure 6.3c). Below that, a further level consists of the software processes of the subsumption architecture (figure 6.3d). In summary then, it would appear that at all these levels of system interaction there is a place for a BBAI-type methodology based on the autonomous control of an agent entity that interacts in an agent-environment system with other agent entities.

6.2.2. Physical Constraints at the Software Processes Agent Level

This view of layers of interacting agent structures on different levels of scale and abstraction is all very well but so far, perhaps, there would appear to be little clear advantage in viewing a system in this way other than as a method of organising such systems. It may be that this is simply a description that could be foisted on any system. However, it is the *behaviour based* approach to system development that is the significant factor. Here there may indeed be benefits, especially when the advantages of parallel and asynchronous architectures are combined with the new global view of distributed control structures as agent-environment behaviour systems (at whatever level of interaction). Another factor that has emerged as being significant as a result of the experimental work of this project is that the physical location and abstraction of processes in a system is important. Most BBAI work focuses on agents, such as mobile robot vehicles, that are physically embodied in their environment, and it might be hard to see that "agent processes" or "agent subsystems" (such as photomultiplier controllers) in a network of transputers are embodied in the same way. However, it has been the case that the physical location and abstraction of interaction from the real-world of subsumption AFSM processes in the laser scanner test-bed have been critical factors in determining the behavioural characteristics of the system (see sections 6 and 7 in chapter 5). It would appear reasonable to suggest that this is a feature that might be found in many complex systems from mobile robots to industrial machinery. This introduces an important part of the thesis: one that concerns the physical location and interaction abstraction of an agent's control processes.

It is well recognised that engineering is in part an art form in which the designer must trade off a number of critical factors in order to achieve the most effective working balance. This was particularly the case in the construction of the control mechanisms of the laser scanner test-bed where the required interaction location of the AFSM processes and the interaction time constraints had to be offset. For example, the photomultiplier control modules utilised a cluster of

closely-coupled AFSMs to deal with the task of controlling the physical aspects of the analogue to digital conversion electronics and formatting data for other more remote AFSMs to use. This was necessary because of the massive amount of sensory input generated for each laser scan stripe (equivalent to the output of up to 1000 analogue-valued sensors). It was not realistic to envisage this quantity of data being sent to various parts of the system in anything like the time required for stable control, due to the comparatively low bandwidth of the transputer communications links. In any case, the control AFSMs (*BallPark* and *NearLin*) did not function on these raw sensor signals alone, although it was still beneficial and necessary for them to be located nearby in order to reduce the time overheads of signal propagation between and across the processors in the network. Other low-level AFSMs (*EdgeMonitor* and *PedestalLevel*) together generated a single signal from the one-dimensional sensor array that indicated the voltage level of the signal pedestal (if a surface was present) tracking the surface edges as they moved across the sensor's field of view. It is clear that:

- The control AFSMs did not require large amounts of raw sensory data to provide stable functionality;
- Any attempt at transferring such large amounts of data along the serial transputer links would cause serious bottlenecks.

As a result of these constraints the pedestal level signal and the EHT voltage signal from the *EHTOutput* AFSM were connected to the higher subsumption control levels, whether or not these were physically nearby. The reduced quantity of data and the tuning of the required signals meant that propagation was not too costly. For example, compare the 1000 words of raw sensor data with the two carefully selected words of these specific signals.

The choice of the above AFSM connection scheme had significant importance for the behaviour of the test-bed system in terms of AFSM processing rate (as dictated by the system's characteristic time). In fact, it can now be seen that the timing of AFSM interaction also changes from layer to layer in the subsumption structure. The lowest levels are necessarily tied to the timing of physical devices which interact with the real world while the higher layers are less so, as (being more abstract in their interactions) they are not so critically linked to the real world. For example, the control page AFSMs of the user interface layer from chapter 5 were set to run with a slower characteristic time than those of the lower levels. This was necessary in order to allow time for storage of run-time data. This attribute is also evident in mobile robot control structures. Here, the low-level reactive response of obstacle avoidance and taxis behaviour must run in relatively tight and fast control loops whereas the actions concerning higher-level behaviour such as the selection between a search for a battery recharging point and some other task are less dependent on rigid synchronisation with the physical world. There is therefore a difference between the fast short time period response to a current situation of the lower levels and the

longer-term effects of higher levels (see [Herman *et al* 90] for detailed discussion along similar lines).

Clearly, as the time domain is expanded in the higher levels of a system the cost of dealing with massive amounts of rapidly-changing sensor and actuator states increases. Thus the engineering trade-off is to balance the degree to which the current sensory situation is merged with the system's situation over time. It is perhaps possible to see abstraction of agent layers in two-dimensional terms, one of interaction (including time) and the other as abstraction from the physical world including sensation and actuation (space).

6.2.3. Information Transfer or Interaction Dynamics

It is tempting to refer to this variation of abstraction in terms of sensory information. As the agent control layers become removed above or away from direct sensory input, it might be claimed that abstraction of sensory information would be required to reduce the signal bottleneck problem which might also be seen as an information bottleneck. However on analysis of the interactions of a single AFSM process this picture becomes less appropriate. We propose that this inappropriateness is not restricted to this level of agent system only but is also just as relevant at the level of physically-embodied agent interaction and the other levels of system suggested above. At the level of a physical agent being controlled by a distributed subsumption architecture, the constituent asynchronously-running AFSM processes can be viewed as individually-acting and interacting agents.

Given this view, it is interesting to emphasis the finite-state-machine nature of these processes. On an individual basis they are simply clocking in a number of continuously-changing inputs and generating an output which may or may not change state depending on the configuration of the inputs (including internal variables which may be seen as direct feedback loops from the finite state machine's output back to its input). This is true for any AFSM process irrespective of its situation in the subsumption control structure and hence irrespective of the degree of abstraction from the physical world of its input and output interaction. As far as an individual AFSM process is concerned, its only world is that of its input and output connections. Any response that is fed back arrives either via other AFSMs or via an actuators→environment→sensor loop, but this difference is not significant except in the eyes of an observer, for example the system designer. This suggests that there is little difference between a "normal" or common set of agent-environment interactions and what may be termed communication. An agent may enter into a pattern of stimulus-response behaviour in its environment. The fact that a significant part of that environment constitutes a second agent entity that is providing the bulk of the stimulation for the interaction is not significant to the internal mechanisms of the first agent.

6.2.4. Layers of Interacting Agent Communities

To summarise, this section suggests that the control structures of real-world situated autonomous agents must take into consideration not only abstraction of the task or behaviour, as is common in the BBAI field, but also abstraction of component or layer interactions. This involves abstraction in both temporal and spatial dimensions:

Spatial: In terms of the physical location and situation of the processes with respect to the physical world.

Temporal: In terms of the characteristic time span over which the process operates.

Factors that this affects in system construction are the quantity and distance of interprocess connections, especially where these are between remotely-located hardware processes. The result is a system architecture similar to that of the layered diagram in figure 6.2 (and [Brooks86]) but also including the more obviously distributed characteristics of a multi-agent community of processes where the agents at this level are clusters of AFSM-like processes. An example is shown in figure 6.4. Factors determining an agent's atomic definition are temporal and spatial, seen in terms of:

- The process location and signal connection distance/quantity trade-off.
- Behaviour and task requirements.

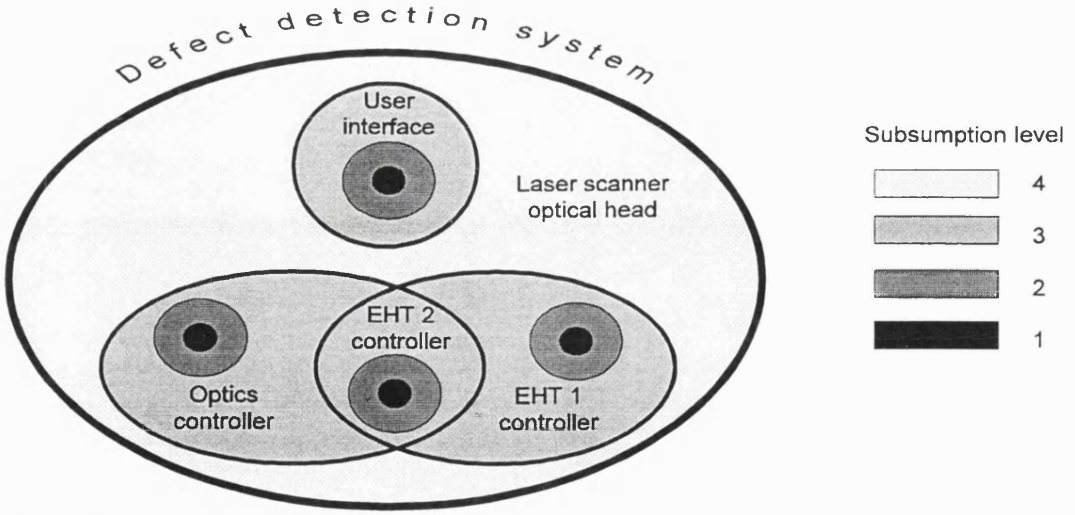


Figure 6.4. A distributed-agent-based behaviour control structure of the laser scanner test-bed. Note the "sphere of influence" of each agent process, which is a function of its physical system location.

In particular, note how the prominence of the user interface layer in figure 6.4 has been reduced from that of the top layer in the original diagram (figure 5.1) to another set of agent processes situated in the host computer in this one. Here the keyboard and screen appear as local sensors and actuators, and communication is from the user interface AFSMs to other subsystem control agent process sets such as the photomultiplier controllers.

6.2.5. Other Multi-Agent Systems Research

The topic of research referred to as Multi-Agent Systems covers a large area. With roots in distributed artificial intelligence, parallel computing and computer networking it has now matured into a recognised and individual field. Distributed Artificial Intelligence work such as that reported in [Hayes-Roth88] included aspects of system control using blackboard data structures for communication between processes or agents. This has been taken further in [Tigli *et al*] where distributed and parallel blackboard control architectures are discussed and [Simmons92] presents issues concerning the concurrent planning of robot motion. Other work concerning interactions within distributed systems is that of [Billard & Pasquale 93] where the effects of communication propagation time on process interaction are examined in terms of agents that have to share, combine or hold onto computing resources. [Jennings & Wooldridge 95] provides a useful and more recent view of the subject of multi-agent technology by introducing and setting the scene for much of the terminology. For example the properties of an agent are listed as including: Autonomy, Social Ability, Responsiveness and Proactiveness with additional characteristics of adaptability, mobility, veracity and rationality. From this and the work presented in this thesis it is apparent that many aspects of BBAI concerning the distributed nature of control architectures also fit into this paradigm.

Multi-agent systems research occurs on two main levels: i) At the system interaction level; for example that of interacting industrial machinery as reported in [Jennings 94] and [Freund & Buxbaum 93] and ii) at the subsystem or component interaction level; such as that reported in [Oliveira *et al* 91], [Habib *et al* 92] and [Lyons93]. In these latter examples architectures consisting of concurrent processes or multiple agents are suggested as tools for robot design although the designs discussed involve a largely functional (classical) breakdown of control structures rather than behaviour based. Conversely the behaviour based work reported in, for example [Parker93], [Mataric92], and [Arkin & Ali 94] deals with multi-agent systems in terms of teams of cooperating robots but the multi-agent or distributed nature of the individual control systems is not dwelt on.

Another active area of agent systems research is that of intelligent software agents that are designed to operate and complete tasks within the domain of the Internet [Riecken *et al* 94]. Although these agents may be said to be mobile and autonomous (within the virtual world of the Internet), the environments that they are situated in differ considerably from those of real-world autonomous systems, at least at the lower levels of sensation and actuation that have been the focus of the work reported so far in this thesis. The subsumption system-component agents discussed in this work are fixed into their niche like the bacteria in a cow's stomach. They are ineffective elsewhere and, as in the case of the bacteria, cannot survive elsewhere. Software agents of the Internet variety have a distinctly parasitic nature; they invade a host system in order to exploit some resource that it has and that they require. The agent view of autonomous systems

proposed in this thesis is of a more symbiotic relationship, with each part of the system benefiting from and in many cases being dependent on the presence and activities of the other. Although there may well be cross-fertilisation of ideas, to some extent, we suggest that this lack of physical interaction within the Internet is likely to prove significant. It is after all one of the facets of BBAI that an agent is *situated in* its world. This we saw in chapter 3 with the spider. While being able to cope magnificently in the complicated hedgerow environment, the spider was completely at a loss in the alien environment of the bathtub, being placed in a domain outside of its "design specification" so to speak.

The topic of this second part of the chapter leads to a proposal of a low-level behaviour based multi-agent view of system construction that may be expanded to include further behaviour based interactions at the higher system levels mentioned above (for example, the user interface layer). The multi-agent system developed with the subsumption architecture to control the laser scanner test-bed has displayed a truly behaviour based approach to both system and subsystem development.

6.3. Conclusions and Introductions

This final section serves both to conclude the first part of the thesis and to introduce the second. The work detailed in chapters 4 and 5 that dealt with the application of a behaviour based control solution to an industrial real-world control problem highlighted a number of factors relevant to both the development of large complex distributed control systems and to further research in BBAI. The final section summarises this and leads onto the next chapter by introducing the need for dynamic adaptation and embedded learning mechanisms in the low levels of behaviour based control architectures.

6.3.1. Benefits of a Layered-Agents View of Autonomous Systems

In [Brooks91] it is suggested that models for intelligence have largely followed the current state of the art in computational machinery, for example:

- The mechanical arithmetic machines of Babbage.
- The computability work of Turing and the Turing Test [Turing50].
- Classical AI of the 1960s and 1970s with implementations testing the "physical symbol system" hypothesis.
- The distributed (classical) AI work typified by [Hayes-Roth88] that attempts to make use of the powers of parallel computer systems while maintaining the essentially centralised and symbolic theme of classical AI.

This progression is also interestingly presented in the robot evolution book [Rosheim94] although with a somewhat different emphasis. The observations of the current chapter seem to be

a continuation of the trend, and we believe that there is considerable evidence that a distributed-agent-based view of a complex system is both more realistic in terms of actual design, implementation and development and better at matching the beneficial characteristics of natural solutions to similar problems [McFarland85] and [Zeki93]. This chapter has highlighted the following points:

- These ideas encompass a view of a complete system from the lowest levels of control upwards. Emphasis is on the starting point of bottom-up development and the necessity to start with the behavioural control of sensor and actuator subsystems.
- Consideration of time and timing of processes and their interactions is of central importance in the development of complex systems.
- The physical location of processes within an agent structure has been identified as being a significant contributing factor to agent-environment behaviour.
- A multilayer view of distributed agent systems is proposed, with levels and scope of levels being determined by abstraction in both temporal and spatial dimensions from the physical world.
- Communication at any level can be seen as a tight actuation-sensation exchange between two or more agents. The sensation of the receiving agent is little more than the characteristic of a particular agent-environment situation which may or may not stimulate a return actuation response, i.e. a reply.

The points above outline five significant factors in system development that are perhaps not currently taken into consideration explicitly during the design of a typical complex distributed system. The bias of the discussion has been towards the point of view of BBAI. As such, the commentary is suggestive of factors that are important in the future design and development of complex behaviour based systems. It has been observed that to date the behaviour based paradigm has not been used for large-scale development, being limited to examples in the form of mobile robot test-vehicles that offer a simplistic though realistic starting point for research on autonomous agents. This thesis has so far provided some indication of the issues that must be addressed if the behaviour based approach to systems control is to be expanded to realistic, real-world applications (in other words, the "real real-world" as seen through complex sensors that are entities in their own right interacting at their own level as agent-environment systems).

The Role of the Designer

The behaviour based approach to control, with its multitude of interconnected and interacting asynchronous rule-based processes has few if any strong design methodologies or techniques apart from the assiduous use of the techniques typified by the many architectures evident in the literature. Chapter 3 overviewed three such architectures for illustration: Subsumption Architecture [Brooks86], PDL [Steels93], and Spreading Activation Networks [Maes89].

However there are others such as AuRa [Arkin90b], and ALFA [Gat et al 94]. There have been few (of which [Colombetti et al 94] is one) attempts to formalise the process of design of behaviour based systems. One of the fundamental results of the BBAI route to systems control has been the development of many tailor-made examples of solutions to specific problems. This, after all, is one of the main reasons for the apparent success of the paradigm; systems are built to function specifically in a particular environmental niche often utilising specific features of the interaction space that allow fast and simple solutions to the control problem. A good example is the can-collecting robot in [Connell88] and [Connell89] that uses combinations of agent and environmental situations to trigger the inevitable sequence of behaviours that constitute collecting a can.

This shrewd utilisation of agent-environment resources (see [Horswill93] for detailed discussion) in guiding the design of an agent's control structure has worked well in the behaviour based demonstrations of the past, but the resulting control solutions are typically unstructured and difficult for a third party to understand. This is despite the layered and parallel structure of such systems which aspires to (and can actually) facilitate the assembly of complex control structures. Architectures like subsumption tend to result in large and elaborate networks of AFSM processes (see figure 5.1 in chapter 5, for example) numbering in the dozens, or hundreds (as yet, larger systems have not been built). This work is tending to suggest that there may be limits to this approach, defined by the ability of a designer to keep track of and utilise the agent-environment situation to best effect.

One of the methods classically used by traditional AI in testing a system is to assess its ability to deal with exceptions or unusual events. Whilst the thrust of BBAI ignores this type of exception provision in order to keep system cost and complexity down, it is still the case that the designer must try to provide the agent with the most wide-ranging and robust set of resources possible. The designer or team of designers must apply their knowledge of the task domain to the intricate details of the target control structure, defining everything from process interconnections to time-constants and actuation profiles. This necessity for manual selection and provision of control resources or behavioural repertoire is a major factor affecting the prospects of further development of behaviour based control techniques. It is apparent that the task is quite probably impossible if anything approaching a human level of behaviour is desired. The techniques of BBAI have led to an impressive collection of control solutions but ultimately face the same problems of scale as other more traditional approaches. However, it has not reached the end of the line yet.

6.3.2. The Problem of Design of Large-Scale Complex Systems

This chapter has concluded by highlighting some fundamental problems in building more sophisticated autonomous agents. The perceptions reported here resulted from discussion and experimental work on an industrially-situated test-bed system where it was emphasised that there

is a need for a greater recognition of the dynamics of agent-environment interaction, specifically through the increasing need to provide localised control and low-level (in behavioural terms) adaptation of sensor and actuator subsystems. It was argued that although sensors and actuators require discrete control at their own level of situation, it is still necessary to take the global behaviour of the agent into account during development. For example, it is desirable to pay attention to the implementation of low-level reflex behaviours physically side by side with sensor and actuator control in order to avoid the unnecessary propagation delays to and from higher-level and remote processing centres.

The work reported in this dissertation along with the accompanying discussion has led to a view of an agent system as being distributed to the extreme, both physically and functionally. It has been suggested ([Pfeifer & Verschure 91], [Mahadevan & Connell 92], [Maes & Brooks 90]) that the problems of designing, developing and tuning these systems to survive and perform tasks in a reliable and robust way becomes astronomically complex given the large amounts of internal and external state in systems that interact with the real-world. As a result there is clearly a need for some means of providing an automatic configuration of a system so that it becomes able to initialise and tune its own interaction parameters as and when it gains experience with environmental situations that the designer has not foreseen. Currently a wide range of multidisciplinary work is being conducted in the areas of autonomous adaptation and learning ([Kaelbling93], [Mahadevan & Connell 92] and [Mataric94]) as well as automatic methods of generating complete agent control structures ([Cliff *et al* 93] and [Harvey *et al* 94]). While some of this work is also covered by the domains of classical AI (for example, [Bratko93] discusses approaches to the inductive learning of rules and decision tree structures) and classical control ([Harris94] and [Miller.W *et al* 90] both of which deal with the application of artificial neural networks to the control of various real-world problems) in an attempt to solve similar scaleability problems, much is specific to the nature and methods of BBAI and is approached with the lessons of BBAI in mind.

The second theme of this dissertation starts with the next chapter, chapter 7. This concerns the equipping of behaviour based control systems with mechanisms of adaptivity and learning ability. Chapter 7 begins by discussing some history and the current state of play of various aspects of adaptivity, learning and automatic generation of control structures and interaction mechanisms of an agent. The second part of chapter 7 then presents a series of ideas for embedding such mechanisms into the subsumption architecture so that adaptivity and learning characteristics are built in as part of the whole system from the bottom up. This is particularly emphasised in the light of the results and conclusions from the first part of the dissertation concerning the need for localised, low-level control at a sub-behavioural or subsystem level in a behaviour based architecture. Chapter 8 then considers the implementation of these ideas in the laser scanner test-bed that was described in chapter 4 and further develops the control structures outlined in chapter 5.

7. Adding Adaptivity and Learning Ability to the Lowest Levels of A Behaviour Based System

This chapter covers the following:

A need for low-level adaptation and learning in behaviour based systems.

Enhancing the augmented finite state machines of the subsumption architecture.

Continuous tuning of critical parameters in AFSM processes.

A complete feedforward AFSM based on an artificial neural network .

7.1. Introduction: A Need For Adaptation and Learning

So far in this dissertation the focus of attention has been on the use of behaviour based control techniques in the domain of truly distributed real-world systems that incorporate complex sensor-actuator subsystems. Illustration has been provided through the use of a test-bed in the form of an industrial laser-scanning sensor which is itself part of a larger distributed system. This has been in contrast to the usual experimental domain of behaviour based artificial intelligence (BBAI) which tends towards the use of mobile robot vehicles. The previous chapter concluded the work on that test-bed with a number of key observations including:

- The necessity for a truly bottom-up development of control structure starting with sensory and actuation subsystems: in other words, with the physical boundary with the real-world at a sub-behavioural level.
- The significance of the physical location of - and time constraints on - a process in the definition of an agent control structure.
- A view of a system as a hierarchy of distributed subsystems of agent-like processes with higher levels increasing in temporal and spatial abstraction.

This chapter now continues by taking up the problem resulting from the complexity of designing such systems to perform in the real-world in a reliable, robust and self-sufficient manner.

The construction of most behaviour based systems typically includes the definition, placement and interconnection of numerous simple processes (see chapter 6). In the case of the subsumption architecture these are augmented finite state machines (AFSM). There is little doubt that these

processes can be seen as elementary, reactive or automatic, components of a larger system which exhibits complex repertoires of behaviour. But there appears to be some criticism in the literature, e.g. [Kirsh91], that discounts the resulting behaviour based systems as being reactive and automatic in the same way as their component parts. Another common suggestion is that so-called hybrid systems that incorporate a symbolic high level and behaviour based low level should be the focus of attention (as discussed by [Franks & Cooper 95]). The difference between a basic reactive system and the more complex behaviour based system is discussed and clarified in [Mataric92c]. Here it is suggested that the internal and external interactions of an agent's multiple control processes result in a system that is significantly more complex than an automaton (see also [McFarland95]). In addition, the specialised nature of the behaviour based development approach does not readily lend itself to a general or clearly defined interface with a more classically composed, higher level AI control system. Rather, specifically tailored solutions to specific problems are in evidence with examples of hybrid control solutions being presented in, for example, [Malcolm95], [Jaeger95], [Holgate & Clarke 95] and [Gaussier & Zrehen 94]. In this discussion it is important to summarise that while current instantiations of behaviour based systems may appear to be simplistic in terms of their task-achieving behaviour, this does not indicate that such systems are restricted due the reactive nature of their component parts.

A "reactive" system is taken to mean one that responds to nothing more than the current agent-environment situation as indicated by its inputs and without reference to any temporal constraints [Mataric92c]. The reason for the interpretation of behaviour based systems being of a purely reactive nature is most often a consequence of the implicit nature of any internal state in such systems. It is the case that as soon as a sensor is sampled the resulting data are in principle out of date, i.e. the world continues changing while the sensor reading remains static, frozen in time. This view is in part the result of the use of digital computation for the processing of such signals [Brooks91] (the issues of digital systems and control were dealt with in chapter 2). Here the point to be made is that the longer the signal data are maintained, the more out of date they are likely to become. Behaviour based systems overcome this by attempting to respond as quickly as possible to such signals through the provision of hardwired or fixed mappings of reflex-like actions. More classically oriented systems (such as [Simmons94]) attempt to transform these signals into symbols in order to support map making, reasoning and planning processes. The longer time spans required by the computational workload of these functions requires that more generalised actions have to be selected, or actions based on larger amounts of data. Meanwhile, in the same time period, a behaviour based system would have resampled the world and updated its actuation output. These issues once led to the statement that "the world is its own best model" in [Brooks90]. The meaning of the quotation is that agents find in the real-world the most valid and up-to-date information concerning their situation and status. Any internal state, from simple records of previous sensory signals to complex symbolic world-models, has a potential for inaccuracy if it is used to generate agent responses to external events without regard for the

present situation. Behaviour based systems address this problem through the layering of behaviours and the temporal abstraction of higher levels using combinations of current agent-environment state and event history as and where necessary.

It is the interconnection and parallel assembly of the reactive behaviour based processes that introduces internal state into the system. For example, the subsumption AFSM processes are connected by wires that signal sensor and AFSM output states. While these wires can change state rapidly, they still constitute a wealth of information that is at any particular instant indicative of the agent's state. This form of internal state may be regarded as an implicit system state and contrasts with the explicit state of, for example, the construction of symbolic maps or event tables. ([vanGelder94] and [Clark94] discuss these issues in terms of agents and the use of representations). Of course, as already mentioned, time does not stand still and hence this time-sliced view of an agent's state is not really a true indication of the system but just one encouraged by the technology of digital computers. This implicit state is continuously changing and in complex real-world agents is unlikely to be of an exactly repeatable pattern, which is in part a reflection of the nature of the real-world environment of such systems. This makes the design of behaviour based systems difficult, with solutions often resulting from much trial-and-error testing of the agent-environment system. For example a hypothetical and typical: *if(left-collision) do turn-right(90)*, coded and adapted by hand to suit a robot's environment. In this example the *90* may refer to an angle for the motor to turn or alternatively refer to a time period. Whatever it is, the designer has put it there because it works and provides the robot with the right response to its current situation. Likewise the *left-collision* condition is typically formatted by the designer to consist of a number of relevant and discrete sensory inputs. As a result it may be argued that the designer has built personal knowledge of the domain into the structure of the agent control system providing the right process connections and constant values to elicit the right agent-environment interaction dynamics ([Brooks91] and [Smithers94]). This designer knowledge can be seen as an embedded property of the architecture of an agent in the form of implicit domain knowledge.

Building implicit domain knowledge into a control architecture is not in itself a bad thing. It is in fact the very essence of system design and can lead to considerable saving in terms of computational resources. Keeping things simple means additionally that there is less to go wrong in a system and it can be easier to track down problems if things do. It is also a sensible engineering approach to start out with an apparently simple but sufficient solution and only add functionality as and when required. But there are also drawbacks. Tuning a system to realise a predetermined response to an environmental situation is fine so long as the environment does not change and that the original tuning was sufficient in the first place. For small numbers of sensors and behaviours the hand tuning option has been demonstrated to be adequate, but there are two significant problems that become apparent for more complex systems:

- The designer is often unable to provide sufficient domain knowledge.

- There will most probably be too many behaviour processes, too many parameters and too many process interconnections to hand-tune (the frame of reference problem [Pfeifer93]).

Therefore there is a distinct need for an automated method of building behaviour based robot control systems that can deal with both the problem of acquiring and maintaining sufficient domain knowledge and the problem of incorporating large numbers of processes generating different types of behaviour into a system. Because the domain knowledge required is very likely to be available only when the robot encounters situations in its world (much hand-tuning of behaviour based systems is done by observing the way robots interact with their environments and then making adjustments) and because the structure of the behaviour control processes can only be developed as and when this knowledge is acquired, there is a clear requirement for a robot to be given mechanisms that i) are able to learn and self-tune parameters and ii) learn, adapt and optimise their actions as the system encounters and experiences environmental situations.

Techniques of adaptive and learning control have been seen widely as a solution to this problem but in many cases have so far failed to be transferred from the idealistic test domain of simulations. Examples that do deal with real-world implementations of learning and adaptation can be seen in [Kaelbling93], [Mahadevan & Connell 92] and [Mataric94] where aspects of discrete state reinforcement learning are experimented with at the level of behaviour selection and adaptation. Reinforcement learning is an on-line technique for unsupervised learning in systems that are in active interaction with their environments. The basic idea is that a reinforcement function generates one or more values that are indicative of the agent's current situation. The reinforcement value is fed back to control processes which may then alter their behaviour depending on the value of the feedback. Other work reported in [Pfeifer & Verschure 93] details a technique for learning associations between complex sensors and actuators in the form of models of classical conditioning ([McFarland85]) applied to individual behaviours. The use of Genetic Algorithms to evolve sets of production rules known as Classifier Systems has been reviewed in [Kaelbling93] and [Beer94]. These systems have been demonstrated to have a potential for control but Kaelbling suggests that their application to real-time embedded systems may not be practical. Also [Nehmzow & Smithers 90] looks at the use of self-organising feature maps for landmark detection. Other work reported in [Clark *et al* 92] looks at on-line tuning of parameters in reactive control architectures.

Many learning and adaptive control problems remain open, with solutions obtained to date having many limitations. Much information exists on experiments with simulated domains but these applications fail to address the more important aspects of robotics that deal with "noisy" sensors and "unreliable" actuators (the adjectives are in quotes since there is no such thing as a perfect sensor and any solution that fails to address these aspects is not a solution) that are a fact of life in real-world systems. In fact, simulated domains tend unwittingly to support a necessary

assumption that is required of a system in order to implement a reinforcement learning scheme such as Q-learning [Watkins89]: namely, that the agent's domain must be modelled as a Markovian Decision Process which according to [Mataric95] is defined as follows:

i) The agent and environment can be modelled as synchronised finite state automata.

ii) The agent and the environment interact in discrete time intervals.

iii) The agent can sense the state of the environment and use it to make actions.

iv) After the agent acts, the environment makes a transition to a new state.

v) The agent receives a reward after performing an action.

In the real-world none of these characteristics can be relied upon. Work that does deal with real-world agent learning is scarce and in the case of examples of reinforcement learning such as Q-learning the learned feature tends to be a relatively high and abstract level of behaviour ([Kaelbling93], [Mataric95] and [Mahadevan & Connell 92]). These recent experiments in using reinforcement learning to learn control strategies for mobile robots, in the best traditions of the behaviour based approach to robot control, have incorporated considerable designer-tuned implicit domain knowledge in order to embed the learning mechanisms into the robot and ensure that they converge to some useful behaviour.

Many reinforcement learning mechanisms also typically suffer from bootstrap problems which are characterised by an initial period in which the agent builds up its interaction processes through environmental exploration and experimentation. During this time the agent's performance is unreliable at best. Other non-symbolic solutions, for example the classical conditioning and self-organisation models (for example [Pfeifer & Verschure 93]) that do have useful bootstrap mechanisms, tend to be very application-specific and centralised. A complication here is that, as concluded in chapter 6, distributed hardware architectures are becoming the norm for robot control with different parts of the robot being controlled locally by their own microprocessors. Another issue then, regarding the solution to providing learning and adaptive control for an agent, is that the design architecture should be able to deal with physically distributed centres of computation which have limited-bandwidth communications and therefore require behaviour to be initiated locally. In such systems global data structures are not available as a component of a cost-effective solution.

The real-world reinforcement learning experiments to date have largely dealt with learning the *when* of doing an action rather than an action itself. This has been a necessary feature of such systems. [Mataric95] used multiple predesigned behaviours and here experiments succeeded in structuring behaviour selection so that groups of mobile robots could effectively interact and forage for objects in their environment. Hard-wired behaviours were used in order to speed up the learning process as well as to provide sufficient abstraction for the reinforcement learning to work. It was suggested that individual behaviour might be learned but would require a framework

for tuning individual parameters. This in part is the focus of the work reported in this chapter and in the second part of this thesis. Other reinforcement learning work reported in [Mahadevan & Connell 92] deals with the automatic programming of a subsumption architecture. In these experiments a number of behaviours are learned but here again considerable domain knowledge is necessarily built into the system in the form of behaviour applicability functions and motor actions as well as the basic structure of the subsumption architecture layers. Both examples also incorporate a number of predefined reinforcement-generating functions. In short, these learning techniques require considerable tailoring in order that their terms of reference may be sufficiently grounded in the robot's domain. This development process is similar to the tailoring required of the behaviour based control architectures that was discussed above. The focus of the work reported in the second part of this thesis has been to explore ways of providing the basic building blocks of the subsumption architecture with an inherent ability to learn input-output associations and subsequently to be able to adapt to long-term changes in the domains of the agent and environment. In this way the benefits may be built into a behaviour based control structure from the bottom up.

Summary

The discussion so far has pointed out that a fundamental characteristic of the behaviour based design of real-world situated systems is the built-in provision of domain-specific knowledge. This is necessary in order that the behaviour control processes are able to operate in a fast and robust manner. It has been suggested that this built-in knowledge is also a potential problem when larger and more complex systems are considered, and hence that systems should be given mechanisms that are able to acquire their own domain knowledge, represented implicitly in control structure and dynamics, as they experience their environments. In other words, the systems should be able to learn. The second part of the above discussion highlighted problems with current real-world robot learning implementations and it was identified there that these examples often failed to address aspects of low-level adaptation in component or sub-behavioural processes. Thus the second part of the "thesis" presented in this dissertation suggests that low-level embedded learning and adaptation are necessary additions to behaviour based robot control architectures, and addresses the following problems in this respect:

i) The techniques of learning and adaptation need to be grounded in the dynamics of real sensors and actuators from the bottom up, or conversely the lowest-level mechanisms such as sensor and actuator controllers require mechanisms of adaptivity to improve stability and performance.

ii) Adaptive and learning mechanisms need to be distributed around multiprocessing hardware architectures, i.e. there is no central resource of information to draw on, and timing between processes cannot realistically be synchronised.

iii) An agent must have a minimum level of ability before and while the learning and adaptive mechanisms are initialised.

These factors are discussed at the same time as we present potential solutions and experiments. The experiments have been conducted on the laser scanning test-bed and use the relevant transputer-based subsumption programming framework (chapter 4). Experiments in the next chapter (8) examine possible improvements to the photomultiplier control subsystems that were the focus of chapter 5. This allows a useful comparison to be made with the earlier work and results.

7.2. Issues in Enhancing the Subsumption Architecture: Embedding Adaptation and Learning

A common feature of many behaviour based control architectures is the utilisation of a number of interlinked parallel processes repeatedly looping and executing condition statements of the form: if *condition* do *action.* Anyone with experience of building behaviour based robots will be familiar with this format; for example see [Brooks85], [Gat *et al* 94], [Mahadevan & Connell 92] and [Pebody91], amongst many others. The success of the behaviour based approach to robot control can be attributed largely to the skill of the designer in selecting the right conditions, action parameters and interconnections for each behaviour process. In order to automate this process the nature of the condition and action must be examined more closely.

The condition predicate part of the statement usually refers to a number of input values to the process that may originate either from a sensor or another internal process. It will typically be either a simple binary-valued variable or the binary result of some comparison of values - for example: *input_a* > *threshold.* As well as thresholds, other common conditions are for tests for a value to be within a certain minimum-maximum range. The critical aspects of the condition when it comes to building in domain knowledge are: i) which inputs are relevant and should be included? and ii) how should these inputs be evaluated to generate a true or false result, e.g. at what value should the threshold be set?

The action part of the statement usually consists of a value that is output from the process. This may either be a value output directly to an actuator or it may be connected to an input of another behaviour process. This value may also be updated as part of the output process. The critical aspects of the action output may be a combination of i) the actual value output if it is locally constant and ii) the size of modification to the output.

From this it would appear that some form of continuous parameter-tuning is required to effect an adaptation of thresholds and action modification, as well as some form of active search of sensory input combinations or input connections, to format both the condition and action part of the rule statement of a control process. In this way the initial rule as specified by the designer may be adjusted and tuned as the agent gains experience in its environment. This chapter

discusses these points using the subsumption architecture for illustration and demonstration. However, it is emphasised that the techniques of domain knowledge acquisition explored here are not specific to this particular control framework and should be of interest to the wider field of behaviour based control systems.

The basic building block of the subsumption architecture is the Augmented Finite State Machine (AFSM) and its interconnections. It takes the form of a finite state machine augmented with a number of input registers plus real-time clock information. The AFSM clocks its inputs regularly, depending on a characteristic time defined for the system. Each input register contains the last data item received irrespective of how many times the register has received something. The rule element of the AFSM translates the inputs and conditionally activates the outputs which are connected to other AFSM input registers and actuators by "wires". Wires may be connected at special nodes. The most common node is the suppression node which allows a second wire to override any activity on a primary or lower wire. Nodes may be stacked up, allowing higher layers of AFSMs to subsume lower layers in the style typical of the subsumption architecture that was illustrated in chapters 3 and 5. Other nodes include an inhibition function and a default function (inverse of suppression). Implementational details were discussed in chapter 4.

In order to implement the automatic tuning of the parameters of an AFSM rule it would be possible to add more AFSMs into the control structure of the system. Each of the critical parameters that collectively contain the designed-in domain knowledge could be turned into a register that is updated, depending on the activities of the robot, by a higher-level AFSM. However, whilst adding robustness to the agent architecture, this approach involves considerably more work on the part of the designer with each new AFSM requiring additional domain knowledge for its implementation. More conditions and constant settings will have to be tuned. It is interesting to recall from chapter 6 that this higher-level domain knowledge would tend to be more abstract in the temporal dimension than the spatial, and the time constraints on these AFSMs would probably be less critical [Pebody94b]. Unfortunately this technique does not lend itself easily to on-line automatic methods. In this instance the learning process would involve the dynamic addition of new AFSMs, connecting them with wires to lower-level AFSM registers, selecting suitable rules that include condition predicates (with possible combinations of threshold and maximum-minimum classification values) and a suitable action output. Clearly a solution is required that can adapt around an existing predefined control structure.

Figure 7.1 shows the proposed format of an enhanced AFSM. The basic AFSM pattern is maintained with the addition of two blocks: a condition-mapping association mechanism and an actuation output arbitration function. The rule acts both to bootstrap the association mechanism and also to ensure that the AFSM has a consistent minimum performance. Another way of seeing this arrangement is as an additional process learning a feedforward response based on a lower-level AFSM. The actuation arbitration function serves to generate the selected AFSM output

which is effectively the result of the "action" part of the AFSM rule. Interestingly this arrangement has parallels with control solutions found in biological systems. For example, the motor-ocular reflexes that control the saccadic movement in the mammalian eye are based on a learned feedforward control that switches out a lower feedback control loop [Robinson90].

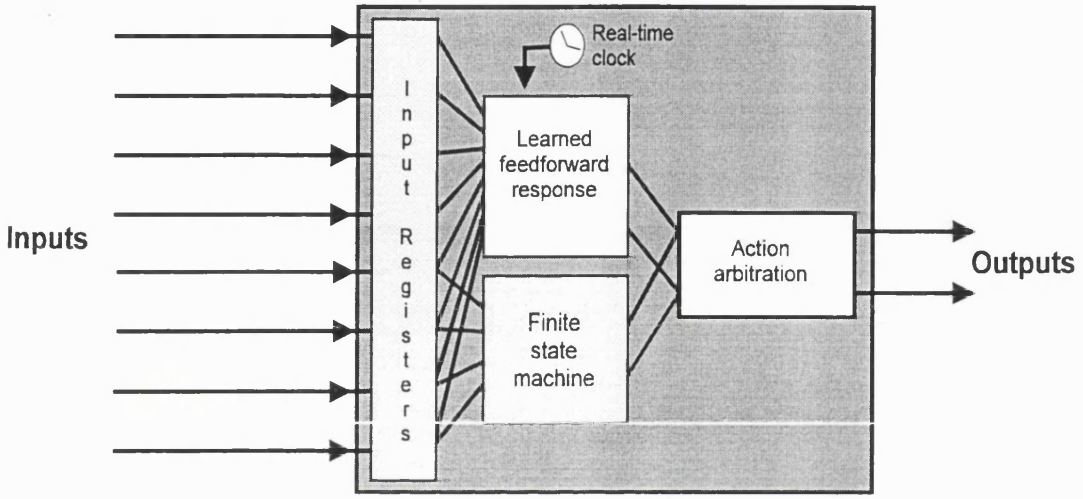


Figure 7.1. Enhancing an AFSM with the addition of a pattern association mechanism.

The rule of an AFSM may be regarded as the formulation of an input-output mapping. The input is some pattern of AFSM input states defined by the condition predicate of the rule and the output is an action that is associated with a particular condition. The problem of parameter selection and tuning may be partially solved with the use of a connectionist or artificial neural network approach. Artificial neural networks are able to learn on-line and also have useful generalisation characteristics. Noisy and incomplete input patterns can still trigger a specific response that was originally associated with more complete data. Both problems identified above - of selecting the input connections and tuning the parameters of the AFSM rule's condition predicate - may be addressed in this way, and are dealt with in section 7.3 below. The problem of tuning the AFSM action output, however, requires a different hybrid approach. We treat it separately with the use of simple optimisation and search algorithms such as hill climbing ([Luger & Stubblefield 89] and [Maza & Yuret 94]) which have previously been experimented with in autonomous agent work [Pierce & Kuipers 91] - albeit only in simulation. The output action tuning aspect of this work is treated separately in section 7.4.

These solutions separate the problem of input state identification and output actuation tuning in order to focus the development of experimental mechanisms. A third possible solution is to utilise a feedforward artificial neural network to map the complete input-output rule function. Section 7.5 details an approach along these lines as a contrast to the approach in the first two sections.

Finally it should be emphasised that this additional computation must occur within the AFSM process and take place within the characteristic time frame of the subsumption control structure. In this way the AFSM interactions across the system that makes up an agent will not be affected by transitions between learning and adaptation phases of an individual AFSM.

Summary

The main job that we assign to the network is to learn an association within the array of inputs based on reinforcement from the AFSM rule activity. The network will be modified continuously, depending on changing environmental situations. The intention is to achieve a form of parameter tuning. Additional inputs may be incorporated to enhance the selectivity of the rule and thus provide an added robustness. Also, redundancy in input state may be utilised in the case of missing or noisy input data. Finally, the behaviour of the enhanced AFSM is dictated by a small number of control parameters that are required to set the update and learning characteristics of the AFSM processes.

When the system starts up, the basic AFSM rule will have domination over the output, but as time goes by the network will take over. This will allow associations between sensors and actuators to be acquired during a learning phase with the nature of the mapping resulting from the activity of the basic rule (akin to a low-level reflex) which acts both to bootstrap the learning process and to provide low-end limits of the system's performance. This has two main benefits: firstly the designer must necessarily provide some domain-specific knowledge and the basic rules allow for a relatively straightforward specification framework; secondly the rules ensure that when the system is first switched on there is some rudimentary reflex behaviour that will prevent the agent from damaging itself though random actions.

7.3. Providing An Input State Mapping: Enhancing the AFSM Rule Condition

Work presented in [Pfeifer & Verschure 93] has demonstrated that self-organising robot controllers can be developed from simple artificial neural networks. Here, a basic reflex action was associated with a secondary sensory source thus enhancing the robustness of the robot's behaviour by learning the extra inputs to the control action. We propose that this approach can be useful in generalising the AFSM rules of the subsumption architecture over the auxiliary inputs as described above. While Pfeifer and Verschure's work focused on the issues surrounding classical conditioning [McFarland85], it provided insight into the nature of continuous adaptive control when applied to real-world robots. More specifically, it had a strong bearing on how such adaptive mechanisms might be initialised (or bootstrapped) so as to ensure that the correct input-output association was acquired. The strategy presented in this chapter for learning and adapting the input-output mapping of an AFSM is based firmly on these adaptive associative networks that are able to learn the associations of complex sensor states from a limited set of simple key inputs.

However, there are a number of factors that must be addressed for these techniques to be embedded in a subsumption-based architecture and used in a more general-purpose manner:

i) The likelihood of having to deal with complex input patterns that are not linearly separable by a single-layer network;

ii) The embedding of the network into the framework of the AFSM to allow a physically distributed control architecture to be implemented;

iii) The temporal sequencing of events as utilised in the work of Pfeifer and Verschure, and typical of classical conditioning, can not be depended upon in all applications, so that an alternative method of allowing a learned response to override a basic reflex is needed.

The fact that the classical conditioning model developed by Verschure ([Pfeifer & Verschure 93]) utilised only a single-layer network potentially left open the problems related to the linear separability of the input space. This problem was first highlighted in [Minsky & Papert 69] with the inability of single-layer perceptron neural networks to implement the simple XOR function which required a complex, and multi-boundary separation in the state space of solutions. More recent accounts can be found in [Wasserman89] and [Masters93] (amongst others). For simple feedforward rule association this is generally not a problem, but it may prove to be so when more complex input spaces are considered. The standard solution to this problem is the utilisation of multilayer networks trained by the backpropagation method. But such networks are notoriously difficult to set up, with training times often being undesirably long and convergence not guaranteed. For this reason work by Nehmzow and Smithers has been studied ([Nehmzow & Smithers 90], [Nehmzow *et al* 89]) which details experiments with self-organising feature maps using Kohonen network algorithms ([Kohonen88]) and later [Nehmzow & McGonigle 94], which combined this early work with additional decoding layers to create networks referred to as Pattern Associators [Rumelhart & McClelland 86]. This is similar to the counterpropagation algorithm originally developed by Hecht-Nielsen [Hecht-Nielsen88] which is detailed in [Wasserman89] as a possible alternative to multilayer perceptrons and the techniques of training by backpropagation of errors.

A series of exploratory experiments conducted with sampled data from the laser scanner test-bed demonstrated that the stability and resolution of the counterpropagation algorithm was significantly superior to a single-layer network and that the convergence time of such networks was within the bounds of usefulness for the application of interest detailed in this chapter; that of enhancing and adapting the functionality of an AFSM control structure during its active life. Additionally, experiments reported in [Nehmzow & Smithers 90] have shown that the counterpropagation solution is capable of mapping a higher dimensionality in input space than that obviously provided by a perceptron network alone.

7.3.1. A Mechanism: Implementing a Two-Layer Association Mapping Network

Figure 7.2 shows the addition of a counterpropagation network to a standard AFSM. The counterpropagation algorithm is a combination of two single-layer algorithms. The first layer is a Kohonen self-organising feature map [Kohonen88] and the second is an Outstar network [Grossberg69], similar in many respects to a single-layer perceptron. The counterpropagation algorithm functions as a look-up table that is capable of generalisation, allowing accurate reproduction of the output despite incomplete or noisy input data. The training process associates input vectors with output vectors which may be either binary or continuous values.

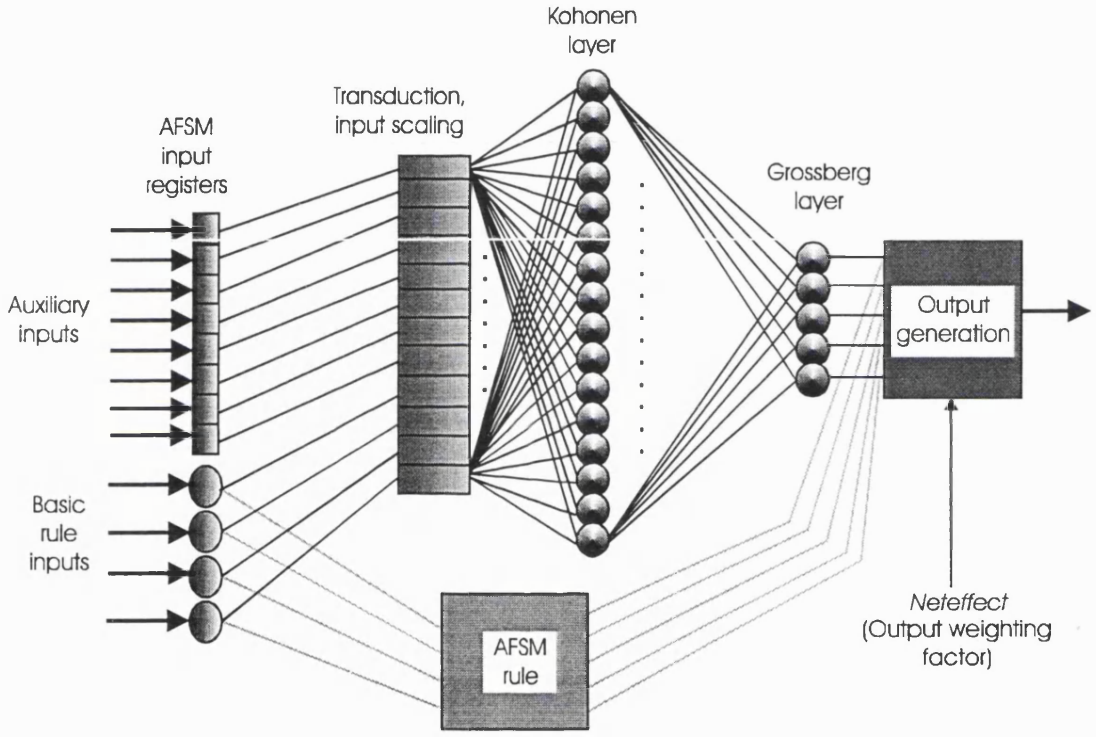


Figure 7.2. An AFSM rule with a counterpropagation network implementation of the association network. Basic input values are routed to the AFSM rule (light grey lines) and the network (black lines); auxiliary inputs are routed to the network only, and are scaled into the range 0-1. In normal response the output of the rule and the output of the Grossberg layer are calculated with output bias being a function of the *neteffect* factor Y. The output generation converts the output vector from the Grossberg-rule combination into an integer AFSM output value.

A number of formats of the main algorithm for the enhanced AFSM process have been experimented with, all utilising the basic format shown in figure 7.2. During every processing cycle of the AFSM the input registers are sampled. A set of key inputs is used to process the AFSM rule and then used further in conjunction with a set of auxiliary inputs to create an input vector for the associative network. This input vector is applied to the network and an output vector generated. Next this output is combined in a weighted way with a similar output vector from the rule and used to look up the AFSM's output action response value which is clocked out onto the wires to other AFSMs or to hardware actuators. Every time an output is generated, the

rule output vector is paired with the network input vector and used to update the weights of the counterpropagation network. This process continues indefinitely until the system is switched off.

The nature of the Kohonen layer and the algorithm for self-organising networks dictates that although small adjustments can be made continuously to fine-tune the network's response, an initial period of high-gain alterations is required to provide a start-up momentum for the self-organising process. During the early stages the network, outputs are ill-defined and most usually noisy. It is during this time that the rule output must provide the main contribution to the AFSM output. However, as time goes by the network becomes settled and is able to take over from the rule. Consequently, the AFSM can be seen to progress through an initial learning phase which relies on the basic rule for AFSM output and then on into an adaptive phase which relies on the network output. It is recommended in [Wasserman89] that to acquire statistically significant clustering of inputs a Kohonen network should be given 500 times the number of cells worth of training cycles. This is currently the time period used in the implementations described here. This is controlled by the maintenance of a variable referred to as the network output weighting factor or *neteffect* which is detailed further below.

The following is a pseudocode listing of the main steps of the algorithm for the enhanced AFSM process. After random initialisation of each of the network layers, the algorithm is repeated for each characteristic time step of the AFSM:

```
Process AFSM rule to get rule vector
Pre-process input vector (i.e. normalise or scale input values to range 0-1)
Apply input to Kohonen layer
Apply situation map vector and rule vector to Grossberg layer to get network output
If AFSM rule is active{
        Update Kohonen layer
        Update Grossberg layer with rule vector
}
Combine rule and network output to generate AFSM output
Output AFSM action value
```

7.3.2. Basic Rule

The basic rule takes a number of key inputs and uses them to generate an output vector R which has l elements, the same number as the output of the counterpropagation network. This vector is scaled and then combined with the Grossberg output layer of the association mapping network in the AFSM output generation function. It acts in the same way as the basic AFSMs detailed in chapters 4 and 5 except that the vector R is used as a lookup reference for the AFSM output register.

7.3.3. Input Scaling

The association network takes as input a vector that includes a set of key inputs as used by the AFSM rule and a set of extra auxiliary inputs as specified by the designer. Real-valued inputs to

Kohonen networks are usually normalised to values in the range 0-1 before being applied. However, the standard normalisation procedure as provided in [Wasserman89] causes the inputs to influence each other in terms of both overall magnitude and changes over time. To avoid this, a simple scaling factor was used for each input. The value of scaling was chosen as an approximate maximum value for each input type so that a scaled input would have a range between 0 and 1. For binary inputs this process is not required.

7.3.4. The Kohonen Layer

Kohonen self-organising networks have been shown to have useful mapping characteristics. The self-organisation of these networks involves the clustering of similar input vectors applied over time. Convergence in the clustering process can be relatively fast, depending on the degree of difference in the input vectors. However, the speed of convergence is also influenced by factors such as the number of cells in the network and the number of distinct clusterings that result from the input.

The following notation is used in this discussion:

X	An input vector with elements x_i
K	The Kohonen network weight matrix with elements k_{ji}, where j indexes the Kohonen elements and i the input vector
Y	An output vector with elements y_j
n	The number of cells in the Kohonen network and output vector Y
m	The number of input elements in the input vector X
G	Kohonen update group vector, a moving subset of K, with elements g_l
u	A variable size of the update group G. Decays to a value of 1 during execution
ξ_K	Gain factor for the weight update in the Kohonen layer

The input-stage layer in Figure 7.2 uses a one-dimensional Kohonen network with a number of weight-storage units in a vector K. The number of units used depends on the required associative mapping. From experimentation it was found that $4l^2$ units was a generally suitable number, where l is the approximate number of input states to be differentiated (effectively the size of the basic rule output vector R). Each unit is a vector with the same number of elements as the input vector and each such vector contains the weights associated with that unit. The weights are updated according to the following scheme (having been initialised to random values in the range 0-1):

1. An input vector (X with elements x_i) is applied to each of the Kohonen-layer units and a measure of appropriateness to the current input is made using a simple dot product of the two vectors. The most active unit is selected and the neighbouring cells in the range +/- $u/2$ in K are selected as members of the update group G.

2. The weights (vector with elements k_{ji}) of the selected unit and the adjacent group of units G are updated according to the following, where ξ_K controls the gain of the weight update and X is the input vector:

$$k_{ji}(t+1) = k_{ji}(t) + \xi_K.[x_i - k_{ji}(t)] \quad (1)$$

3. The update gain ξ_K and the size of the update group u are reduced, and steps 1 and 2 repeated for other input vectors. The group size is reduced gradually during the learning phase until only one unit is selected for updating and the update gain is reduced to a value of 0.1.

The calculation of the output (vector Y with n y_j elements) of the Kohonen layer is made in the same basic way as in many other artificial neural net algorithms:

$$y_j = \sum_{i=1}^{m} k_{ji}.x_i \quad (2)$$

All elements of the output vector Y are reset to zero except for a group of the p most active. This grouping is independent of the group G used in the weight update procedure. A set with size of approximately $p = 0.1n$ of the number of cells in the Kohonen layer was generally found to be the most effective in providing a distinct output signature.

During the learning phase the update group size of the Kohonen layer G_u is reduced from an initial size of half the number of cells in the Kohonen layer to one cell only and ξ_K, the gain in the weight update function, is reduced from an initial value of 0.7 down to 0.1. These figures were selected as being generally suitable after extensive testing of various network configurations and values. Once the update gain has decayed and the group size is reduced to 1, the learning phase is over. During the subsequent adaptive phase the Kohonen layer still adjusts the network weights, but only in small increments, with the learning rate ξ_K being left at a constant 0.1. This provides a continuous and gradual adaptation to any changes in the agent-environment situation.

7.3.5. The Grossberg Layer

The purpose of this layer is to match patterns generated by the Kohonen layer and map them onto an output as trained by the activity of the AFSM rule. The number of units o in the Grossberg layer is matched to the number of basic rule-action states. The basic rule output is in the form of a vector R which has s binary-valued units. An action output from the rule is indicated in the vector R by a unit value of 1 (all others set to 0). The output elements z_l and finally out_l of the Grossberg layer output are calculated in the manner of a single-layer perceptron using the following (from [Wasserman89]):

$$z_l = \sum_{j=1}^{o} w_{lj}.y_j \quad (3)$$

$$out_l = \frac{1}{1+e^{-z_l}} \quad (4)$$

In these equations and below, we have:

Y	An output vector from the Kohonen layer with elements y_i
n	The number of cells in the Kohonen network and output vector Y
W	Grossberg weight matrix with weight elements w_{li}
Z	Grossberg output vector with elements z_l
o	The number of cells in the Grossberg layer W and output vector Z
out	Counterpropagation final output vector with elements out_l
ξ_G	Grossberg layer weight update gain factor. Set to a constant value of 0.2

The weight update function of the Grossberg layer was of the standard single-layer perceptron form with ξ_G, the learning rate constant, set to 0.2 in both learning and adaptive phases. This ensured that the layer responded quickly to continuing changes on the Kohonen layer during the ongoing adaptive phase. At all times the weights in the Grossberg layer were updated using the difference between the current output Z (with elements z_l) and the output from the AFSM rule vector R (with elements r_l) and the input vector Y from the Kohonen layer. The weights j of element l of the Grossberg layer were updated using the following (taken from [Wasserman89]):

$$\Delta w_l = \xi_G.[r_l - z_l].r_l.(1 - r_l) \quad (5)$$

$$w_{lj(t+1)} = w_{lj(t)} + \Delta w_l.y_j \quad (6)$$

7.3.6. AFSM Output Generation

The output is a function of both AFSM basic rule output and the counterpropagation network output (see figure 7.2). The influence of each depends on the phase of the system (learning or adaptive) and this is indicated by a variable *neteffect* which provides an output weighting factor Ψ. This value grows exponentially from 0 at start-up to a value of 1 at the end of the learning phase (see the next section for further details) and is indicative of the output influence of the association network. The AFSM output vector *AFSMout* is a weighted combination of the rule vector R (with elements r_l) and the counterpropagation output vector *out* (with elements out_l) with values in the range 0-1 and is calculated for each element l as follows:

$$AFSMout_l = (1 - \psi).r_l + \psi.out_l \quad (7)$$

The AFSMout vector is used to select the actual AFSM output. This is the unit with the highest activation that is greater than a threshold of 0.5.

7.3.7. When to Learn and When to Adapt

The transition from an initial learning phase to that of adaptation is controlled by the growth of the *neteffect* variable referred to as the network output weighting factor (Ψ above). The rate of growth of this variable is an important factor in the performance and development of the AFSM and as such has been the focus of considerable experimentation. The reason for introducing the

weighting factor is that Pfeifer and Verschure's work was based on modelling the phenomenon of classical conditioning, which relies on a temporal difference between the basic and associated actions. It relied on the associated response occurring before the built-in response. The decay used here ensures that the associated action becomes the dominant output whether or not there is a timing factor involved. The network output weighting factor Ψ is initialised at a value of 0 and as time passes is gradually updated after every learning event according to:

$$\psi = \frac{e^{\left(\frac{f.t}{T}\right)}}{T}, \tag{8}$$

where T is the time period (number of events) selected for the learning phase, t is a specific event number in the range [1,T], and f is a scaling factor to ensure a growth from 0 to 1 in the time period. f is initialised as the natural logarithm of T. This simple non-linear weighting function, used by the AFSM output generation function to calculate the AFSM output (described above in section 7.3.6), was chosen because it was found from experimentation that the self-organisation of the Kohonen network layer did not progress in a linear fashion. It was the case that although the update group size u and update adjustment gain ξ_K were reduced linearly, the output accuracy of the Kohonen layer did not behave accordingly. The graph in figure 7.3a shows a plot on a simple incremental time base with a T value of 1000, as would occur if the system were to update the network at every time step (which does not in fact happen; see below). The continuous plot depicts the decay of the update group size of the Kohonen layer and the gain of the update function while the grey lines indicate the value of the network weighting factor. Once the network weighting value becomes 1, the learning phase is over and the system remains in the adaptive phase. Note that the Kohonen learning gain does not decay to 0 but remains at a value of 0.1.

The plot in figure 7.3a, however, is not the whole story. It is often the case that an AFSM in a subsumption control structure will not trigger an output signal (or in the case of our implementation, a rule output vector containing only zeroes is generated). When this happens it is often useful to set the AFSM either to output the previous signal state or to generate an output at random. The justification for the choice is that this will prevent the system from falling into a deadlock, as often in real-world situations doing something is better than doing nothing. This is particularly the situation in mobile-robot work where an extra movement may get the vehicle out of a sticky situation such as a particularly tight dead-ended passageway. This technique has also proved useful in the development of the laser scanner test-bed control programs. This trick, though, presents a potential problem for the learning of the condition part of the rule. If the AFSM is occasionally outputting random signals, any network learning algorithm presented with these atypical input→output pairings will fail to obtain convergent values for the network weights. Consequently the identification of a suitable range of situations for learning is necessary. It may be argued that these extra and sporadic AFSM outputs are beneficial and that they should also be learned, since this might provide an impressive ability to learn and recall occasional

beneficial events. However, this would require considerable computation and the holding of an extra volume of data and would run the risk of overloading the AFSM process (given the current processing resources). It is also a complex extension, and as such would unduly increase the complexity of this initial work.

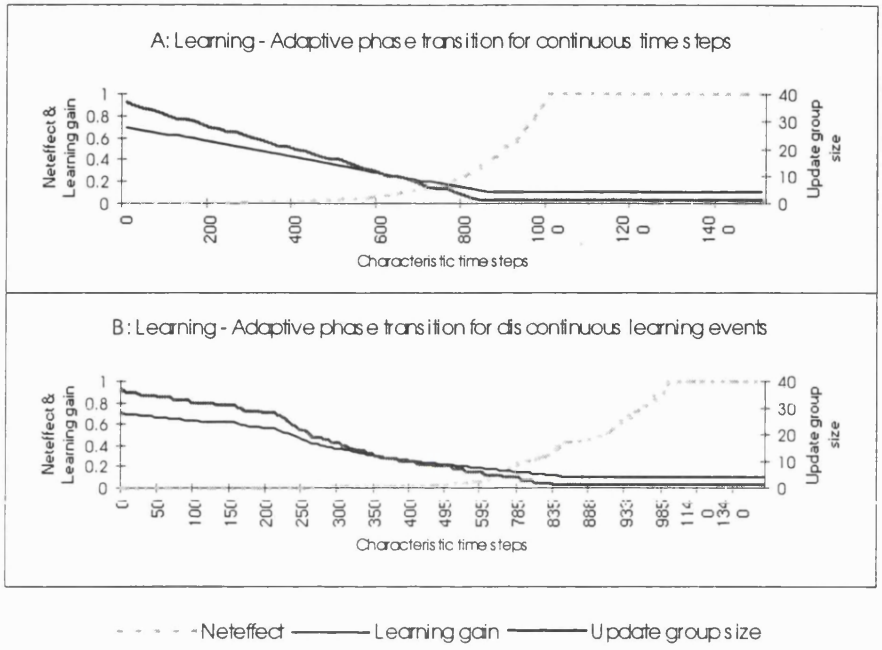


Figure 7.3. Plot showing a continuous transition from learning to adaptive phase: A) assuming that every characteristic time event of an AFSM provides a valid output state and B) in the real sampled case of non-continuous AFSM events. The drop-off around 121 steps resulted from a reduction in stimulation frequency from the test surface (when the surface was changed for one with fewer breaks).

Thus the issue regarding when to learn is apparently clear. It is when the AFSM rule makes a positive state identification. However, one further problem now arises. If the agent-environment situation is relatively constant, then it is likely that the AFSM will spend more time learning a limited number of input-output mappings during the learning phase. This problem can be partially solved by setting the learning event to occur only when the AFSM rule output changes state and not every time a signal is output that is the same as at the previous time step. Obviously this only solves the problem partially. If the system is not presented with a sufficiently diverse and interesting set of environmental conditions during the learning phase, it cannot be expected to cope with variation later in its operational life despite limited continuous adaptation of input-output mappings during the adaptive phase.

So now the issue of when to learn is less apparent. Should the AFSM input mapping be learned every time the rule makes a state identification or only when it identifies a new state? It turns out in our experiments that the most effective solution is to keep things simple and implement a learning event every time the AFSM rule identifies a particular state. This is mainly because of the nature of the rule and the identification of real-world states. It is possible that short transitive conditions may trigger a change of AFSM state but not necessarily be the best

examples of that state for the purposes of learning. Learning only these transitions is obviously not a good thing. The other factor that is significant here is that no matter how careful the designer is in setting up conditions for learning to occur, this is all made irrelevant if the real-world does not present enough variation during the learning phase.

The final result is that the time taken for the learning phase to pass into the adaptive phase is determined not by an internal time constant but by the activity of the complete agent-environment system. AFSMs that are connected to sensors and actuators will have their learning rates dictated by the frequency of environmental events and their activity. Some AFSMs will be completely embedded in amongst others and so in this case the timing of the learning phase will be determined by internal activity as well as (ultimately) external activity from agent-environment interaction. For the initial learning period it is helpful if the system is presented with as rich a set of input stimuli as possible, since it is at this time that the system is maximally responsive to learning various input characteristics. This will not only ensure a more thorough repertoire of learned input conditions but will also allow a faster rate of learning since the number of learning events required will pass more quickly. The difference between the learning rates of two AFSMs is evident in the examples in the next chapter. One, being more active than the other, generally goes into its adaptive phase much more quickly than the other. Figure 7.3b shows a more realistic example of the growth of the network output weighting factor.

7.3.8. What to Learn

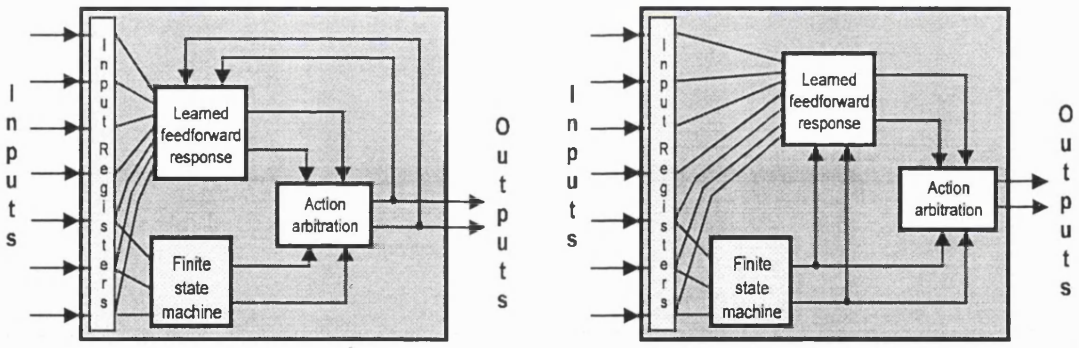


Figure 7.4. Illustration of two different schemes for updating network weights. At left is the feedback of the complete weighted AFSM output. The feedback of the rule output only is shown on the right.

There remains one aspect of the input-output association learning mechanism that has not been addressed: the source of the output that is used as part of the input-output pair during the updating of the Grossberg network weights. This may originate from one of two places; either the direct output from the basic rule or the combined rule-network output that is the final AFSM output (see figure 7.4). The second of these would start off as the basic rule output but, as the AFSM gained experience and the output weighting factor grew, the source of the network update

would change to that of the network output. Experimentation has shown that the usage of the weighted AFSM output may result in a better immediate performance level than if the basic rule output only is used. However, gradually over a period of time, the performance degrades and the AFSM output becomes incoherent. Conversely, although the basic rule may not achieve such high initial performance, it is generally the case that the system maintains its ability indefinitely once the adaptive phase is entered.

On analysis, the reasons for these phenomena are relatively straightforward. In the first learning case, the initial network output is trained from a combination of the basic rule that is combined with an increasingly superior network output (as will be demonstrated in the next chapter). Eventually the network output takes over the reinforcement activity completely. Over time, however, activity in the network that classifies situations differently from the rule causes the weight clusterings of the Kohonen layer to lose their sharpness so that the Grossberg layer is unable to differentiate them. On the other hand, the constant use of the rule output for providing the weight update irrespective of the "age" of the AFSM provides a stable response to environmental situations and guarantees a minimum level of performance which is continually reinforcing the network layers in the same way, throughout the life of the AFSM. It is the case that the removal of the rule output from the source of training stimuli results in a network for which the fixed-point of the learning dynamics is the state in which it receives maximum reward. When used as the source of training, the rule acts to prevent this fixed-point convergence by maintaining the response to the AFSM's basic inputs.

7.4. Tuning the Output Registers: Continuous Adaptation by Hill Climbing

The previous section dealt with the provision of an adaptable input-condition mapping mechanism for a subsumption AFSM. The counterpropagation network was intended to provide an enhanced decoding of the AFSM input state learning to incorporate additional inputs over and above those required for the basic rule functionality. This mechanism, however, only served to increase the resolution of the action selection and did not manage to affect the response action at all, other than on the particular issue of when it was selected. This section looks at a scheme that attempts to provide the AFSM output values with a degree of adaptability or automatic tuning. Again, as before, the main reasoning behind our approach is that the designer can only make an approximation of the correct values of the outputs, or the correct adjustment factors used to modify the outputs.

This requirement for adaptability was introduced at the beginning of the present chapter, and is a result of the design task that necessitates the provision of implicit domain knowledge to provide the AFSM with an effective actuation (whether the output is connected to other AFSM inputs or directly to an actuator). An example of the criticality of how this is done can be seen in

the *BallPark* AFSM of the laser scanner control structure of chapter 5. The operation of this AFSM is to identify a number of photomultiplier states that are outside the basic near-linear state. Depending on the identified state the *BallPark* AFSM outputs a value that causes the output EHT voltage to be increased by a fixed step size. The output values for each state have optimum values that depend on the characteristics of a particular photomultiplier and its response to incoming light. Consequently the EHT step size associated with the quiescent signal characteristic might be optimised so that the near-linear region is reached as quickly as possible. If the step size is too big, overshoot will probably occur; if it is too small, the signal will take too long to adjust. As a photomultiplier ages, its response characteristics change. Thus, additionally, a system should be able to adapt to these changing circumstances, which are of a more long-term nature than the speedy response provided by normal control techniques.

The concepts presented in this section are intended only to serve as an indication of a desirable solution. The work so far in this area has been limited to simple experimentation with some initial ideas. It is nevertheless included in the this part of the dissertation to complete the story, so to speak, of adding adaptivity to the basic building blocks of a subsumption architecture and thereby introducing these mechanisms at the lowest possible level of an agent's control structure.

7.4.1. An Adaptation Mechanism

The output of an AFSM takes the form of an execution of the command *output(register)* where the register is one of the AFSM output connections as declared in the program code (see appendix A). The value of the register may be constant (e.g. it may simply be a binary output indicating a true condition) or it may be modified in some way depending on the outcome of the AFSM rule. In the first case the constant value is determined by the designer and is most often the target of some empirical tuning. In the second case the update function that affects the output register generally utilises some constant (designer-provided) value. The purpose of the mechanism outlined here is to adapt this constant value, as initially set by the designer, on a continuous basis with the adaptation being dependent on some form of performance measure or reinforcement value. In other words, the problem is one of optimising the register value such that the maximum reinforcement is attained over time. Reinforcement optimisation is dealt with in many sources, in terms of control in [Miller.W *et al* 90] and [Harris94] and in mobile robots [Mataric95], [Mahadevan & Connell 92] and [Sutton91]. It is also treated in terms of heuristic search problems in sources such as [Luger & Stubblefield 89] and [Charniak & McDermott 85] which outline hill-climbing search as one of the simplest and earliest optimisation algorithms. Despite its well-known limitations, hill climbing has several advantages, and with some modification (for example [Maza & Yuret 94] and [Cvijovic & Klinowski 95]) it has been made into a more robust technique.

The nature of the problem dealt with here is one of tuning, or optimising, a single parameter over a long time period. Although an AFSM will typically contain many "magic constants", all of

which may benefit from dynamic optimisation, the potentially n-dimensional optimisation space can be reduced to a direct product of one-dimensional spaces by treating each constant individually in time. In other words, time-slicing the AFSM optimisation problem and dealing with one dimension at each instant. The solution to the problem outlined here relies on the fact that the adaptation of the AFSM output registers can take place over considerable time periods (determined ultimately by the speed and nature of environmental changes) that may be in the order of tens of minutes or more, so that the tuning of each individual register in an AFSM will occur in a sequence with each register taking its turn.

The mechanism presented here is based loosely on a steepest-gradient hill-climbing optimisation process which works by making a change to a parameter and then monitoring the effects of making that change. If the situation improves, then the change is kept and a further change in the same direction is made for further adaptation. If the situation worsens, then the change is retracted and the direction of adaptation altered. In a one-dimensional space this is relatively straightforward. The main criteria that dictate the nature of the output register optimisation process are the following:

- The optimisation function must be computationally cheap both in terms of time and resource usage. An AFSM is supposed to be a relatively simple process used as a low-level building block for higher-level control functionality and there are likely to be many such optimisation processes going on in many of the AFSMs in the system.
- Real-world factors must be taken into account such as the frequency of activity of the AFSM. If an AFSM is inactive for a considerable proportion of time, any tuning or optimisation to an output register might end up being based on other system dynamics and not the result of the local alteration of parameters.
- A reinforcement feedback is required in order to provide an "assessment" of the AFSM's performance resulting from an adjustment of a parameter. In a distributed system, global reinforcement is not always possible. Ideally therefore, an AFSM should be able to generate its own reinforcement value internally depending on its own assessment of performance.

For the dynamic adaptation of a single output register the following is executed after each count of a preset number of AFSM characteristic time steps referred to as the adjustment period. The reinforcement period is the time span over which the reinforcement value is accumulated. The value of this parameter is the result of a trade-off between filtering out short-term transitive events and being able to respond to changes (whether beneficial or not) in time to maintain the system's performance. The basic computation is as follows:

```
Repeat after each adjustment_period{
        Calculate reinforcement value for last reinforcement_period
        If reinforcement has gone down since value was adjusted{
                Reset to original value
                Reverse direction of adjustment
        }Else reinforcement has gone up{
                Save new value as a function of register mean activity
        }
        Make new adjustment to register value as a function of tuning_factor, current value,
                                        mean reinforcement and mean register activity
}
```

The above sequence is executed with a frequency that is preset by the designer, not at every characteristic time step. The period must be set such that sufficient time is provided to monitor the change in system performance that is a result of adjustment to the value of the register in question. This algorithm takes a number of factors into account: (i) The size of the trial adjustment, in the first instance, is determined by a preset parameter as a percentage of the original value in the register; (ii) The size of the trial adjustment is also dependent on the size of the reinforcement signal (which is always in the range 0-1). The larger the reinforcement, the smaller the change made. This is to ensure that the method homes in on an optimum value and does not oscillate around it; (iii) In the case of an increase in reinforcement the actual change to the register value is not the full amount of the initial change but a proportion depending on the activity of the register during the time period of the adjustment trial. This eliminates, as far as possible, the number of changes that are based on effects of other processes and events in the agent-environment system. The expression for a single register trial update tv is:

$$tv = rv + f.rv.(1-r) \tag{9}$$

The actual register value rv update, given an increase in reinforcement, is:

$$rv = tv.\eta \tag{10}$$

where:

rv = Current value in the register

tv = Trial value in the register

h = Mean activation adjustment during last period t_a $(0 < h < 1)$

f = Update factor $(0 < f < 1)$

r = Mean reinforcement over last reinforcement period t_r $(0 < r < 1)$

Further, the information required to maintain this process for one adaptive register is:

Mean activity:	Variable holding register activity during adjustment period
Original register value:	Variable holding previous best register value before adjustment is made
New register value:	Variable holding current register value including trial adjustment
Reinforcement:	Variable holding previous reinforcement associated with original register value

The following three parameters are used to tune the update characteristics:

Reinforcement period t_r:	Time span over which reinforcement is accumulated
Adjustment period t_a:	Number of AFSM characteristic time steps over which adjustments are tested
Update factor f:	Percentage of current register value for test alteration.

The values assigned to these three parameters must be tuned for the environment in which the system is intended to operate (but they are of a fairly general nature and it will be seen in chapter 8 that the initialisation is relatively simple). For example, an inspection surface with gaps and spaces will mean that a reinforcement value of 1 (100%) is impossible to achieve. Consequently a higher-valued update factor may be required to ensure that register updates are of a sufficient magnitude to have an effect on the system's behaviour. In addition, the reinforcement and adjustment periods must reflect the periodicity and rates of change of the inspection surface.

The pseudocode above dealt with a single output register value. However, it is likely that an AFSM will contain multiples of such registers. To deal with this situation each adaptive output register is given one adjustment event in turn. Consequently the set of variables listed above must be maintained for each output register along with a pointer to indicate the output register currently under adjustment. The result is that the AFSM output registers are adjusted one by one, with changes being made that take into account the activity of the AFSM and the optimality of the current register value in terms of a reinforcement value. It is up to the designer to preset the time interval between adjustment events, although this might be adapted dynamically by a higher-level subsumption control layer in a future implementation. Chapter 9 discusses further additions to this work.

7.4.2. Reinforcement Functions

A reinforcement value is an arbitrary number that ranges from an indication of "good" to "bad". The particular representation varies from implementation to implementation. For example, in [Mahadevan & Connell 92] values of the range +/-3 were used, -3 being a sign of negative reinforcement and +3 positive. In the work reported here a binary indication was used. At each characteristic time step a 1 was used to indicate a good situation with any other situation (bad or indifferent) being indicated by 0. As a result, an average reinforcement value could be calculated, generating a real number between 0 and 1. This fraction provided an indication of the proportion of time in which the system was in a good state; e.g. an average of 0.8 indicated than for 80% of the sample time the system was in a good state.

As already indicated above, the source of a reinforcement value in a distributed system is not straightforward. It is also the case that the reinforcement must be related directly to the performance of the AFSM at the level of a community of interacting AFSM processes rather than entirely at the level of the physical agent-environment system. Since the relationship between real-world events and the action of a particular AFSM in the agent control structure is at best tenuous, it was necessary to base the generation of reinforcement on internal AFSM activity alone. For example, the *BallPark* AFSM is responsible for driving the photomultiplier into the near-linear condition. It may happen that a change in the environment, and not any actuation of *BallPark,* results in the desired near-linear photomultiplier state being achieved. Alternatively, another part of the system such as the *NearLin* AFSM may act to achieve this state. Consequently any

reinforcement that is generated must be grounded in the activities of the AFSM to which it is being applied. It may be that in the future more complex tactics might be used in the form of weighted reinforcement from external sources. But in general the weighting would have to reflect the closeness (or conversely the remoteness) of the reinforcement source and take into consideration other possible contributions towards the reinforcement value - something that is difficult to quantify.

We have thus generated the reinforcement value for an AFSM by setting a flag every time the process goes into a pre-nominated state and then maintaining a rolling average of the reinforcement level over a preset period (reinforcement period t_r above). This has resulted in a reinforcement value with a range of 0-1. For example, the *BallPark* AFSM in the near-linear signal region is an indication that the AFSM has succeeded in its local task. Consequently every time the AFSM is in the near-linear state a reinforcement value of 1 is incorporated into the mean reinforcement value for the register adaptation function. This mechanism is easily and cheaply incorporated into the existing AFSM rule mechanism, and in fact was very similar to the method used in chapter 5 for providing a means of quantifying the performance of various control structures. Issues of experimentation arose in order to assess the relative merits of using the main network-driven AFSM output or the output of the basic rule only. It was the case that the basic rule, being a designer-defined function, was the most stable source of reinforcement and so provided the most useful solution. This was mainly because of the relative rather than absolute nature of the reinforcement value and the fact that the behaviour of the designed-in basic AFSM rule remained unaltered throughout the system's life, resulting in a constant source of system self-assessment.

7.5. Complete AFSM Input-Output: Feedforward Counterpropagation

Finally in this chapter a further possible solution to the problem of building learning and adaptivity into a subsumption architecture AFSM process is detailed. It is in fact little more than an extension of the mechanism presented in section 7.3. Rather than using the artificial neural network approach to learn only the AFSM input state mapping as an action selection (or state identification) mechanism, the output of the network is used here to provide directly a continuous/real value AFSM register output. This output is weighted by the *neteffect* variable in the same way as that above, but here the real value is used as the output rather than the contents of an output register. During learning events the actual output value from the AFSM basic rule is used rather than the state identification value that was previously used to activate the output register. In this way the network learns to provide a complete input-output AFSM state mapping. Possible advantages here are that: (i) the real output values from the network will reflect to some extent a continuous mapping of output space and (ii) a simpler enhanced AFSM mechanism is

used compared to that of the combined enhanced input and adaptive output mechanisms of the previous two sections.

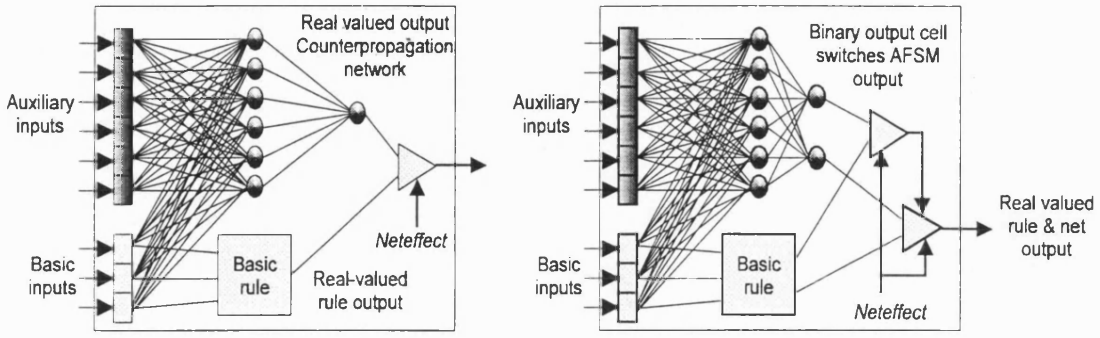


Figure 7.5. A complete network solution to the AFSM enhancement problem. A: A continuous output solution and B: An activated output, with occasional output being enabled by a second output network cell.

An AFSM Architecture

Figure 7.5a shows the basic architecture for this format of AFSM. The same two-layer counterpropagation network as outlined in section 7.3 was used but the output Grossberg perceptron layer has been replaced with a single continuous-valued output cell. During learning, the output value from the basic rule is presented to the network whenever the rule has been triggered positively. The *neteffect* parameter provides a similar weighting of output, but rather than weighting the selection of output register it provides a direct weighting of output value. Again an exponential output function was used on the output-layer cells and this necessitated the provision of a transfer function to translate the network output from its normal range of 0-1 into the required AFSM output range. This function is application-dependent and must be set up by the application designer. For example, in order to provide a range between a negative minimum and positive maximum the following function may be applied to the output of a network where m is the positive maximum value of the output range and out_j is the output of the perceptron layer cell j (in this case there is only one) generated by equation 4 above:

$$action = 2\mu.(out_j - 0.5) \tag{11}$$

The *action* output value is then weighted with the *neteffecty*parameter and the rule output r as follows:

$$afsm\_op = \psi.action + (1-\psi).r \tag{12}$$

The use of a single cell in the output layer is only suitable if the AFSM provides actuation at all characteristic time steps; in other words, if at all times a register output is transmitted onto the AFSM output wire. However, this is not often the case. For example, when an AFSM subsumes a lower level with a suppressor connection then it must only make an output at particular relevant events. In section 7.3 this aspect was not an issue since the counterpropagation network was learning a mapping of input space which was effectively an output-selection mechanism and a

"non-output" was just one of the set of possible actions to be selected. In order to allow for this here, a second output cell is added to the perceptron network, the job of which is to act as an output activation switch to the other output cell (see figure 7.6b). This is treated in the same way as the other perceptron output cell but the output function is a simple binary 0 or 1. Consequently the output processing of equation 9 is not needed, although the output is still weighted by *neteffect* to arbitrate between a similar rule-generated activation variable and the network output.

The Learning Phase

The learning of the output layer is, in the same way as that of section 7.3, a continuous process with a constant learning-rate factor. Equations 5 and 6 above are again used. However, an inverse output transfer function is required to translate the rule output into the required network output.

$$training\_op = 0.5 + \frac{rule\_op}{2\mu} \tag{13}$$

In the case of the AFSM activation output a binary training value is used in exactly the same way as in section 7.3.5.

7.6. Summary

This chapter started by bringing to attention the requirement that a behaviour based controller should have mechanisms of adaptation at the lowest levels. A need for the self-organisation of these systems was identified to enhance the domain knowledge and structure provided by a designer. It was suggested that various methods of learning and optimisation could be useful in this respect. The first two main sections then presented a solution to these problems with attention centred around the basic rule and structure of a subsumption AFSM. The condition and action part of the rule were treated separately with an artificial neural network being used to learn and then adapt a feedforward mapping of the rule's condition. A simple optimisation strategy based on hill climbing was used for the action part of the rule. The third main section then dealt with a network-only implementation of the AFSM input-output mapping, replacing the output registers with a single continuous-valued output network layer. Issues of when and what to learn were also discussed, with particular emphasis on the problems associated with real-world situations.

The next chapter outlines a series of experiments on the use of the above ideas on the laser scanning test-bed (described in chapter 4). These experiments deal with issues including the location of the enhanced AFSM processes and the integration of these devices within a standard subsumption control structure based on that developed in chapter 5. A gradual and incremental approach is followed with each of the above mechanisms being added in turn and compared with the basic system.

8. Experiments in Embedded Learning and Adaptation

This chapter covers the following:

Experiments on learning AFSM input state recognition.

Experiments concerning choice of adaptive actuation parameters.

Enhanced-adaptive AFSMs and feedforward networks.

The previous chapter suggested a number of additions to the augmented finite state machine (AFSM) processes of the subsumption architecture which were intended to provide low-level learning of input mappings and continuous adaptation of actuation parameters. This chapter deals with the testing of these ideas in the form of a series of experiments that use the laser-scanning test-bed from chapter 4 and a basic subsumption control structure that is adapted from the one described in chapter 5. The experiments reported here serve to illustrate the characteristics and performance of the laser-scanner system controlled by the new enhanced and adaptive AFSM processes. Their results are compared with the behaviour of the system reported previously in chapter 5.

Firstly, this chapter deals with the issues of applying the counterpropagation networks, which were presented in the previous chapter, to the inputs of AFSMs within the photomultiplier control structure that was the focus of chapter 5. In the first instance only one AFSM, *BallPark*, is modified and tested in experiments. The issues of when and what to learn that were discussed in the previous chapter are illustrated before we proceed to a description of an implementation of multiple enhanced AFSMs in section 8.1.2. The second part of this chapter details the application of the action output optimisation mechanism, also described in the previous chapter. Here again at first only one AFSM is modified, but then subsequently others. In the final series of experiments reported here, these two mechanisms are combined and the resulting control structure compared with that of the continuous output network solution detailed at the end of chapter 7 in section 7.5. Results from each set of experiments are discussed as the chapter progresses, and factors and assumptions affecting each subsequent set of experiments are outlined. At the end of this chapter, in section 8.4, results from the series of experiments are given along with some application-related conclusions and discussion. Discussion regarding the techniques in general is saved until chapter 9.

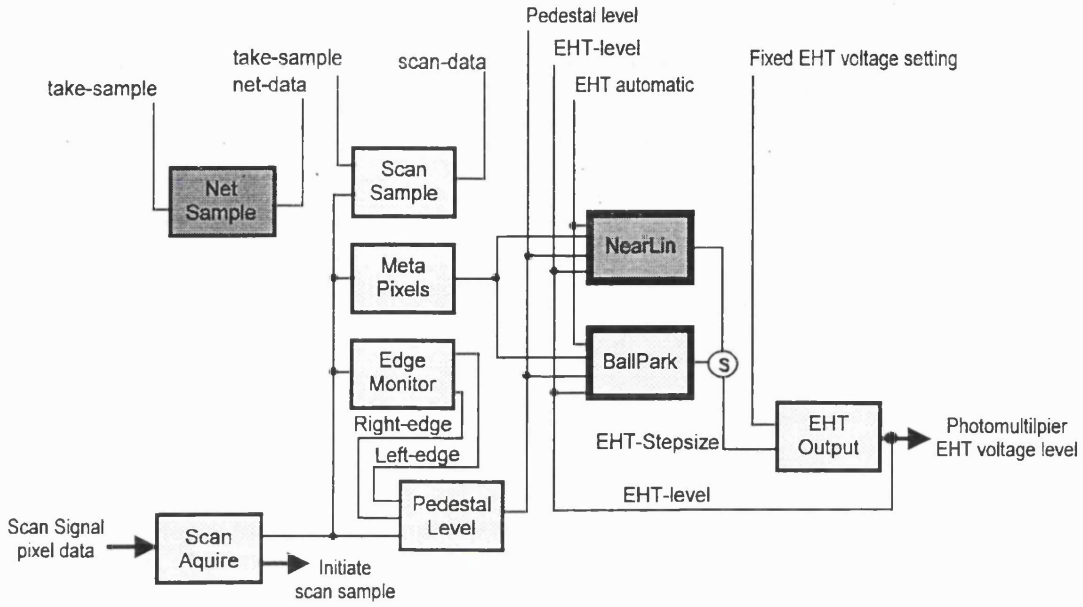


Figure 8.1. The control structure used for a single photomultiplier channel and as the basic framework for the experiments reported in this chapter. Note the insertion of the level-0 *MetaPixelGen* AFSM, added to format scan-stripe pixel data for higher-level AFSMs.

Figure 8.1 shows the basic subsumption control structure for the single photomultiplier channel that was used previously in chapter 5. In this current chapter the *BallPark* and *NearLin* AFSMs are the focus of further development and are used as the main targets for experimentation. To support the enhancements, which deal in part with an expanded input space, a low-level AFSM has been added as part of the level-0 scan-acquisition behavioural node. This is the *MetaPixelGen* AFSM. It is introduced below. This architecture was tested in its various configurations on the same set of surfaces that were used in chapter 5. These are shown again in figure 8.2.

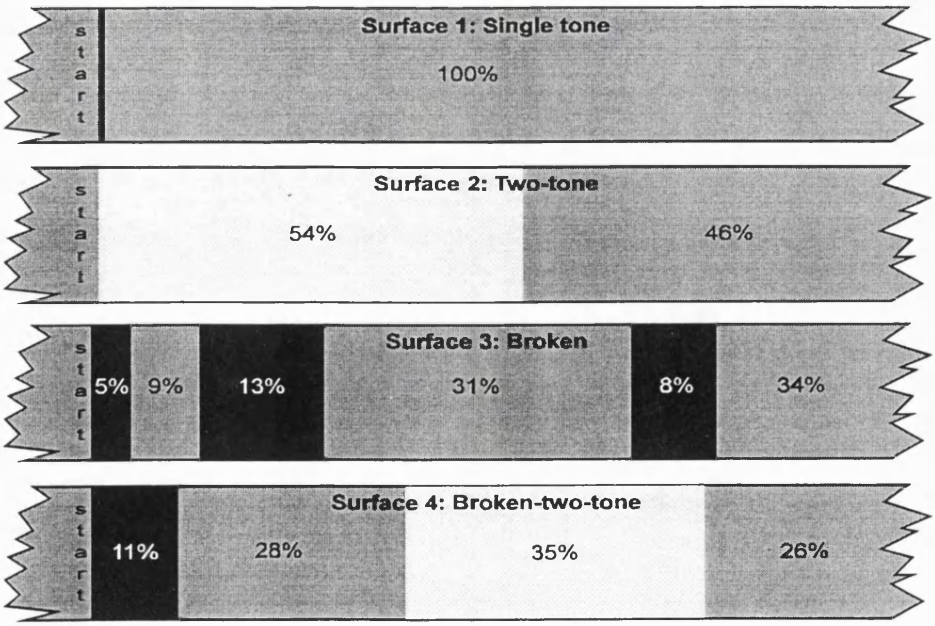


Figure 8.2. Characteristics of the surfaces used on the rotating test-bed.

8.1. Experiments in Learning Enhanced State Recognition

This section outlines the implementation and set-up of a series of experiments that test the utilisation of learning a feedforward association mapping of an AFSM input. The first part details the enhancement of the *BallPark* AFSM. The second part of this section details the additional enhancement of the *NearLin* AFSM and thus addresses issues concerning the performance and stability of incorporating more than one enhanced AFSM process into an asynchronous control structure.

The Experiments

A number of experiments were run in both single enhanced AFSM and multi-enhanced AFSM control configurations to test the following:

1. Effects of different learning-phase interactions.
    1. Broken surface stimulation using test surface 3 (figure 8.2).
    2. Continuous surface stimulation; test surface 2 (figure 8.2).
2. Effects of introducing different surfaces other than those used during learning. These different surfaces are taken from the set shown in figure 8.2.
3. Effects of removing AFSM inputs (simulating system breakdown).
    1. No basic rule inputs
    2. No auxiliary inputs
    3. 1 in 2 scan pixels missing
    4. 1 in 3 scan pixels missing
    5. 1 in 4 scan pixels missing
    6. 1 in 5 scan pixels missing
    7. 1 in 6 scan pixels missing
    8. 1 in 8 scan pixels missing
    9. 1 in 11 scan pixels missing

For each set of experiments a control run was done using the non-enhanced AFSM architecture from chapter 5 (figure 8.1). The data recorded was then used in order to quantify differences in performance of the new configurations.

Firstly, the system was trained with an intuitively-chosen surface, one that provided the highest frequency of variation to the photomultiplier. This was the broken-surface sample of drum 3. Once the learning phase was complete, the system's performance with missing inputs was tested according to the patterns above. The system was then tested with the other surfaces: continuous, two-tone and broken two-tone. Next the system was retrained using the two-tone surface sample of drum 2 and then tested with the different continuous, broken and broken two-tone surfaces of drums 1, 3, and 4. After each alteration of system configuration a period of at least 1000 characteristic time steps (approximately 1 minute, given a characteristic update rate of 15 Hz) was provided in order for the system to have time to settle in to the different set of circumstances.

During the tests a continuous measurement of *BallPark* and *NearLin* AFSM output state was recorded to provide comparative data in the same way as that used in chapter 5. Further, three

recordings of the various weight vectors in the *BallPark* network were made, once when the system was first started, secondly after the learning phase was over and then finally after a further period equal to that of the learning phase (the actual time of which depended on the stimulation and consequently the activity of the AFSM in question). This scheme was adopted in order to examine any further adaptation that occurred after the learning phase was over. Finally, at intervals during the system's run-time (including initialisation, after learning phase, after a period of stable adaptation and then once for each of the experimental changes in input and surface characteristics) an intensive set of 1000 samples was made over a period of approximately 4000 characteristic time steps (approximately 4 minutes) in which the following system parameters were recorded:

EHT voltage output level,
Photomultiplier scan-level,
BallPark AFSM rule output ,
BallPark AFSM output (including network contribution),
BallPark net output weighting,
BallPark network input vector,
NearLin AFSM rule output ,
NearLin AFSM output (including network contribution),
NearLin net output weighting,
NearLin network input vector.

Assessing the Results

In chapter 5 the results of the behaviour based laser-scanner control implementations were presented in terms of the scan-level output of a photomultiplier channel. A region of +/-0.06V was used and a count was maintained during runtime for every characteristic time step that the *NearLin* AFSM occupied in this "signal in range" state. Because the control solutions in this chapter are significantly more complex in terms of the number of factors affecting the system's performance, we selected several different measures for use that focus on two levels of system activity.

At the level of individual AFSMs, namely *BallPark* and *NearLin*, the values of reinforcement that are built into the adaptive output mechanisms (chapter 7.4) are used as local measures of AFSM performance. This applies even in the experiments that do not utilise a reinforcement function directly during their operation because the value is easy to compute and provides a useful indication of the system's behaviour. However, it will be seen below that these measures of the quantity of reinforcement alone are not completely adequate as indicators of performance. These values typically do not reflect aspects such as the stability and usefulness of the photomultiplier scan signal, or in other words the degree to which the system is succeeding at its overall task. A measure or measures that indicate the performance at the level of the performance of the complete photomultiplier subsystem are required. The most obvious choice here is to use the characteristics of the scan-level signal itself; a less obvious source of information is in the nature of the EHT voltage output from the level-0 *EHTOutput* AFSM.

A system-level analysis of the photomultiplier scan-level signal is not as straightforward as at first might be envisaged. As a single measure over a sample period the mean of the scan-level may be calculated. This will be closer to 1 volt for the more successful systems and so provides a useful indicator of general system performance. However, this figure will not take into account extremes of signal variation. The scan-level signal is in effect an incoming pattern of peaks and troughs that indicate the amount of laser light collected by the optical components on the front-end of the photomultiplier tube. While the control AFSMs are attempting to maintain a fixed-point signal level for the scan stripe at 1 volt, activity in the outside world almost guarantees that this level is not maintained for long. Consequently the dangers in using the level of the scan signal as a measure of system performance manifest themselves in the fact that the signal emanates from a sensor device, the purpose of which is to reflect activity in the world and not an internal state. Any change in signal level is most likely to be a result of unpredictable external activity and not directly of system actuation. Even processing such data to acquire the standard deviation of a sample does not really provide any useful guide. Although it could be claimed that a smaller standard deviation would indicate a more stable signal, this would only be true if the environment were stable. Since this was not the case with the set-up used in these experiments the standard deviation of scan-level signal was not used as a measure of system performance.

Additionally then, the voltage output from the *EHTOutput* AFSM was monitored. While the mean level of the scan signal did prove useful to a limited extent, the mean level of the output voltage was relatively meaningless (except perhaps as an indirect indication of ambient light level). The standard deviation, however, is more useful as it provided an indication of the extent to which the EHT voltage was being adjusted by the controller about the mean level (whatever that might be). By monitoring the standard deviation of the output over time it is possible to gain a better idea of the stability of the system, with a smaller standard deviation typifying a more stable system. This can be likened to a hyperactive characteristic at one extreme and that of being lazy (with the appropriate benefits) at the other. So long as the scan-level signal is maintained, it can be appreciated that the lazy system will be doing the job more efficiently than a hyperactive one. In conclusion, however, it must be pointed out that care must be exercised when using the standard deviation of system activity as a measure of system performance. It is the case that a good controller may exhibit a high standard deviation when it is coping with a complex and changing situation while a poor controller might have a very low standard deviation when confronting an unchanging one. Clearly a like for like comparison must be made, to account for the systems situation. In the case of the laser scanner this means comparing the performance of different control solutions on the same test-bed surface sample (figure 8.2).

The measures used to report the results of this set of experiments are thus: (i) *BallPark* reinforcement over time, (ii) *NearLin* reinforcement over time, (iii) Mean scan signal level and (iv) The standard deviation of the EHT voltage output. Other plots and graphs will be used to

show time series data and state-space plots of system activities to enhance the quantitative evidence of the above listed factors.

Relative Performance and Reinforcement Values

Finally it should be emphasised that for some of the result formats, such as the use of a reinforcement value, a 100% success rate was not achievable. This applied to the tests run on the broken and two-tone surface samples which contained blank or missing areas that did not reflect the laser light. Surface D had 27% of its area blanked out and so in consequence the best that any controller applied to this problem could hope to achieve was a result of 73% surface in view. For the *BallPark* AFSM this generally corresponded to a reinforcement value of around 0.73, so the values of more than 0.5 witnessed in these tests are reasonably impressive. Allowing for the blank spaces, this is in fact a success rate of around 70%. Given this, it can also be appreciated better that the *NearLin* AFSM which, only being able to function in the near-linear signal region, and having a reinforcement performance of 0.3, was in fact performing well with a success rate of around 60%. In addition, allowing that the reinforcement was given when the signal was within a region of +/- 0.06Volts either side of the required 1-volt signal set-point, this result is encouraging.

8.1.1. The Use of A Single Enhanced AFSM

In this set of experiments the *BallPark* AFSM of the subsumption control structure of the photomultiplier was enhanced with an associative network to learn the feedforward mapping of the AFSM input-output state transitions and to associate the basic functionality of the rule with a set of auxiliary inputs. The counterpropagation network was used as described in chapter 7, along with the basic rule of the *BallPark* AFSM from chapter 5 for which the table in figure 8.3 gives the state conditions.

AFSM state	Identifying characteristic	*BallPark* output
0. Inactive	No identifiable input condition	Inactive
1. Quiescent	9 < Scan-level < 18	Step size = 200 Volts
2. Near-linear	Scan-level gradient polarity = EHT output gradient polarity	No output, desired state
3. Fold-back-A	Scan-level gradient polarity ≠ EHT output gradient polarity	Step size = -20 Volts
4. Fold-back-B	Scan-level < 9	Step Size = -100 Volts

Figure 8.3. Table of *BallPark* AFSM input-output state mapping. Signal levels are in ADC counts where 1 count = 0.02 Volt.

System Set-up

The rule part of *BallPark* uses various combinations of current scan-level, current EHT level, the gradient of the scan-level over time and the gradient of the EHT voltage over time as input. These were sufficient to identify approximately the four photomultiplier states (quiescent, near-

linear, fold-back-A and fold-back-B). However, as can be seen in figure 8.4, an array of coarse-grained pixels, called meta-pixels, formed by averaging sets of the main 1000-element scan-level pixel data, is also highly indicative of the state though difficult to characterise in terms of explicit rule conditions. Thus for this enhanced AFSM the scan-level, scan-level gradient, EHT level and EHT-level gradient are basic rule inputs and the meta-pixel array is used as an auxiliary input. The meta-pixels are generated by a new level-0 AFSM: *MetaPixelGen*. This AFSM splits each sampled scan stripe into 16 large or coarse-grained pixels by storing the average level of every 62 real pixels. A 16-element vector was chosen since it provided a clear characterisation of the scan-stripe state and seemed to strike a good balance of level of detail. It was particularly important to minimise the data transfer between AFSM processes as well as the computation required to process the counterpropagation algorithm within the characteristic time frame of the AFSM. The input of the enhanced *BallPark* AFSM is shown in the table in figure 8.5.

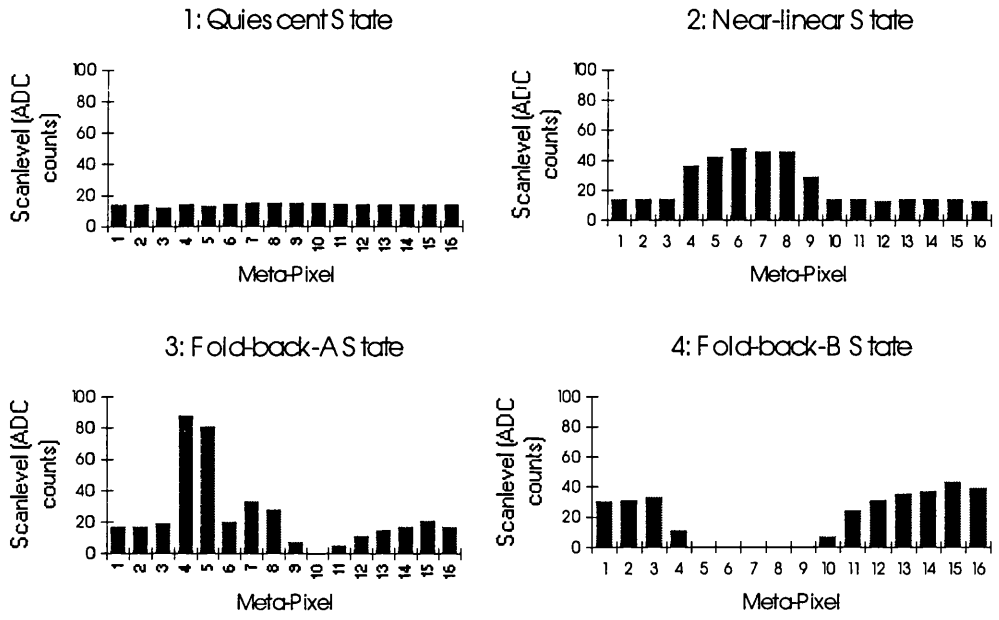


Figure 8.4. Laser-scan stripe states as characterised by a 16-element meta-pixel vector.

The Kohonen layer of the *BallPark* counterpropagation network was given an array of 64 processing cells (or artificial neurons). This was in accord with the equation from chapter 7: number of units = $4n^2$ where n (in this case 4) is the number of differentiable input states. The Grossberg layer was given 4 cells to represent the four photomultiplier signal states that the *BallPark* AFSM had to deal with. This was effectively a binary representation of the photomultiplier state with one bit for each state. Each of the four units was connected to the outputs of the Kohonen layer. The training of the *BallPark* associative network was set up so that the basic rule function would also generate a 4-element vector. If no bits were set, then the rule had failed to identify a specific state. This was used to determine if a network learning event should take place. Otherwise the rule output vector was used to train the Grossberg layer by

associating the Kohonen-network output with the rule output. With a Kohonen network size of 64 elements the learning phase, and thus the network output weight factor growth rate, was set (as described in chapter 7.3.1) at 500x64=32000 training events which, with a system characteristic time of 66mS (15 Hz), was approximately 35 minutes of real-time.

Signal Name	BallPark Input	Signal Name	*BallPark* Input
EHT Level	1	Meta-Pixel 6	11
EHT Gradient	2	7	12
Scan-level	3	8	13
Scan-Level Gradient	4	9	14
Meta-Pixel 0	5	10	15
1	6	11	16
2	7	12	17
3	8	13	18
4	9	14	19
5	10	15	20

Figure 8.5. *BallPark* AFSM input vector.

The execution of the new *BallPark* AFSM was moved to the T805 transputer that was connected to the two photomultiplier controllers (see chapter 4, figure 4.6). This was done to make use of the second transputer's floating-point arithmetic unit and larger amounts of memory. The transputer's subsumption program structure was thus broken into a collection of five behaviour nodes: (i) the server node, (ii) optics-controller node, (iii) photomultiplier level-0 for channel A and (iv) channel B and (v) the code for the *BallPark* AFSM.

8.1.2. The Activity of an Enhanced AFSM

Figure 8.6 shows the output clusterings generated by the Kohonen self-organising network layer of the associative network in the *BallPark* AFSM. It compares the input vector, which consists of both basic rule inputs and the extra auxiliary inputs, with the output of the Kohonen layer. It can be seen that the Kohonen outputs are clearly different for each of the AFSM states as presented by the basic rule. It was these outputs that were presented to the Grossberg layer. It can be seen that in this activity, the first part of the associative network performed completely adequately.

Figure 8.7 shows a series of *BallPark* outputs for sensor transition over the test surface. The left column of graphs show the development of the association network output in comparison with the AFSM rule output. It can be seen that the network output eventually develops a similar response to the rule but differs in small ways, particularly in the speed of state transition and filtering out of transitive state changes. The right-hand column shows the merged AFSM output compared to that of the rule output. Here it can be seen that, as expected, the rule output dominates in the early stages. However, by the end the output has become that of the network output. It can be seen clearly there that the an association has caused the early indication of state 4 on two occasions, characterised by spikes of the thin black curve at around samples 250 and 380.

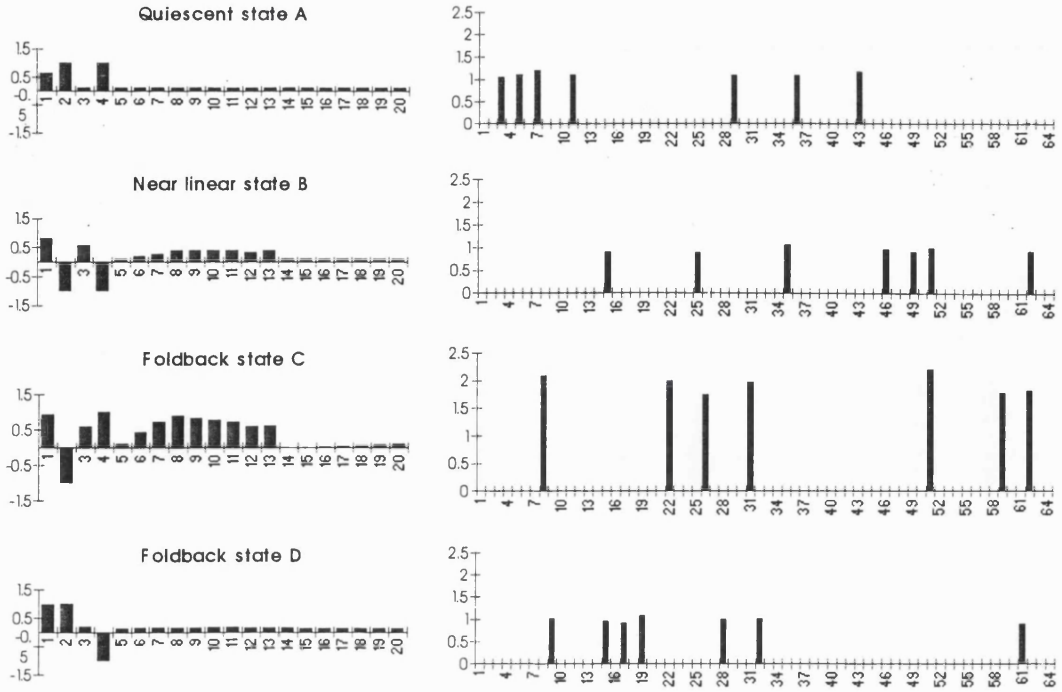


Figure 8.6. Each of the four photomultiplier states as reflected by the *BallPark* inputs. The graph at left shows the *BallPark* input vector with the 4 basic inputs leftmost and 16 meta-pixel inputs. The right-hand graph is the mapping output of the Kohonen self-organising network.

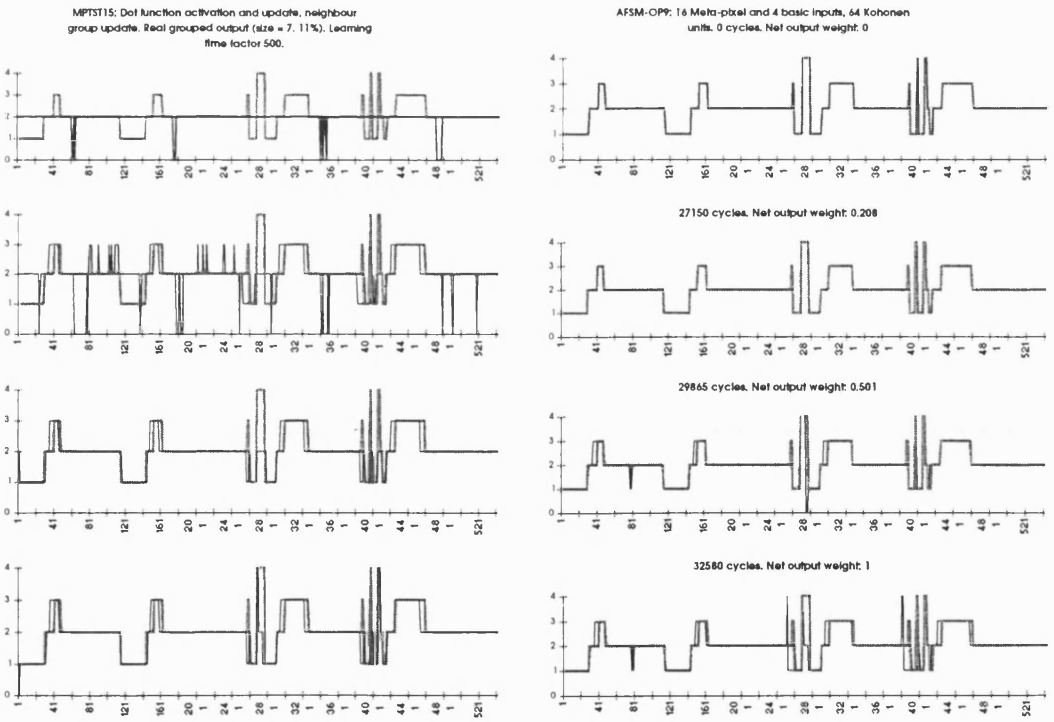


Figure 8.7. State value output plots for the *BallPark* AFSM. Left: Comparing basic rule output with associative network output at all times and Right: Comparing the combined rule-net output with the basic rule output. Thick lines are basic rule output, thin are network output (left) / AFSM output (right).

Experiments in dealing with unusual circumstances: Altering the test surfaces

The results of switching from the rule-based action-state identification to that of the network are shown in the various graphs of figure 8.8 for a number of different surfaces. The first test involved the comparison of the two trained sessions with that of the completely reactive system of chapter 5. The first training set used the broken-surface sample of drum 3 for the training period; the second used the continuous but varying surface of drum 2. After training, the system was tested on other drum surfaces in order to asses robustness when confronted with unusual circumstances.

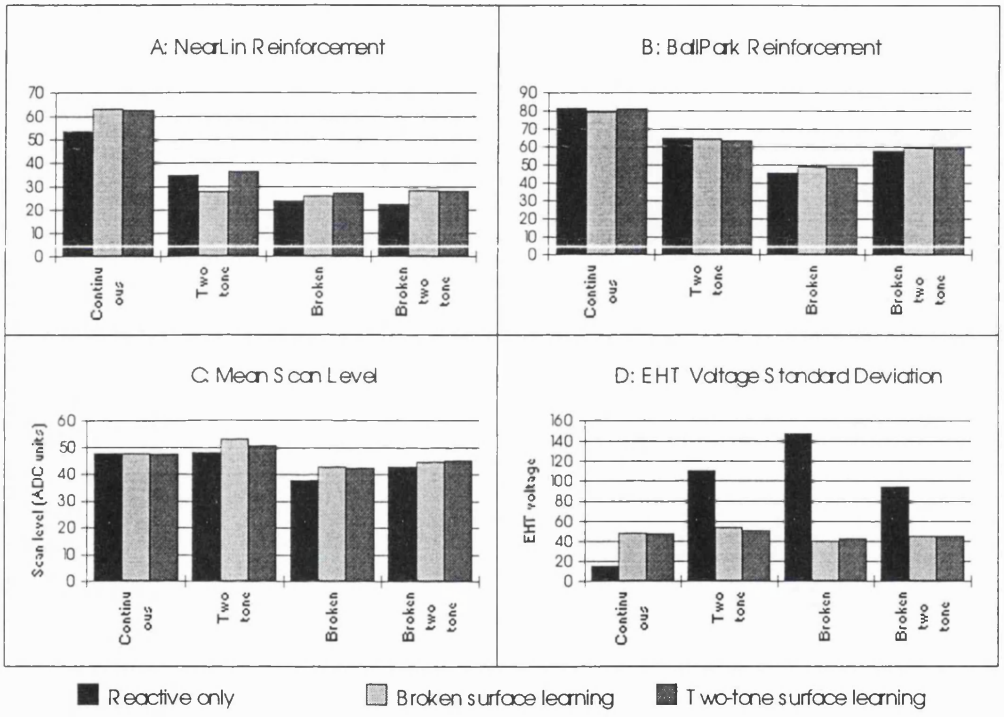


Figure 8.8. Graph A: *NearLin* AFSM state output, percentage of time that scan signal was in an acceptable range of the control set point (+/- 0.6V). Graph B: *BallPark* AFSM state output, percentage of time in near-linear state. Graph C: Mean level of the scan signal for each surface in the test. Graph D: Standard deviation of the EHT voltage output showing controller activity.

Graph B compares the percentage of time that the *BallPark* AFSM was in the near-linear output state (i.e. accumulating reinforcement value) for various surface samples and for the two learning regimes outlined in the experimental description above. It can be seen that there is in fact little difference between the systems that were trained on different surfaces. It can also be seen that there is only a small trend towards the learned systems having higher percentages of time in the near-linear state. The fact to note about this graph is that the learned response is not significantly worse than the reactive rule-based system. Examination of the *NearLin* output state in graph A, however, shows a more marked difference. Despite the fact that the *NearLin* AFSM did not support an input association network, it can be seen that the percentage of time for which

it was in the signal-in-range state increased markedly for the learned responses. This may be attributed to the more timely action selection of the *BallPark* AFSM.

The performance at the system level can be seen most clearly by comparing the standard deviations of the control-voltage outputs of the *EHTOutput* AFSM in graph D of figure 8.8. It is apparent that the standard deviation of the basic rule-controlled system is significantly larger than the enhanced AFSM outputs in all cases. This suggests an improved stability of the enhanced *BallPark*-controlled photomultiplier system as a result of reduced actuation from more appropriate action selection. In other words, a more appropriate classification of system state. Further analysis of the mean scan signal level, however, provides little further indication of the state of affairs, although the small increases in mean level for the broken and two-tone surfaces again indicate an improvement in the enhanced *BallPark* systems. The mean scan-level is markedly lower for the two broken-surface types, but this is due to the fact that the broken gaps inherently reduce the mean from that of the desired control set-point as a result of the scan signal spending considerable periods in the quiescent state as the gaps pass by.

Experiments in Robustness: Reducing AFSM Inputs

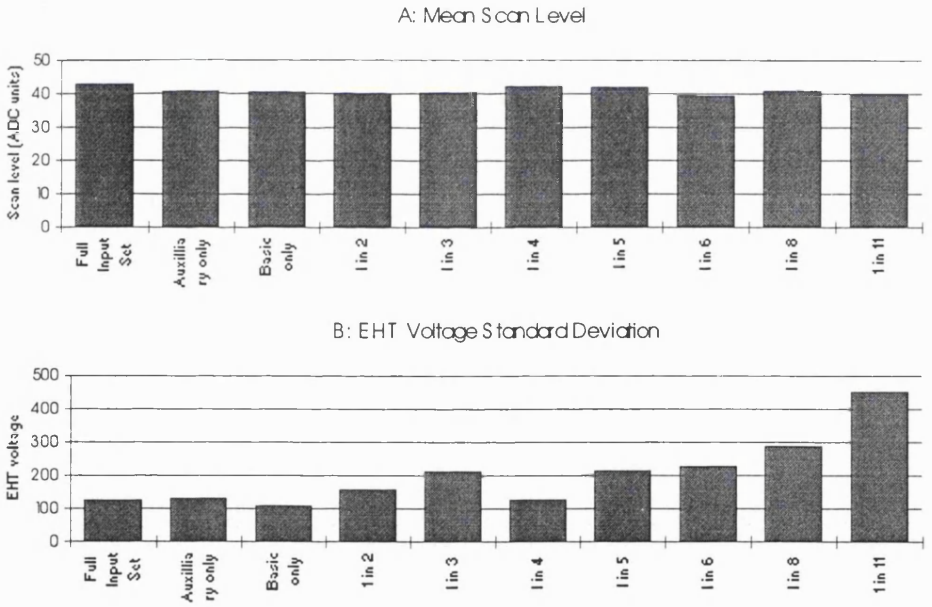


Figure 8.9. System performance with reduced *BallPark* AFSM inputs. Graph A shows that the mean scan-level remains more or less unaffected. Graph B: Standard deviations of the increasing EHT voltage show that the system gradually becomes more unstable as the inputs are reduced.

The graphs in figure 8.9 show the relative performances of the various tests in input reduction for the series of experiments on the single enhanced AFSM. Firstly it should be noted that when the basic rule inputs of the *BallPark* AFSM were removed, the system's performance was not significantly reduced. This indicates that the network's auxiliary inputs have indeed been assimilated into the network and that it is not simply providing a retranslation of the basic rule. In

fact, it can be seen that as the inputs to the network were progressively removed (by setting them to zero, see above) the degradation in system stability, as indicated by the standard deviations of the EHT voltage, was remarkably gradual with even a minimum of two network inputs proving to be sufficient for stable control. The reason for this apparent impressive robustness, however, results from an artificial construct in the structure of the experiment. It was the case that the basic inputs to the *NearLin* AFSM were the same as those used for *BallPark* and during the various stages of the experiment the *NearLin* inputs were not disabled in the same way. Therefore the *NearLin* this process continued to act unimpaired, providing a subsuming controlled output despite the gradual degradation of *BallPark* output. Because of this, the experiments in the next section provide a better indication as to the robustness of the system when faced with disappearing input state information. More detail will be provided there.

Summary

In general it can be seen from these experiments that a distinct, if not massive, improvement in system behaviour was gained through the use of an enhanced *BallPark* AFSM. While the statistical evidence may not seem to be overly impressive, it can be seen more clearly in scan-level/EHT-voltage state-space plots in figure 8.10 of the actual scan signal and EHT control data that, after learning, a more stable response is achieved. The measurement of the preferred output state or reinforcement level for each of the AFSMs (see section 8.1: Assessing the Results) did not account for the frequency of state transition. A system may spend time oscillating between two states and still accrue a similar reinforcement as a smoother-acting system. This in itself is perhaps indicative of two things: (i) That the reinforcement functions used as measures in these experiments were too simplistic and (ii) that the ever-present problem of credit assignment ([Barto90] and [Minsky61]) applies as much at the level of AFSM processes in a parallel control architecture as it does at higher levels. This is discussed further in the final section of the present chapter.

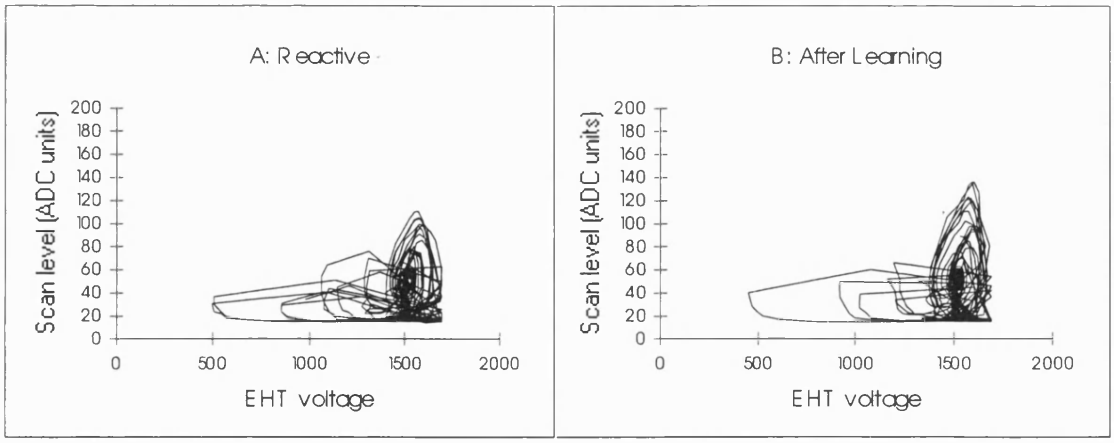


Figure 8.10. EHT control output plots for A: A reactive system on the broken surface and B: After the learning phase has passed, again on the same broken surface.

8.1.3. Multiple Enhanced AFSMs

In these experiments the control structure developed above with one enhanced AFSM was expanded to include two, the second being the *NearLin* AFSM. The other AFSMs in an individual photomultiplier-channel control mechanism did not lend themselves to the utilisation of the learning and adaptivity enhancement provided by the inclusion of the feedforward association network. Indeed these other AFSMs (*ScanAquire, PedestalLevel, EdgeMonitor* and *EHTOutput*) were of such a character that there was no obvious benefit in adding further mechanism to them. This is discussed again in the conclusions in chapter 9.

System Set-up

AFSM state	Identifying characteristic	NearLin output
0. Inactive	No identifiable input condition	Inactive
1. Negative set-point error	Near-linear state and scan-level < 47	*step_size* = ADJUST_FACTOR x (set_point - scan_level)
2. Positive set-point error	Near-linear state and scan-level > 53	*step_size* = ADJUST_FACTOR x (set_point - scan_level)
3. Signal in range	Near-linear state and 47 < Scan-level signal < 53	No output, desired set-point achieved

Figure 8.11. Table of *NearLin* AFSM input-output state mapping. Signal levels are in ADC counts where 1 count = 0.02 Volt.

The *NearLin* AFSM has a different input requirement from that of the *BallPark*, although at first sight it uses the same set of basic rule signals. From the table in figure 8.11 it can be seen that the inputs are used firstly to identify a near-linear state and then subsequently the scan signal level is used to select a positive or negative step-size output function. From examination of the meta-pixel array in figure 8.3 it can be seen that within the near-linear region there is little to differentiate between a signal level higher or lower than the set point of 50 analogue to digital converter units (1 volt). For this reason the network input scaling factor was used to enhance the meta-pixel's relative size so that the range around the set point was enhanced.

The Experiments

With the addition of associative network mechanisms to the *NearLin* AFSM and the retention of the *BallPark* AFSM from the previous work the photomultiplier controller reported here incorporated two enhanced AFSMs. The new *NearLin* AFSM was also relocated to the secondary T805 transputer to make use of the onboard floating-point arithmetic processor and additional memory. The main object of this set of experiments was to demonstrate the stability of utilising more than one, essentially asynchronous, enhanced AFSM in the same control structure. The experiments for this control structure were run along the same lines as those for the single enhanced AFSM. First the effects of different learning environments were observed, again with

learning applied to the broken surface and the two-tone surface. After each learning phase the systems were tested with the same set of secondary surfaces as before. Next, the robustness of the system to disappearing inputs was tested - again in the same way as in the previous section.

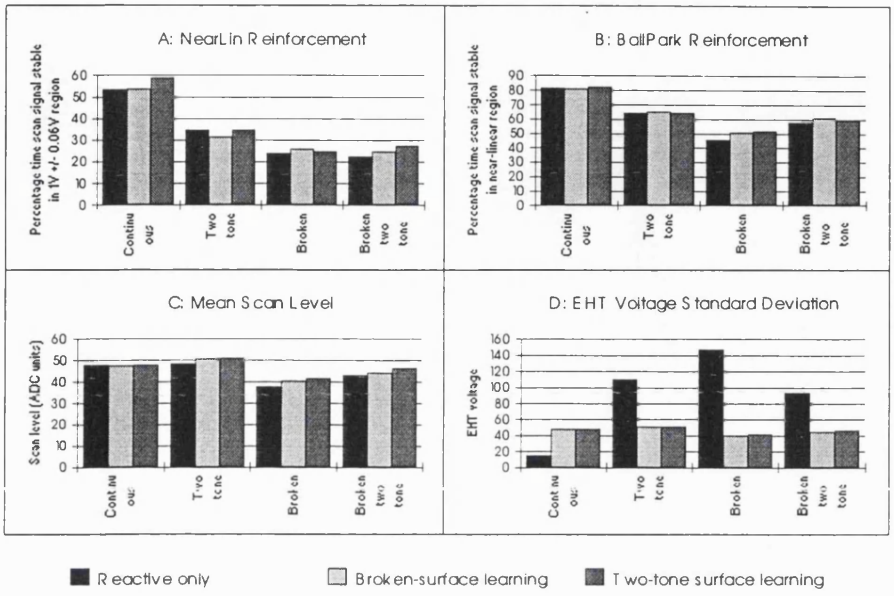


Figure 8.12. Comparative performance of multi-enhanced AFSM control of the photomultiplier subsystem. Graph A: Percentage of time that the *NearLin* AFSM state output was in acceptable range of the scan-signal set point level. Graph B: Percentage of time that the *BallPark* AFSM output was in the near-linear state. Graph C: Mean scan signal level for each surface sample set. Graph D: Standard deviation of EHT voltage outputs showing controller activity.

8.1.4. The Activity of Multiple Enhanced AFSMs

Different Learning Phases and Altered Test Surfaces

The results of the first series of experiments concerning the different learning phases are shown in the graphs of figure 8.12. It can be seen from comparing these graphs with those of figure 8.8 that there was very little difference in the performance characteristics of these two systems. In fact the data would suggest that there was little benefit from the added complexity of the second enhanced AFSM. However, the fact that the system remained stable while both AFSM processes were going through their learning and adaptive phases demonstrates that the localised association of additional inputs can be made to occur successfully in a distributed fashion. It is perhaps hardly surprising that the performance change was so small when it is remembered that the source of the entire subsystem activity is the same set of inputs all originating from the same sensor signal, albeit with different interpretations and emphasis on different dynamics of the signal. This stability may be seen by comparing the scan-level/EHT-voltage state-space plots in figures 8.13 for the broken-surface training and 8.14 for the two-tone-surface training.

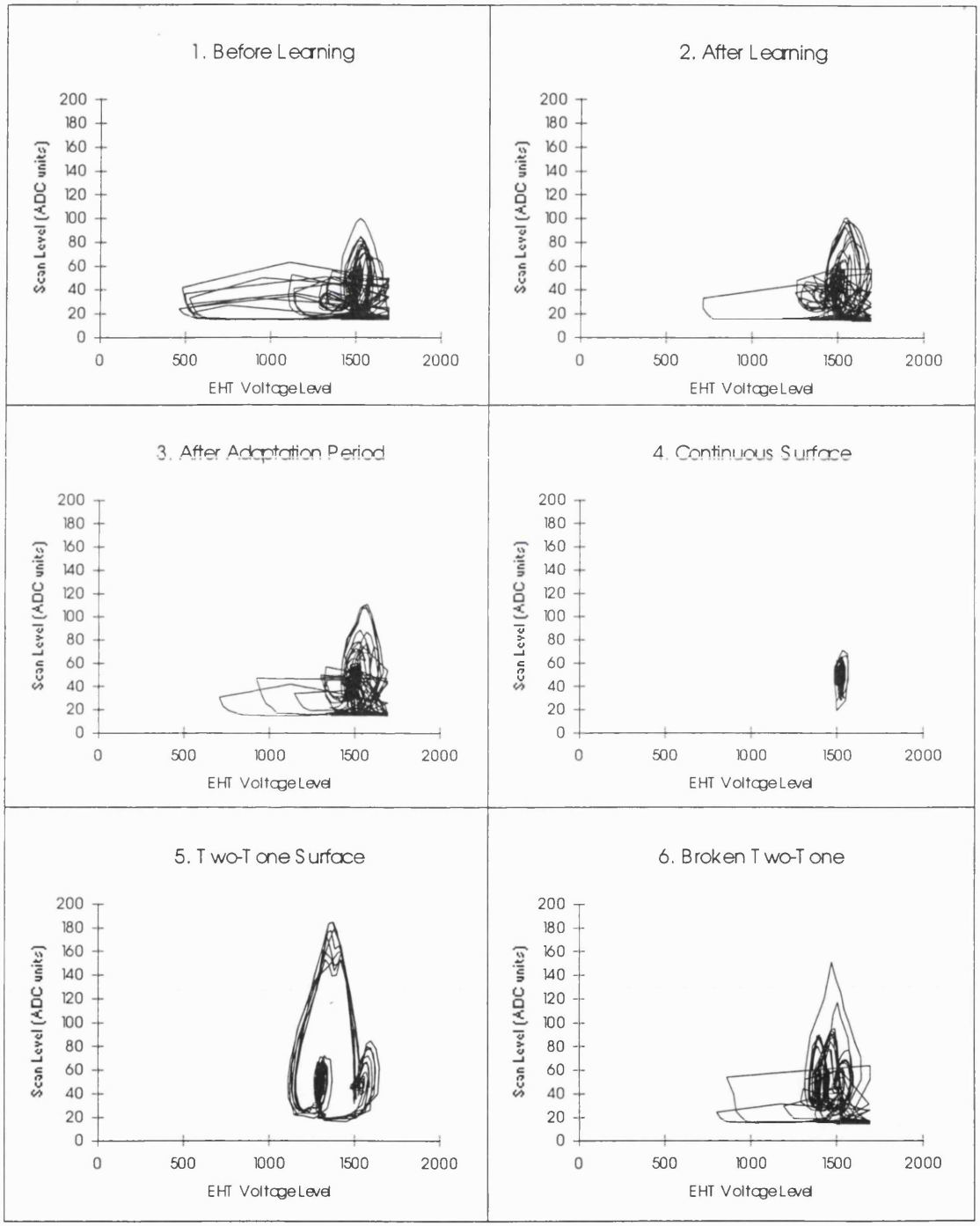


Figure 8.13. Scan-level / EHT-voltage state-space plots for broken-surface learning. 1. Before learning, 2. After learning, 3. After a period of adaptation, 4. Continuous surface, 5. Two-tone surface, 6. Broken two-tone surface.

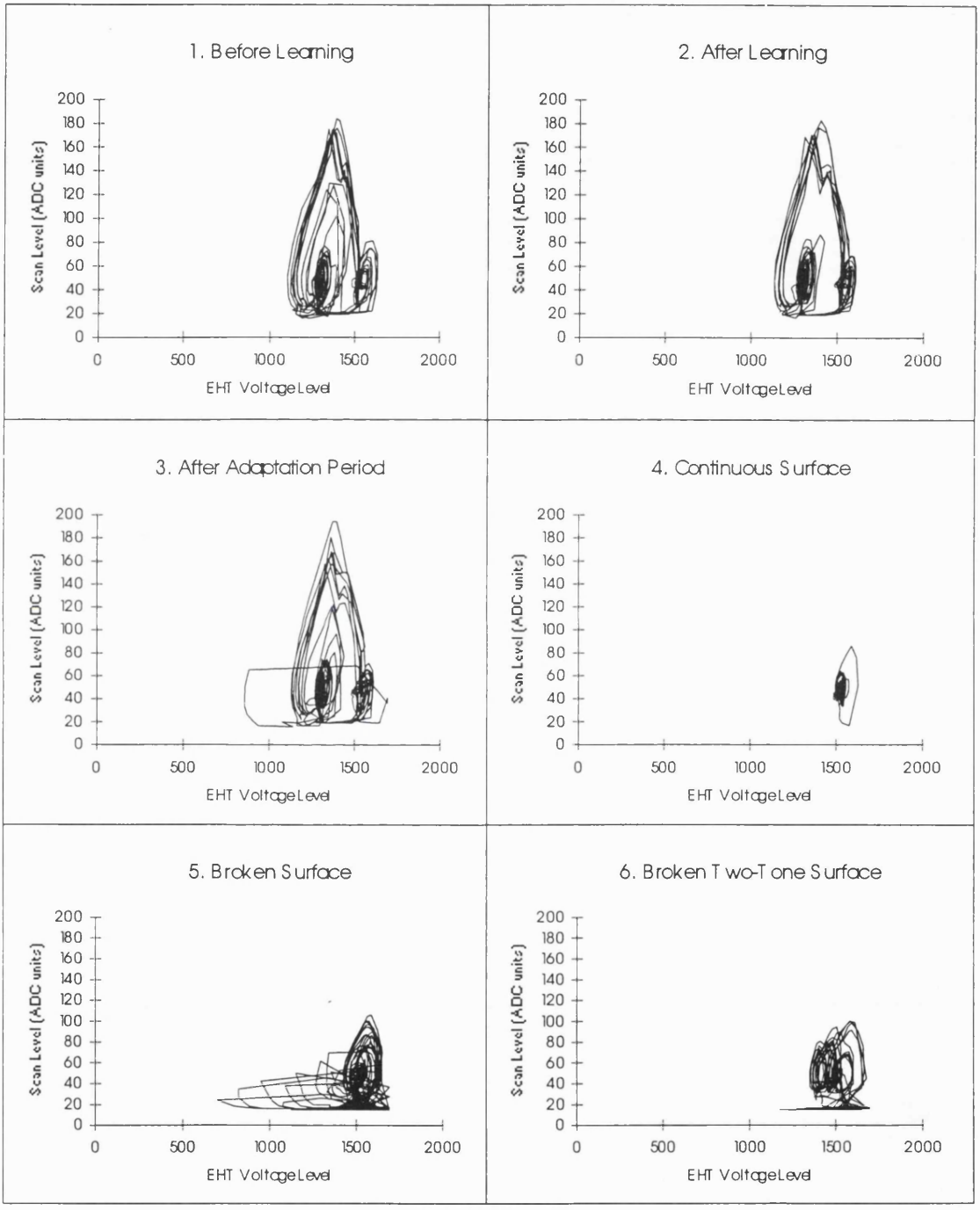


Figure 8.14. Scan-level / EHT-voltage state-space plots for two-tone surface learning. 1. Before learning, 2. After learning, 3. After a period of adaptation, 4. Continuous surface, 4. Broken surface, 5. Broken two-tone surface.

Reducing AFSM Inputs

The apparent performance of this version of the system appears to be worse than that for the *BallPark*-only version. This is entirely due to the fact that in this case when inputs are disabled they are removed from both the *BallPark* and *NearLin* AFSMs with the result that the complete system degrades. This is actually a more realistic test than that of the single enhanced AFSM system above in section 8.1.3.

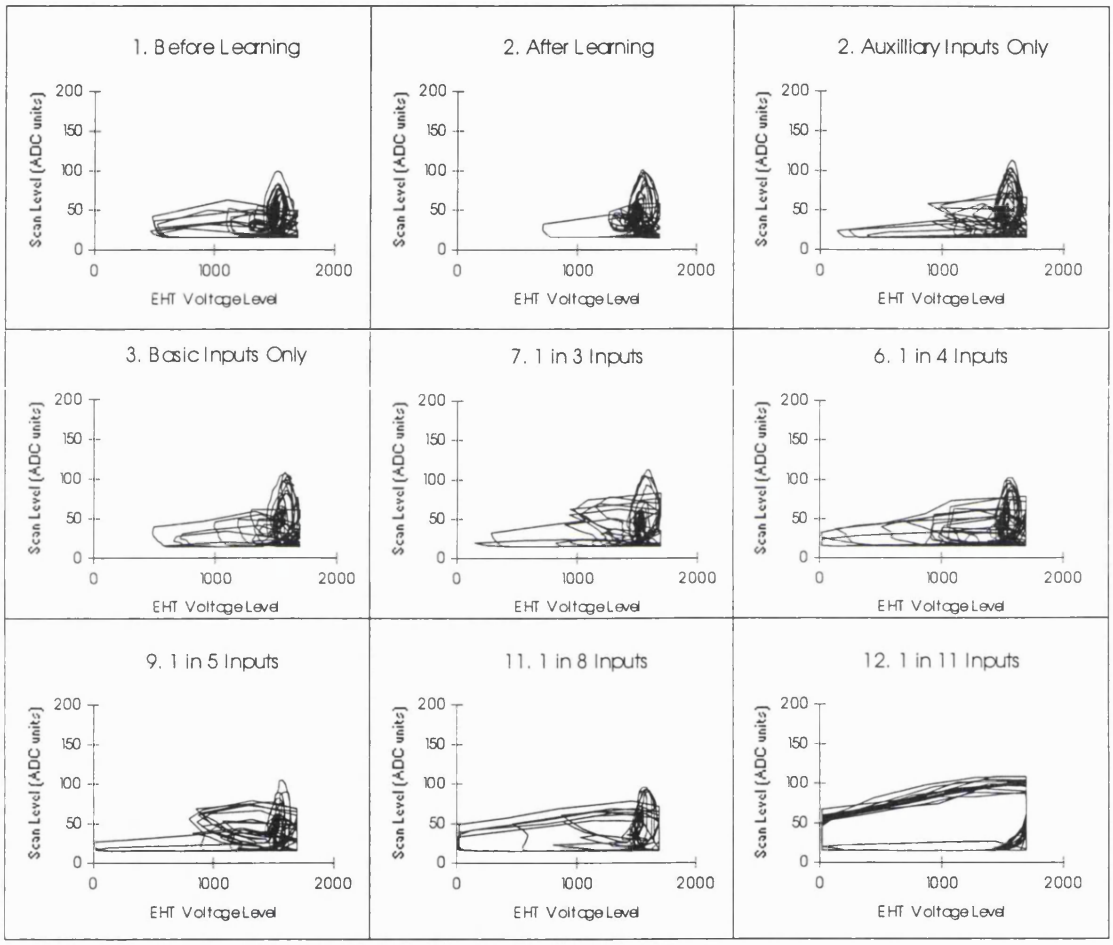


Figure 8.15. Scan-level / EHT voltage state-space plots showing degradation of performance with removal of AFSM input connections.

From the graphs in figure 8.16 it can be seen that a similar set of results was obtained as those for the first set of experiments above although the degradation of the *NearLin* AFSM performance due to missing inputs is more marked. Both AFSMs can be seen to perform satisfactorily when the basic rule inputs are removed and they are forced to rely exclusively on the learned response to the auxiliary inputs. It is interesting to note the apparently consistent performance in terms of preferred state (graph A and B) as inputs are disabled until a seeming collapse of competence when only one or two inputs remain. This robustness is however somewhat artificial in that the inputs are switched to 0 when we remove them. In actual fact it is quite likely that a failed input would manifest itself either as an indiscriminate fluctuation or a

fixed value and it is unlikely that the robustness under these circumstances would be as strong. Further experimentation in this area would be desirable but is beyond the scope of this initial exploratory work. Figure 8.15 shows scan-level / EHT voltage state-space plots for each test, where here again the degradation of performance stands out clearly.

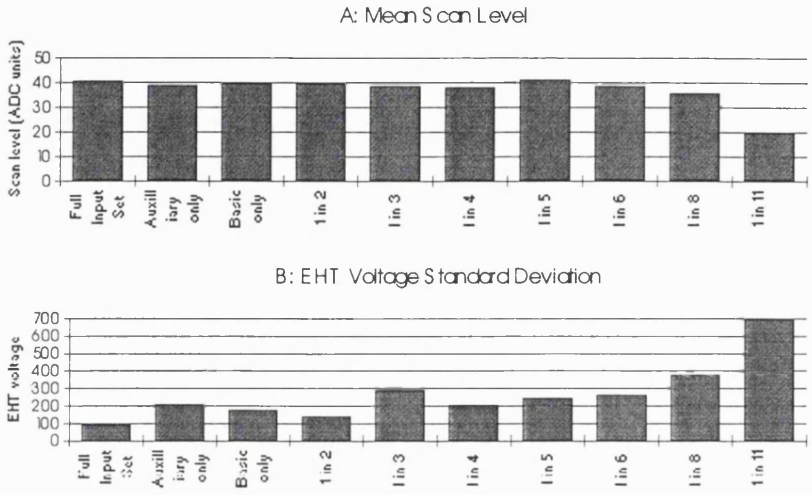


Figure 8.16. System performance with reduced *BallPark* and *NearLin* AFSM inputs. Graph A: The increase in the standard deviations of the EHT voltage shows that the system gradually becomes more unstable while Graph B shows that the mean scan-level remains more or less unaffected.

Summary

The main point to emphasise from this last set of experiments is that the system has remained stable despite considerable amounts of locally altering control processes. Issues of sequencing the learning phases of different AFSMs in different parts of the system are an important aspect that may affect the behaviour of the system significantly. This is discussed in more detail in the next chapter. Finally, it should be emphasised that the enhancement mechanism presented in this section only affects the input-state recognition of an AFSM and does not adapt the actual output values in any way. Consequently any improvement is the result of a more effective action selection and nothing more. Adaptive outputs are the focus of the next section.

8.2. Experiments in Adaptive Actuation Parameters

This section examines the use of the adaptive output registers detailed in chapter 7, section 7.4. Again, as for the experiments with the enhanced input AFSMs above, the first section below first deals with the use of a modified *BallPark* AFSM before the direction continues on to the operation of a system with more than one such mechanism in section 8.2.2.

8.2.1. Tuning Output Parameters Of A Single AFSM

As a first example of the use of the output adaptation mechanism, the three *BallPark* AFSM output registers; *Quiescent_step-size*, *Foldback_A_step-size* and *Foldback_B_step-size* were

provided with the requisite software infrastructure. With initial values set at the level used in all previous experiments, the adaptation mechanism was given a head start in that it did not have to find a suitable value from scratch. This is reasonable since in normal circumstances a designer is in a position to insert some sort of educated estimation of suitable initial values, and in any case the mechanism is intended to provide a continuous tuning of the register value rather than a complete initialisation from scratch. The *BallPark* AFSM utilises three output registers that, when activated, specify an EHT voltage increase or decrease, the magnitude of which depends on the current state of the photomultiplier-output scan signal. The following are the constant settings used for these registers in the experiments in section 8.1:

Quiescent output register:	EHT step size = 200 Volts
Fold-back-A output register:	EHT step size = -20 Volts
Fold-back-B output register:	EHT step Size = -100 Volts

In the following experiments these values are turned into variables which are modified on a continuous basis by the output-register adaptation mechanism.

The Experiments

The register-adaptation mechanism presented in chapter 7 utilised three behaviour-tuning variables in the form of: *Reinforcement Period* t_r, *Adjustment Period* t_a and *Adjustment Factor* *f*. The experiments that follow explore a space of possible settings of these parameters by using them to vary the output tuning mechanism's adaptivity from fast with large changes to slow with gradual changes. These are of course imprecise constraints and the interaction of the system within its environment is of primary importance in determining the best values for these settings. Consequently the activity on the sample surface provides a means of comparing the relative effects of different combinations of parameter settings and subsequent register-update characteristics.

The broken surface was used for the duration of these experiments. The unit that rotated the surface sample was set to provide a repeating pattern at a rate of one sample per (approximately) 30 seconds; in other words, the drum was rotating at 2 revolutions per minute. With a system characteristic frequency of 15 Hz this meant that the surface pattern repeated at a rate of approximately once in every 450 characteristic time steps. With this in mind the table in figure 8.17 shows the points in the 3-dimensional space of the three relevant tuning parameters used for experimentation in 12 tests referenced as numbered in the table. The use of a reinforcement period that was considerably larger than that of the repeating surface pattern was chosen in order to test the idea that the longer averaging-time function for the reinforcement value would provide a smoothing-out of any transient peaks and troughs in system performance that might result from irregularities in the outside world. The update periods were chosen to demonstrate the benefits from rapid and short-term experimental periods compared to longer periods, in this case equivalent to approximately two surface passes. A range of adaptation factors was selected for each combination of reinforcement and update periods.

	Test number	Update Period 100	Test number	Update Period 1000
Reinforcement Period 1000	1	Adaptation 0.2	4	Adaptation 0.2
	2	Adaptation 0.5	5	Adaptation 0.5
	3	Adaptation 0.8	6	Adaptation 0.8
Reinforcement Period 5000	7	Adaptation 0.2	10	Adaptation 0.2
	8	Adaptation 0.5	11	Adaptation 0.5
	9	Adaptation 0.8	12	Adaptation 0.8

Figure 8.17. The parameter space of the adaptive output mechanism for the laser-scanning test-bed. Period times are shown in characteristic time steps: 1 count equals approximately 0.067 seconds at 15Hz.

Each system configuration in figure 8.17 was run continuously for more than 100,000 characteristic time steps with periodic sampling of reinforcement values and register values. Every 50,000 time steps, 1000 detailed traces of system status were made for a continuous period of 4000 time steps as in the previous section. This extensive test period, in the order of 110 minutes per test, was used so that any transient behaviour would be filtered out from the long-term results.

Assessing the Results

The adaptation mechanisms implemented here are of a continuous nature, within the constraints of pre-set update-frequency parameters that dictate the speed of adaptation. Consequently the change in system performance over time is of interest in assessing the results of these experiments. To facilitate this, the reinforcement values returned from the *BallPark* and *NearLin* AFSMs along with the value of each output register were sampled at regular intervals over the complete time span of each experiment. At longer intervals complete traces of system activity were recorded to provide similar information to that used in the previous section, although in this case only serving to provide a snapshot of a continuously evolving system.

The plots in figure 8.18 show the reinforcement over time of test run 7 along with the values of the output registers. The changes in output values can be seen to match up with corresponding changes in reinforcement. However, the comparison of data in this format is difficult. The results have therefore been collected in the form of mean reinforcement values and ranges of reinforcement values for each test run. The graphs in figure 8.19 show this information for each of the test runs of table 8.17. It can be seen that a number of patterns have emerged as a result of different parameter settings. The material that follows will deal with the parameter variations (reinforcement period, update period and update factor) separately. Before this, however, it is worth pointing out the difference between an observed increase in *BallPark* reinforcement and a simultaneous decrease in *NearLin* reinforcement. This effect on the *NearLin* AFSM response may be seen as a secondary result of the parameter adjustments. As previously, it should be expected that the performance of *NearLin* would at least improve gradually in line with *BallPark*.

However, the results here emphasise the activity of the *BallPark* AFSM since the experiments concern its own local reinforcement and interaction.

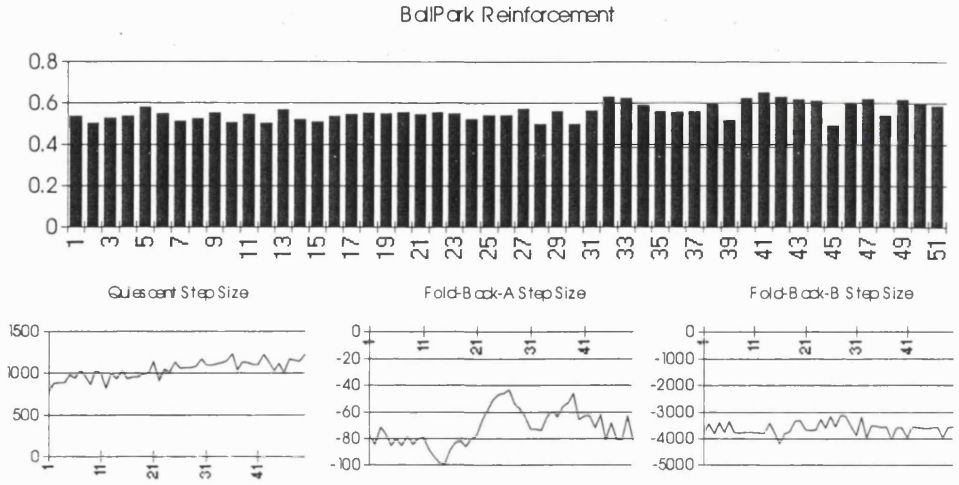


Figure 8.18. Reinforcement and *BallPark* output register values over time during test run 7. Samples were taken every 1000 characteristic time steps for the duration of the test run.

Update Factor

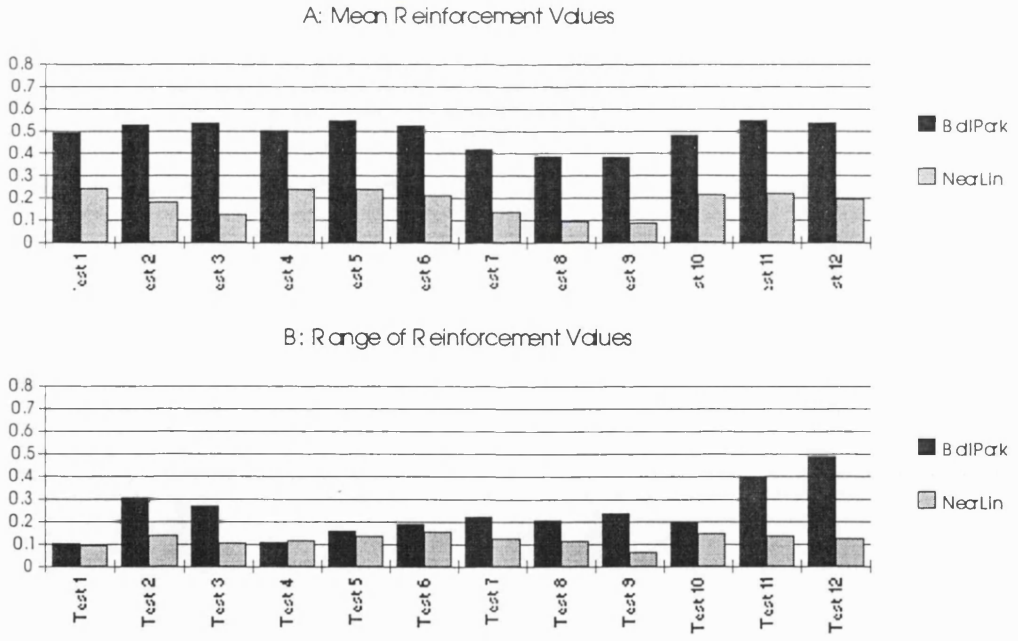


Figure 8.19. A: Mean reinforcement (success) levels and B: Range values for both *BallPark* and *NearLin* AFSMs (although only *BallPark* was adaptive).

The range of values tested for the *update factor* was 0.2, 0.5 and 0.8. This parameter had the effect of biasing the size of any exploratory alteration to a register's value. Since this exploration is an on-line process the effects of a "bad move" are reflected in the output of the system. In this case graph B in figure 8.19 can be examined to see the effective range of reinforcement values.

The range reflects the distance between the maximum and minimum recorded reinforcement. As a result, therefore, the stability of the system in terms of maintenance of a usable scan signal can be assessed. It is generally the case that the larger values of the update factor lead to larger ranges of reinforcement, which is intuitively understandable as the larger update factors correspond to a larger experimental variation in output register value. The larger range had some effect on the level of the mean reinforcement value but in general a higher mean value is indicative of a more successful system. From comparison of the range and the mean values it would appear that a factor of 0.5 provided the best compromise of maximising the mean level whilst keeping a relatively small and manageable reinforcement range.

Reinforcement Period and Update Period

The effect of increasing the *update factors* in graph B in figure 8.19 can be seen clearly to cluster the observed behaviour into the four groupings determined by the *reinforcement period* and *update period.* For example, tests 1,2 and 3 resulted in a gradual increase in *BallPark* mean reinforcement whilst the *NearLin* reinforcement dropped steadily. This pattern recurs across the graph but with differing levels of mean reinforcement. From the mean levels the results are seemingly consistent across the four test sets with the exception of test set 3. This is characterised by a large *reinforcement period* and small *update period* of 5000 and 100 respectively. It is evident that the rapid rate of change realised by the small setting of the *update period* was not given enough time to manifest in the long *reinforcement period* (5000 characteristic time steps at 15Hz is 5½ minutes, 100 steps is just under 7 seconds).

Further insight into the system's behaviour may be attained by examining the reinforcement ranges in figure 8.19, graph B. Here it can be seen that the three tests in set 2 have a consistently smaller reinforcement range than the others. Set 3 also has reasonably small range but compared with the reinforcement levels attained (graph A) this is to be expected. Test sets 1 and 4 both demonstrate systems with still relatively high *reinforcement period / update period* ratios and it can be seen that here again the ranges achieved were quite large, especially in the case of the higher *update factors*. It can be concluded from these results that the most stable systems had an equal *reinforcement period* and *update period.* The reasoning that suggested that a longer reinforcement period might be beneficial is therefore disproved. It is the case that the matching of the two parameters provides the most suitable response for the reinforcement over the time period, allowing the register adjustment to be tuned to the relevant time period. The idea that the longer reinforcement period would be beneficial due to an inherent filtering of transient performance changes was not upheld.

Finally in this section the relationship between the *update* and *reinforcement periods* and events in the agent's environment should be emphasised. It is quite likely that the 1000/1000 reinforcement/update period parameter setting proved successful because this coincided with two passes of the test surface. It is therefore the case that even with localised adaptation at the level of

individual AFSM processes the activity of the system and the physical world are of fundamental importance.

The Winners

The next section deals with the use of adaptive outputs in more than one AFSM. The experiments will build on the results of this section by concentrating on development of systems that use the better parameter sets from the results discussed here. From the graphs in figure 8.19 it can be seen that generally test sets 4, 5 and 6 are the most effective both in terms of higher reinforcement and smaller reinforcement ranges. Consequently the set-up of test 5 will be used along with (for comparison) that of test 1, which actually achieved a respectable reinforcement level while maintaining the lowest reinforcement range with values of 0.49 and 0.1 respectively.

8.2.2. Multiple-AFSM Output Tuning

The object of making the test runs reported here is to demonstrate the performance of a system with more than one adaptive AFSM. Again, it is the *NearLin* AFSM that is used as the second target for modification. Two test runs demonstrate the system's performance with the initialisation shown in the table of figure 8.20. The tests were run on the broken surface in the same manner as those in 8.2.2 above.

Test 13	Reinforcement Period	1000
	Update Period	100
	Update Factor	0.2
Test 14	Reinforcement Period	1000
	Update Period	1000
	Update Factor	0.5

Figure 8.20. Table showing adaptive output parameters for multiple adaptive AFSM demonstration. Parameters are taken from results obtained from section 8.2.1. BP = *BallPark* AFSM settings and NL = *NearLin* AFSM settings.

System Set-up

The *BallPark* AFSM remains the same as that used above. The *NearLin* AFSM, however, required the addition of the adaptive mechanism for a single output parameter which served to determine the response of the AFSM to varying scan-levels by scaling the step size for the output EHT voltage (as sent to the *EHTOutput* AFSM in level 0). The ADJUST_FACTOR used in the output rule effectively determines the gradient of the step-size output in comparison with scan-level. Changing the ADJUST_FACTOR will effectively change the step-size output for a given input condition. This factor was used previously as a preset constant with a value of 5 during the calculation of the EHT voltage step in the near-linear region of photomultiplier operation:

$$step\_size = \text{ADJUST\_FACTOR} \times (\text{set\_point} - \text{scan\_level})$$

By making this parameter adaptable we intended that the *NearLin* AFSM would be able to tune itself automatically to differing environmental conditions including surfaces, ambient light and sensor-surface distance.

Results

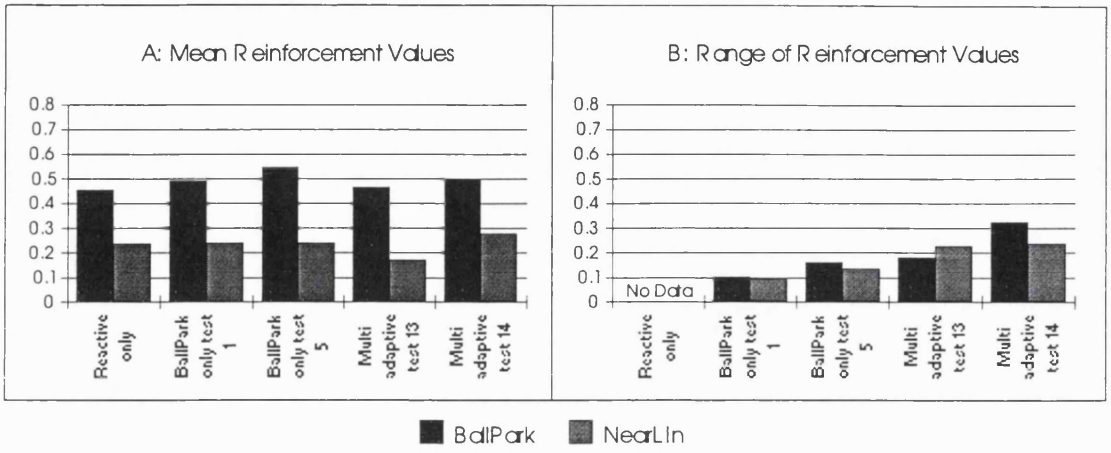


Figure 8.21. Mean reinforcement and range for the two multiple adaptive AFSM experiments. Tests were run using the broken-surface sample number 3. (see figure 8.2 above).

Figure 8.21 shows the results of the two test runs compared with their *BallPark*-only counterparts in the single adaptive AFSM tests reported in the previous section and with the reactive AFSM system of chapter 5. In graph A it can be seen that the reinforcement values for the two test runs differ significantly in format. The first test (13), based on test 1 of the last period with small update period and relatively large reinforcement period, demonstrates a drop in performance, while that in test 14 with its equal timing parameters is most notable for its distinctly improved *NearLin* performance. Although in this result the *BallPark* AFSM appears to have dropped in the amount of time spent in its preferred state, the global view of the system is represented better by that of the scan-level state of *NearLin*. This performance, however, must be qualified by the apparent increase in range of reinforcement received over the test period. The effect of this adaptation to the *NearLin* register is to retune the controller continually in response to varying scan-level/EHT-voltage gradient. This was highlighted in chapter 4 as the main non-linear characteristic of the photomultiplier tube, which is affected by changing environmental conditions such as surface type, ambient light and surface-sensor distance.

It is clear that the extra activity of the *NearLin* AFSM added to the variability in environment to which the *BallPark* AFSM had to adapt. While in the previous experiments only one AFSM was adapting while the rest remained constant, in this series of tests much more was happening within the system. By examining the output plots of the test-14 reinforcement over time and the corresponding adjustments to register values (figure 8.22), the degree of adaptation and change within the system can be appreciated. There, the fact that the performance of the photomultiplier control channel as a whole not only remained stable but actually improved is noteworthy. Finally,

the poor performance of the test-13 configuration can be attributed to its fast update rate. From the earlier series of experiments using this configuration it was noted that the performance was marginal as evidenced by the relatively large ranges of reinforcement values (figure 8.16a, tests 1, 2 and 3). In this last instance we believe that it was not possible for the adaptation mechanism to compensate for the increased complexity of the control system by the small update-factor parameter of 0.2.

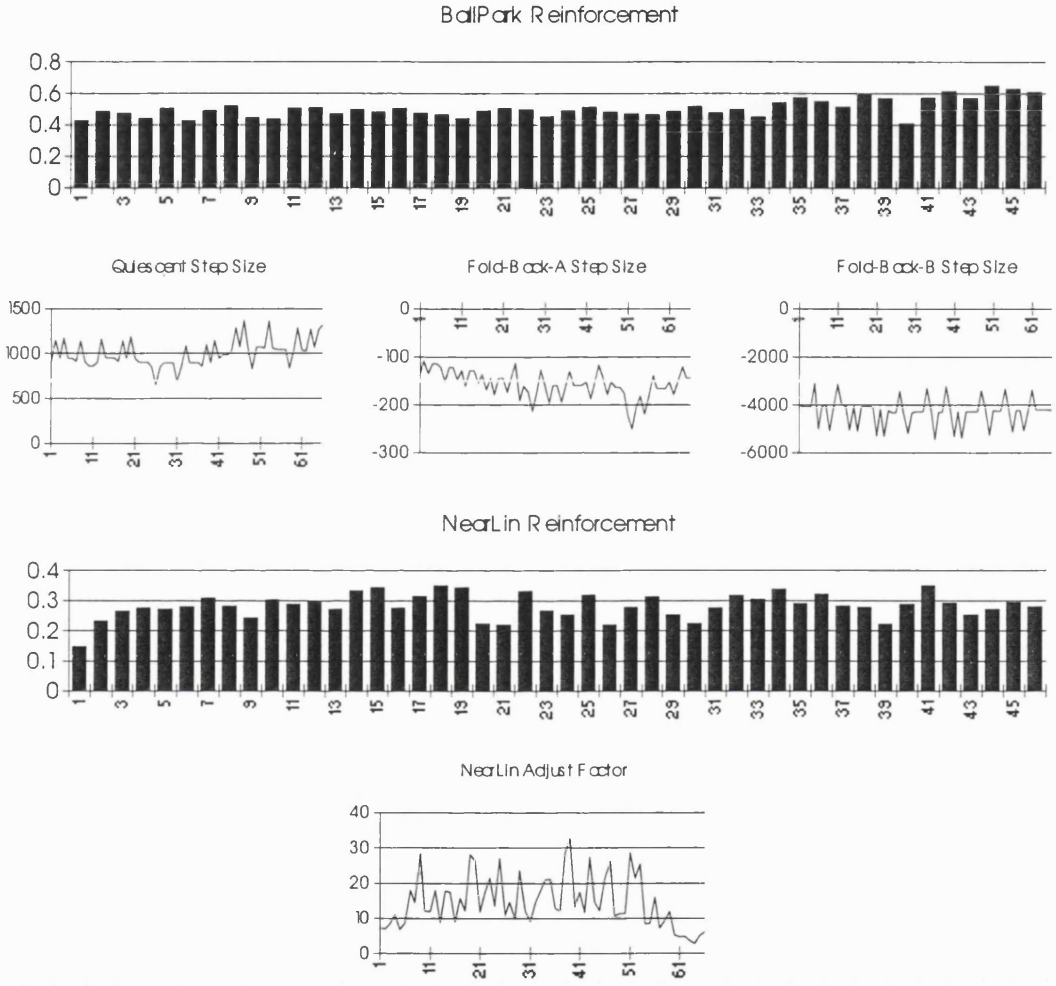


Figure 8.22. Test-14 reinforcement over time in conjunction with *BallPark* and *NearLin* AFSM output register values. On the broken-surface sample 3 samples were taken every 1000 characteristic time steps over the duration of the test run.

Figure 8.23 shows scan-level / EHT-voltage state-space plots for four samples made during test 14. It can be seen that there is not necessarily a smooth and consistent trend towards improved performance, although in plot 4, after 120 000 steps, this was actually the case. This apparent irregularity in performance was due to the experimental nature of the hill-climbing register-adjustment mechanism and the fact that the changes made were not always beneficial. In general the results show that the update period must be long enough to filter out transient environmental effects while at the same time being short enough to provide a rapid readjustment

of parameters should the system performance deteriorate. This apparent catch-22 might be solved in the future by cutting short the update period should things become too unstable and reinstating the parameter levels early.

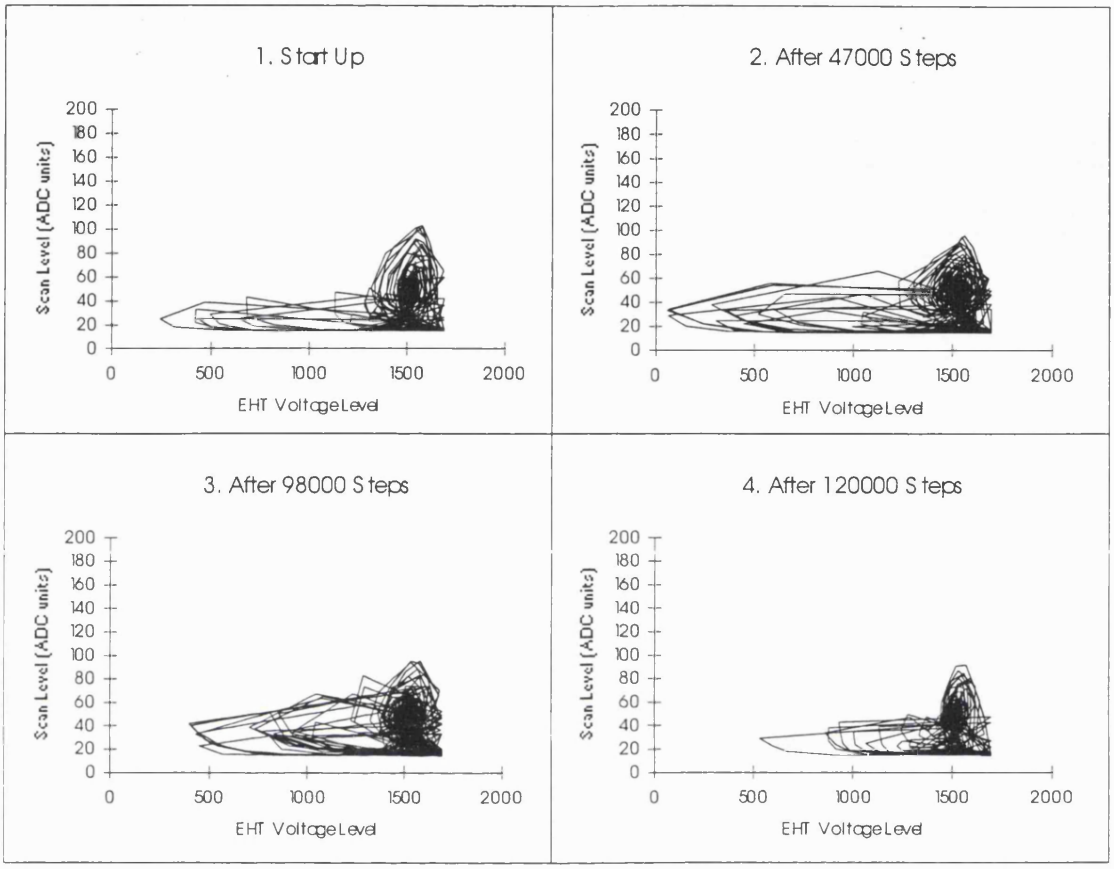


Figure 8.23. Scan-level / EHT voltage state-space plots for three samples of test run 14 and the broken-surface sample 3.

From these experiments the configuration of test-14 was chosen as the most suitable candidate for the final test illustrated in this chapter, that of the complete system of multiple adaptive AFSMs including both input-state recognition from the feedforward associative networks and the adaptive output registers.

8.3. Bringing it All Together

In this section the enhanced AFSM input associative mapping provided by the counterpropagation network in section 8.1 is combined with the output-adaptation mechanism of section 8.2. This is compared with a further possible solution to the problem of enhancing an AFSM in the form of a network-only solution, that of the real-valued output counterpropagation net as detailed in section 5 of chapter 7. Potentially both of these implementations provide a similar result in terms of enhancing the input-output mapping of an AFSM, but the latter does not allow automatic tuning of output registers. Rather, it allows a varying output depending on input-

output mapping strengths and network weights. The experiments below will compare the merits of these two approaches.

Figure 8.24 shows the control structure of the *BallPark* and *NearLin* AFSMs for the new feedforward network enhanced system. Note that the *NearLin* AFSM subsumes *BallPark* and so utilises a counterpropagation network with a two-element output layer, one to generate the output value and the other to act as an output gating switch. Apart from this structural difference all network configuration parameters are the same as those used in the first (input only) implementation.

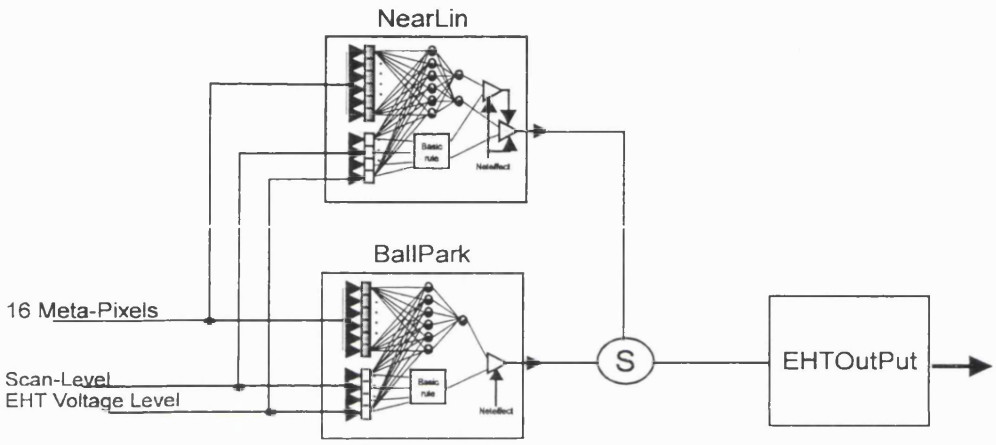


Figure 8.24. BallPark and NearLin AFSM network control configuration.

Experiments

The two systems were each trained with the broken-surface sample and then tested with the same range of other surfaces (continuous, two-tone and broken two-tone) as used in section 8.1. Likewise a similar set of reduced input tests was performed by disabling the inputs to the networks in the same way as in section 8.1. Traces of the data relevant to system performance were made as before. Tests were run with the separate enhanced/adaptive AFSM processes and then with the complete feedforward-network AFSM process.

Continuous Output AFSM Activity

Figure 8.25 shows a time series of output from the continuous-output network AFSMs after a learning period on the broken-surface sample. The plots here show the *EHTStepSize* value output to the *EHTOutput* AFSM and illustrate that the rigid step-like output of the rule is superseded by a more flexible output. While the *BallPark* AFSM outputs generally match the basic rule output, it is interesting to note the expanded range of the *NearLin* AFSM actuation which was

additionally controlled by the learned output switch (figure 8.25). We deal with this in more detail in the discussion of results immediately below.

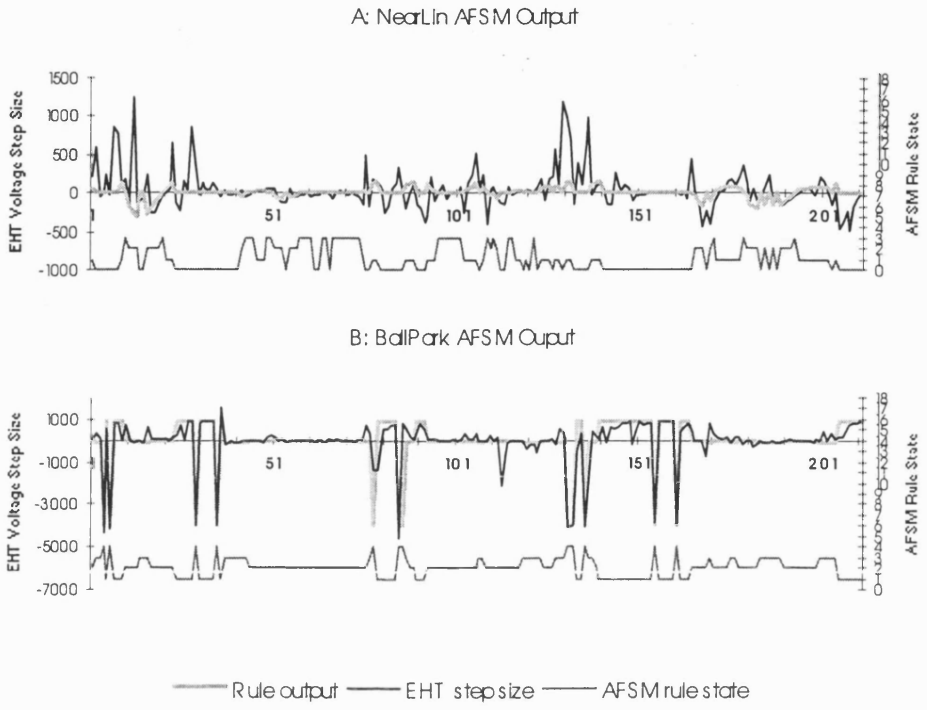


Figure 8.25. Comparison of step sizes in EHT voltage as output by A: *NearLin* and B: *BallPark*. Basic rule output is compared with learned EHT step size. Plots relate to the broken-surface (surface sample number 3).

Results

Figure 8.26 shows result graphs of the tests and compares previous tests with the two results (full feedforward network and enhanced input, adaptive output AFSMs) obtained here. It can be seen that the enhanced-input, adaptive-output configuration resulted in the higher reinforcement values for both the *BallPark* and *NearLin* AFSMs by a considerable margin. This was paid for by an accompanying small increase in the range of the reinforcement values. It is interesting to note the difference in AFSM output signals for these two configurations of controllers. It can be seen in figure 8.25 that the continuous-valued output AFSMs provided a much more gradual change in EHT step-size compared to the fixed-register-determined output of the enhanced/adaptive AFSMs. This is reflected in the relative sizes of standard deviations of EHT voltage output from the *EHTOutput* AFSM seen in figure 8.26. As a result the continuous-value output AFSMs appeared to have a much smoother control behaviour than that of the enhanced/adaptive type.

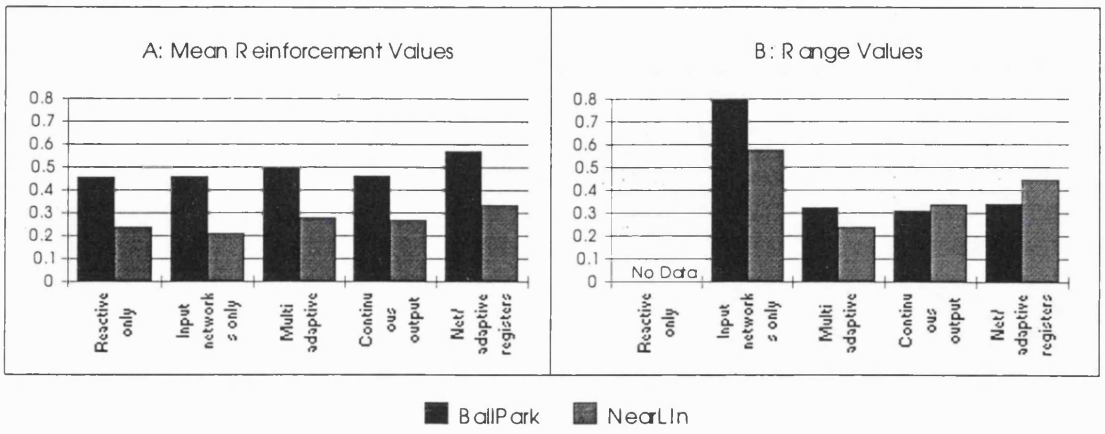


Figure 8.26. Results of the Enhanced/Adaptive AFSM versus continuous-valued output AFSM tests compared with the best performers of the other tests in previous sections. As before, tests were conducted on the broken-surface sample.

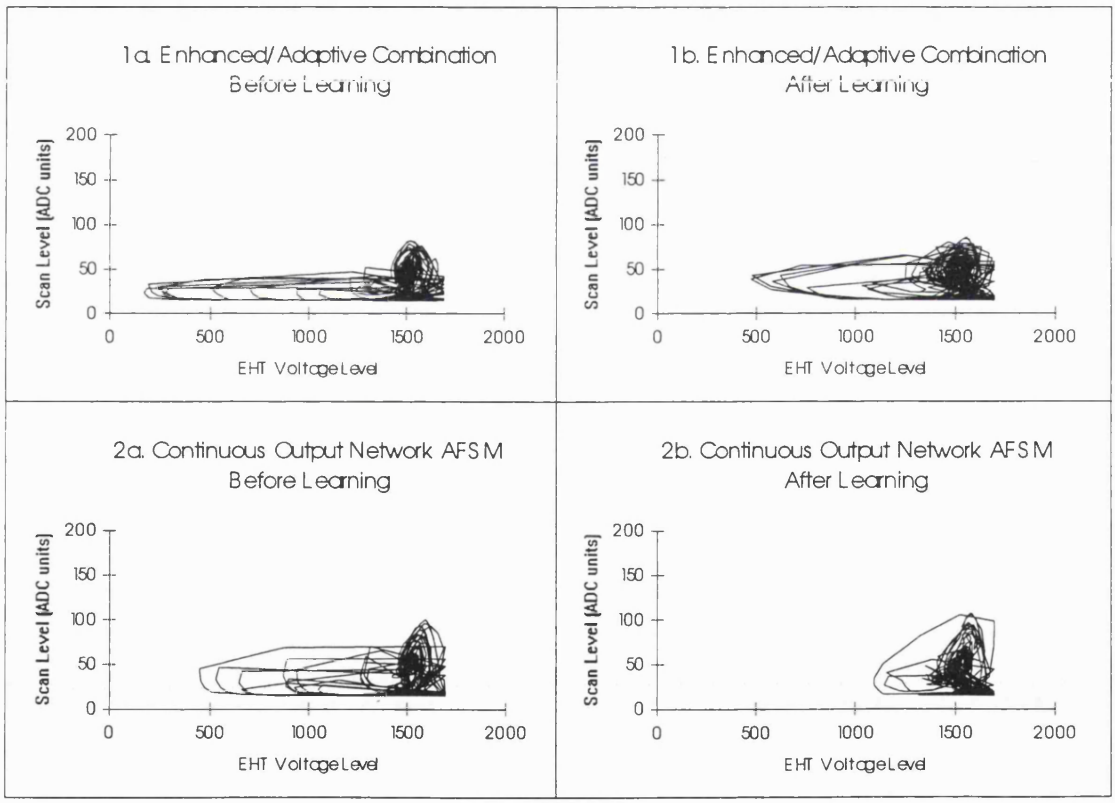


Figure 8.27. Scan-level / EHT-voltage state-space plots for the three test runs compared to the reactive-only system. 1. Enhanced/Adaptive AFSM, 2. Continuous-valued output AFSM. Plots relate to the broken-surface sample (number 3).

It is therefore a matter of debate as to whether one technique should be chosen as a winner over the other, although comparison of the scan-level/EHT-voltage state-spaces in figure 8.27 provides another useful source of information. Here it would appear that the continuous-output-network AFSMs were significantly better. However, the answer to the question of the relative performances of these two techniques is most likely to be highly application-dependent. Some

other form of control activity would have to be implemented to address these issues, but that would be too large a programme to be within the scope of this thesis.

8.4. Summary of Results

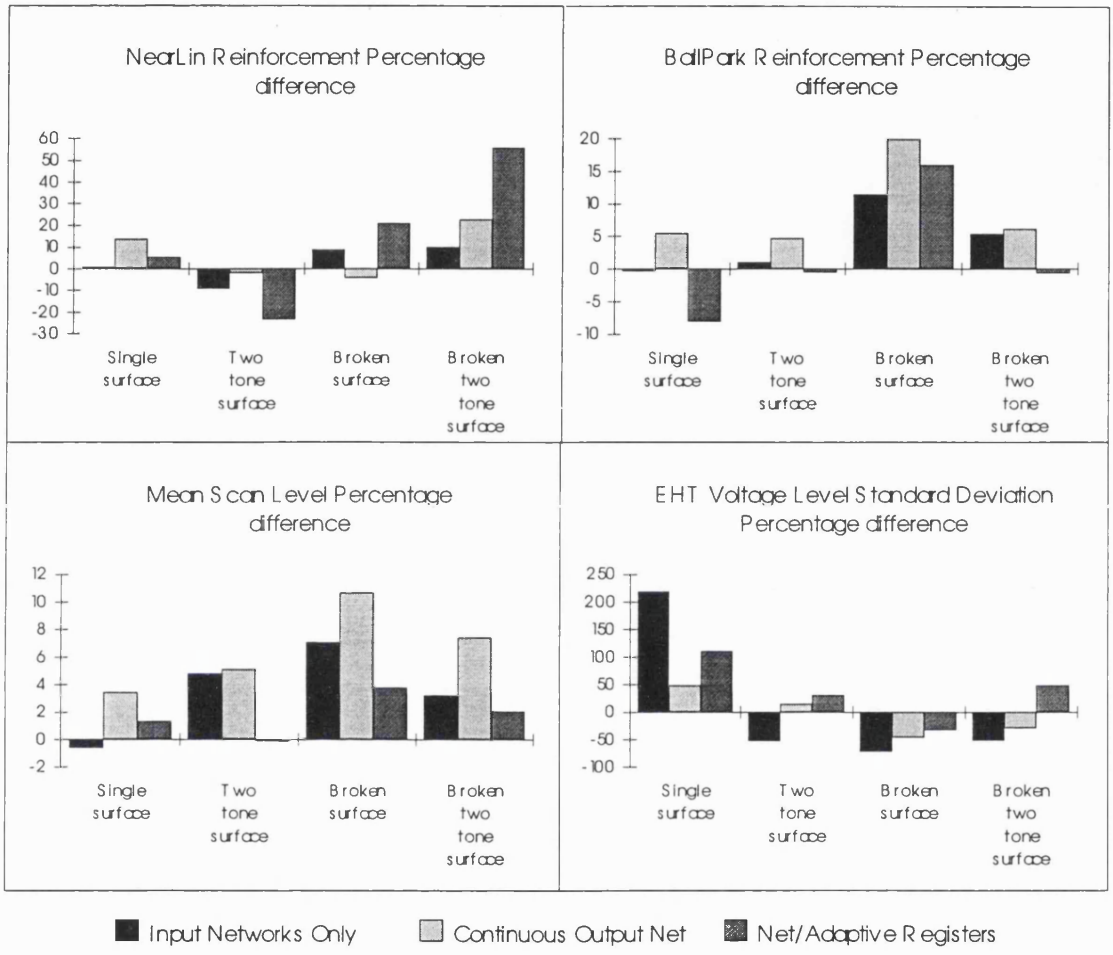


Figure 8.28. A comparison of "best of each" reinforcement results from the experiments in this chapter with comparisons in terms of percentage difference with respect to the performance of the basic component system from chapter 5. All results are taken from the broken-surface test sets (surface 3).

This chapter has presented an empirical exploration of some implementational aspects of learning and adaptivity in the sub-behavioural mechanisms of a behaviour based control structure. More specifically a number of additions to the basic AFSM building blocks of the subsumption architecture were made in an effort to provide a degree of automatic configuring and continuous adaptation to the ongoing environmental situation. Firstly the addition of a simple feedforward artificial neural network to the input-state identification part of an AFSM was tested and then followed by the addition of an automatic adaptation of the AFSM output registers. Finally these two mechanisms were combined and tested against a third method for providing adaptivity in an AFSM process, the continuous-valued output counterpropagation network. Figure 8.28 summarises the results by comparing the performances of the various implementations in terms of reinforcement values for individual AFSM processes and also at a system level in terms of mean

scan-level and the standard deviation of the EHT voltage output. The results for the broken-surface training examples are shown and the performance of each configuration on the other main test surfaces is compared. It can be seen that a clear improvement has been achieved through the addition of these mechanisms. In fact, an extra aspect not emphasised by the graphs in figure 8.28 is that of the improved robustness and resistance to noise and reduced inputs, as evidenced in figure 8.15 and figure 8.16.

In the last section the performance of the continuous-valued output counterpropagation, in terms of mean reinforcement values, did not appear greatly impressive. However, given the comparative graphs in figure 8.28 that show more detail in terms of a system-level examination using mean scan-level and the standard deviation of the EHT voltage output, the results are more notable. In the final analysis the choice of mechanisms is probably one that is application-specific and there is certainly nothing to preclude the use of both techniques at different points in the same control structure. It may be that in the long term the "pure" network approach will provide more scope for further research through the use of more advanced artificial neural network techniques. But it may also be beneficial to keep the adaptive output function, which is not very expensive in computational terms, and use this to adapt the basic AFSM rule which is in turn used to update the feedforward network as before. The adaptation then becomes a background mechanism only, manifesting itself through producing changes in behaviour via its influence on modification of the network weights. Future work of this nature is discussed in more detail in the next chapter.

Multi-Agent Interactions

One of the original aims of this work was to explore the feasibility of embedding learning and adaptation mechanisms into the functionality of a low-level behaviour based control process. The subsumption architecture was used as a target for the research in part because it provided a framework of asynchronous control processes, the augmented finite state machines. The issues of synchronisation have been largely left out of the discussion so far, but one or two factors have materialised as being important in this respect. These concern firstly the nature of setting up the parameters of the adaptation and learning mechanism so that the inter-AFSM processes will interact in a sustainably stable manner. The second factor concerns any explicit synchronisation strategy that the designer builds into the system. For example, a handshake type of signal between two AFSMs (which was included in the user interface layer described in chapter 5). It was the case with the *BallPark* and *NearLin* AFSMs in the photomultiplier control structure in this chapter that it was not possible to guarantee to maintain any interaction built in by the designer as part of the AFSM basic rule once the network had taken over the state identification/action selection process. Consequently the design and initialisation of an enhanced/adaptive AFSM process had to be such that any change in interaction could be compensated for elsewhere within the distributed system. Each component must be able to cope with continually changing characteristics of any interaction of processes. The problem is that the design task now includes

the local initialisation of internal process adaptation mechanisms. The agent-community-oriented view of systems at this level consequently becomes more relevant. We therefore take up this view as a topic of discussion in both the next and final chapters.

9. Discussion of Adaptation and Learning

This chapter covers the following:

Discussion based on the laser-scanner experiments.

General outlook and continuing development of enhanced-adaptive AFSMs.

The experiments reported in the previous chapter served to illustrate and highlight the application of learning and adaptive capabilities into the lowest levels of a behaviour based artificial intelligence (BBAI) control architecture. Chapter 7 built up the case and reasoning regarding the need and desirability for such mechanisms, which resulted from previous experience in the first part of this thesis and elsewhere ([Pebody91], for example). The present chapter aims to draw some conclusions from that work with regard to this second theme of the thesis.

This chapter is split into two main sections. The first part deals with conclusions and discussion of direct relevance to the laser-scanning test-bed control application. There are many loose ends and points of interest that resulted from the work reported in chapter 8 and several of these show promise for further development and experimentation. The second part of this chapter brings together aspects of a more general nature. In particular, the results are presented within the context of BBAI at large.

Chapter 8 has shown that through the addition of a number of relatively simple mechanisms a given behaviour based control structure may be made more robust and more effective. The enhanced systems were shown to provide a more stable scan signal over a greater variety of conditions as well as being resilient to degradation of sensory input (as illustrated in the previous chapter in sections 8.1.4 and 8.4). The application of learned feedforward input state-space mappings that provided an enhanced selection of actions (chapter 8.1), along with the automatic tuning of critical output parameters to locally situated asynchronous control components (chapter 8.2 and 8.3), was shown to be not only achievable without loss of system stability but also to be of observably consistent and positive benefit to the system on a global level (chapter 8.4). While some of these additions have been used previously in unique behaviour based control solutions (classical conditioning [Pfeifer & Verschure 91] and counterpropagation [Nehmzow & Smithers 90]), the work reported in this thesis has emphasised an architectural or developmental framework approach to the construction of such systems that has not been in evidence in earlier work. We hope that the techniques and lessons discussed here will be of use in, and provide stimulation for, further behaviour based control experimentation along these lines

9.1. Discussion on the Laser Scanner Experiments

The experiments and control structures presented in the previous chapter did not necessarily provide demonstrations of the best possible solution to the control of the an active laser-scanning inspection system. Rather they were implanted in order to test the mechanisms and ideas detailed in chapter 7 which in turn resulted from the demonstration of behaviour based control reported and discussed in chapters 5 and 6. Nevertheless, it has been found that the solutions arrived at are still of a usefully robust nature and can provide a serviceable system for use in a real industrial situation. There is of course much remaining scope for improvement and further development, but first it is necessary to discuss some important factors that affected the experiments and implementations of this work.

9.1.1. Factors Affecting Performance

Test Surfaces

One of the most important factors affecting the development and results of this work was the range and nature of sample test surfaces available for experimentation. In chapter 5.1.1 the provision of target inspection surfaces was detailed, and again in chapter 8 the variety of samples was shown (in figure 8.2). While only a subset of these was used for the reported experiments, there were in fact several other desirable surface types, each with varying characteristics selected to provide some significant obstacle to the laser-scanner control system. The surface types that appear in the experiments reported in this thesis were chosen carefully in order to provide a manageable and representative range of situations and characteristics.

The surface samples were also somewhat limited in their size and extent. The pattern repeated every 490mm (as the surface rotated beneath the laser-scanner). Although the surface rotation unit had variable speed, effectively providing a varying resolution of feature size, it was found that for most work the slowest speed available should be used in order to ensure that sufficient feature variation was available within the imposed surface-length-time constraints.

Low-Power Laser

Another factor that affected the extent of the experiments was the comparatively low power of the laser light source in the sensor. It was found that under many circumstances the laser light would be saturated by ambient light. This had several effects, both good and bad, on the experiments. For example, the low power of the laser meant that the photomultiplier devices tended to be running at the limits of their operational range. This was useful in that it increased the tendency of the outputs to go into the foldback states thereby, increasing the requirement for active control. A disadvantage was that it limited the variety and types of inspection surfaces that could be used.

Sensor Input Space

For experimentation with BBAI control techniques, a suitable balance is required in terms sensory complexity and practical manageability. Too simple a system leads to an insufficient richness of problem, experiments become contrived and the outcome is often inconclusive, while complex systems can tend to lose the focus of the experiment. At first sight we thought that the laser scanner test-bed might fall into the former category of over-simplicity since the range of available actuation of the system is, in the final analysis, only a one-dimensional problem space. But the non-linear nature of the control problem, in conjunction with the rich and critical input space, leads to what we can make an argument for calling an almost optimal level of complexity given the time constraints on this project work. Each photomultiplier controller input space consisted of an array of up to 1000 pixels each with a value in the range 0 - 255 representing the intensity of detected laser light from the inspected surface. For the task in hand, this provided an interesting and yet manageable set of sensors and was probably one of the main contributing factors to the focus and timely completion of this work.

Multi-Channel Sensor System

The experiments reported here illustrate the use of only one sensor channel while the test-rig provided two. Work has been conducted on a two-channel system which was implemented successfully by simply copying the first channel's software to the second and providing a set of high-level user interface AFSMs as implemented in chapter 5. It was found that the adaptive AFSM processes were able to adjust for the different characteristics of the second photomultiplier channel (no two systems are alike) - which, in the case of the original reactive AFSM, would have required considerable hand-tuning of critical parameters. This illustrates a significant payoff for the cost of developing the first adaptive enhanced AFSM control structure. It is possible to envisage the transfer of these software modules to as many different photomultiplier channels as necessary, and in this aspect the results are certainly encouraging. However, for the purposes of continued enhancement of this application, the second channel was not used extensively due mainly to the bottleneck of the top server-level transputer and its inability to handle the necessary storage of large amounts of runtime data concerning the trace of the system's status.

Industrial Situation

The implementation of an industrial laser-scanning inspection system test-bed for experiments in behaviour based control provided an unusual problem domain. As such, the industrial framework contributed towards the shaping of the solutions reported in this thesis. The nature of the control of the sensor required a system that was able to analyse a signal that reflected characteristics of the physical environment and then to make changes to maintain the quality of the signal. The problem of differentiating between changes resulting from external influences and those from the controller itself was real and in fact characterised in a straightforward way one of

the fundamental problems that are the focus of much of the research effort on the control of autonomous systems.

9.1.2. Further Application-Specific Development and Enhancement

Utilising the Temporal Dimension

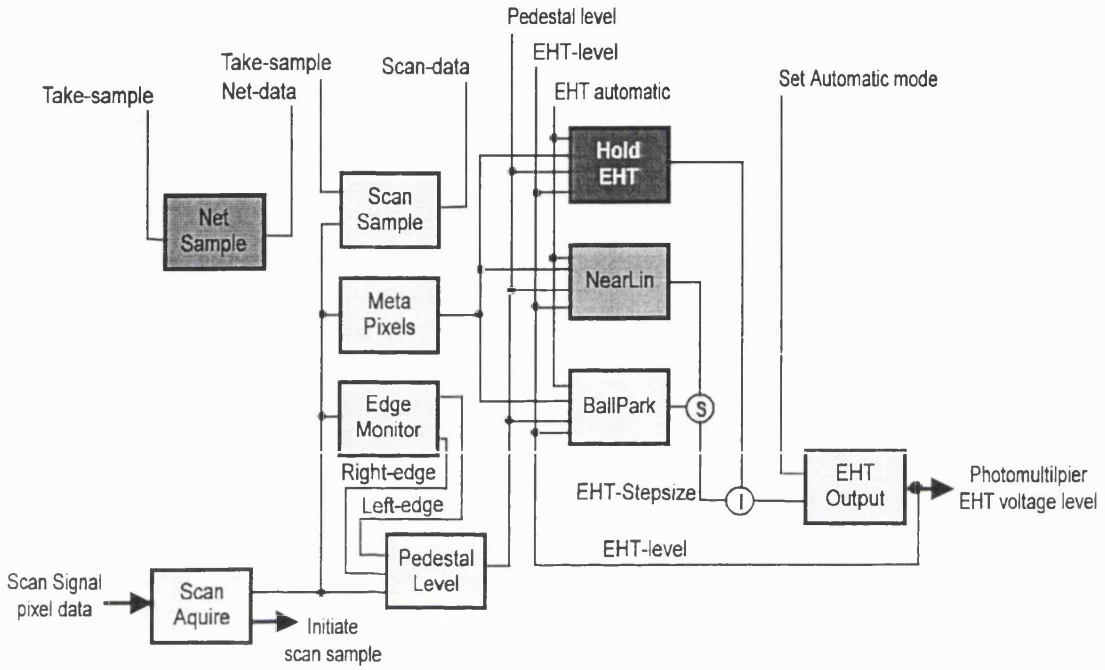


Figure 9.1. A third subsumption layer using temporal information to inhibit the lower levels of EHT voltage stepping during "recognised" periods of surface disappearance.

Further development of the photomultiplier control strategy for a single channel may be achieved by adding a third subsumption level onto the existing control structure as shown in figure 9.1. The idea is that this enhanced AFSM learns to override and inhibit the output of EHT voltage step commands during periods when there is no surface present to inspect. This would prevent the hunting of EHT voltage that is characteristic of this transition period as the controller continuously ramps up the EHT voltage in a search for any detected laser light. When the surface actually returns, the EHT voltage setting is typically far from the right value, and a period or readjustment is needed while the signal set-point is re-acquired. By learning surface characteristics over time, this layer should be able to recognise and predict regularly-occurring breaks and hold the EHT voltage at a fixed level until the surface reappears. Obviously a strategy is required both for the occasions when mistakes are made and when the surface does not reappear as expected. After this the hunting strategy would take over again. The implementation of this expansion into the domain of temporal activity could be implemented in a number of ways, either through the simple maintenance of a shifting array of scan-level values that is updated at each characteristic time step (see [Pebody94] for an implementation of a *TimePixels* AFSM) or through the use of recurrent or feedback artificial neural networks such as Hopfield networks and

bidirectional associative memory nets (see [Wasserman89] for an overview and further references). The incorporation of other mechanisms is discussed in more detail in section 9.2.2 below.

Inter-Channel Communication

Finally, it is worth suggesting here that a large area of interest might be opened up through the idea of experimenting with inter-channel communication. As one possibility, the agent-like properties at the level of the photomultiplier control subsystems might be utilised in order that they "compare notes" on activities. For example, if one channel is spending large amounts of time in a foldback state it may benefit from "looking" at the state of the other channel. This may result in either the recognition of some component failure or a change in control strategy to compensate for the new situation. A possible scheme is illustrated in figure 9.2. The mechanisms of associative learning, including a temporal aspect, could prove useful here in an arrangement similar to the original work that was reported in [Pfeifer & Verschure 91] (although in this case the temporal association was a result of the physical properties of sensor devices). This communication layer may well serve to bring the channels of the system together to some extent as a higher-level agent system, presenting a combined front to other parts of the system.

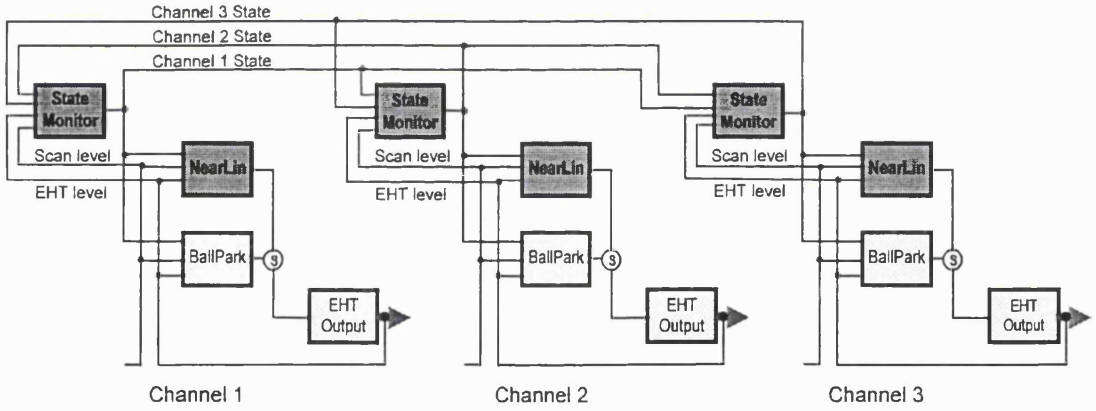


Figure 9.2. Showing a possible inter-channel communication between photomultiplier controllers.

9.2. A More General Outlook and Continuing Development

Having dealt with application-specific continuations of experimentation and development along the lines of the laser-scanning inspection system, this section now turns to a more general examination and discussion of the enhanced and adaptive AFSM mechanisms within the framework of BBAI.

9.2.1. Discussion on Enhanced and Adaptive AFSM Processes

At the beginning of chapter 7 the characteristics of behaviour based system development were discussed and the nature of the exercise of constructing agent control structures from parallel and distributed processes was outlined. The importance in this procedure of embedding designer

domain knowledge was emphasised as a significant factor contributing to a system's capabilities. This designer knowledge was identified as manifesting itself in the control structure in the form of rule specifications including critical parameters within processes and sensor and actuator interconnections between processes. It was pointed out that the design of such systems was often a lengthy and largely empirical procedure with designers spending much time on the tuning of critical behaviour-defining parameters of distributed rule processes. This observation was in part the result of the earlier work reported in the first chapters of the present dissertation and in part the result of observation of other work in the BBAI community typified by [Arkin90b], [Ferrell94], [Maes89], [Mahadevan & Connell 92], [Mataric95], [Gat *et al* 94] and [Steels93]. The use of on-line learning and adaptivity was identified earlier in chapter 6 as a necessary component of these systems, though at a sub-behavioural level.

The Original Requirements

A solution to these problems was proposed in the form of a number of functional additions at the basic building-block level of a behaviour based control architecture. Techniques of unsupervised on-line learning and adaptation were suggested to provide a sub-behavioural level of automatic system configuration and continuous tuning. For experimental purposes the subsumption architecture was chosen and the augmented finite state machines used to support the additional functionality, although it was stressed that the ideas were intended to be useful in other BBAI control architectures and not to be specifically subsumption-oriented.

The additions to the subsumption architecture focused on the internal mechanisms of the AFSM building blocks. It was intended that the basic interconnection and interaction of the subsumption architecture should remain unaffected. In particular, the aspects of synchronisation between AFSMs and the use of the loose synchronisation provided by the characteristic time factor were considered of fundamental importance, as was a consistent characteristic time at all stages of the system's operation. Also, the single-element input registers and the suppression, inhibition and default wire nodes were not to be altered in any way. The result was a set of three proposed additions to the basic rule element of the subsumption architecture.

i) The utilisation of a simple feedforward artificial neural network to learn an association between the AFSM inputs and output selection and activation. The AFSM rule was used to bootstrap the basic rule inputs with the AFSM outputs and to provide the integration of a further set of auxiliary inputs.

ii) Tuning of output register values and other critical parameters through the utilisation of a simple hill-climbing optimisation approach applied separately to each target variable.

iii) The use of a real, continuous-valued output artificial neural network similar to that used in (i) above but providing a complete input-output mapping of the basic AFSM rule, again with the incorporation of additional auxiliary inputs.

The first two of these proposals were complementary in their effects while the third, a simple network extension of the first, was intended to supersede or replace the second. Chapter 8 reported on a series of experiments that tested the use of these mechanisms extensively.

Results

The solutions to the above mentioned requirements certainly fitted into the format of the subsumption architecture AFSMs in an almost seamless manner. From an external point of view, in terms of the interaction dynamics there was no difference in the timing and response speed between a normal and an enhanced/adaptive AFSM. Both types maintained the characteristic-time period that was pre-set for the system (as verified by a continuous characteristic-time monitor process that ran as part of the user interface layer; see chapter 5). The mechanisms of learning and adaptation did function successfully as on-line and unsupervised, processes although it was necessary to ensure that during the learning phase a sufficiently "interesting" set of environmental conditions was experienced by the system. Subsequent adaptive performance was in general improved in terms of the local reinforcement, global scan level and EHT voltage output performance measures used, as well as in the resulting increases in robustness against degraded inputs. It was found that the adaptive architecture increased the amount of domain knowledge that the agent could acquire for itself, making it is less dependent on the initial control structure and parameters imposed by the designer.

The resulting mechanisms have provided what is best described as sub-behavioural learning and adaptation. That is, the mechanisms have been embedded into the behaviour based control system at the level below that of the general behaviour-achieving structures which, in the case of the subsumption architecture, manifest as behaviour levels that consist of a number of interacting AFSM processes. It was found that the design process, in terms of the breakdown of system behaviour, was little different from that of the fundamental behaviour based technique, although extra specification and design of enhanced/receptive AFSM input sets was required. Examples of such details include the selection and transduction of auxiliary inputs and the choice of critical output-register parameters needed to achieve the best-performance result. However, in the final analysis the design task is made more simple if it is remembered that the provision of basic rules with large numbers of extra real-valued inputs is a massive task not normally considered as viable for the basic AFSM. Consequently it should be stressed that these new mechanisms are not required in all parts of a subsumption control structure. In the case of the laser-scanning inspection system, many basic AFSM processes remained in operation and did not present any potential for enhancement.

There were several instances in which the only contribution of the "new improved" versions of AFSM processes to the global behaviour of the system was the costly reduction of overall stability. It was usually the case that these processes suffered from an unsuitable initialisation of parameters. The set-up of the update and reinforcement period parameters in the experiments of

the adaptation process in chapter 8.2 is an instance of this effect. The potential for instability is not an unexpected outcome. When the number of variables in a system is increased, the potential for instability is also increased. It is interesting to note that as the enhanced/adaptive AFSMs completed their learning phases and their behaviour became less predictable, the various adaptation mechanisms were able to cope with this and still provide a global as well as local improvement in performance.

This was reflected in the following ways:

i) The necessity of using a diverse number of factors to assess the performance of the system.

ii) The increasingly obvious difference between local AFSM process behaviour and that of the global system.

iii) The use of both local and global AFSM reinforcement performance measures: local in the form of preferred local state, and global in the form of the mean scan level over time and EHT output voltage standard deviation.

Consequently the appearance of the system as a community of interacting agents was reinforced and the fixed view of the system as a number of distributed reactive control processes became less realistic.

System Design and Development

The demonstrations of the enhanced input and adaptive output subsumption AFSM processes in chapter 8 are the result of a gradual and incremental development that was driven by the need to test ideas and explore potential solutions to problems. This is not a typical procedure as far as the development of actual applications is concerned. As a result, the question of whether this work actually does provide a useful technique for simplifying or at least helping in the development of autonomous systems remains open in some respects. An important point to emphasise here is that although a number of low-level factors are automatically taken care of by automatic mechanisms, this very utilisation leads directly to a new set of critical parameters that affect the behaviour of the system significantly. These in turn need to be set up by the designer. In the final analysis none of the proposed frameworks removes the task of system design; this would perhaps be wishing too much. The design task is however altered in nature and is perhaps more complex as a result. The systems that can be built with these added learning and adaptive components may be significantly more elaborate than it was previously possible to achieve. The management of large arrays of sensory information, for example, may be assimilated into an AFSM by comparatively simple specification of a basic rule set and then allowing the feedforward network component to take care of the details of associating the large amounts of sensor state information into the activity of the AFSM (although the designer must still exercise care with the selection of input sets so that the auxiliary inputs to an AFSM are at least relevant in some way and do not sap computational resources unnecessarily). Likewise the iterative task of

tuning critical parameters in the output of an AFSM may be left to an automatic process, the designer only having to specify a near guess at a suitable value. The designer is thus able to develop a much larger system than previously possible, as well as one that has a significantly more complicated interaction with its world.

Control Parameters

This issue of increasing the number of critical parameters in a system leads on to some interesting points. It was claimed above that the extra mechanism necessary to provide for an automatic and on-line learning behaviour leads to a new set of critical parameters which must be set up in order for the system to behave in a satisfactory way. These new parameters do however supersede to a large extent the criticality of the lower-level values which were of the threshold, max.-min. range and gradient-defining variety. This can be seen in the more abstract effect of the new parameters on system performance. For example, the four factors that affect speed of adaptation to change in the world of the output adaptation mechanism: reinforcement period t_r, adjustment period t_a, and the update factor f as well as the time period of the *neteffect* (ψ) variable in the input mapping network. The AFSM mechanisms that used these parameters proved to be relatively robust to initialisation values. More significant, apparently, were the magnitudes of the parameters in relation to each other (see results in the previous chapter 8.2, for example). It was the case that an acceptable though possibly sub-optimal performance could be achieved with only a small amount of thought on the part of the designer.

The main feature of the new group of pre-set parameters for the system is that they are nearly all related (either directly or indirectly) to the generation of credit for system activity, especially when experimental adjustments are being made. The solutions experimented with here were necessarily simplistic in order to keep the complexity and computational requirements of a single AFSM process at an acceptably low level. However, the problem of credit assignment is one of the main factors affecting automatic adaptation and learning in systems and has been recognised as such, probably since [Minsky61] was published.

Untested Factors

An important aspect of behaviour based control is the implementation of systems that are able to deal with events in the time domain rather than just behave in a reactive way to the current state of an agent-environment system. The experiments reported here did not deal with the temporal dimension other than in the accumulation of domain knowledge over time by the learning and adaptation mechanisms. However, this does not address the nature of temporal activity and sequencing of action that is in question here. The further extensions to the photomultiplier control structure detailed above in section 9.1 as a third subsumption layer would start to deal with this point, while the next part of this chapter below presents some ideas for implementing these factors within the enhanced adaptive AFSM processes. This issue is really less the concern of the work reported in this part of the thesis concerning photomultiplier control

than it is of the field of BBAI as a whole. The provision of behavioural mechanisms that exhibit some form of predictive or temporal look-ahead ability without recourse to central representation and symbolic manipulation is discussed in [Maes89] and [Churchland95] among other places.

Another factor that was not fully addressed in the reported work was that of the benefits of using the counterpropagation network in preference to the more common multi-layer network with learning by backpropagation. A comparison of performance along these lines would be useful, but in general it is an issue of implementation rather than a factor that bears on the principle in question addressed here; that of enhancing a subsumption AFSM process with the use of a feedforward artificial neural network mechanism. It is the case that there are some features of counterpropagation, mostly concerning the input Kohonen layer, that might be considered as weak points in the system. For example in [Wasserman89] it is suggested that the benefits of the shorter weight-settling times of the counterpropagation network might be offset by lower generality of the learned response. Even so, for the purposes of this work it has proved to be a robust mechanism (which was also the case reported in [Nehmzow & Smithers 90]) and the reasoning for its selection that was outlined in chapter 7 concerning computational cost and weight convergence has been demonstrated to be sound. It is clear that this mechanism would be a useful starting point for further research and other applications of enhanced subsumption architectures.

9.2.2. Other Techniques

The implementations presented in this thesis utilised two techniques of on-line knowledge accumulation and adaptation in the form of a simple two-layer feedforward artificial neural network and a simple multi-dimensional search using a steepest-ascent hill climbing type of approach. Before dealing with other viable techniques it is worth pointing out that within the domains of the techniques used there is an enormous amount of material and work, constituting research fields in their own right. For example, the field of artificial neural network research is massive and there are countless sources of ideas here alone. The work in this thesis has used ideas from [Wasserman89] and [Masters93] amongst other more applied sources such as [Nehmzow & Smithers 90] and [Pfeifer and Verschure 91]. Likewise methods of state-space search also constitute a massive research field, and again material is in abundance. Useful references here have been [Charniak & McDermott 85], [Luger & Stubblefield 89], [Maza & Yuret 94] and [Cvijovic & Klinowski 95]. Nevertheless, for the experimental work reported in this thesis the constraints imposed by the required simplicity of an AFSM process and the potential for large-scale parallelism meant that uncomplicated solutions were implemented in order to keep computational overheads to a minimum.

The addition of the mechanisms described in chapter 7 to the basic subsumption AFSM process structure was implemented in a modular way, so that (if desired) other association mapping or adaptation techniques could be built in with little extra work. One such interesting

experiment would be, as mentioned above, the implementation of a standard multilayer backpropagation neural network to compare with the behaviour of the currently-used counterpropagation network. Consequently an advantage of the system developed and presented in this dissertation is that it would be relatively easy to configure and experiment with other algorithms, and even other non-network-oriented learning mechanisms. Other areas of research that might be of some interest in this respect are principally characterised by the fields of Genetic Algorithms, Fuzzy Logic and Machine Learning, as well as other projects within the BBAI field. These fields are discussed briefly below.

Genetic Algorithms

Genetic algorithms (GAs), as outlined in [Goldberg89] and widely reported in conference proceedings such as [Varela & Bourgine 91] and [Cliff *et al* 94] are most often used for off-line development and optimisation, some of which has been directed at autonomous systems. Examples are [Harvey *et al* 94], [Beer & Gallagher 92], [Manela & Campbell 95], [Ram *et al* 94] and [Koza91]. The last of these details a GA evolution of a simulated subsumption based control structure. However, GA procedures do not lend themselves ideally to control solutions where continual on-line adaptation is required within the life-span of a single agent. GAs typically operate in large populations with ineffective solutions being weeded out as the representative phenotypes are killed off. Clearly if only one agent is available as a target system, death for any reason is unacceptable.

Work that does transfer GA control solutions to real-world robots has been reported in a number of papers that have explored the combination of simulation and real-world robot trials. For example [Nolfi *et al* 94] details control structures that are evolved with phenotypic plasticity (i.e. control structures that can adapt during their lifetime). In this way an instance of a control solution for a mobile robot has been shown to demonstrate an ability to adapt to different environments. Other GA work, [Lund95], deals with the evolution of artificial neural network controllers and the parallel evolution of the fitness criteria that dictates the selection of control solutions. This results in a dynamic fitness landscape that can adapt to different environments. Finally [Mondada & Floreano 94] also report on promising GA work that evolves artificial neural network control solutions to the perception and actuation of small mobile robots.

The contribution of GA techniques to the work reported in this thesis is most likely to manifest itself in the form of off-line development of the control structure for an agent. Most interesting here would be experimentation with the ideas reported by Nolfi and colleagues. It may be that the parameters affecting learning and adaptation along with the configurations themselves could be expressed as a particular genotype and optimised through evolution. But this would require considerable off-line experimentation or the provision of some sort of simulated domain.

Fuzzy Logic

Fuzzy logic (and, more significantly, fuzzy control) is another large research domain and has much to offer in the area of adaptivity and learning in control systems. A significant amount of work is reported in sources typified by [Kim.S-H *et al* 93] and [Kim.S-W *et al* 93] as well as [Harris94] and [Cox94] which provide recent collections of applications and techniques. It would be desirable to experiment with these techniques in the framework of the AFSM processes used in our work, particularly in the light of recent results on neuro-fuzzy controllers [Yamaguchi *et al* 94] and reports in [Godjevac94] concerning a behaviour based implementation of fuzzy control in a real-world mobile robot. The fuzzy-systems field remains an open area of interest with respect to the ideas presented in this thesis. The only reason why these ideas were not experimented with in the course of this project was that there was not enough time available.

Machine Learning

Machine learning is a wide and diverse field and it is most likely that some will claim that it encompasses all of the previous domains mentioned separately above. However, machine learning here is taken to refer to areas of more classically oriented AI learning that tends towards techniques of inductive logic programming as a form of relational learning as well as the learning of decision rules, e.g. as in [Bratko93]. The work reported in this thesis has taken on board many aspects of reinforcement learning but arguably in a different format to that typical of the machine learning literature, examples of which date back to research reported in [Minsky61] and [Tsypkin71].

One area of machine learning with particular relevance to the area of autonomous systems is that of Q-learning, originally proposed in [Watkins89] and extensively reported in [Kaelbling93], [Mahadevan & Connell 92], [Mataric95] and [Sutton91], amongst others. All of these deal with the robotics domain. However, the nature of this learning process is governed by the fact that the problem space to be learned is typically one described as a "Markovian Decision Process" (as discussed at the beginning of chapter 7 and in [Mataric95] and [Kaelbling93]). While at the behavioural level of control learning has been demonstrated using these techniques, we suggest that they are not so appropriate for the situations that have been the focus of this thesis, i.e. those of sub-behavioural learning and adaptation. One example that does implement a similar level of adaptation to that described here is that reported in [Clark *et al* 92]. While that work is very application-specific and does not appear to be readily suited to truly distributed control systems, it does at least look at the problem from a behaviour based architectural perspective.

Related BBAI Work

Techniques of learning and adaptation are widely researched within the field of BBAI. However, much of this work is still in the simulation stage and has not been demonstrated on real-world systems. For example, [Digney & Gupta 94] and [Weiß95] present distributed and

modular approaches to reinforcement learning, while [Gruau94] presents an off-line method of defining control systems using modular neural networks and genetic algorithms. Being simulations, the processes detailed in these works are not genuinely distributed, and so there are no allowances for the characteristics of physical process distribution that were outlined in chapter 6. Additionally Digney and Gupta state explicitly that: "...the robot model neglected dynamics and the sensors and actuators were assumed ideal.". It is the subsequent transfer of this type of work to the real-world that provides emphasis for the need for further research into the lower sub-behavioural level of automatic configuration and adaptation that has been the focus of this second part of the thesis.

Examples of behaviour based work that does address the problems associated with adaptive control architectures in real-world situations can be found in [Clark *et al* 92]and [Gaussier & Zrehen 94]. The work of Gaussier and Zrehen demonstrates a modular neural network approach to on-line learning and adaptation at a behavioural level in a real mobile robot, but the lower-level control structures in this example are not subject to adaptation. In contrast, the work of Clark *et al* does deal with the problem of low-level adaptation and also refers to a real-world robot control system. Still, this scheme would appear to be of a specialised parameter-tuning nature and specific to the application in hand. Consequently the work reported in this dissertation complements these accomplishments towards the development of autonomous control systems by demonstrating the real-world application of learning and adaptation that is embedded in the component parts of a general-purpose behaviour based control architecture.

9.2.3. Continuing Architectural Development and Experimentation

The work that has been reported in this thesis has been used to illustrate the development and construction of the distributed control components of an autonomous system. As is often the case with such work, the possibilities and avenues of interesting development are more numerous and extensive than the project time permits for complete exploration. Section 9.1 above has already outlined a number of interesting developments that might be implemented to further the performance of the laser-scanning inspection system in particular. This section outlines some more general ideas and concepts that would provide interesting continuation from the point of view of implementing sub-behavioural learning and adaptive capabilities in a behaviour based control architecture.

Regression to the Learning Phase

It was suggested in chapter 7 that one of the benefits gained from using the basic rule of the subsumption AFSM process to bootstrap the associative network is that there is an inbuilt and known base-level performance of the control process. It was claimed there that this enabled the system to have a guaranteed minimum performance. To date this has been implemented with the basic AFSM rule that tunes the associative network output layer throughout the life of the

system, i.e. the adaptive phase. This allows a continual adaptation to gradual change in the output of the Kohonen self-organising input network as the agent-environment situation changes over longer periods of time (in the order of several hours or more of operation). Nevertheless it has been the case during some experiments, usually triggered by a sudden change of test surface characteristic, that the system is unable to readapt in this way. Consequently the learning phase had to be restarted from scratch. This leads to two potential features worthy of development: (i) An automatic regression mechanism to reset the *neteffect* parameter back to zero, thus restarting the learning phase and (ii) once point one is working, experimentation with partial regression of the learning mechanism.

Automatic reset of the *neteffect* variable could be achieved relatively easily by adding an AFSM or two to monitor the system's performance (perhaps in comparison with other photomultiplier channels in the case of the laser-scanner situation) and externally affect the *neteffect* parameters of lower-level AFSMs. But the partial regression may not be achieved so easily. In the case of a reset, the network weights could simply be reinitialised to random numbers, but in the case of a partial regression the networks would be expected to retain some of their prior configuration and only partially readapt to the new agent-environment situation. This is not an insurmountable problem. Indeed the work reported in [Pfeifer & Verschure 91] used a decay factor in the network update functions on a continual basis in order that there was an inbuilt "forgetting factor" in the network's operation. Although this was implemented in a relatively simple (single layer) perceptron, it is not inconceivable that a similar mechanism could be built into the two layers of a counterpropagation network.

Sequenced and Layered Learning

The use of the two enhanced AFSMs in chapter 8 gives only a flavour of what might be achieved in larger systems. Still, there are aspects of the learning process that are not clearly understood, e.g. the nature of what happens or ought to happen when an AFSM undergoes its learning phase. If other AFSMs in the system play a significant role in the local control structure's behaviour, then it may be beneficial for higher-level AFSMs to hold back and wait for the lower-level learning phases to finish before they themselves begin the initialisation procedure of their own networks. In this way these higher-level AFSMs may benefit from a more effectively operating lower-level system on which to base their new input-space mappings. While this is all well and good, the reverse may also be true. It may be that the lower levels could benefit from waiting until the higher levels have completed their learning phases, for exactly the same reasons.

In order to effect a sequencing of learning phases an interconnecting signal would be required, daisy-chained between AFSMs so that they would indicate their progress through the learning phase. This signal could actually be the *neteffect* parameter. In this way local groups of AFSM processes would be able to monitor each other's progress. Given the indeterminacy about which AFSM would benefit the most from waiting to learn, it might transpire that a free-for-all policy

would be the best general approach. This would be even more the case if the AFSMs were able to regress and relearn if need be. It is also definitely the case that AFSMs will learn at different rates anyway. This is because the growth of the *neteffect* parameter is dependent on the frequency of the AFSM's interaction with its environment, and most AFSMs interact at different frequencies by the nature of their differing functions. Consequently it may be that the best policy is to leave a system to settle of its own accord.

In this vein there has been considerable work on the modelling in artificial neural networks of secondary and tertiary conditioning, and more besides. Secondary conditioning is the phenomenon of classical conditioning that occurs on top of or after a primary conditioning association between stimuli has been formed; likewise tertiary conditioning would occur after a secondary association. The mechanism used for the associative mapping in this work was based strongly on material reported in [Pfeifer & Verschure 91] and [Pfeifer & Verschure 93] which has dealt with models of classical conditioning. This phenomenon is also discussed in [Edelman87] and [McFarland85]. It is possible that the behaviour of low-level enhanced AFSMs may be used to bootstrap higher levels in the same way, with a more structured interdependency of stimuli and response than the somewhat loose grouping suggested above.

Further Sources Of Reinforcement

The adaptive outputs of the *BallPark* and *NearLin* AFSMs used a simple reinforcement function which monitored the state of the AFSM in question and set a flag when a predefined preferred state was entered. The reinforcement value was generated by taking the average of the values of this flag over a fixed time period. The result was a value in the range 0 - 1 with a 1 indicating a 100% occupation of the preferred state. This proved to be useful in the case of the laser-scanner test-bed but it may be beneficial at some point in the future to consider two further aspects: (i) The autonomous alteration of the preferred state that generates the reinforcement; (ii) The augmentation of the AFSM's internal reinforcement with that of an external source such as a higher-level AFSM.

Both the reinforcement function for the adaptive output stage and the effect of the *neteffect* parameter may be augmented externally via other sources of reinforcement. In general it would seem reasonable to postulate that both positive and negative reinforcement might be applied usefully to the register adaptation and *neteffect* mechanisms. The registers currently utilise an internal positive reinforcement in the form of the preferred-state flag, while the *neteffect*, being reduced to a lower value by an external-negative valued influence, would provide a regression to a more active learning phase as discussed above. Other possible sources of reinforcement would generally be application - and implementation - dependent although it is possible to envisage some benefit from monitoring the frequency of an AFSM's activity and quantity of unclassified states. One factor worth emphasising at this point is that remote sources of reinforcement may become more difficult to assign to local AFSM actions. This was one reason why only the preferred-state

reinforcement was used in the experiments in chapter 8. It may, for example, be possible to set up a local area of influence around a group of AFSMs such that if one attains a particularly high reinforcement measure, the others reduce the degree to which they adapt. But this introduces a further complication via aspects of communal interaction (which were also highlighted at the end of chapter 6.2) and this along with the other factors mentioned here is clearly an area for considerable continued research. [Sutton91] provides useful discussion on relevant issues in reinforcement learning in autonomous agents, as do [Kaebling93] and [Miller.W *et al* 90].

Further Aspects of Automatic Configuration

The counterpropagation network used in the enhanced AFSM to implement an associative learning mechanism allowed an additional set of auxiliary inputs over and above those required for the basic function of the AFSM to be included in the process of recognition of an AFSMs input state. This alone provided a degree of flexibility in design, since the format of the auxiliary inputs was left to the activity of the Kohonen self-organising input network. However, it was still necessary for the designer to pre-specify all of the connections and the configuration of the network layers. Work reported in [Fritzke91] details a self-organising-self-organising network, or in other words a Kohonen network that adapts its structure in response to the activities of the self-organising process. The result is an optimisation of the network size in response to the probability distribution of the inputs. Other work details a similar process for backpropagation networks; for example, [Khorasani & Weng 94] and [Salomé & Bersini 94]. Using these techniques it may be possible to provide a mechanism that searches the local neighbourhood of other AFSMs actively for other input connections that may correlate usefully with the AFSM's activities. This additional activity might be similar to that developed and demonstrated in [Maes & Brooks 90].

Mobile Robot Implementation

This work has been, to much positive advantage, conducted with the aid of an experimental test-bed constructed around an industrially situated laser-scanning inspection system. The advantages of this choice have been stated already (e.g. see chapter 4). It would be nice to be able to transfer some of these ideas to a mobile robot test-bed to experiment with the interactions of an enhanced subsumption architecture in a more diverse agent-environment system and task domain. The focusing effect of the laser-scanner test-bed has been a useful source of discipline in ensuring that experiments were thoroughly implemented. But in the final analysis, the majority of BBAI work is carried out on mobile robot test-beds and to compare the ideas of this thesis with other control, learning and adaptive architectures it would probably be necessary to use a mobile robot and real physical environment and task set.

9.3. Summary

This chapter has concluded the second part of the thesis with discussion of the implementation of embedded learning and adaptation at the sub-behavioural levels of a behaviour based control

architecture. This direction of development was taken after observing that the control of a distributed system using the techniques of BBAI resulted in the need for some form of designer-independent means of tuning critical parameters and classification of complex sensory input spaces. This was implemented in the framework of the subsumption architecture, which resulted in a distinctly communal control structure made up of a number of asynchronously-interacting AFSM processes. The agent-like nature of these processes that was highlighted in chapter 6 became even more evident when, given their own mechanisms for learning input-output mappings and adapting their actuation parameters, these process started to self-organise into a more effective community or (at the next level up) system. These observations build on those developed in chapter 6. The next chapter concludes this thesis by bringing the two parts together.

10. Autonomous Systems In the Real-World

This chapter covers the following:

Distributed behaviour based systems.

Systems as layers of agent entities.

Distributed sub-behavioural learning and adaptation.

Other points of view.

Conclusion.

This dissertation has reported on the course of a series of experiments that examined the use of techniques from behaviour based artificial intelligence (BBAI) in an industrial application domain. The reasoning behind this work was twofold. In the first place it was recognised that the field of BBAI has provided many novel and interesting solutions to problems of control of complex systems but that virtually all of the examples of the techniques have utilised mobile robot test-beds for experimentation - with little trace of industrially situated work, other than that of mobile robots that happened to operate in an industrial environment. We decided that it would be interesting to look at the use of these techniques in another test situation, and selected the industrially situated laser-scanning product inspection system as a significant and somewhat unusual example in part because of its modular and distributed hardware architecture. The second reason for the work was an observation that systems, including those of the BBAI mobile robot test-beds, are becoming increasingly complex and utilising several or many special-purpose hardware processing units (typically in the form of embedded microprocessors). The view of these systems in terms of a single behaviour-generating agent acting in the physical world was becoming overstretched and the multiple centres of activity, often physically distributed within the same system, were perhaps not being given the degree of attention required in order to elicit the best performance in terms of reliability and robustness of the main system. One of our aims was to gain and report some insight into the nature of constructing behaviour based control structures for systems of this kind containing such forms of complexity.

Because of the way in which the structure of the thesis has been presented in two parts, this chapter concludes by bringing the two main themes of the work together. In chapters 4, 5 and 6 the implementation of and subsequent experimentation with the behaviour based control strategy for an industrially situated laser-scanning inspection system was detailed. This work involved the

development of control systems for the laser-scanning sensory subsystem of the main inspection equipment, and prepared the way for the second part of the thesis by identifying the need for a distributed sub-behavioural level of learning and adaptation in BBAI control structures. In that second part, chapter 7 posited some solutions to these requirements while chapters 8 and 9 illustrated and discussed experimentation in this area on the laser-scanning test-bed.

This chapter concludes by discussing the most important results that have emerged from the reported work. In particular, the viewpoint of distributed behaviour based systems and its emphasis on control in terms of the interactions of physical real-world systems is dealt with. Next, the need for BBAI to address lower levels of system interactions is discussed. This is used to introduce the idea that the control of autonomous systems may be seen usefully as a series of levels of agent communities acting on different levels of interaction abstractions. The use of learning and adaptation is then reviewed in order both to summarise the case for its use and to reinforce the agent-like view of asynchronous control processes. Finally, some alternative viewpoints are briefly outlined.

10.1. Distributed Behaviour Based Control Systems

The fundamentals of BBAI were covered in chapter 3. Factors including situatedness, groundedness, physical embodiment, non-symbolic computation, emergence and (not least) bottom-up development were described and highlighted. These characteristics typify a number of diverse examples of research into autonomous systems, ranging from robotics and engineering [Arkin90] to neuroscience and cognitive science [Giszter94]. Each of these fields, although its practitioners may not necessarily consider it to be a particular branch of artificial intelligence, does nevertheless contribute to a common theme. Together, they have generated a large body of knowledge about the control of real-world situated systems.

The exploration of the use of the subsumption architecture as a framework for the control of the laser-scanning inspection test-bed in chapter 5 led to some interesting observations of system design, using such techniques, which were detailed in chapter 6. Firstly in this implementation it was observed that the constraints imposed by the physical as well as functionally distributed target system demanded the development of a control structure that consisted of a number of almost independent subsystems. "Independent" here refers to the degree with which components within the subsystems interact amongst themselves compared to interaction with other parts of the system. Additionally, it was found that the nature of each subsystem was distinctly agent-like in its own right but that the typical task and behaviour requirements were at a level below that of the main agent system. Consequently, a layered hierarchy of agent-communities was proposed, with the definition of each agent-community layer being a function of its temporal and spatial abstraction from the real physical world.

10.2. A Lower Bottom to the AI

Chapter 6 highlighted the ongoing trend of increasingly complex control systems both within the BBAI field and in technology at large. Systems typically consist of multiple centres of computation and localised processing of control functions. There are examples that cover both BBAI ([Smithers94], [Ferrell94], [Nehmzov & McGonigle 94], [Nourbakhsh *et al* 95] and [Balch *et al* 95]) and the world in general (such as manufacturing systems and automated banking systems). It was suggested that the embedding of low-level processing often resulted in a fixed-strategy control of active sensors and sophisticated actuator devices. This assignment of subsystem control to the sidelines, it was claimed, led to a view that sensors and actuators were just interfaces to the world.

This attitude seems to be particularly prevalent in the BBAI literature (see for example [Ferrell93], [Steels94], [Smithers94], [Ruyssinck92], [Kweon *et al* 92], [Vereertbrugghen93]). It can be attributed to the early experiments in BBAI that utilised mobile robot test-beds which provided relatively simple, often binary, sensors and actuators. The strength of the early BBAI work was in the demonstration that these systems were capable of complex and even sophisticated behaviours in cluttered and unstructured environments despite (or more likely as a result of) the simplistic sensor-actuator functionality. Work is now focusing on BBAI systems with increasingly complex sensors and actuators which often provide and respond to real or continuous-valued signals (although often within the limits of digital to analogue and analogue to digital conversion devices) where the simple control strategies are no longer proving to be so robust. More effort is now being put into sensor and actuator transduction. In fact, complete subsystems are being constructed with processors that are dedicated to these increasingly important tasks. Figure 10.1 below is repeated from chapter 6. It shows again in 10.1b that the result of setting aside sensor and actuator control as being "someone else's problem" can lead to the subtle concealment of the true distributed structure that is an important characteristic of BBAI systems.

The behaviour based control structure developed for the laser-scanner test-bed, however, did not conform to this regressive tendency. Even so, in the first instance it was not by conscious design. The hardware architecture and characteristics of the transputer network that provided the computational resources for the test-bed effectively dictated the architecture in figure 10.1a. As a result, a new sub-behavioural level was identified in the system: a level that concerned the control of complex sensor devices such as the photomultiplier controllers, and actuators such as the test surface rotation motor control which automatically detected stalled states and acted to restart the motor. The processes at this level were built up around a small number of subsumption AFSMs which were referred to in chapter 4 as behavioural nodes. These, as behavioural nodes, acted on an asynchronous and local basis to achieve their own task agenda. Being physically embodied,

these behavioural nodes have a case for being viewed as agents in their own right. This is exactly the interpretation that we propose.

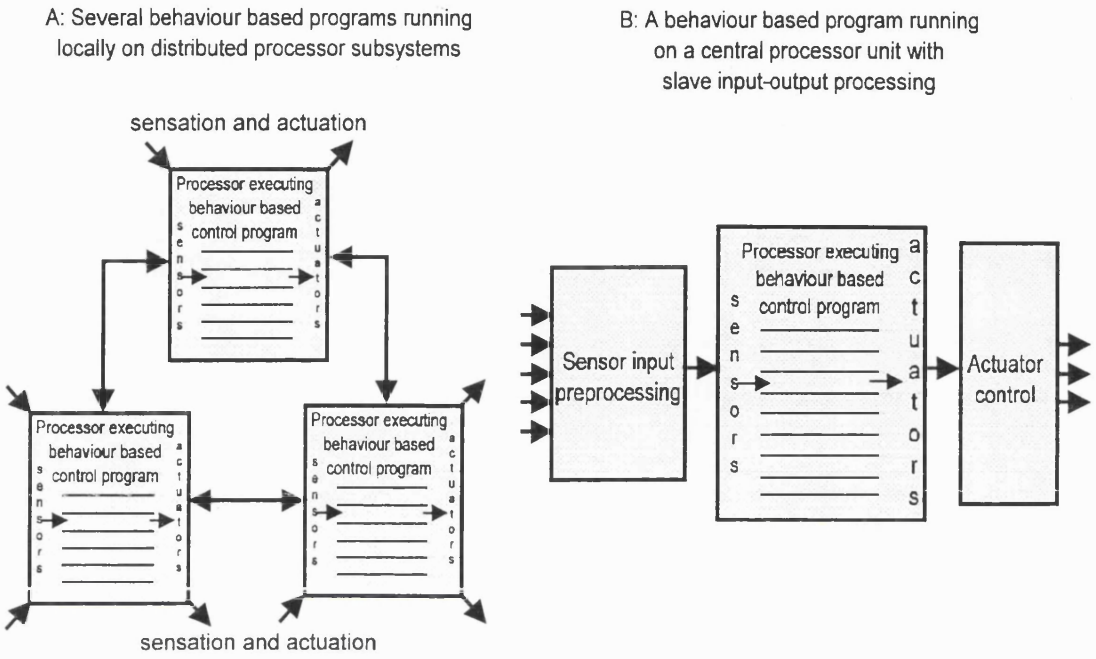


Figure 10.1. Two interpretations of behaviour based systems. A: a genuinely behaviour based approach to low-level sensor and actuator control. B: showing the software implementation on a classical hardware architecture and the subtle and hidden maintenance of a pipelined control flow.

10.3. Layers of Agent Entities

On examining the processes of the laser scanner implementation, we recognised that there was a pattern dictated by the time and space dimensionality and abstraction of different layers in the system. This was discussed in chapter 6. It would appear that there is a closed loop of reasoning which follows: High-level control at the behavioural level (and above) tends to deal with abstract representations of sensor and internal state including dynamical patterns that are the result of extended interaction over the lifetime of the system (e.g. the proposed level 3 of the photomultiplier controller detailed in chapter 9). Lower levels, such as those that have been the subject of this thesis, are more directly coupled with the physical world. For example, the control of active sensor devices and actuators takes place within a shorter temporal span and possibly has to deal with a greater spatial domain (as for the *PedestalLevel*, *EdgeMonitor* and *MetaPixelGen* AFSMs in the laser scanner). The low-level systems are embedded in the physical interaction of the agent and its environment, and as a result must be able to filter out massive amounts of rapidly changing and irrelevant information from their "view" of the world in order to provide higher levels with stable and more slowly changing information. They must do this for two reasons: firstly, the higher levels are normally physically remote from the low levels and the transfer of large amounts of information in suitable time periods is not practical; secondly, the higher levels require information of a more general nature in order expand the temporal domain in

more abstract terms. The latter is true in part because they are forced to interact on a longer time scale due to their remoteness from the physical world and also because of the benefits gained from taking the temporal domain into account. This brings us back to the beginning of the loop. Figure 10.2 shows this in pictorial format with the low-level processes of an agent acting with a large spatial resolution and feeding back to higher levels that have an increased temporal resolution and different terms of reference from those of the lower layers.

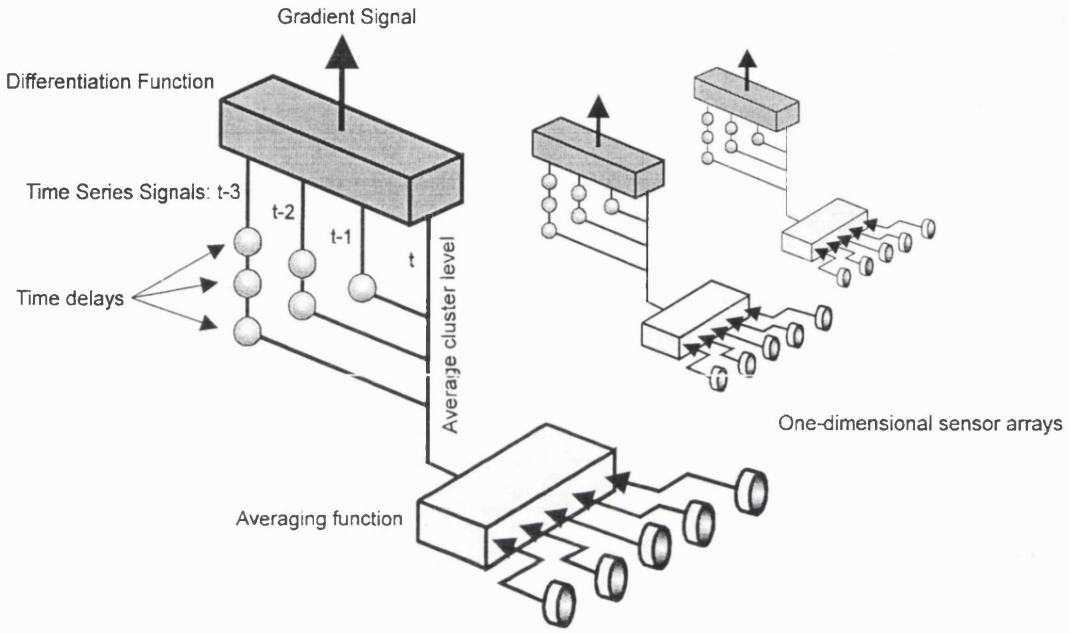


Figure 10.2. Low-level spatial resolution turning into high-level temporal resolution.

This layered abstraction based on the temporal and spatial interactions of processes is also suggested and implemented in mobile robot control in [Albus81] and [Herman *et al* 90]. Further, it is similar to the discussion in the Society of Mind [Minsky85], although the ideas presented in that reference are largely unsubstantiated there in terms of physical real-world examples. Other sources too recognise this phenomenon which, if not considered explicitly in system development, results implicitly from work on system construction and development. Work that does address these issues can be found in [Causse & Christiensen 94]. The relevant message of the first part of this thesis was that behaviour based systems must also take this gradation of interaction into account, but in a way that maintains the essential characteristics of the field. We have done this in our work. A resulting observation is that at each level the interacting processes may be viewed as communities of interacting agents or agent-processes (depending on the viewpoint of the observer), not only at that of the physical-world agent environment level but as also at layers of activity below that, within subsystems of individual agents.

This layered multi-agent community view of a system emerges naturally in many examples. Starting at the physical component level it can be seen that there is a clustering of activity that has already been identified as the sub-behavioural level. The next level up is that of the larger

task and behaviour-achieving agent that interacts in the world in the manner of a robot or animal. Above this it is possible to identify a level of interacting communities, such as groups or teams of people or more obviously the communal insects such as bees, ants and termites. In this vein BBAI work in multi-robot systems has been reported in [Parker92], [Balch *et al* 95], [Mein91], [Mataric95], [Steels89] and [Steels94], among others. A similar upward expansion of agenthood can be attributed to the more technological domain of the industrially situated-laser scanner system. The bottom level is the activity of the controlling AFSM processes that interact and manifest themselves at the subsystem level of the photomultiplier and optics controllers along with the user interface. The next layer up is that of the hardware parts of the laser-scanner, the optical head and the front-end defect-detection host computer. Going one level up again, there is the level of interacting machines in the machine room of the factory.

Given this layered picture, it is reasonable to postulate that the general techniques, design philosophies and frameworks of BBAI systems may well prove to be sufficient for system development at all these levels of agent abstraction. It must be emphasised that there is still much work to be done to investigate the effect and value of this postulate, though, not least in the automation of the design and development process that was introduced and explored in the second part of this thesis.

10.4. Distributed Sub-Behavioural Learning and Adaptation

The second part of the thesis led on from the first as the consequence of a perceived need for automation in the design and development of complex BBAI control structures at the various levels of abstraction and terms of reference identified above. Aspects of embedded learning and adaptation were examined in order to assess the effect of building them into a BBAI control architecture at a component or sub-behavioural level. The results were demonstrated in the framework of the subsumption architecture with the respective enhancements and additions to the basic augmented finite state machine (AFSM) processes. The use of learning and adaptivity has been reported extensively in the BBAI literature, but almost entirely at the level of the behaviour of the main agent in response to its environment. Our work has concentrated on the actual mechanisms that maintain the behaviour - mechanisms that are normally hardwired at the design stage.

It was found that the enhanced/adaptive AFSM processes could be built beneficially into existing subsumption control structures and that two or more such processes could be utilised with no adverse effects in terms of system stability. On the contrary, it was found that the systems with more that one enhanced/adaptive AFSM were generally more effective on a global system level than their non-enhanced counterparts in terms of both robustness and task achievement. However, it was also noted that it was not necessary to enhance all of the AFSMs within a system; some cases existed where no obvious benefit would result from the added

complexity (e.g. the AFSMs in the scan acquisition subsystem of the photomultiplier control structure in chapter 5 section 2).

In general the robustness of a system is increased if subsystems are designed to be as self-sufficient as possible and are able to utilise their own "knowledge" of their local domain in order to function better. "Knowledge" in the case of the BBAI systems implemented for this thesis refers to the implicit domain knowledge that is present in the parameter values and structure of the various levels of an agent's control mechanism. Although a system that consists of a number of communicating, interacting and asynchronous (i.e. with no imposed artificial timing constraints that can lead to deadlock between processes) agents may perhaps not achieve an optimal performance in any given situation, it is nevertheless inherently more robust due to the decreased dependence on specific relationships with other parts of the system.

10.5. Some Other Points Of View

The lower-level AI based control advocated in the thesis is not intended to replace the techniques of classical control. At the same time it is not augmenting a classical control regime. The nature of AI system design has so far generally failed to address the lowest of the conceptual levels of activity that have been highlighted by the work reported in this thesis and the domain has largely been left to be filled by existing tried and tested control techniques. This trend has resulted in the development of some hybrid systems that utilise a two-part combination of both methodologies: AI and classical control (an example of which is detailed in [Herman *et al* 90]). These architectures have in general been the focus of much recent attention. However, there is room within the AI framework and BBAI in particular for attention to lower-level control issues that are not suited for treatment by classical techniques. It is suggested that any hybridisation at these lower levels would seem to be more likely to occur beneficially with a merging of techniques rather than from a basic top-down two-part split.

The problems typified by real-world domains such as robotics consist of tasks that contain unpredictable events, changing environments and systems that are difficult to model [Franklin & Selfridge 90]. The work reported in this thesis has taken the issues of intelligent-agent-based control down to that of sensory subsystems. This effectively removes the need for the two-part split between high-level AI and low-level control development that is a fundamental characteristic of hybrid systems. Although both classical control and the planning and modelling frameworks of classical AI are tried and tested methodologies that are used far too widely and successfully to be dismissed, the positive results of the experiments on the laser-scanner inspection system reported in this dissertation have shown that the low level AI, intelligent-agent-based approach is at least viable and shows promise for use in applications as well as in future research.

10.6. Conclusion

The results of the second part of the thesis reinforce the agent-based system view of an autonomous system's control structure that is advocated in the first. By enhancing some of the component parts of a behaviour based control architecture at a *sub-behavioural* level, we achieved an improvement in global system stability and robustness. The additional functionality included the embedding of a number of different processes that ran in the background of the basic mechanism, providing automatic configuration (*learning*) and *adaptation*. These processes were shown in chapter 8 to provide also an automatic association of complex input information with a basic system operation, and a continuous tuning of output and actuation characteristics. At the level of the work reported here, the basic architectural building-block was an inherently asynchronous component of a behaviour-achieving control structure. The addition of an ability to learn useful input-space configurations and to adapt output continuously to ongoing changes in the immediate surroundings and situation gives these basic processes a significant boost in the direction of autonomy. For example, it was reported that designer-specified relationships between two such processes were often changed dynamically later as they gained experience in their environment. Thus it can be appreciated that these processes are acquiring their own repertoire of skills and abilities on top of and sometimes in place of those originally laid out by the designer.

It was stressed that these "sub-behavioural" components were operating as part of a behaviour-achieving structure, but it is now apparent that they are exhibiting their own form of behaviour as dictated initially by the designer but to an increasing extent driven by their ongoing interactions with surrounding processes. At this level of system abstraction, below that of the physical agent system (such as a mobile robot or product inspection system) there is clearly a lower-level community of interacting agents, each with its own unique behaviour. This presents a very different view of a system from that generally taken in classical design and classical AI and one which currently does not provide any kind of formal methodology, though [Smithers94] and [Beer94] give examples of work directed toward this end based on dynamical systems theory while [Colombetti *et al* 94] presents an attempt at setting out a formal approach to systems design and development called "Behaviour Engineering".

This multi-level, distributed and multi-agent view of complex systems gives a different perspective from that normally taken by designers. Any use of it in the future should contribute to and augment understanding of the physical and behavioural requirements for complex systems development, in a realistic environment. This thesis has illustrated many aspects of this novel approach to system development. Chapter 9 indicated the lines of future research that we consider to be most useful on the basis of our experience with the laser-scanning test-bed application, and each of these fall within the three main result areas that have been emphasised in this concluding chapter: i) the need for a layered agent-based approach to the distributed control of real-world autonomous systems (section 10.1 and 10.3), ii) the presence of sub-behavioural layers of

interaction and activity below that currently addressed by BBAI systems (section 10.2), and iii) the benefits of embedding mechanisms of learning and adaptivity within all these agent-levels including, especially, the lowest level aspects of sensor and actuator interaction (section 10.4).

References

Albus81	*Brains, Behaviour, and Robotics.* James S. Albus. Byte Books. 1981.
Arkin & Ali 94	*Integration of Reactive and Telerobotic Control in Multi-agent Robotic Systems.* Ronald C. Arkin and Khaled S. Ali. From Animals to Animats 3. Proceedings of the Third Conference on Simulation of Adaptive Behaviour 1994. Cliff, Husbands, Meyer and Wilson editors. MIT Press.
Arkin87	*Motor Schema-Based Mobile Robot Navigation.* Ronald C. Arkin. Proceedings of the IEEE International Conference on Robotics and Automation, Pg. 264-271. April 1987.
Arkin90	*The Impact of Cybernetics on the Design of a Mobile Robot System: A Case Study.* Ronald C. Arkin. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 20. No 6. Pg. 1245-1257. November/December 1990.
Arkin90b	*Autonomous Navigation in a Manufacturing Environment.* Ronald C. Arkin and Robin R. Murphy. IEEE Transactions on Robotics and Automation. Vol. 6. No. 4. Pg. 445-454. August 1990.
Arreguit & Vittoz 94	*Perception Systems Implemented in Analogue VLSI for Real-Time Applications.* X. Arreguit and E. A. Vittoz. Proceedings: From Perception to Action Conference. September 1994. Pg. 170-180. IEEE Computer Society Press. Editors P. Gaussier and J-D. Nicoud.
Balch *et al* 95	*Io, Ganymede, and Callisto. A Multiagent Robot Trash-Collecting Team.* Tucker Balch, Gary Boone, Thomas Collins, Harold Forbes, Doug MacKenzie and Juan Carlos Santamáña. AI Magazine, Vol. 6, No. 2. Pg. 39-51. Summer 1995.
Barto90	*Connectionist Learning for Control.* Andrew G. Barto. In Neural Networks for Control, Miller, Sutton and Werbos editors. MIT Press 1990.
Beckers *et al* 94	*From Local Actions to Global Tasks: Stigmergy in Collective Robotics.* R. Beckers, O. E. Holland and J-L Deneubourg. In Proceedings of Artificial Life 4. R Brooks and P Maes editors. MIT Press 1994.
Beer & Gallagher 92	*Evolving Dynamical Neural Networks for Adaptive Behaviour.* Randall D. Beer and John C. Gallagher. Adaptive Behaviour Vol. 1, No. 1, Pg. 91-122. Summer 1992. MIT Press Journals.
Beer90	*Intelligence as Adaptive Behaviour.* Randall D. Beer. Academic Press Inc. Harcourt Brace Javanovich, Publishers, Boston. 1990.
Beer94	*A Dynamical Systems Perspective on Agent-Environment Interaction.* Randall D. Beer. Artificial Intelligence 1994. Publishers: Elsevier Science B.V. North Holland.
Berghuis *et al* 93	*A Robust Adaptive Robot Controller.* Harry Berghuis, Romeo Ortega and Henk Nijmeijer in IEEE Transactions on Robotics and Automation, Vol. 9, No 6. December 1993.
Billard & Pasquale 93	*Effects of Delayed Communication in Dynamic Group Formation.* Edward A. Billard and Joseph C. Pasquale. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, No 5, September/October 1993.

Bissell94 — *Control Engineering.* C. C. Bissell. Tutorial Guides in Electronic Engineering 15. Chapman and Hall, London. 1994.

Boden94 — *Autonomy and Artificiality.* Margaret A. Boden. In AISB Quarterly, No 87. Pg. 22-28. Spring 1994. Journal of the Society for the Study of Artificial Intelligence and Simulation of Behaviour. University of Sussex, Brighton, UK

Braitenberg84 — *Vehicles. Experiments in Synthetic Psychology.* Valentino Braitenberg. MIT Press 1984.

Bratko93 — *Machine Learning in Artificial Intelligence.* Ivan Bratko. Artificial Intelligence in Engineering. 8. Pg. 159-164. Elsevier Science Publishers Ltd. North Holland. 1993.

Brooks & Stein 93 — *Building Brains for Bodies.* Rodney A. Brooks and Lynn Andrea Stein. MIT Artificial Intelligence Laboratory Memo No. 1439. August 1993.

Brooks86 — *A Robust Layered Control System For A Mobile Robot,* Rodney A. Brooks, IEEE Journal of Robotics and Automation. Vol. RA-2, No 1, Pg. 14-22. March 1986.

Brooks86b — *Achieving Artificial Intelligence Through Building Robots.* Rodney A. Brooks. MIT Artificial Intelligence Laboratory Memo no. 899. May 1986.

Brooks89 — *A Robot that Walks; Emergent Behaviours from a Carefully Evolved Network.* Rodney A. Brooks. MIT Artificial Intelligence Laboratory Memo 1091. February 1989.

Brooks89b — *Fast, Cheap and Out of Control: A Robot Invasion of the Solar System.* Rodney A. Brooks and Anita M. Flynn. Journal of The British Interplanetary Society, Vol. 42, Pg. 478-485. London. 1989.

Brooks90 — *Elephants Don't Play Chess.* Rodney A. Brooks, Robotics and Autonomous Systems 6 (1990).

Brooks90b — *The Behaviour Language Users Guide.* Rodney A. Brooks, MIT AI Memo 1227 April 1990.

Brooks91 — *Intelligence without reason.* AI memo 1293, Massachusetts Institute of Technology AI Laboratory. 1991.

Brooks94 — *Coherent Behaviour from Many Adaptive Processes.* Rodney A. Brooks. From Animals to Animats 3. Proceedings of the Third Conference on Simulation of Adaptive Behaviour 1994. Cliff, Husbands, Meyer and Wilson editors. MIT Press.

Burns88 — *Programming In Occam2.* Alan Burns. Addison-Wesley Publishing Company, Wokingham, England. 1988.

Causse & Christiensen 94 — *Hierarchical Control Design Based On Petri Net Modelling for an Autonomous Mobile Robot.* O. Cause and H. I. Christiensen. Laboratory of Image Analysis, Aalborg University, Fr. Bajers Vej 7, DK-9220 Aalborg East, Denmark. 1994.

Charniak & McDermott 85 — *Introduction to Artificial Intelligence.* Eugene Charniak and Drew McDermott. Addison-Wesley Publishing Co. Reading Massachusetts. 1985.

Chen & Ni 93 — *Dynamic Calibration and Compensation of a 3-D Laser Radar Scanning System.* Y. D. Chen and J. Ni in IEEE Transactions on Robotics and Automation, Vol. 9, No 3. June 1993.

Chomsky65 *Aspects of the Theory of Syntax.* Noam Chomsky. MIT Press, 1965.

Churchland95 *The Engine of Reason, The Seat of the Soul.* Paul M. Churchland. MIT Press. 1995.

Clark *et al* 92 *Learning Momentum: On-line Performance Enhancement for Reactive Systems.* Russell J. Clark, Ronald C. Arkin and Ashwin Ram. Proceedings of the IEEE International Conference on Robotics and Automation. May 1992.

Clark94 *Autonomous Agents and Real-Time Success: Some Foundational Issues.* Andy Clark in workshop proceedings: Dynamics and Representation in Adaptive Behaviour and Cognition. Dept. of Computer Science and Artificial Intelligence, University of San Sebastian, Spain. Editor Tim Smithers 1994.

Cliff & Miller 95 *Tracking the Red Queen: Measurements of Adaptive Progress in Co-Evolutionary Simulations.* Dave Cliff and Geoffrey Miller. Advances in Artificial Life: Proceedings of the Third European Conference on Artificial Life June 1995. Springer.

Cliff *et al* 93 *Exploration in Evolutionary Robots.* Dave Cliff, Phil Husbands and Inman Harvey. Adaptive Behaviour, Volume 2 number 1. Summer 1993. MIT Press Journals.

Cliff *et al* 94 *From Animals to Animats 3.* Edited by David Cliff, Philip Husbands, Jean-Arcady Meyer and Stewart W. Wilson Editors. Proceedings of the Third International Conference on Simulation of Adaptive Behaviour. MIT Press, Bradford Books 1994.

Colombetti et al 94 *Behaviour Analysis and Training: A Methodology for Behaviour Engineering.* M. Colombetti, M. Dorigo and G. Borghi IEEE Transactions on Systems, Man and Cybernetics Vol. 26, Issue 6 (1996).

Connell88 *A Behaviour Based Arm Controller.* Jonathan H. Connell. MIT AI-Memo 1025. June 1988.

Connell89 *A Colony Architecture for an Artificial Creature.* Jonathan H. Connell. MIT AI technical report 1151. June 1989.

Cooper *et al* 95 *Symbolic and Continuous Processes in the Automatic Selection of Actions.* Richard Cooper, Tim Shallice and Jonathan Farringdon. In Hybrid Problems, Hybrid Solutions. Editor John Hallam. Proceedings for the Tenth Biennial Conference on AI and Cognitive Science organised by the Society for the Study of Artificial Intelligence and Simulation of Behaviour, UK. 1995.

Cornell *et al* 94 *ANIT. A System for Perceptual Subsumption and Intelligent Vision System.* Stuart M. Cornell, John E. W. Mayhew and Sam Harrison. Research Paper, AI Vision Research Unit, University of Sheffield. UK.

Cox94 *The Fuzzy Systems Handbook.* Earl Cox. Accademic Press Ltd.,Oval Rd, London. NW1 7DX. 1994.

Critchlow85 *Introduction to Robotics.* Arthur J. Critchlow. Collier Macmillan Publishers.

Crowley *et al* 94 *Integration and Control of Reactive Visual Processes.* James L. Crowley, Jean Marc Bedrune (IMAG-LIFIA, 46 Ave Félix Viallet, 38031 Grenoble, France), Morten Bekker and Michael Schneider (Image Analysis Lab, Aalborg University, Fr. Bajers Vej 7, DK-9220 Aalborg, Denmark). In Proceedings European Conference of Computer Vision, Stockholm, May 1994.

Cruse91 — *Coordination of Leg Movements In Walking Animals.* Holk Cruse. From Animals to Animats 1. Proceedings of the first Conference on Simulation of Adaptive Behaviour 1991. MIT Press.

Cvijovic & Klinowski 95 — *Taboo Search: An Approach to the Multiple Minima Problem.* Djurdje Cvijovic and Jacek Klinowski. In Science Vol. 267. February 1995.

DeSchutter & Nuyts 93 — *Birds use self-organised social behaviours to regulate their dispersal over wide areas: evidences from gull roosts.* G. DeSchutter and E. Nuyts. In Proceedings of the Third European Conference of Artificial Life 1993.

Dickmanns & Muller 96 — *Scene Recognition and Navigation Capabilities for Lane Changes and Turns in Vision-Based Vehicle Guidance.* E. D. Dickmanns and N. Muller. In journal: Control Engineering Practice. Volume 4, Number 5. Pg. 589-599. 1996.

Digney & Gupta 94 — *A Distributed Adaptive Control System for a Quadruped Mobile Robot.* Bruce L. Digney and M. M. Gupta. In From Animals to Animats 3. Proceedings of the Third Conference on Simulation of Adaptive Behaviour 1994. MIT Press.

Donnet & Smithers 90 — *Lego Vehicle: A Technology for Studying Intelligent Systems.* Jim Donnet and Tim Smithers. Technical report , University of Edinburgh, Department of Artificial Intelligence. 1990.

Dorf92 — *Modern Control Systems*, 6th edition. Richard C. Dorf, Addison-Wesley 1992.

Downs *et al* 95 — *A Neural Network Decision-Support Tool for the Diagnosis of Breast Cancer.* Joseph Downs, Robert F. Harrison and Simon S. Cross. In Hybrid Problems, Hybrid Solutions. Editor John Hallam. Proceedings for the Tenth Biennial Conference on AI and Cognitive Science organised by the Society for the Study of Artificial Intelligence and Simulation of Behaviour, UK. 1995.

Dreyfus93 — *What computers still can't do: A critique of artificial reason.* H. L. Dreyfus. MIT Press 1993.

Eastman61 — *Go, Dog. Go!* P. D. Eastman. Collins and Harvill publishers, New York, New York. 1961.

Edelman *et al* 92 — *Synthetic Neural Modelling Applied to a Real-World Artefact.* Gerald M. Edelman, George N. Reeke, Jr., W. Einar Gall, Giulio Tononi, Douglas Williams and Olaf Sporns. Proceedings of the National Academy of Sciences USA. Vol. 89. August 1992.

Edelman87 — *Neural Darwinism.* Gerald M. Edelman. Basic Books. 1987.

El-Gamal *et al* 92 — *Reflex Control for an Intelligent Robotic System.* M. El-Gamal, A. Kara, K. Kawamura and M Fashoro. In Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robotics and Systems. Raleigh, NC. July 7-10 1992.

Ferrell93 — *Robust Agent Control of an Autonomous Robot with Many Sensors and Actuators.* Cynthia Ferrell, Masters Thesis, MIT Artificial Intelligence Laboratory 1993.

Ferrell94 — *Robust and Adaptive Locomotion of an Autonomous Hexapod.* Cynthia Ferrell. Proceedings: From Perception to Action Conference. September 1994. IEEE Computer Society Press. Editors P. Gaussier and J-D. Nicoud.

Fikes & Nilsson 71 — *STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving.* Richard E. Fikes and Nils J. Nilsson. Artificial Intelligence. 2. Pg. 189-208, 1971.

Fraichard & Laugier 93 — *Dynamic Trajectory Planning, Path-Velocity Decomposition and Adjacent Paths.* Th. Fraichard and C. Laugier. Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93). Vol. 2.

Franklin & Selfridge 90 — *Some New Directions for Adaptive Control Theory in Robotics*, in Neural Networks for Control. Eds. W. T. Miller, R. S. Sutton and P. J. Werbos. MIT press 1990.

Franklin & Selfridge 90 — *Some New Directions for Adaptive Control Theory in Robotics.* Judy A. Franklin and Oliver G. Selfridge. In Neural Networks for Control, W.T. Miller, R.S. Sutton and P.J. Werbos editors. MIT Press 1990.

Freund & Buxbaum 93 — *A New Approach to Multi Agent Systems Control In Robot-Based Flexible Manufacturing Work Cells.* E. Freund and H-J. Buxbaum. Proceedings: IEEE/RSJ International Conference on Intelligent Robot Systems. July 1993.

Fritzke91 — *Let it Grow - Self-Organising Feature Maps With Problem Dependent Cell Structure.* Bernd Fritzke. Proceedings of the International Conference of Artificial Neural Networks, 1991.

Gat *et al* 94 — *Behaviour Control for Robotic Exploration of Planetary Surfaces.* Erann Gat, Rajiv Desai, Robert Ivlev, John Loch, and David P. Miller. IEEE Transactions on Robotics and Automation. Vol. 10. No. 4, August 1994.

Gat93 — *On The Role of Stored Internal State in the Control of Autonomous Mobile Robots.* Erann Gat. AI Magazine. Vol. 14, No. 1, Spring 1993.

Gaussier & Zrehen 94 — *Complex Neural Architectures for Emerging Cognitive Abilities in an Autonomous System.* Philippe Gaussier and Stéphane Zrehen. Conference proceedings: From Perception to Action 94. IEEE Computer Society Press 1994.

Giszter94 — *Reinforcement Tuning of Action Synthesis and Selection in a "Virtual Frog".* Simon Giszter. From Animals to Animats 3. Proceedings of the third Conference on Simulation of Adaptive Behaviour 1994. MIT Press.

Godjevac94 — *A Comparative study of Fuzzy Control, Neural Network Control and Neuro-Fuzzy Control.* Jelena Godjevac. In: Fuzzy set theory and advanced mathematical applications, edited by Da Ruan, published by Kluwer Academic Publishers, January 1995.

Goldberg89 — *Genetic Algorithms in Search, Optimisation and Machine Learning.* David E. Goldberg. Addison-Wesley. 1989.

Grossberg69 — *Some networks that can learn, remember and reproduce any number of complicated space-time patterns.* Journal of Mathematics and Mechanics 19. 1969.

Gruau94 — *Automatic Definition of Modular Neural Networks.* Frédéric Gruau. Adaptive Behaviour, Vol. 3. No. 2, Fall 1994.

Habib *et al* 92 — *Simulation Environment for an Autonomous and Decentralised Multi-Agent Robotic System.* Maki K. Habib, Hajime Asama, Yoshiki Ishida, Akihiro Matsumoto and Isao Endo. Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems. Raleigh, NC. July 7-10 1992.

Hallam94 — *Hybrid Problems, Hybrid Solutions.* Editor John Hallam. Proceedings for the Tenth Biennial Conference on AI and Cognitive Science organised by the Society for the Study of Artificial Intelligence and Simulation of Behaviour, UK. 1995.

Harnad90 — *The Symbol Grounding Problem.* Stevan Harnad. Phyisica D, Volume 42, pages 335-346. Elsevier Science Publishers. B. V. North Holland. 1990.

Harris94 — *Advances in Intelligent Control.* Editor: C. J. Harris. Publishers: Taylor and Francis Ltd., 1994.

Harvey *et al* 94 — *Seeing the Light: Artificial Evolution, Real Vision.* Inman Harvey, Phil Husbands and Dave Cliff. From Animals to Animats 3. Proceedings of the Third Conference on Simulation of Adaptive Behaviour 1994. MIT Press.

Hayes-Roth88 — *A Blackboard Architecture For Control.* Barbara Hayes-Roth, Readings In Distributed Artificial Intelligence, 1988. Edited by Alan H. Bond and Les Gasser.

Hayes79 — *The Logic of Frames.* Patrick J. Hayes. Frame Conceptions and Text Understanding. Edited by D. Metzing. Walter de Gruyter and Co. 1979.

Hecht-Nielsen88 — *Applications of Counterpropagation Networks.* R. Hecht-Nielsen. Neural Networks 1. 1988.

Herman *et al* 90 — *Intelligent Control for Multiple Autonomous Undersea Vehicles.* Martin Herman, James S. Albus and Tsai-Hong Hong. In Neural Networks for Control, W.T. Miller, R.S. Sutton and P.J. Werbos editors. MIT Press 1990.

Holgate & Clarke 95 — *An Adaptive State Machine for Use in Unsupervised Parallel Learning Systems.* Christopher J. Holgate and Thomas J. W. Clarke. In Hybrid Problems, Hybrid Solutions. Editor John Hallam. Proceedings for the Tenth Biennial Conference on AI and Cognitive Science organised by the Society for the Study of Artificial Intelligence and Simulation of Behaviour, UK. 1995.

Horswill93 — *Specialization of Perceptual Processes.* Ian D. Horswill, PhD thesis, MIT Artificial Intelligence Laboratory, 1993.

Ibbett & Topham 89 — *Architecture of High Performance Computers Volume I and II.* R. N. Ibbett and N. P. Topham. Macmillan Computer Science Series. 1989.

Jaeger95 — *An Introduction To Dynamic Symbol Systems.* Herbert Jaeger. In Hybrid Problems, Hybrid Solutions. Editor John Hallam. Proceedings for the Tenth Biennial Conference on AI and Cognitive Science organised by the Society for the Study of Artificial Intelligence and Simulation of Behaviour, UK. 1995.

Jakobi *et al* 95 — *Noise and the Reality Gap: The Use of Simulation in Evolutionary Robotics.* Nick Jakobi, Phil Husbands and Inman Harvey. Advances in Artificial Life: Proceedings of the third European conference on Artificial Life, June 1995. Publishers: Springer.

Jennings & Wooldridge 95 — *Applying Agent Technology.* Nicholas R. Jennings and Michael Wooldridge. Applied Artificial Intelligence: An International Journal, Taylor & Francis London, 9 (4) 1995, 351-361.

Jennings 94 — *Cooperation in Industrial Multi-Agent Systems.* N. R. Jennings. Series in Computer Science, Vol 43, World Scientific Publishing, Singapore (ISBN 981-02-1652-1), 1994.

Jochem & Baluja 93	*A Massively Parallel Road Follower.* Todd M. Jochem and Shumeet Baluja. In Proceedings of: Computer Architectures for Machine Perception, 1993 (CAMP '93).
Jones & Flynn 93	*Mobile Robots, Inspiration to Implementation.* Joseph L. Jones and Anita M. Flynn. Publishers: A K Peters, Wellesley, Massachusetts. 1993.
Jones *et al* 91	*Distributed process control system survey.* Jones, Tinham, Reeve, Readman and Mawkin. Journal: Control and Instrumentation Vol. 23, Iss. 5, Pg. 30-71, Date: May 1991 Country of Publication: UK
Kaelbling93	*Learning in Embedded Systems.* Leslie Pack Kaelbling, Bradford Books MIT Press 1993.
Khorasani & Weng 94	*Structure Adaptation in Feed-Forward Neural Networks.* K. Khorasani and W. Weng. IEEE International Conference on Neural Networks Vol. III 1994.
Kim.S-H *et al* 93	*On Developing an Adaptive Neural-Fuzzy Control System.* Seong-Hyun Kim, Yong-Ho Kim, Kwee-Bo Sim and Hong-Tae Jeon. Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems. July 1993.
Kim.S-W *et al* 93	*A New Fuzzy Adaptive Controller Using a Robust Property of Fuzzy Controller.* Seung Woo Kim, Eun Tai Kim and Mignon Park. Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems. July 1993.
Kirsh91	*Today the Earwig, Tomorrow Man?* David Kirsh. Artificial Intelligence 47. Elsevier Science Publications B.V. North Holland. 1991.
Knick & Schlegel 94	*AMOS: Active Perception of an Autonomous System.* Manfred Knick and Christian Schlegel. Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems 1994.
Kohonen88	Self-Organisation and Associative Memory. T. Kohonen. Springer-Verlag 1988.
Koza91	*Evolution of Subsumption using Genetic Programming.* John R. Koza. In: Towards a Practice of Autonomous Systems, proceedings of the First European conference on Artificial Life. Francisco J. Varela and Paul Bourgine eds. MIT Press 1992.
Kweon *et al* 92	*Behaviour-Based Intelligent Robot in Dynamic Indoor Environments.* In So Kweon, Yoshinori Kuno, Mutsumi Watanabe and Kazunori Onoguchi. Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems. July 7-10 1992.
Langer *et al* 94	*A Behaviour-Based System for Off-Road Navigation.* D. Langer, J. K. Rosenblatt and M. Hebert. IEEE Transactions on Robotics and Automation. Vol. 10 No. 6. December 1994.
Long89	*A review of temporal logics*, D.P. Long in The Knowledge Engineering Review 4(2). 1989.
Luger & Stubblefield 89	*Artificial Intelligence and the Design of Expert Systems.* George F. Luger and William A. Stubblefield. Benjamin/Cummings Publishing Company, Inc. 1989.

Lund95 — *Specialization under Social Conditions in Shared Environments.* Henrik Hautop Lund. In: Advances in Artificial Life: Proceedings of the Third European Conference on Artificial Life June 1995. Springer.

Lyons93 — *Representing and Analysing Action Plans as Networks of Concurrent Processes.* Damian M. Lyons. IEEE Transactions on Robotics and Automation, Vol. 9, No 3. June 1993.

Ma *et al* 95 — *A Heuristic for General Rule Extraction from a Multilayer Perceptron.* Zhe Ma, Robert F. Harrison and R. Lee Kennedy. In Hybrid Problems, Hybrid Solutions. Editor John Hallam. Proceedings for the Tenth Biennial Conference on AI and Cognitive Science organised by the Society for the Study of Artificial Intelligence and Simulation of Behaviour, UK. 1995.

Maes & Brooks 90 — *Learning to Coordinate Behaviours.* Pattie Maes and Rodney A. Brooks. Proceedings AAAI 1990 conference.

Maes89 — *How To Do The Right Thing.* Pattie Maes, Connection Science, Vol. 1, No 3, 1989.

Maes89b — *The Dynamics of Action Selection.* Pattie Maes. In proceedings IJCAI-89. 1989.

Maes91 — *Adaptive Action Selection.* Pattie Maes. Technical report MIT AI-Lab and MIT Media-Lab. 1991. NE43-825.

Maes91b — *The Agent Network Architecture (ANA).* Pattie Maes. In proceedings: AAAI Spring Symposium on Integrated Intelligent Architectures. AAAI Press, 1991.

Maes91c — *A Bottom-Up Mechanism For Action Selection in an Artificial Creature.* Pattie Maes. In: Animals to Animats 1. Proceedings of the first Conference on Simulation of Adaptive Behaviour 1991. MIT Press.

Mahadevan & Connell 92 — *Automatic Programming of Behaviour Based Robots Using Reinforcement Learning.* Sridhar Mahadevan and Jonathan Connell, Artificial Intelligence 55, Pages 311-365. 1992.

Malcolm *et al* 89 — *Symbol Grounding via a Hybrid Architecture in an Autonomous Assembly System.* Chris Malcolm and Tim Smithers. In: New Architectures for Autonomous Agents, Edited by Pattie Maes. MIT Press and North Holland.

Malcolm *et al* 89b — *An Emerging Paradigm in Robot Architecture.* Chris Malcolm, Tim Smithers and John Hallam. University of Edinburgh Department of Artificial Intelligence research paper no. 447.

Malcolm95 — *The SOMAS System: A Hybrid Symbolic and Behaviour-based System to Plan and Execute Assemblies by Robot.* Chris Malcolm. In Hybrid Problems, Hybrid Solutions. Editor John Hallam. Proceedings for the Tenth Biennial Conference on AI and Cognitive Science organised by the Society for the Study of Artificial Intelligence and Simulation of Behaviour, UK. 1995.

Manela & Campbell 95 — *Designing Good Pursuit Problems as Testbeds for Distributed AI: A Novel Application of Genetic Algorithms.* Mauro Manela and J. A. Campbell, Department of Computer Science. University College London. 1995.

Marr82 — *Vision.* David Marr. W.H. Freeman and Co. 1982.

Mason93 — *Kicking The Sensing Habit.* Matthew T. Mason. AI Magazine Spring 1993.

Masters93 — *Practical Neural Network Recipes in C++.* Timothy Masters. Academic Press Inc. 1993.

Mataric91 — *Navigating With a Rat Brain: A Neurobiologically-Inspired Model for Robot Spatial Representation.* Maja J. Mataric. From Animals to Animats 1. Proceedings of the first Conference on Simulation of Adaptive Behaviour 1991. MIT Press.

Mataric92 — *Designing Emergent Behaviours: From Local Interactions to Collective Intelligence.* Maja J. Mataric. Proceedings: From Animals to Animats, second international Conference on Simulation of Adaptive Behaviour 1992. MIT Press.

Mataric92b — *Integration of Representation Into Goal-Driven Behaviour-Based Robots.* Maja J. Mataric. IEEE Transactions on Robotics and Automation. Vol. 8, No. 3. June 1992.

Mataric92c — *Behaviour-Based Systems: Key Properties and Implications.* Maja J. Mataric, IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems. 1992.

Mataric95 — *Interaction and Intelligent Behaviour.* Maja J. Mataric. PhD Thesis, MIT Artificial Intelligence Laboratory technical report 1495. 1995.

Maza & Yuret 94 — *Dynamic Hill Climbing.* Michael De La Maza and Deniz Yuret. AI Expert. March 1994.

McFarland 94b — *Animal Robotics - From Self-Sufficiency to Autonomy.* David McFarland. Proceedings: From Perception to Action Conference. September 1994. IEEE Computer Society Press. Editors P. Gaussier and J-D. Nicoud.

McFarland85 — *Animal Behaviour*, David McFarland. Publishers: Longman Scientific & Technical 1985.

McFarland94 — *Towards Robot Cooperation.* David McFarland. From Animals to Animats 3. Proceedings of the third Conference on Simulation of Adaptive Behaviour 1994. MIT Press.

McFarland95 — *The Biology of Behaviour - Criteria for Success in Animals and Robots.* David McFarland. The Biology and Technology of Intelligent Autonomous Agents. Editor: Luc Steels. Published by Springer, NATO ASI Series.

Mein91 — *Co-operative Behaviour In Uniformly and Differentially Programmed Lego Vehicles,* Richard Mein, MSc Thesis, Department Of Artificial Intelligence, University Of Edinburgh.

Meyer *et al* 92 — *From Animals to Animats 1.* Edited by Jean-Arcady Meyer, Herbert L. Roitblat and Stewart W. Wilson. Proceedings of the second Conference on Simulation of Adaptive Behaviour 1992. MIT Press.

Meyer *et al* 93 — *From Animals to Animats 2.* Jean-Arcady Meyer, Herbert L. Roitblat and Stewart W. Wilson Editors. Proceedings of the Second International Conference on Simulation of Adaptive Behaviour. MIT Press, Bradford Books 1993.

Miller.W *et al* 90 — *Neural Networks for Control.* Eds. W. T. Miller, R. S. Sutton and P. J. Werbos. MIT press 1990.

Miller93 — *A Twelve-Step Program to More Efficient Robotics.* David P. Miller. AI Magazine, Spring 1993.

Miller94 — *The Long Term Effects of Secondary Sensing.* David P. Miller. AI Magazine Vol. 15, No. 1. Spring 1994.

Minsky & Papert 69 — *Perceptrons.* M. L. Minsky and S. Papert. MIT Press, Cambridge MA. 1969.

Minsky61 — *Steps Toward Artificial Intelligence.* Marvin L. Minsky. Proceedings of the Institute of Radio Engineers 49. Reprinted in Computers and Thought, editors: E. A. Feigenbaum and J. Feldman. New York: McGraw-Hill.

Minsky81 — *A Framework for Representing Knowledge.* Marvin Minsky. In: Mind Design, Edited by J. Haugeland. MIT Press 1981.

Minsky85 — *The Society of Mind.* Marvin Minsky. Touchstone Books, Simon & Schuster Inc. publishers. 1985.

Mondada & Floreano 94 — *Active Perception, Navigation, Homing and Grasping: An Autonomous Perspective.* Dario Floreano and Francesco Mondada. In Proceedings: From Perception to Action Conference. September 1994. Pg. 170-180. IEEE Computer Society Press. Editors P. Gaussier and J-D. Nicoud.

Morán *et al* 95 — *Advances in Artificial Life.* F. Morán, A. Moreno, J.J. Merelo and P. Chacón editors. Proceedings of the Third European Conference on Artificial Life. Springer 1995.

Moravek88 — *Mind Children. The Future of Robot and Human Intelligence.* Hans Moravec. Harvard University Press. 1988.

Musliner *et al* 94 — *The Challenges of Real-Time AI.* D.J. Musliner, J. Hendler, A.K. Agrawala, E.H. Durfee, J.K. Strosnider and C.J. Paul. IEEE Computer, 28(1).

Nehmzow & McGonigle 94 — *Achieving Rapid Adaptations in Robots by Means of External Tuition.* Ulrich Nehmzow and Brenden McGonigle. From Animals to Animats 3. Proceedings of the third Conference on Simulation of Adaptive Behaviour 1994. MIT Press.

Nehmzow & Smithers 90 — *Map Building using Self-Organising Networks in "Really Useful Robots".* Ulrich Nehmzow and Tim Smithers. University of Edinburgh Department of Artificial Intelligence Research Paper No. 489.

Nehmzow *et al* 89 — *Really Useful Robots.* Ulrich Nehmzow, J. Hallam and Tim Smithers. From proceedings: Intelligent Autonomous Systems 2, Editors: T. Kanade, F.C.A. Groen and L.O. Hertzberger. 1989.

Newell & Simon 76 — *Computer Science as Empirical Enquiry: Symbols and Search.* Allen Newell and Herbert A. Simon. Communications of the Association for Computing Machinery, 19. March 1976.

Newell81 — *The Knowledge Level.* Allan Newell. Journal of Artificial Intelligence, 18 (1). 1981.

Nilsson84 — *Shakey the Robot.* Nils J. Nilsson (ed.), SRI AI centre technical note 323, April 1984.

Nolfi *et al* 94 — *Phenotypic Plasticity in Evolving Neural Networks.* Stefano Nolfi, Orazio Miglino and Domenico Parisi. Proceedings: From Perception to Action Conference. September 1994. Pg. 146-157. IEEE Computer Society Press. Editors P. Gaussier and J-D. Nicoud.

Nourbakhsh *et al* 95 — *Dervish, An Office-Navigating Robot.* Illah Nourbakhsh, Rob Powers and Stan Birchfield. AI Magazine, Vol. 6, No. 2. Summer 1995.

Oliveira *et al* 91 — *A Multi-Agent Environment in Robotics.* Eugénio Oliveira, R. Camacho and C. Ramos. Robotica, Vol. 9, pages 431-440. 1991.

Osorio *et al* 94 — *Insect Vision and Olfaction: Different Neural Architectures for Different Kinds of Sensory Signal?* D. Osorio, Wayne M. Getz and Jürgen Rybak. From Animals to Animats 3. Proceedings of the third Conference on Simulation of Adaptive Behaviour 1994. MIT Press.

Parker92 — *Adaptive Action Selection for Cooperative Agent Teams.* Lynne E. Parker. From Animals to Animats, second international Conference on Simulation of Adaptive Behaviour 1992. MIT Press.

Parker93 — *Designing Control Laws for Cooperative Agent Teams.* Lynne E. Parker. Proceedings of the 1993 IEEE Robotics and Automation Conference. May 1993.

Pebody91 — *How To Make A Lego Vehicle Do The Right Thing*, Miles Pebody, MSc Thesis, Department Of Artificial Intelligence, University Of Edinburgh.

Pebody94 — *Robots Have Only One Frame of Reference: The Real World.* Miles Pebody. In "Models or Behaviours - Which Way Forward for Robotics?" Proceedings of the Society for the Study of Artificial Intelligence and Simulation of Behaviour (AISB) workshop, edited by Ruth Aylett, Leeds, April 1994.

Pebody94b — *Acting to Sense: The lowest levels of a Subsumption Architecture.* Miles Pebody. Conference proceedings: From Perception to Action 94. IEEE Computer Society Press 1994. Editors P. Gaussier and J-D. Nicoud.

Pfeifer & Verschure 91 — *Distributed Adaptive Control: A Paradigm for Designing Autonomous Agents.* Paul F.M.J. Verschure and Rolf Pfeifer. In: Towards A Practice of Autonomous Systems, Proceedings of the First European Conference on Artificial Life. Francisco J. Varela and Paul Bourgine eds. 1992.

Pfeifer & Verschure 93 — *Categorisation, Representations and the Dynamics of System-Environment Interaction: A Case Study in Autonomous Systems.* Paul F.M.J. Verschure and Rolf Pfeifer. Proceeding of the Second International Conference on the Simulation of Adaptive Behaviour, Bradford books MIT Press 1993.

Pfeifer88 — *Artificial Intelligence Models of Emotion.* Rolf Pfeifer. Cognitive Perspectives on Emotion and Motivation. Editor V. Hamilton. Kluwer Academic Publishers 1988.

Pfeifer93 — *Studying Emotions: Fungus Eaters.* Rolf Pfeifer. Advances in Artificial Life: Proceedings of the Third European Conference on Artificial Life June 1995. Springer.

Pfeifer95 — *Cognition - Perspectives from Autonomous Agents.* Rolf Pfeifer in The Biology and Technology of Intelligent Autonomous Agents. Editor: Luc Steels. Published by Springer, NATO ASI Series.

Pfeifer96 — *Building "Fungus Eaters": Design Principles of Autonomous Agents.* Rolf Pfeifer in: From Animals to Animats 4. Proceedings of the Fourth Conference on Simulation of Adaptive Behaviour 1996. P. Maes *et al* editors. MIT Press.

Pierce & Kuipers 91 — *Learning Hill-Climbing Functions as a Strategy for Generating Behaviours in a Mobile Robot.* David Pierce and Benjamin Kuipers. From Animals to Animats 1. Proceedings of the first Conference on Simulation of Adaptive Behaviour 1991. MIT Press.

Pinhanez95 — *Evaluating an Active Camera Controlled by a Subsumption Architecture.* Claudio S. Pinhanez. The Biology and Technology of Intelligent Autonomous Agents. Editor: Luc Steels. Published by Springer, NATO ASI Series.

Porter92 — *A Subsumption Architecture Development Environment.* Ian Porter, MSc Thesis, Department Of Artificial Intelligence, University of Edinburgh.

Pratt87 — *Thinking Machines: The Evolution of AI.* Vernon Pratt. Pubs. Oxford, B. Blackwell. 1987.

Prem95 — *Grounding and the Entailment Structure in Robots and Artificial Life.* Erich Prem. In: Advances in Artificial Life: Proceedings of the third European conference on Artificial Life June 1995. Publishers: Springer.

Ram *et al* 94 — *Using Genetic Algorithms to Learn Reactive Control Parameters for Autonomous Robotic Navigation.* Ashwin Ram, Ronald Arkin, Gary Boone and Michael Pearce. Adaptive Behaviour Vol. 2. No. 3. Winter 1994.

Riecken *et al* 94 — *Intelligent Agents.* Communications of the ACM July 1994. Vol. 37 no. 7.

Robinson90 — *A Computational View of the Oculomotor System.* David A. Robinson, Computational Neuroscience, Editor Eric L. Schwartz, MIT Press 1990.

Rodd & Deravi 89 — *Communication Systems for Industrial Automation.* Michael G. Rodd and Farzin Deravi. Prentice Hall (UK) Ltd. publishers 1989.

Rodd & Verbruggen 92 — *Expert Systems in Advanced Control - Myths, Legends and Realities.* M. G. Rodd and H. B. Verbruggen. Tutorial Paper, Institute for Industrial Information and Technology, Department of Electrical and Electronic Engineering, University of Wales, Swansea, UK.

Rokey & Grenander 90 — *Planning for Space Telerobotics: The Remote Mission Specialist.* Mark Rokey and Sven Grenander. IEEE Expert, June 1990.

Rosen87 — *Real Brains, Artificial Minds.* Bob Rosen. Edited by John L. Casti and Anders Karlqvist. New York; London : North-Holland. 1987. ISBN 0444011552.

Rosheim94 — *Robot Evolution. The Development of Anthrobotics.* Mark E. Rosheim. Published by John Wiley and Sons, Inc. 1994.

Rumelhart & McClelland 86 — *Parallel Distributed Processing, Vol. 1.* David Rumelhart, James McClelland and the PDP Research Group. MIT Press 1986.

Rutkowska94 — *Emergent Functionality in Human Infants.* Julie C. Rutkowska. From Animals to Animats 3. Proceedings of the third Conference on Simulation of Adaptive Behaviour 1994. MIT Press.

Ruyssinck92 — *Lola 2 Technical Reference Manual.* Piet Ruyssinck. Autonomous Agents Group, AI Laboratory, Vrije Universitiet Brussel.

Salomé & Bersini 94 — *An Algorithm for Self-Structuring Neural Net Classifiers.* Tristan Salomé and Hugues Bersini. IEEE International Conference on Neural Networks Vol. III 1994.

Schell & Dickmanns 94 — *Autonomous Landing of Airplanes by Dynamic Machine Vision.* F. R. Schell and E. D. Dickmanns. In journal: Machine Vision and Applications. Volume 7, Number 3. Pg. 127-134. 1994.

Schwarz93 — *The Chambers Dictionary.* Edited by Catherine Schwarz. Chambers Harrap Publishers Ltd., 1993.

Simmons92 — *Concurrent Planning and Execution for Autonomous Robots.* Reid Simmons. IEEE Control Systems. February 1992, Pages 46-50.

Simmons94 *Structured Control for Autonomous Robots.* Reid G. Simmons. IEEE Transactions on Robotics and Automation. Vol. 10 No. 1. February 1994.

Smithers91 *Lecture Notes in Knowledge Representation and Inference II.* University of Edinburgh Department of Artificial Intelligence 1991.

Smithers92 *Taking Eliminative Materialism Seriously: A Methodology for Autonomous Systems Research.* Tim Smithers. In: Towards A Practice of Autonomous Systems, proceedings of the first European conference on Artificial Life. Francisco J. Varela and Paul Bourgine eds. 1992.

Smithers94 *Why Better Robots Make it Harder.* Tim Smithers. From Animals to Animats 3. Proceedings of the third Conference on Simulation of Adaptive Behaviour 1994. MIT Press.

Smithers94b *What the Dynamics of Adaptive Behaviour and Cognition Might Look Like in Agent-Environment Interaction Systems.* Tim Smithers. Workshop proceedings: Dynamics and Representation in Adaptive Behaviour and Cognition. Dept. of Computer Science and Artificial Intelligence, University of San Sebastian, Spain. Editor Tim Smithers 1994.

Smithers94c *Dynamics and Representation in Adaptive Behaviour and Cognition.* Editor Tim Smithers. Workshop proceedings. Dept. of Computer Science and Artificial Intelligence University of San Sebastian, 1994.

Smithers95 *On Quantitative Performance Measures of Robot Behaviour.* Tim Smithers. The Biology and Technology of Intelligent Autonomous Agents. Editor: Luc Steels. Published by Springer, NATO ASI Series.

Steels89 *Cooperation Between Distributed Agents Through Self-Organisation.* Luc Steels. AI Memo No. 89-5. Artificial Intelligence Laboratory, Vrije Universiteit Brussel, June 1989.

Steels90 *Exploiting Analogical Representations.* Luc Steels. Robotics and Autonomous Systems 6. Elsevier Science Publishers B.V. North Holland. 1990.

Steels92 *The PDL Reference Manual.* Luc Steels. Vrije Universiteit Brussel AI Memo 92-5. March 1992.

Steels93 *Building Agents out of Autonomous Behaviour Systems.* Luc Steels in The 'Artificial Life Route To Artificial Intelligence' Building Situated Embodied Agents. L Steels and R Brooks (eds.) 1993. New Haven: Lawrence Erlbaum Associates.

Steels94 *A Case Study in the Behaviour-Oriented Design of Autonomous Agents.* Luc Steels. From Animals to Animats 3. Proceedings of the Third Conference on Simulation of Adaptive Behaviour 1994. MIT Press.

Steels95 *Intelligence - Dynamics and Representations.* Luc Steels. The Biology and Technology of Intelligent Autonomous Agents. Editor: Luc Steels. Published by Springer, NATO ASI Series.

Sutton91 *Reinforcement Learning Architectures for Animats.* Richard S. Sutton. From Animals to Animats 1. Proceedings of the first Conference on Simulation of Adaptive Behaviour 1991. MIT Press.

Tigli *et al* — *A Reactive Multi-Agents System As Mobile Robot Controller.* J.Y. Tigli, M Occello and M.C. Thomas. I3S Laboratory, Université de Nice, Sophia Antipolis, 250 Rue Albert Einstein, Sophia Antipolis, 06560, Valbonne. France.

Tsypkin71 — *Adaptation and Learning in Automatic Systems.* Ya Z. Tsypkin. Academic Press. 1971.

Turing50 — *Computing machinery and intelligence.* Alan M. Turing, A.M. Mind 59:433-460. 1950.

Tyrrell94 — *An Evaluation of Maes' Bottom-Up Mechanism for Behaviour Selection.* Toby Tyrrell. Adaptive Behaviour Vol. 2, No. 4, Pg. 307-348. Spring 1994. MIT Press Journals.

vanGelder94 — *What Might Cognition Be, If Not Computation?* Tim van Gelder. Workshop proceedings: Dynamics and Representation in Adaptive Behaviour and Cognition. Dept. of Computer Science and Artificial Intelligence, University of San Sebastian, Spain. Editor Tim Smithers 1994.

Varela *et al* 91 — *Towards a Practice of Autonomous Systems.* Francisco J. Varela and Paul Bourgine editors. Proceedings of the First European Conference on Artificial Life. MIT Press 1991.

Vereertbrugghen93 — *Design and Implementation of a Sensor-Motor Control Unit for Mobile Robots.* Dany Vereertbrugghen. Licentie Thesis, AI Laboratory, Vrije Universiteit Brussel.

Vertommen95 — *Reporting Experiments on Integration of Learning Algorithms and Reactive Behaviour-Oriented Control Systems on a Real Mobile Robot.* Fillip Vertommen. The Biology and Technology of Intelligent Autonomous Agents. Editor: Luc Steels. Published by Springer, NATO ASI Series

Wasserman89 — *Neural Computing: Theory and Practice.* Philip D. Wasserman. Van Nostrand Reinhold. 1989.

Watkins89 — *Learning from Delayed Rewards.* C.J.C.H. Watkins. PhD. thesis, Kings College, Cambridge.

Webb & Smithers 91 — *The Connection Between AI and Biology in the Study of Behaviour.* Barbara Webb and Tim Smithers. In: Towards A Practice of Autonomous Systems, proceedings of the first European conference on Artificial Life. Francisco J. Varela and Paul Bourgine eds. 1992.

Webb90 — *Computer Simulations in the Study of Intelligent Behaviour.* Barbara Webb. University of Edinburgh Department of Artificial Intelligence Discussion Paper No. 101.

Webb93 — *Modelling Biological Behaviour or "Dumb Animals and Stupid Robots".* Barbara Webb. In Proceedings for the third European Conference of Artificial Life 1993.

Webb95 — *Robotic Experiments in Cricket Phonotaxis.* Barbara Webb. From Animals to Animats 3. Proceedings of the third Conference on Simulation of Adaptive Behaviour 1994. MIT Press.

Webber & Bisset94 — *Competition and Cooperation - A Model for Behaviour-Based Robot Controllers.* A. D. Webber and D. L. Bisset. Proceedings: From Perception to Action Conference. September 1994. IEEE Computer Society Press. Editors P. Gaussier and J-D. Nicoud.

Weiß95	*Distributed Reinforcement Learning.* Gerhard Weiß. In: The Biology and Technology of Intelligent Autonomous Agents. Editor: Luc Steels. Published by Springer, NATO ASI Series.
Wiener48	*Cybernetics.* Norbert Wiener. Publishers: John Wiley and Sons 1948. (second edition MIT Press 1961)
Winston77	*Artificial Intelligence.* Patrick H. Winston. Addison-Wesley, Reading Massachusetts. 1977.
Yamaguchi *et al* 94	*Self-Organising Control Using Fuzzy Neural Networks.* T. Yamaguchi, T. Takagai and T. Mita. In *Advances in Intelligent Control.* C. J. Harris Ed. Pubs. Taylor and Francis Ltd., 1994.
Yamauchi and Beer 94	*Sequential Behaviour and Learning in Evolved Dynamical Neural Networks.* Brian M. Yamauchi and Randall D. Beer. Adaptive Behaviour journal, Volume 2 number 3. Winter 1994. MIT Press Journals.
Yuta & Premvuti 92	*Coordinating Autonomous and Centralised Decision Making to Achieve Cooperative Behaviours Between Multiple Mobile Robots.* Shin'ichi Yuta and Suparerk Premvuti. Proceedings: IEEE/RSJ International Conference on Intelligent Robot Systems. July 1992.
Zeki93	*A Vision of the Brain.* Semir Zeki. Blackwell Scientific Publications. 1993.
Zorpette94	*Autopilots of the Deep.* Glenn Zorpette. Institute of Electrical and Electronic Engineers Spectrum journal. August 1994.

A. A Transputer Environment for Building Subsumption Control Architectures

A.1. The Subsumption Architecture

The Subsumption Architecture was amongst the first frameworks to emerge from the AI community that advocated the bottom-up design methodology for intelligent robotics. It has since been continuously under development and a programming language called the Behaviour Language has been designed to support it at a higher level. This work was started by Rodney Brooks of the Massachusetts Institute Of Technology and has lead the way in the up and coming field of Behaviour Based Artificial Intelligence.

The Subsumption Architecture provides a mechanism which enables the real-time control mechanisms of autonomous mobile robots to be built up in layers starting with very basic, reactive response and reflexes. New layers of the system can be added without the need to modify already operational lower ones. One of the main characteristics of the system, one that typifies that of all BBAI research, is the nature of the interaction of the layers with sensors and actuators and thus the real-world. Rather than using a functional approach which leads to a sequential pipe-line mechanism of: sensors→sensor processing→map building→action planning →action realisation→actuators (or some such), the Subsumption Architecture results in a massively parallel mechanism with modules which have access to virtually any sensory or internal information. The exact interface for each module is specified and built into the system as part of the design process (although work has been reported that examines mechanisms for learning these interconnections dynamically [Maes & Brooks 90]). The resulting redundancy enhances the systems performance by making it more robust and more able to respond quickly (as a reflex) to critical situations.

The architecture focuses around a number of interconnected finite state machine processes that have been augmented with real-time clock information, hence: Augmented Finite State Machines: AFSM. The AFSMs are interconnected via single element buffers and "wires". Information is not stored in the buffer, it simply contains the most recently arrived message, it is up to the AFSM to respond to this in some suitable way as messages can be lost if a new one arrives before the old has been used.

Each AFSM uses its real-time clock information to maintain a regular processing cycle. The finite state machine inputs are sampled and processed to provide the next output state once every clock period. This period is constant for all AFSMs in the network, being fixed at compile time.

Thus the AFSMs are loosely synchronised in that they all process at the same rate. However, the exact stage of an AFSMs process can not be determined since there is no signal between the machines to provide this information. It is in this respect that the network is asynchronous. Therefore an AFSM reads its input state and calculates its output which is then transmitted. Because of the asynchronous aspect of the AFSMs the longer an output remains active the more likely it is to have an effect. However it may also be that the response of an AFSM must change quickly in order to reflect the state of its inputs. These are some of the aspects that must be considered during the design of Subsumption networks.

The inputs and outputs of a particular AFSM may be inhibited, suppressed or given a default value by a wire output of another AFSM. This is the only way in which wire may interconnect. Figure A.1 presents a single AFSM.

Inhibition is applied to a module's output and prevents the module's output from reaching its destination.

Suppression effects the input to an AFSM. Its effect is similar to that of an inhibition but additionally the suppressing AFSM forces a new value onto the wire.

Default provides a value for a wire that is not currently active. It is in effect an inverse of suppression.

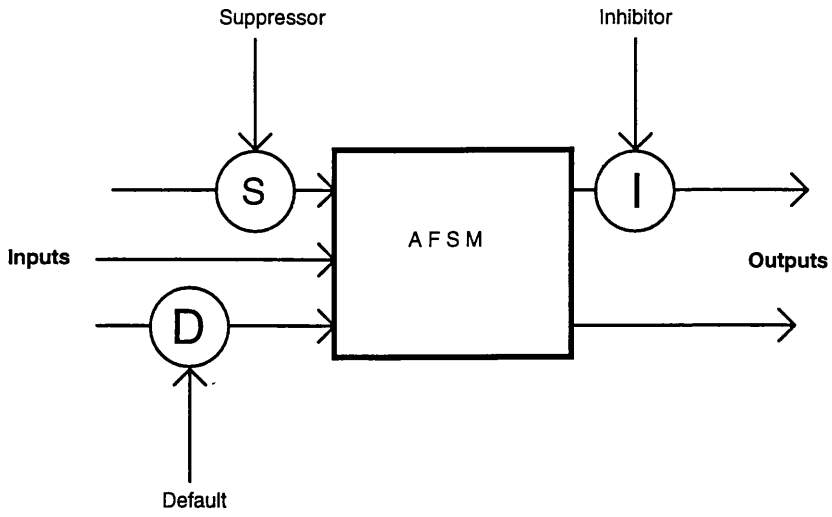


Figure A.1. An augmented finite state machine and its interconnections.

A single layer of a system consists of several AFSMs and as higher layers are added the lower layers become totally embedded in the system. The higher layers influence the lower layers by inhibition and suppression and also by their effect on the actuators and hence the vehicle's situation in the world. More detail on programming agents using the Subsumption Architecture

and details of robots that have been built can be found in (amongst others) [Brooks86b], [Brooks90] and [Brooks91].

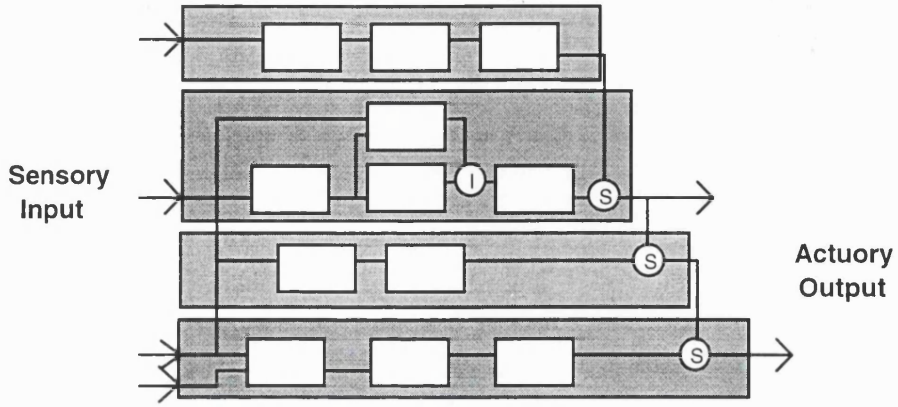


Figure A.2. An example Subsumption network showing the layered development of such systems with each layer being built up out of multiple AFSMs.

A.2. Subsumption on Transputers

This section begins by briefly describing the transputer microprocessor. For further details the reader is guided to one of the many transputer programming text books (for example [Burns88]). After this introduction the section continues with the implementation of the subsumption architecture onto a transputer based system.

A.2.1. The Transputer

Transputers are high performance, reduced instruction set (RISC) microprocessors designed to be easily incorporated into parallel networks of multiple processors. Each transputer can run time-sliced processes the scheduling of which is implemented in the processors hardware rather than in some higher level kernel or operating system. A single transputer has a standard type address space for local memory and other local input/output ports. In addition to this four bi-directional high speed (up to 20Mbits/s) links for connections to other transputers are provided with on chip control hardware for each link. Figure A.3 shows a simplified layout of both a single transputer and a simple array of nine.

Software on a transputer can communicate with that on a neighbouring transputer via a channel protocol that is implemented over the hardware links. This channel protocol is used to communicate between all processes in a transputer system whether they are located on the same processor or not. This means that at the process level the hardware division of transputers is not significant, processes communicate in the same way whether they are located on the same processor or on a neighbouring processor. Standard routing software is available that provides a network with message routing between non-neighbouring transputers. This message passing

between processes is a fundamental characteristic of the transputer. Whilst global data may exist between processes running on a single transputer there is no means of providing a complete network with global data. In fact the use of global data is generally not encouraged.

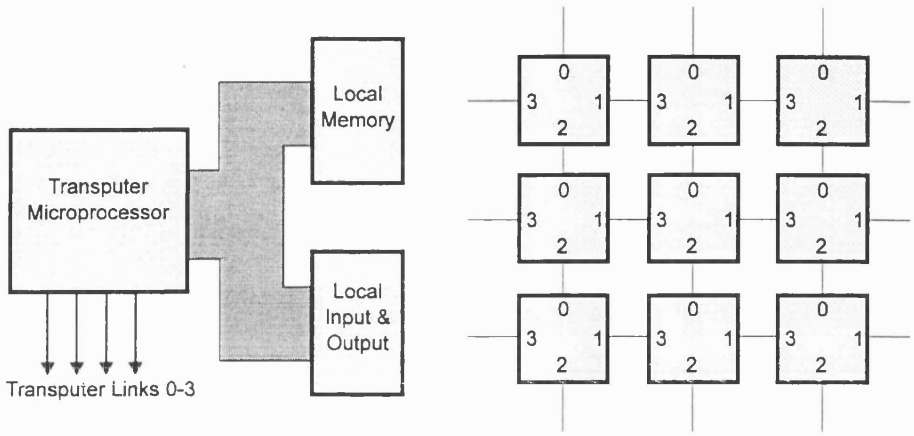


Figure A.3. Showing details of a single Transputer (left) and an array of interconnected Transputers (not showing local address space)

Interconnected by a grid of physical links large arrays of transputers can be built up relatively easily and in many configurations (the grid shown above in figure A.3 is a simple example). It is also relatively straight forward to add new transputers to an existing system in order to increase the available processing resources. Multiple transputers can be used in close proximity or they can be distributed around a larger system using the high-speed serial links for communication (for a more detailed discussion see [Ibbett & Topham 89]).

A.2.2. Augmented Finite State Machines and Wires

It would appear that the layered AFSM approach of the Subsumption Architecture will adapt well to the distributed processing architecture of the transputer. In the same way that each AFSM is an isolated processing machine only connected by its input and output registers, each transputer process only shares data with other processes via channels. The communication between discrete transputers and multiple processes running on them would thus appear to be a useful characteristic when implementing the wires of a Subsumption network. There is no marked difference between hardware processors and software processes and this allows the distributed machines of the Subsumption Architecture to map well onto a transputer network. Another aspect that supports the conjected suitability of the Subsumption Architecture is the fact that the characteristics of enhancing a system by adding further behavioural layers is similar in principal to adding extra transputer based hardware in order to increase a systems processing resources.

Transputer processes are inherently synchronous and this posed a problem with an architecture along the lines of that implemented by Brooks in which all AFSMs are fundamentally

asynchronous. Another problem was how to implement the functionality of the wires and the different connections which in the Subsumption architecture which takes place as a property of the wires themselves. Transputer links have no function except to transfer data. It turned out that both of these aspects could be solved by implementing each AFSM as a pair of transputer processes: An input process and an AFSM process. These appear in figure A.4.

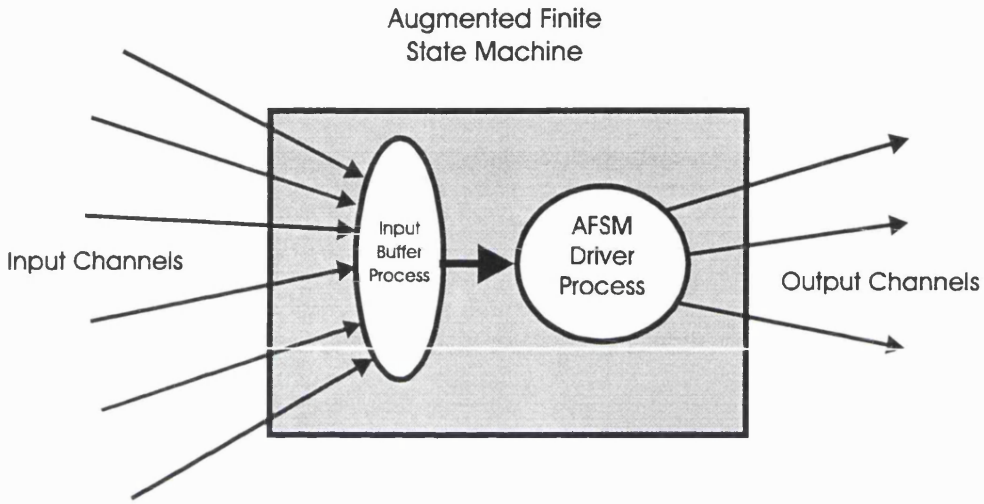


Figure A.4. Transputer processes making up a Subsumption augmented finite state machine.

The diagram above shows the two AFSM transputer processes and the transputer channels that connect them. The subsumption wires and their connection node function is implemented by the input buffer process. Hence each wire appears simply as a transputer channel. The subsumption connection node is not obviously distinguishable by viewing the transputer network channels alone. The following describes each process in more detail.

Input buffer Process

The input buffer process has two functions. The first is to provide the desynchronisation between the transputer AFSM processes and the second is to implement the functionality of the wire connections nodes (Inhibit suppression and default). Figure A.5 shows how a number of AFSM outputs each suppress those of lower AFSMs and eventually effect an output AFSM. This is implemented in transputer processes as four separate channels all connected to the input buffer of the final destination AFSM process pair. It is this input buffer that performs the arbitration between the various suppressor nodes.

Channels into the input buffer are grouped into blocks of wires as shown in the example in figure A.5. The feature that brings these wires together is that they all have a common destination, they are all connected either directly or by wire nodes to the destination AFSM D. The wires of each block are grouped and processed together because the value at the destination

input buffer depends in the activity of all these wires and this must be taken into account when AFSM registers are updated. A simple order of priority is used for the suppress and default nodes with the highest (in the case of suppress) active AFSM taking priority over lower ones. This priority is set when the system is built and the priorities are sorted taking into account the characteristics of the wire block. For example default nodes are effectively an inverse of suppress resulting in a reduced priority for the wire concerned rather than increased.

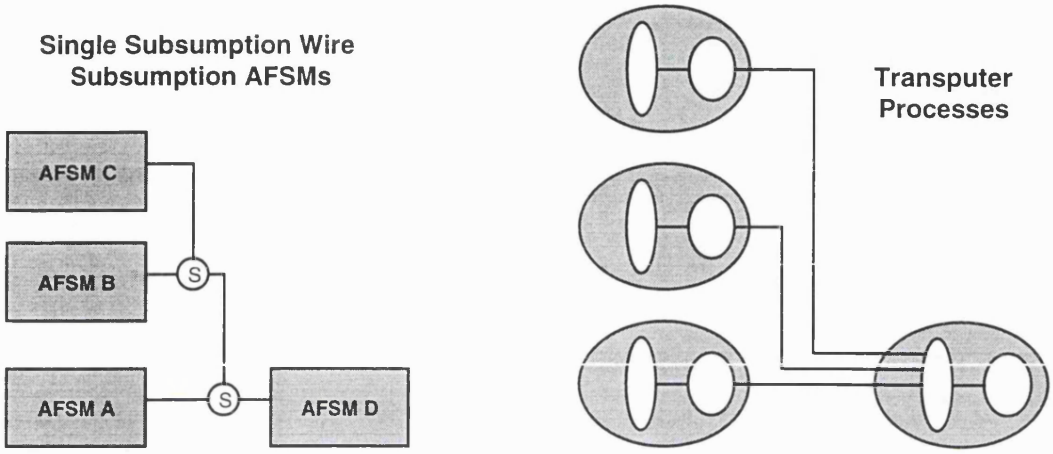


Figure A.5. A set of wires with suppression nodes left as found in a Subsumption network and right as implemented in Transputer processes.

Within the input buffer process is a main loop that continuously polls the input channels wire block by wire block testing to see if any have data and servicing them if they do. A channel must be read in order that the sending process can continue operating. It is this that desynchronises the system and prevents deadlock between processes. At the end of each iteration, the internal channel that connects to the AFSM driver process is tested to see if the AFSM driver is requesting an update of its internal registers. Finally before repeating the loop the process is put on hold in order to give other processes CPU time. This is presented in the pseudo-code listing below.

```
Do forever:
        Test each input wire set for active channels
                if active then read in data
                if not suppressed
                        update register
        Test AFSM driver update request
if active send all register data and status to AFSM driver
        Reschedule the process
Loop End
```

Input buffer process pseudo-code.

AFSM Process

This process performs the actual function of the AFSM as designed by the programmer. A loop repeatedly executes the relevant program instructions with the characteristic time period that is determined at compile time. This ensures that all the AFSMs in the network are loosely

synchronised. The pseudo-code bellow outlines the operations. During the execution of the AFSM process the input registers are updated automatically. However, outputs must be explicitly commanded by the AFSM function that is written by the designer.

```
Do forever:
        Store real-time clock reading for use in characteristic time calculation
        Complete handshake with input buffer process to update registers
        Execute AFSM functionality
        Test time and suspend process until system characteristic time is up
Loop end
```

AFSM process pseudo-code.

A.3. A Behaviour Language

This section outlines the method of specifying and programming a network of augmented finite state machines to run on a transputer system and hence build a subsumption architecture control structure. The syntax is loosely founded on that of the behaviour language developed by Rodney Brooks and others at the Massachusetts Institute of Technology, however rather than being based on the LISP programming language this version is built around C. This is because the available transputer programming tool-set uses a parallel dialect of the C language. The implementation bellow provides a subset of the behaviour language, as presented in [Brooks90b], which can then be translated into standard C, compiled linked and configured for execution on a transputer system.

The body-forms of the AFSMs are specified as a number of "*whenever*" rules or simply as a block of C program code. If C code is used it must be ensured that there are no loops included that may take longer than the systems characteristic time to execute. If this does happen the synchronisation of the network will be disrupted. Instead, iteration should be built in to the function of the AFSM which is also more in keeping with the philosophy of the subsumption architecture. AFSMs are generated from level zero (i.e. non-nested) *whenever* statements and *defmachine* statements only. As in the behaviour language the interconnecting subsumption wires are defined using the *connect* statement. Details follow.

Single Rules:

Each AFSM is implemented as an individual process that runs concurrently with all other AFSMs in the system. The most simple way of creating one of these in the Behaviour Language is to specify a *whenever* construct of the form:

```
whenever( condition){
        body-form;
}endwhenever;
```

This creates a process that continuously loops once each period of the systems characteristic time, monitoring the specified condition and when found to be true executes the body form. In the

Transputer 'C' version the body form can either take the form of further nested *whenever* statements or a number of 'C' statements (which should not include any loops).

Nested Rules:

Whenever statements may be nested but lower levels are not turned into separate processes, they are built into the process of the first, top level whenever rule term. Each whenever rule is executed as a continuous loop that ensures the correct functionality and dynamics of the AFSM (see section A.2 above). In normal cases once entered the whenever will execute infinitely and so in the case of the example below, once condition_1a is found to be true the original condition_1 will no longer be tested. This can be remedied in two ways described bellow.

```
whenever( condition_1){
        whenever( condition_1a){
                body-form_1a;
        }endwhenever;
}endwhenever
```

Exclusive Rules:

The exclusive form causes all *whenever* forms on the same nest level to be actively tested each cycle of the system's characteristic time. This allows more than one *whenever* condition to be monitored and different actions taken depending on the result:

```
exclusive{
        whenever( condition_1){
                body-form_1;
        }endwhenever;
        whenever( condition_2){
                body-form_2;
        }endwhenever;
}endexclusive;
```

This effectively creates a loop that tests both conditions and then executes the body-form of the first condition to become true. Exclusive *whenevers* may also be nested and they may also be combined with normal *whenever* forms.

Terminating A Whenever Execution:

The continuous loop executed by a *whenever* form may be terminated with the use of a *donewhenever()* statement:

```
whenever( condition_1){
        whenever( condition_1a){
                whenever( condition_1aa){
                        body-form_1aa;
                        donewhenever( level);
                }endwhenever;
        }endwhenever;
}endwhenever;
```

The *donewhenever* causes the program to exit the current level of *whenever* form and to return to testing the condition of the specified level. The argument *level* can be used to specify the number of levels to jump out. The default is to simply exit the current *whenever* level.

Machines That Can Be Interlinked:

The *whenever* form on its own is of little use in building subsumption networks as it does not provide for the declaration of AFSM registers which allow the interconnection of wires. The *defmachine* construct allows this by having an AFSM name declaration field and a register declaration field.

The machine name is used in conjunction with register names in the generation of Transputer channel implementations of the Subsumption wires. The *decls*: field contains the register declarations which are standard 'C' variable type definitions with an optional initialisation value for each (the default initialisation value is zero). The *rule*: field contains a *body-form* as outlined above with *whenever* and *exclusive* forms and standard C code. These make use of the declared registers.

```
defmachine( name){
        decls: type1 name1 [ = initial vaiue1], typeN nameN [ = initial valueN];
        rule: body-form;
}endmachine;
```

Connecting The AFSMs:

Wires are generated from *connect()* statements. These may be located anywhere in the program code so long as they appear at the beginning of their own line. The *connect* statement is of the form:

```
connect( source, destination [, destination2, destinationN] );
```

Where: source and destination are of the form: **AFSM-name.Register-name**

As indicated a single source register may be connected to an unlimited (within reason) number of destination registers however the reverse is not true, only one register can input to a destination unless Inhibit, Suppression or Default connections are used. In which case the connect statement take on the form:

```
connect( source inhibit( destination));
connect( source suppress( destination));
connect( source default( destination));
```

Sending Out Register Contents:

The *output()* statement uses the parallel 'C' *ChanOut()* function to transmit the contents of a specified register over its wire-channel. The form is:

```
output( register_name);
```

Register Status:

The *received()* and *ifreceived()* frunctions return an integer value of 1 (TRUE) if the specified register has been updated since the last time it was read.

```
received( register_name);

ifreceived( register_name){
```

```
        body-form;
    }
```

A.3.1 Multi-Transputer Programs

The Inmos 'C' development tool-set requires that at least one "main()" function be complied and linked for each target transputer. This is easily achieved by writing separate behaviour programs and translating them into 'C' separately. The only new behaviour statement that is necessary is *external(reg.afsm)*, which is used in place of the source or destination part of a *connect* statement.

For wires originating locally use:

```
connect( source, external( destination name), destination2... destinationN);
connect( source,  external(suppress( destination name)), destination2, ....
destinationN);
```

Wires that originate from another transputers must be declared as *external* inputs:

```
connect( external( source name), destination1,...,destinationN);
```

The configuration of the transputer network is however more complex and care must be taken to match up the transputer channels in the *.CFS file with the references in the behave *.CON output file.

NB When building multi-transputer programs care must be taken to ensure that the 'C' compiler uses standard data types for all transputers in the network. For example the default size of an "int" data type on a 16bit transputer (e.g. T222) is 16 bits whilst on a 32bit transputer (e.g. T425) it is 32 bits. This can lead to problems with the channel protocol. The afsmglobs.h file includes a data type definition for INT32 which specifies a 32 bit integer ("long int"). Using this for all integer data connections between AFSMs will prevent this problem from arising. Alternatively specifying either "long int" or "short int" for both transputers will also be satisfactory. Floating point and character data types do not cause problems.

Array Data And AFSM Connections

It is possible to specify AFSM registers to be arrays of up to 65535 bytes. These are dealt with slightly differently by the behaviour translator program which requires the following special attention:

- A typedef statement must define the array i.e: `typedef int array[200];` defines an array of 200 integers. This "array" type can then be used to declare registers in machine afsms: `array dataarray.`
- All array registers must have names ending in "array" e.g. "dataarray".

A.4. An Example Program

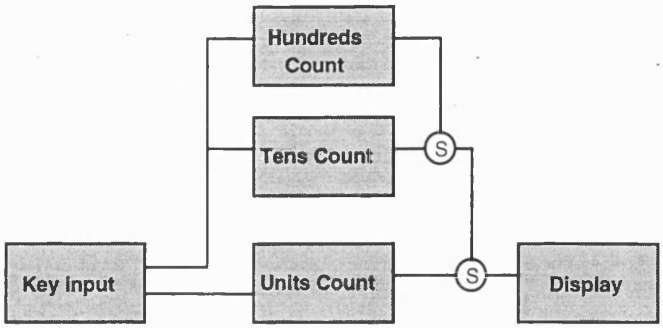


Figure A.6. Showing the counter AFSMs of the test program.

This program is a test program written to be executed on one transputer that is resident on the personal computer development system expansion card. Since the only input and out put available is the screen and keyboard these are used to control and display the outputs of a several counter AFSMs as shown in the diagram bellow. This program is also available as a multi-transputer version with the key input and display being run on one transputer and the counter AFSMs on another.

The following is the behaviour language program listing:

```
/*****************************************************************/
/*                                                               */
/*    COMMUNICATING MACHINE TEST.                                */
/*                                                               */
/* Basic Suppression test.                                       */
/* Layers of counter afsms output to a display afsm, bottom is */
/* a units counter which is subsumed by a tens counter is a '2' */
/* key is hit etc.                                               */
/*****************************************************************/
#include<iocntrl.h>
#include<ctype.h>
/*****************************************************************/
/* KEYBOARD afsm reads the PC key board and if a digit is read   */
/* its decimal value is sent.                                    */

defmachine( keyboard){
decls: int keyin, int numin;

connect( keyboard.numin, units.resetval);
connect( keyboard.keyin, tens.sendflag);
connect( keyboard.keyin, hundreds.sendflag);

rule:
        whenever( 1){
                keyin = pollkey();
                if( isdigit( keyin)){
                        numin = keyin - 48;        /* Convert ascii to decimal.*/
                        output( numin);
                }
                if( isalpha( keyin)){
                        output( keyin);
                }
        }endwhenever;
}endmachine;
```

```
/*****************************************************************/
/*****************************************************************/
/* DISPLAY machine sends value to screen.*/

defmachine( display){
decls: int countvalue;

rule:
        whenever( received( countvalue)){
                printf("THIRD: Count value = %d\n", countvalue);
        }endwhenever;

}endmachine;
/*****************************************************************/



/*****************************************************************/
/* UNITS machine sends value to third machine.*/

defmachine( units){
decls: int count, int resetval;
connect( units.count, display.countvalue);

rule:
        whenever( 1){
                if( count%4 == 0){
                        output(count);
                }
                count++;
                ifreceived( resetval){
                        count = resetval;
                }
        }endwhenever;
}endmachine;
/*****************************************************************/
/*****************************************************************/
/* TENS afsm caounts in tens and if a 't' is received from the   */
/* keyboard then the count value is output to the DISPLAY afsm.  */
/* the counter to that value.                                    */

defmachine( tens){
decls: int count, int sendflag;
connect( tens.count, suppress(display.countvalue));

rule:
        whenever( 1){
                count = count + 10;
                if( (char) sendflag == 't'){
                        if( (count/10)%2 == 0){
                                output( count);
                        }
                }
        }endwhenever;
}endmachine;
/*****************************************************************/
/*****************************************************************/
/* HUNDREDS afsm caounts in 100s and if an 'H' is received from */
/* THE keyboard then the count value is output to the DISPLAY    */
/* AFSM. the counter to that value.                              */

defmachine( hundreds){
decls: int count, int sendflag;
```

```
connect( hundreds.count, suppress( display.countvalue));

rule:
        whenever( 1){
                count = count + 100;
                if( (char) sendflag == 'h'){
                        output( count);
                }
        }endwhenever;
}endmachine;
/*********************************************************************/
/*********************************************************************/
/* End Of Behave Source File */
/*********************************************************************/
```

B. Pseudocode Listings

B.1. Pseudocode for Chapter 5

B.1.1. Motor and Laser Control Subsystem

MotorControl AFSM:

```
connect this stalled counter register to external optics control page stalled counter register
connect this actual motor speed register to external optics actual motor speed register
connect external optics control page set motor speed register to this motor set speed register

whenever( ALLWAYS){
        If photomultipler channel 1 is active and motor speed setting > 0
                calculate channel 1 average scanlevel over last 5 AFSM cycles
                if average scan level is the same as the last average scan level
                        set possible_stall_chan1 flag
                end if
        end if
        If photomultipler channel 2 is active and motor speed setting > 0
                calculate channel 2 average scanlevel over last 5 AFSM cycles
                if average scan level is the same as the last average scan level
                        set possible_stall_chan2 flag
                end if
        end if

        stall_possible = possible_stall_chan1 flag OR possible_stall_chan2 flag
        if stall_possible
                increment stalled counter
                if stalled counter > 5
                        reset stalled counter to 0
                        increment motor speed
        else
                reset motor speed
        end if

        output actual motor speed register
        output motor stalled state register
}
```

LaserMonitor AFSM:

```
connect laser power register to external optics control page laser power register

whenever( ALLWAYS){
        read laser power
        output laser power register
}
```

B.1.2. Photomultiplier Control Modules

ScanAquire AFSM:
connect scan buffer to external EHT control page, PedistalLevel AFSM and EdgeMonitor AFSM

```
whenever( ALLWAYS){
        initiate scan buffer read
        wait for scan acquisition
        if error then output error signal
        Increment sample counter
        output scan buffer register
}
```

PedestalLevel AFSM:
connect scanlevel to external EHT control page, BallPark AFSM and NearLin AFSM

```
whenever(  scan buffer received){
        scan level = mean scan level between left scan edge and right scan edge
        output scanlevel  register
}
```

EdgeMonitor AFSM:
connect leftedge register to PedestalLevel AFSM left scan edge register
connect rightedge register to PedestalLevel AFSM right scan edge register
connect external BallPark state register to this state register

```
whenever( state is not foldback){

        search scan buffer for left positive edge
        search rest of scan buffer for right negative edge
        output left edge register
        output right edge register
}
```

BallPark AFSM:
connect this EHT step size register to EHTOutPut AFSM step size register
connect this state register to EdgeMonitor state register

```
whenever( system in automatic mode){

        if scanlevel < FOLDBACK_B max limit OR EHT voltage at hi limit
                state = FOLDBACK_B
                stepsize = FOLDBACK_B EHT voltage step
        else
                if  scan level < QUIESCENT_MAX limit and scan level < QUIESCENT_MIN limit
                        state = QUIESCENT
                        stepsize = QUIESCENT EHT voltage step
                else
                        if scanlevel gradient = EHT voltage gradient
                                state = NEAR LINEAR
                                stepsize = 0
                        else
                                state = FOLDBACK_A
                                stepsize = stepsize = FOLDBACK_A EHT voltage step
                        end if
                end if
        end if
        output state register
        output stepsize register
}
```

NearLin AFSM:
connect this EHT step size register to suppress EHTOutPut AFSM step size register

```
whenever( system in automatic mode){
        if scan level > QUIESCENT_MAX limit and scanlevel gradient = EHT voltage gradient
                if scan level < setpoint - BOUNDARY
                        stepsize = adjust factor x (setpoint - scanlevel)
                else
                        if scan level > setpoint BOUNDARY
                                stepsize = adjust factor x (setpoint - scanlevel)
                        else
                                do nothing, scan level is within range.
                        end if
                end if
        end if
        output step size register
}
```

EHTOutPut AFSM:
connect EHT voltage register to external EHT control page, BallPark AFSM and NearLin AFSM

```
whenever( ALLWAYS){
        if received fixed voltage setting
                EHT voltage = fixed voltage setting
        else
                if EHT voltage + stepsize > 0
                        EHT voltage = EHT voltage + stepsize
                else
                        EHT voltage = 0
                end if
        end if

        if EHT voltage < hardware voltage limit
                set EHT voltage
        else
                set voltage to hardware limit
                EHT voltage register = hardware limit
        end if
        output EHT voltage level register
}
```

B.1.3. User Interface

MasterPage AFSM:
connect page view select register to other display page AFSMs

```
whenever( ALLWAYS){
        if page display is selected
                display user interface and page status
        end if
        process user input requests
}
```

TimeMonitorPage AFSM:

```
whenever( ALLWAYS){
        poll AFSM timer channels for characteristic time overruns
        if overruns detected and page display is selected
                display overrun status
        end if
}
```

DualChannel Control AFSM:

```
connect fixed EHT voltage register to suppress all
        external photomultiplier channel EHTOutPut AFSMs
connect set automatic register to suppress all
        external photomultiplier channel BallPark and NearLin  AFSMs

whenever( ALLWAYS){
    if page display is selected
        display channel status summary for all channels
    end if
    process user input requests
}
```

PhotomultiplierControlPage AFSMs:

```
connect fixed EHT voltage register to external photomultiplier channel EHTOutPut AFSM
connect set automatic register to external photomultiplier channel BallPark and NearLin  AFSMs

whenever(ALLWAYS){
    if page display is selected
        display channel status
    end if
    if status monitor enabled
        save status data to file
    end if
    process user input requests
}
```

OpticsControlPage:

```
connect motor speed register to external MotorControl AFSM set motor speed

whenever(ALLWAYS){
    if page display is selected
        display motor and laser status
    end if
    process user input requests
}
```

B.2. Pseudocode for Chapter 8

Enhanced BallPark AFSM:

```
connect this EHT step size register to EHTOutPut AFSM step size register
connect this state register to EdgeMonitor state register
connect status registers to external user interface EHT control page

whenever( system in automatic mode){

        if scanlevel < FOLDBACK_B max limit OR EHT voltage at hi limit
                rule vector = 0,0,0,1
        else
                if  scan level < QUIESCENT_MAX limit and scan level < QUIESCENT_MIN limit
                        rule vector = 1,0,0,0
                else
                        if scanlevel gradient = EHT voltage gradient
                                rule vector = 0,1,0,0
                        else
                                rule vector = 0,0,1,0
                        end if
                end if
        end if
        Pre-process input vector (i.e. normalise or scale input values to range 0-1)
        Apply input to Kohonen layer to get situation map vector
        Apply situation map vector and rule vector to Grossberg layer to get final output
        If AFSM rule active{
                Update Kohonen layer
                Update Grossberg layer with rule vector
                update neteffect variable
        end if

        if AFSM final output active
                generate step size and state register value
                        state = NEAR LINEAR
                        stepsize = 0 OR
                                state = FOLDBACK_A
                                stepsize =  FOLDBACK_A EHT voltage step OR
                                        state = QUIESCENT
                                        stepsize = QUIESCENT EHT voltage step OR
                                                state = FOLDBACK_B
                                                stepsize = FOLDBACK_B EHT voltage step
                output step size register
        end if
        generate state indication

        output status information registers
        output state register
}
```

Enhanced NearLin AFSM:

```
connect this EHT step size register to suppress EHTOutPut AFSM step size register
connect status registers to external user interface EHT control page

whenever( system in automatic mode){
        if scan level > QUIESCENT_MAX limit and scanlevel gradient = EHT voltage gradient
                if scan level < setpoint - BOUNDARY
                        rule vector = 1,0,0
                else
                        if scan level > setpoint BOUNDARY
                                rule vector = 0,1,0
                        else
                                rule vector = 0,0,1
                        end if
                end if
        end if
        Pre-process input vector (i.e. normalise or scale input values to range 0-1)
        Apply input to Kohonen layer to get situation map vector
        Apply situation map vector and rule vector to Grossberg layer to get final output
        If AFSM rule active{
                Update Kohonen layer
                Update Grossberg layer with rule vector
                update neteffect variable
        end if

        if AFSM final output active
                generate step size register value depending on identified AFSM state
                        rule output = adjust factor x (setpoint - scanlevel) OR
                                rule output  = adjust factor x (setpoint - scanleve) OR
                                        do nothing, scan level is within range
                output step size register
        end if

        output status information registers
}
```

Adaptive BallPark AFSM:

```
connect this EHT step size register to EHTOutPut AFSM step size register
connect this state register to EdgeMonitor state register
connect status registers to external user interface EHT control page

adaptive registers:        quiescent_step_size - adaptreg[0]
                           foldback_A_step_size - adaptreg[1]
                           foldback_B_step_size - adaptreg[2]

whenever( system in automatic mode){

        if scanlevel < FOLDBACK_B max limit OR EHT voltage at hi limit
                state = FOLDBACK_B
                stepsize = quiescent_step_size - adaptreg[2]
        else
                if  scan level < QUIESCENT_MAX limit and scan level < QUIESCENT_MIN limit
                        state = QUIESCENT
                        stepsize = quiescent_step_size - adaptreg[0]
                else
                        if scanlevel gradient = EHT voltage gradient
                                state = NEAR LINEAR
                                stepsize = 0
                        else
                                state = FOLDBACK_A
                                stepsize =  foldback_A_step_size - adaptreg[1]
                        end if
                end if
        end if

        if adjustment_period is over
        Calculate reinforcement value for last reinforcement_period for register - adaptreg[n]
                If reinforcement has gone down for current register since value was adjusted
                        Reset current register adaptreg[n] to original value
                        Reverse direction of current register adaptreg[n] adjustment

                Else reinforcement has gone up
                        Save new value as a function of current register adaptreg[n] mean activity
                end if

                set pointer to next register for adjustment adaptreg[n+1]
                Make new adjustment to new current register adaptreg[n] value as a function of
                        tuning_factor,
                        current value,
                        adaptreg[n] mean reinforcement and
                        adaptreg[n] register activity
        end if

        output state register
        output stepsize register
}
```

Adaptive NearLin AFSM:
connect this EHT step size register to suppress EHTOutPut AFSM step size register

adaptive registers: adjust factor

```
whenever( system in automatic mode){
        if scan level > QUIESCENT_MAX limit and scanlevel gradient = EHT voltage gradient
                if scan level < setpoint - BOUNDARY
                        stepsize = adjust factor x (setpoint - scanlevel)
                else
                        if scan level > setpoint BOUNDARY
                                stepsize = -adjust factor x (scanlevel - setpoint )
                        else
                                do nothing, scan level is within range.
                        end if
                end if
        end if

        if adjustment_period is over
                Calculate reinforcement value for last reinforcement_period for adjust factor register
                If reinforcement has gone down for adjust factor setting since value was adjusted
                        Reset adjust factor register adaptreg[n] to original value
                        Reverse direction of adjust factor register adjustment

                Else reinforcement has gone up
                        Save new value as a function of adjust factor register mean activity
                end if

                Make new adjustment to adjust factor register value as a function of
                        tuning_factor,
                        current value,
                        mean reinforcement and
                        register activity
        end if

        output step size register
}
```

Enhanced - Adaptive BallPark AFSM:

```
connect this EHT step size register to EHTOutPut AFSM step size register
connect this state register to EdgeMonitor state register
connect status registers to external user interface EHT control page

adaptive registers:        quiescent_step_size - adaptreg[0]
                           foldback_A_step_size - adaptreg[1]
                           foldback_B_step_size - adaptreg[2]

whenever( system in automatic mode){
        if scanlevel < FOLDBACK_B max limit OR EHT voltage at hi limit
                rule vector = 0,0,0,1
        else
                if  scan level < QUIESCENT_MAX limit and scan level < QUIESCENT_MIN limit
                        rule vector = 1,0,0,0
                else
                        if scanlevel gradient = EHT voltage gradient
                                rule vector = 0,1,0,0
                        else
                                rule vector = 0,0,1,0
                        end if
                end if
        end if
        Pre-process input vector (i.e. normalise or scale input values to range 0-1)
        Apply input to Kohonen layer to get situation map vector
        Apply situation map vector and rule vector to Grossberg layer to get final output
        If AFSM rule active{
                Update Kohonen layer
                Update Grossberg layer with rule vector
                update neteffect variable
        end if

        if AFSM final output active
                generate step size and state register value
                state = FOLDBACK_B
                stepsize = quiescent_step_size - adaptreg[2] OR
                        state = QUIESCENT
                        stepsize = quiescent_step_size - adaptreg[0] OR
                                state = NEAR LINEAR
                                stepsize = 0 OR
                                        state = FOLDBACK_A
                                        stepsize =  foldback_A_step_size - adaptreg[1]
                output step size register
        end if

        if adjustment_period is over
        Calculate reinforcement value for last reinforcement_period for register - adaptreg[n]
                If reinforcement has gone down for current register since value was adjusted
                        Reset current register adaptreg[n] to original value
                        Reverse direction of current register adaptreg[n] adjustment

                Else reinforcement has gone up
                        Save new value as a function of current register adaptreg[n] mean activity
                end if
                set pointer to next register for adjustment adaptreg[n+1]
                Make new adjustment to new current register adaptreg[n] value as a function of
                        tuning_factor,
                        current value,
                        adaptreg[n] mean reinforcement and
                        adaptreg[n] register activity
        end if

        output status information registers
        output state register
}
```

Enhanced - Adaptive NearLin AFSM:
connect this EHT step size register to suppress EHTOutPut AFSM step size register
connect status registers to external user interface EHT control page

```
adaptive registers:        adjust factor

whenever( system in automatic mode){
        if scan level > QUIESCENT_MAX limit and scanlevel gradient = EHT voltage gradient
                if scan level < setpoint - BOUNDARY
                        rule vector = 1,0,0
                else
                        if scan level > setpoint BOUNDARY
                                rule vector = 0,1,0
                        else
                                rule vector = 0,0,1
                        end if
                end if
        end if
        Pre-process input vector (i.e. normalise or scale input values to range 0-1)
        Apply input to Kohonen layer to get situation map vector
        Apply situation map vector and rule vector to Grossberg layer to get final output
        If AFSM rule active
                Update Kohonen layer
                Update Grossberg layer with rule vector
                update neteffect variable
        end if

        if AFSM final output active
                generate step size register value depending on identified AFSM state
                        rule output = adjust factor x (setpoint - scanlevel) OR
                                rule output  = adjust factor x (setpoint - scanleve ) OR
                                        do nothing, scan level is within range
                output step size register
        end if

        if adjustment_period is over
                Calculate reinforcement value for last reinforcement_period for adjust factor register
                If reinforcement has gone down for adjust factor setting since value was adjusted
                        Reset adjust factor register adaptreg[n] to original value
                        Reverse direction of adjust factor register adjustment

                Else reinforcement has gone up
                        Save new value as a function of adjust factor register mean activity
                end if

                Make new adjustment to adjust factor register value as a function of
                        tuning_factor,
                        current value,
                        mean reinforcement and
                        register activity
        end if
        output status information registers
}
```

Continuous Output Net BallPark AFSM:

```
connect this EHT step size register to EHTOutPut AFSM step size register
connect this state register to EdgeMonitor state register
connect status registers to external user interface EHT control page

whenever( system in automatic mode){

        rule active = 0
        if scanlevel < FOLDBACK_B max limit OR EHT voltage at hi limit
                state = FOLDBACK_B
                rulestepsize = FOLDBACK_B EHT voltage step
                rule active = 1
        else
                if  scan level < QUIESCENT_MAX limit and scan level < QUIESCENT_MIN limit
                        state = QUIESCENT
                        rulestepsize = QUIESCENT EHT voltage step
                        rule active = 1
                else
                        if scanlevel gradient = EHT voltage gradient
                                state = NEAR LINEAR
                                rulestepsize = 0
                                rule active = 1
                else
                                state = FOLDBACK_A
                                rulestepsize =  FOLDBACK_A EHT voltage step
                                rule active = 1
                        end if
                end if
        end if
        Pre-process input vector (i.e. normalise or scale input values to range 0-1)
        Apply input to Kohonen layer to get situation map vector
        Apply situation map vector and rule vector to Grossberg layer to netoutput
        If AFSM rule active
                Update Kohonen layer
                Update Grossberg layer with rulestepsize
                update neteffect variable
        end if

        AFSM stepsize output = (neteffect x netoutput) + (1-neteffect x rulestepsize)

        output AFSM stepsize output
        output status information registers
        output state register
}
```

Continuous Output Net NearLin AFSM:
connect this EHT step size register to suppress EHTOutPut AFSM step size register

```
whenever( system in automatic mode){

        rule active = 0
        if scan level > QUIESCENT_MAX limit and scanlevel gradient = EHT voltage gradient
                if scan level < setpoint - BOUNDARY
                        rulestepsize = adjust factor x (setpoint - scanlevel)
                        rule active = 1
                else
                        if scan level > setpoint BOUNDARY
                                rulestepsize = adjust factor x (setpoint - scanleve )
                                rule active = 1
                        else
                                rulestepsize = 0
                                do nothing, scan level is within range.
                                rule active = 1
                        end if
                end if
        end if

        Pre-process input vector (i.e. normalise or scale input values to range 0-1)
        Apply input to Kohonen layer to get situation map vector
        Apply situation map vector and rule vector to Grossberg layer to get netoutput
        If AFSM rule active
                Update Kohonen layer
                Update Grossberg layer with rulestepsize
                update neteffect variable
        end if

        AFSM stepsize output = (neteffect x netoutput) + (1-neteffect x rulestepsize)

        output AFSM stepsize output
        output status information registers
}
```