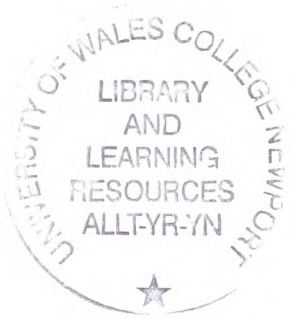


Integration of Navigation with a Behavioural Control Scheme

Thesis submitted to the University of Wales for the degree of:

Doctor of Philosophy

Christopher Ashley James Tubb B.Eng.

Mechatronics Research Centre,

Department of Engineering

University of Wales College, Newport.

December 2000

DECLARATION

This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree.

Signed.....(candidate)

Date....17th November 2000.....

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated.

Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed.....(candidate)

Date.....17th November 2000.....

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed.....(candidate)

Date.....17th November 2000

SUMMARY

This thesis describes a navigational strategy for automatically guided vehicles and a behavioural control implementation using a modified zero order Sugeno fuzzy inference engine. Animals are examined as a model of intelligent behaviour and behavioural control implementations discussed and compared. The features and the requirements of a behavioural control implementation are identified and modifications to Sugeno inferencing are then described which allow these to be met. A successful implementation is then presented. The navigation task is examined, as are various methods of representing space. Special structural features of constrained spaces are identified. A route based navigation system exploiting these features is then developed. The strategy combines the representation of space with the task description into an easily communicated message. This navigational strategy is then successfully integrated with the behavioural control implementation presented earlier. The influence of spatial structure on the complexity of the navigation task is investigated with a view to the categorisation of space. A definition of maze-spaces is developed from this. Conclusions are then drawn on the themes of the work, and suggestions made for further investigations.

CONTENTS

LIST OF FIGURES

1: INTRODUCTION TO THE RESEARCH

There are three main themes running through this work, used to synthesise a navigation system for automatically guided vehicles [AGV] that are used within constrained spaces of the type found within the human environment.

This work describes an implementation strategy for behavioural controllers based on the principles of fuzzy logic. The architecture developed ensures the properties of behavioural control are conserved while exploiting the advantages of using fuzzy logic for the description of "intelligent" tasks. The result is an intuitive system of control that may be easily applied to the task of navigation in an indoor environment. The architecture is extensible and may integrate controller elements with different structures.

The special problems associated with navigation are investigated, and the requirements for successful navigation outlined. This is extended to identify the unusual properties of constrained environments that may influence the structure of navigating agents. A system of navigation is then proposed as a suitable mechanism for control of a vehicle in space. The use of fuzzy theory and a simple control architecture, together with a reduction in system complexity achieved through tight coupling with the features of the environment in which the vehicle is intended to operate, is exploited to allow scripts to form the major spatial representation method.

Issues about the suitability of behavioural control for a variety of tasks are raised and discussed, as are comparisons with animal behaviour and its suitability as a model for behaviour in intelligent systems.

Methods of identifying the structure of spaces are suggested, and used to develop a definition of 'maze-spaces' of a type suitable for the proposed navigation system.

1.1 Motivation

Machines that have the ability to operate with little human supervision have many industrial, military, commercial and scientific uses. High degrees of flexibility also improve the usefulness of machines.

The level of supervision required is dependent on the complexity of the environment and the level of intelligence that the agent displays. Immediately a machine is required to move through an environment, the complexity of the control task increases drastically. To achieve autonomous operation in mobile robots or vehicles requires novel control systems and some method of representing space.

To exhibit usefulness in an industrial or commercial environment an autonomous vehicle must be directed towards a task or goal. For maximum flexibility an operator should be able to define this task simply, which the vehicle will then perform independently, without elaborate commissioning procedures, environmental modification, or extended machine-training times. Optimum flexibility and usefulness are likely to be achieved where the system task may be reconfigured rapidly and simply.

The aim of this work is thus to identify methods by which an automatically guided vehicle may be designed to demonstrate an optimum balance between autonomy and usefulness and suggest a controller architecture by which this may be achieved. This architecture should support a method of easily communicating a new task to the machine from a scheduler or human operator, while maintaining performance. As the movement is the primary feature of an AGV, navigation has been chosen as the main focus of activity.

The objectives by which this aim is to be reached are to:

Investigate navigation in both animals and machines and identify those aspects that are relevant to navigation of agents within the chosen environment.

Identify the features of the environment that effect navigation and determine if these may be exploited to aid the navigation process.

Investigate possible behavioural control architectures, with consideration of the integration of a navigational system.

Describe a navigational system that may be used within the chosen environment and integrated with a control architecture to produce a automatically guided vehicle.

1.2 Approach

The work described here is an attempt to achieve the objectives stated above, independence from supervision, task directed-ness and flexibility, while supporting the communication of navigational goals. In pursuit of these objectives a number of contributing elements have been considered.

The suitability of behavioural techniques for the control of an autonomous vehicle is discussed as is the use of fuzzy logic as the implementation technology for this controller. Adoption of fuzzy techniques has the advantage of making the transfer of expertise from a human designer or operator to the vehicle simple, but also generates some interesting questions on the nature of the controller itself. The use of standard fuzzy inference engines provides a simple method to integrate behaviours, which is mathematically well understood.

A script based navigation technique is presented. This is made possible by the spatial simplification of the human environment. The environment is reduced to a network of locations connected by paths. These paths are physical, and exist due to the requirements humans place on their environment. The coupling of the navigation techniques to the

environment extends to the task of location. The vehicle uses generic features as guides in the use of the navigational script.

1.3 Definition of common terms

The following terms are used extensively throughout this work. To prevent confusion between the many interpretations of each that may be employed by those working in the fields of robotics, machine intelligence and control, brief definitions as used are given. These usages are partial quotations from the Oxford English Dictionary (1971):

Autonomous:

Making or having ones own laws

1) Of or pertaining to autonomy

2) Possessed of autonomy, self-governing, independent

3) Conforming to its own laws only & not subject to higher ones.

Autonomy:

2) Autonomous condition...b)organic independence

Automatic:

Self-acting, having the power of motion or acting within itself

1)..."Nothing can be said to be automatic" Sir Humphrey Davy

Foster "...modern meaning totally opposite- understood to mean..."

2) Self-acting under conditions fixed for itself...

3) Of animal Actions like those of mechanical automatons; not accomplished by volition or consciousness.

4) Not characterised by active intelligence, merely mechanical

Route:

Way taken in getting from starting point to destination.

Send along particular route

Path:

Footway, track, line along which person or thing moves

Course:

Onward movement in space/time. Direction of going. Line of conduct or action, series or sequence. Line of bits (e.g. course of bricks)

As can be seen the dictionary some of these definitions are a little ambiguous, and in the case of "automatic", contradictory. For this reason the Oxford definitions have been interpreted in this work to mean:

Autonomous:

Operating in a totally independent manner, in pursuit of internal goals only.

Automatic:

Self-acting under fixed conditions in a manner not characterised by active intelligence, mechanical

Route:

Way taken in getting from starting point to destination.

Path:

Line along which an agent moves.

Course:

The series of actions that make up a route.

1.3.1 Cataloguing behaviour

The use of the term behaviour throughout the literature is extensive; confusion may easily follow, as its employment is flexible. Following is a brief explanation of the terms used to describe the behaviour of a system employed within this work. The major part of the definition is drawn from Grier (Grier and Burk 1992) and relates to ethology and animal behaviour studies.

1.3.1.1 Behavioural Hierarchies

A behavioural system is one that responds to the environment in which it exists by performance of some action. The manner in which a system acts, in its entirety, is the *behaviour*, this is sometimes referred to by such terms as the *system behaviour* or *over-all behaviour* for clarity.

By watching animals, or using knowledge of the construction of machines it can be observed that behaviour is made up of a series of smaller units of function. The smallest of these are referred to as *Ethons*, *Sub-behaviours* or *behavioural sub-units*. These can be thought of as the atomic units of behaviour.

Interaction of these fundamental units produces *Action patterns* and *Behavioural acts*.

Note that in the world of ethology, the terms *Ethon, Action Pattern* and *Behavioural act* are all equivalent. As it is possible to have a deeper knowledge of the structure of a behavioural controller these definitions may be expanded.

The behaviours or *elements* of a system are the set of components of the system behaviour. *A behaviour* is a member of this set, so may be either an action pattern or ethon.

A diagrammatic illustration of these relationships is given in Figure 1.1

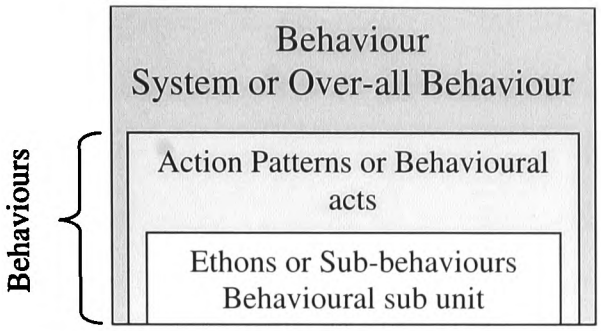


Figure 1.1 Hierarchy of behaviours

1.3.1.2 Sequences

Sequences of action are an important technique in both animals and machines. These are referred to as *behaviour sequences* or *behaviour patterns.*

1.3.1.3 Nomenclature: Action and Consequence.

There are two main techniques for naming behaviour. These are "description by action" and "description by consequence".

Description by action:

> The action that is exhibited by a behaviour is used to define that action pattern. This would be characterised by names such as "Phono-taxis" (turning towards a sound), "extend leg" and "turn right".

Description by consequence:

> The name of a behaviour is drawn from the consequence or goal of that behaviour. Examples of this method of description would be "wall following", "Seek shelter" and "hunt".

Grier (Grier and Burke 1992) states that consequential descriptions are more useful in cataloguing the behaviour of animals, but that the definitions are open to interpretation. A description of an action, though easy to interpret, provides no indication of the purpose of that pattern of action. A description by consequence may easily be applied to an action

pattern that in reality has no relation to the achievement of a goal, but is merely coincident. The goal may not even exist, being imposed on the animal or machine by the prejudice of the observer. The anthropomorphisation of behaviour by an observer is widely reported as anecdote (Warwick 1997), this can be a problem for the object study of behaviour. It appears that human observers often see human-like traits and consciousness exhibited by systems that probably contain neither, behavioural edifices being interpreted as conscious behaviour patterns.

Using behaviours in mechanical systems however benefits from both approaches to description. Description by action may be useful for lower levels of a controller, while description by consequence is far more convenient when there are sequences and patterns of behaviour under consideration. This naming convention also has advantages during the design phase of a controller, as it enables the controller to be specified as a series of consequences, to which achieving actions may later be added, allowing a more abstract design approach.

1.4 Overview of the thesis

The thesis is divided into chapters as follows:

Chapter 2 Animal Behaviour

Describes some of the properties of animal behaviour and illustrates why this is considered a suitable model for the design of intelligent or autonomous systems.

Chapter 3 Behavioural Control

Contrasting behavioural controllers with both other control paradigms and animal behaviour as explored in Chapter 2. This chapter defines behavioural control through the use of examples and the discussion of important behavioural control architectures. Implementation issues are considered and the advantages and disadvantages of adopting behavioural control techniques are investigated.

Chapter 4 Navigation and Spatial Representation

Concepts of space are considered and various representations of space outlined. These are then used to determine the properties a representation might need if it is to be used for navigational purposes. The chapter then investigates the navigation process, describing the requirements and various methods by which navigation may be achieved. Focus is concentrated on the relationship these navigational methods hold with the representation of space and with features of the environment.

Chapter 5 Fuzzy behavioural control architecture.

Chapter 5 details the development of a fuzzy behavioural controller. It addresses the issues of implementing a behavioural controller in fuzzy logic and suggests some modifications that are required to a standard zero order Sugeno inference system to allow full behavioural control implementations. The result of this development is an architecture employing a hierarchical combination of fuzzy inference systems to produce a working controller. This architecture is able to display dynamic prioritisation, inter-element inhibition and support and to integrate elements that are of a different structure. Deliberative elements may also be integrated simply without disruption of the behavioural controller, allowing the architecture to form the basis of layers and hybrid control strategies.

Chapter 6 Navigation via Script in Constrained Environments.

Some of the concepts developed in chapter 4 are expanded to provide a mechanism suitable for navigation of an automatic vehicle operating within a constrained environment such as a building. The technique benefits from exploiting features of these environments. The technique merges the communication of the goal to the vehicle with the representation of space used to navigate by embedding sufficient information in the message. This message is a route, or list of locations and associated actions. The technique exploits the features of the environment to simplify the specification of routes, and their interpretation. The system is simple, and may be easily understood by English speaking humans and machine alike.

Chapter 7 Identification of Maze Spaces

The methods described in Chapters 5 and 6 rely heavily on the agent operating in a spaces that exhibits certain features. Before these techniques may be used on a vehicle in a real situation, it must be ensured that the environment in which it is to operate also exhibits these features. This chapter looks at spaces suitable for the operation of the navigation system described, and determines features that allow them to be identified. From this investigation a definition is developed.

Chapter 8 Categorisation of Spaces

The structure of a space has an impact on the effectiveness of navigational techniques. Different techniques can exhibit different levels of performance when operated in spaces with differing properties. In addition the comparative measurement of performance between navigational techniques requires that they be tested in the same environment. Where this is not possible some measure of the structure of the space and its navigational difficulty needs to be made. This chapter proposes measures that may form the basis of an index of navigational complexity.

Chapter 9 Conclusions and issues for further investigation

The final chapter of the work summarises and identifies those important issues raised throughout the work. Areas where it is believed further investigation would be fruitful are also listed.

Appendix 1 Fuzzy sets and fuzzy logic

Appendix 1 provides an overview of fuzzy logic listing features and techniques and describing some of the possible structures and operators used.

Appendix 2 Asimov's three laws of robotics

Lists the laws of robotics as developed by Isaac Asimov. The laws are also placed in a literary context.

Appendix 3 Example direction sets

Appendix 3 provides three examples of directions sets, as used to communicate a route. These sets are taken from real world situations. Some analysis of the structure of the sets is used to highlight concepts employed in the navigational system described in chapter 6.

1.5 References

Grier J.W., and Burk T. 1992. *Biology of Animal Behaviour.* St. Louis U.S.A.: Mosby Year Book.

Warwick K. 1997. *March of the Machines.* London: Century.

2: ANIMAL BEHAVIOUR

The requirements of survival have forced some amazing abilities to be obtained by animals. It appears that the more science and engineering try to imitate the animal kingdom, the more respect it demands. The following chapter provides an introduction to animal behaviour that concentrates on those aspects that have relevance to the design and construction of intelligent systems.

2.1 Introduction

The desire to emulate animal behaviour is rooted in nature's ability to perform tasks that appear to be difficult to even define with traditional mathematical control and computing techniques. While machines can be built that perform better than humans in some areas, it has become a common complaint in the literature that those abilities that are the simplest for man and beast are the ones that elude science. This can be seen in the different levels of success of chess computers and delivery robots. A delivery robot has many more practical applications than "Deep Blue". With their far greater utility it would not be unreasonable to expect to see many more successful delivery robots than mechanical grand masters. This however, is not the case; chess computers were almost ubiquitous a decade ago while independently operating delivery robots are still rather scarce. The highly structured environment of the chessboard makes techniques such as searching through a game model feasible. Chess computers do not need to cope with noisy sensors, the detail of motor action or making decisions in real-time. A delivery robot, operating where there are many more variables and parameters, is harder to create. Animals however primitive, do exhibit the types of competence that we require of our machines. Observation of animal

behaviour and interpretation of the mechanisms that produce it help provide paradigms for the control of machines.

2.2 Animal behaviour and machines

When observing animals the complexity of the natural world means that there is some difficulty in determining exactly what the fundamental behaviour is, and how it has been affected by interaction with the wider environment. The more complex the behaviour the more pronounced this effect becomes. Simple animals such as insects exhibit behaviour that can be observed and analysed less arduously than higher taxonomies. This is one reason much work in the mechanical emulation of natural behaviour has concentrated on insects and insect like machines. The attempts at modelling the behaviour of such simple animals may be specific (Lund et al., 1997), or more loosely based on an abstract perception of the type of activity observable in simple organisms (Mitchell et al., 1994)

Regardless of the simplicity of the animals that are being observed, the usefulness of analysing natural behaviour is not greatly diminished due to present technology being concerned with tasks and goals of a low level of complexity. There are some exceptions to this, such as the COG and KISMET projects at The Massachusetts Institute of Technology < http://www.ai.mit.edu/projects/humanoid-robotics-group/> where there is an attempt to create an artificial humanoid, capable of interacting fluidly with those in its vicinity. This work however has been based on principles learnt through extensive building of small insect like machines.

Although most of the design that has been implemented in COG is an attempt to produce an analogue of systems found in the natural world, the implementation of these systems are vastly different to those found in nature, as human technology has a different set of strengths and weaknesses. Motors and muscles may perform equivalent functions but are operationally and technologically very different.

The traffic in information has not been one way from the biologist to the engineer. The construction and use of small mechanical systems using biological inspiration has also helped to illuminate some aspects of animal behaviour and brought insight into the workings of animal minds (Webb 1994)(Webb 1996)(Lund et al., 1997)(Lambrinos et al., 1998)(Scutt 1994).

There are some properties of animal behaviour that appeal greatly to the designer of machines. Animals are incredibly robust, not only physically, but also to variability of environment. Early attempts to produce meaningful behaviour automatically were limited to highly constrained environments (Malcolm and Smithers 1990)(Agre and Chapman 1990). As Artificial Intelligence has moved away from the traditional symbolic processing and search techniques towards more biologically analogous systems, the robustness of the machines produced to the complexity of the environment has increased.

2.3 Behavioural mechanisms in animals

The manner in which animals behave would appear to be quite different to the mechanisms that have traditionally been used in machines and automatic systems. A recurring theme of both animal behaviour studies and observations made while designing intelligent systems is that novel solutions often out perform those of a more obvious and mathematical nature.

In traditional intelligent systems the environment is measured and used with some degree of prior knowledge to construct an internal representation of the world. A decision making engine is then fed the world state and a goal condition, and expected to produce a plan. When executed this plan will result in the goal being achieved.

Animals however seem to behave in a rather less structured manner. Rather than a homogenous system, like that outlined above, the behaviour of animals appears to be defined by a collection of behaviour sub units, each concerned with a small subset of the actions and responses that the animal will make during its everyday life. Each of these

behaviour sub-units is likely to operate on a subset of the available environmental inputs. Each sub-behaviour is a self-contained processing unit, or Motor Schema (Arkin 1998).

The "design method" which produced animal behaviour is also markedly different to the design methodologies commonly adopted for the construction of machines. If behaviour is assumed to be an extension of the animals phenotype (Dawkins 1989), rather than the product of some vital essence or 'ghost in the machine', it may be thought of as the product of a perpetual cycle of tinkering and testing in the hands of natural selection. A designoid [appearing to be designed] process that builds on previous behavioural prototypes by addition, but rarely removes existing elements unless they lower the fitness of that species.

As mentioned above, the complexity of animal behaviour makes determination of the mechanisms that produce it difficult. There are however several identified mechanisms to account for the behaviour observed in animals.

2.3.1 Motor Schema

One view of the creation of animal behaviour is through the interaction of a system of motor schemas.

A motor schema is a pattern of action, which is observed in, or followed by an animal or machine. Arkin gives a list of five different definitions in his 1989 paper (Arkin 1989). Each motor schema is associated with a perceptual schema. When a perceptual schema is realised, the associated motor schema is activated. It can be seen that there is very little difference between a motor schema and a behavioural sub-unit. An important property of motor schema theory is the identification of perceptual and motor schemas. This allows the triggering event, or stimulus, and process to be explicitly identified.

Motor schema outputs have a definite form. All the schemas that compete or interact will have the same form of output. These outputs are then combined to produce a suitable compromise. Production of actual motor action will require a further transformation of

this signal to a form that can be applied to the muscles. Experimental evidence for this abstraction of motor signals is reported in (Bizzi et al., 1991).

2.3.2 Reflexive behaviour

Some aspects of everyday animal behaviour are heavily stereotyped. These reflexes represent perhaps the simplest "modules" of animal behaviour.

Reflexes use only small numbers of neurones, so much so that they may be controlled from outside the central nervous system. The existence of reflexes reinforce the view that animal behaviour may be thought of as being distributed around the whole of the animal's nervous system. In the knee-jerk reaction, a single sense neurone is directly connected to a single motor neurone. A signal from the sense neurone will trigger the motor neurone directly into action. The neural circuit comprising a single synapse (Chulder).

Reflexive behaviours are effectively "hardwired" into the animal, and by definition beyond conscious control (Jeffrey 1995). The presence of the correct triggering stimulus will cause the reflex behaviour to be performed. Large parts of the animal's processing system are bypassed. For example, the neuronal system that implements the reflex may indeed be local to the muscles that it controls. The response that causes a human to keep balance by extending a leg backward when leaning forward is produced reflexively by the spinal cord. Afferent neurones are then required to report this action to the brain (Dock 1996).

The strength of response in a reflex action is determined by the strength of the stimulus and the way it is administered. Generally the strength of the response is proportional to the strength of the stimulus applied. Interestingly other stimuli or voluntary actions may affect the performance of the reflex by either inhibition or facilitation of the action. The modulating stimulus may be almost undetectable and the relevance of the modulating voluntary behaviour to the reflex may be slight (Ison 1995). The level of reflex modulation may be used to investigate brain function; specifically the amount of mental analysis carried out on the modulating stimulus may be determined.

Reflexes are an important part of animal behaviour, especially in lower animals. The simplicity of the computation circuitry, and the proximity of this to the muscles implicated in the response make them some of the fastest operating behaviours, ideal for safety and housekeeping tasks.

2.4 Robustness

As specific neuronal paths control individual reflexes, it is possible for several behaviours to be active at any time. In addition, this dependence on different processing hardware for individual behaviours may lead to increased robustness. Damage to a path or paths does not affect all behaviour. Although only the simplest of behaviours are directly wired in this way, the huge connectivity of neuronal systems means that all behaviour is robust. Evidence of this and its beneficial effect on the robustness of the behaviour can be illustrated by some cases of extreme cranial trauma. In 1848 Phineas Gage, a New England (USA) railway worker was involved in an industrial accident. An explosion caused a 1m long, 30mm diameter, tamping-rod weighing 5kg to be blown through his head, entering through the left cheek and finding egress through the top of the skull. Gage was still talking immediately after the accident, and made a full physical recovery within a few months (McNally 1998). Gage appeared normal after his wound had healed, although there was an element of behaviour modification or loss for some higher level functions, which manifest as a personality change. Few mechanical systems could withstand such physical damage.

2.5 Behaviour generation

Reflexes not only support a distributed view of behaviour, but also imply that much behaviour is created using very simple transformations, rather than complex decision making algorithms. A reflex is a simple mapping between stimulus and action. There is no concept of goal to be stored or worked towards. Any models of the environment are

stored implicitly in the transfer function. Provided applied stimuli remain within expected limits, the behaviour of each sub unit will remain predictable.

These characteristics support the required high level of robustness within the behaviour of an animal. The highly distributed nature of the reflexive actions reduces the chances that trauma to any area of the nervous system will be catastrophic or that inappropriate data will be propagated throughout the behaviour of the animal. The dispersal of representation to simple mappings distributed around simple behaviours performs a similar effect, preserving the information held by the animal.

2.6 Behavioural Sub-units

Along with stereotyped actions, more complex behaviour is also generated by the synthesis of many simple behavioural modules. An illustration of this modular nature of behaviour may be observed in the egg retrieval response of the greylag goose (Grier and Burk 1992). The goose makes a shallow nest on the ground, the low sides of which mean it is not uncommon for eggs to roll out. If the goose spots an egg outside of the nest, it follows a simple, stereotyped procedure to retrieve it. The neck is extended over the egg, then the beak lowered so that the goose may drag the egg back towards the nest, using the underside of the bill. Retracting the neck pulls the egg back into the nest. To prevent the egg from rolling around the side of it's bill, the goose employs a side to side oscillatory action. Finally when the egg has been drawn back into the nest, it is tucked under the bird.

What is interesting about this behaviour is that even when the goose has lost control of the egg, or the egg is removed by an experimenter, it will continue to withdraw its neck. Finishing the retrieval task with the same flourish that would have placed the egg in safely under the bird's plumage had the retrieval been a success. The goose will then restart the whole retrieval behaviour from the beginning. The goose can see that it has lost the egg, but is incapable of breaking its behavioural mould, the retrieval response being a single

behavioural sub-unit. Once initiated such responses must be completed. The greylag goose is not the only ground nesting bird to exhibit such behaviour.

Behavioural modules may appear to be inefficient, ineffective and even humorous, but the assembly of animal behaviour from such simple atomic units does not limit the repertoire of the animal. Fixed actions of behaviour, employed in specific circumstances lower the amount of calculation that must be made by the animal, and increases the number of situations to which the animal may respond without requiring a large increase in intelligence.

Contrary to lowering the effectiveness of the animal, fixed action patterns provide common situations with automatic responses reserving the difficulties of synthesising new behaviour for novel situations.

2.7 Selection of animal behaviour

If animal behaviour is thought of in the way outlined in this chapter, then it is an event driven system. There are occasions however when the same animal will not always respond to the sensory stimulation in the same way. An animal also exhibits state, which will affect the behaviour it performs. The motivation of an animal determines which set of behaviours it is likely to employ. Motivation is a compromise between the internal and external conditions acting upon an agent.

An animal will be less inclined to drink, if it no longer thirsts, and may stop feeding on the approach of a predator, when a more appropriate action would be to take flight. Alternatively an animal that has few conflicting motivational factors may drink when presented with a source of water, even if its thirst is not great. In most animals the desire for immediate survival is the strongest motivational force, as would be expected, followed by such desires as feeding and drinking, which though just as important are rarely urgent.

These differing motivational forces are managed by a mechanism of prioritisation, inhibition and sensitisation to activation.

The prioritisation of behaviours has been studied in the carnivorous inter-tidal snail *Pleurobranchia* where the neurophysiological control of some behaviours is partially understood (Ridley 1995). In this snail the escape response has highest priority. Withdrawal of the oral veil is a response to being touched. While the oral veil is withdrawn the snail cannot feed. If the snail is sated, or there is no food present the withdrawal response is activated on touching the animal, if the snail is hungry and feeding this is not the case. Eggs form a part of the snails' diet, for this reason the feeding response is inhibited while the snail is laying its eggs. "Righting" is the snail's response to being turned over. Figure 2.1 shows a schematic representation of the interaction between the snail's behaviours.

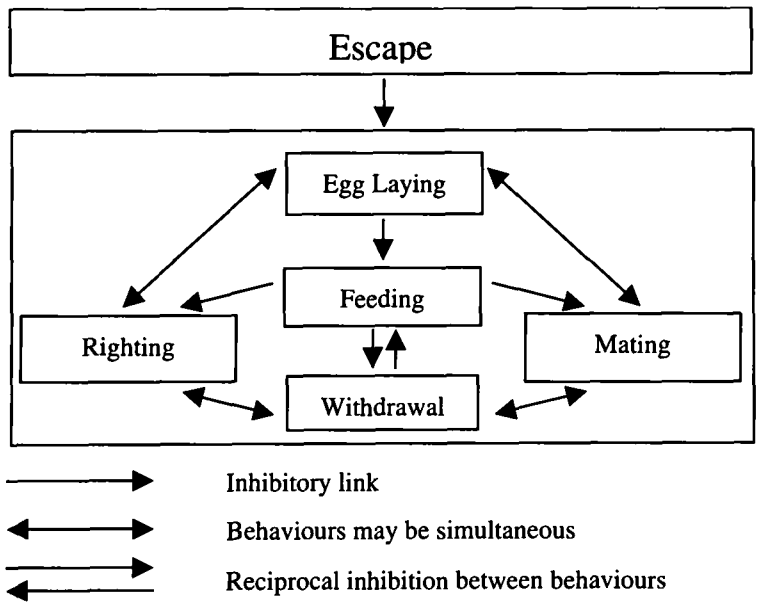


Figure 2.1 Behaviour Selection in Inter-tidal snail *pleurobranchia*.

2.7.1 Triggering behaviour

The perceptual schemas that trigger behaviour may be quite primitive without adversely affecting the performance of the animal in its natural environment. The feeding response of toads illustrates this (Ridley 1995). Several criteria must be met before a toad will attack

an item: firstly the target must be moving. A toad kept in an environment where real food items are present, but all tethered will starve to death; the size of the target must also be correct. If the target is too large the toad will shy away to protect itself. The size criteria of these responses are dependent on the area of arc of the target in the toad's field of view. The toad seems to attack all moving objects below a certain size, while it will shy away from all those above it. Thus a simple relationship may be stated between area of arc of a moving target and the toads response to that target. Assuming there is no effect on the toad's motivation caused by the presence or absence of hunger, selection of feeding behaviour may be modelled by just two prepositional statements:

IF *target is moving* AND *target is smaller than threshold size* THEN *attack*

IF *target is moving* AND *target is bigger than threshold size* THEN *defend*

The simplicity of the toad's perceptual schema is evident from the way that it may be easily fooled in the laboratory. However even with this simplicity it is sufficient for the toad to live successfully in its natural environment. In the natural environment of the toad, food is mobile and within a certain size range. As there are few other items that correspond to these criteria within the environment of the toad, such simple selection criteria are sufficient.

It is worth noting that the feeding response of the toad is more than just a simple muscular action, the toad must orient to the target and strike. It is, however, a single motor schema, a sequence of actions that is triggered by a simple external condition.

2.7.2 Super normal release and vacuum behaviour

The simplicity of the triggering stimulus and generation of behaviour from primitives may further be illustrated by the existence of supernormal releasers and vacuum behaviours.

Supernormal release occurs when an artificial stimulus elicits response more effectively than its natural counterpart. Some birds will try to incubate an oversized artificial egg in

preference to their own (Grier and Burk 1992), an effect presumably caused by the simplicity of the "egg template" within the perceptual schema that triggers incubation behaviour. It is possible that if there were a greater number of "oversized egg" objects in the bird's natural environment, this supernormal release would be selected against by evolution, disappearing with time from the bird population.

When the correct stimulus is lacking a behavioural pattern may still be performed. This is a vacuum behaviour. Examples of such are insectivorous birds catching, killing and eating imaginary flying insects when deprived of food, rats trying to make nests using their own tails when suitable materials are not present, or dogs pretending to bury a bone on a carpeted floor, completing the task with nose tamping of the imaginary back fill! The common link between these vacuum behaviours is the presence of the motivation to perform the behaviour, but a lack of the appropriate stimulus. The motivational forces that trigger these vacuum behaviours may be overcome if other environmental circumstances favour different motivational states, or trigger other behaviours. Effectively the internal state of the animals must be such that there is an internal predisposition to the vacuum behaviour, and not to other behaviours. The state of the environment must be such that there are no strong external behaviour triggering conditions; the vacuum behaviour will then be performed.

The existence of vacuum behaviours has great implications for models of animal activity. In the examples given above, each of the behaviours is a complex sequence of activity, not merely a stereotyped motor action. The fact that these behaviours exist as single, though not necessarily indivisible units supports a view of behaviour as consisting of a collection of patterns of action that are defined by a script like structure from smaller atomic actions. These structures, like scripts, are not recipes or programmes for the construction of behaviour, but indications of what behavioural patterns need to be implemented at a time. Once a 'script' has been instantiated the sequence of events is continued, showing that a level of commitment to the behavioural pattern has been made. An animal that is engaged

on an action pattern has a tendency to continue this pattern. This commitment to a series of actions that make up a coherent pattern has important implications for achieving more abstract goals by use of directed sequences of simple behaviour.

2.8 Behavioural Sequences

Many aspects of animal behaviour are greatly augmented by the ability to sequence primitive sub-behaviour. A rather beautiful illustration of the power of behavioural sequencing is the spider's web. Observation suggests that these complex designs can be constructed by following a few simple rules. Whether the spider has some abstract concept of what a web is or a web template is not known, but by following simple, rules, a spider could produce an orb web without either.

An orb web comprises a frame, radial spokes and a catching spiral. Construction starts with the frame and spokes (Ridley 1995).

The spider first spins a thread and allows one end to blow in the breeze until it becomes entangled. When this happens she fixes the end she holds at her present position before starting a new thread also fixed at this point. She walks back along the first, wind blown thread (Figure 2.2a), until she reaches the position where it is entangled, where she attaches the new thread.

The spider then retraces her steps along this new thread. At approximately the centre she attaches another thread. This point will become the hub of the web. She then allows herself to fall whilst spinning (Figure 2.2b). When she reaches an object this latest thread is fixed producing a Y-shaped support for the web. The radial spokes are then added. To do so the spider returns to the hub, attaches a new thread and heads for the outside of the web. When this is reached she attaches the thread producing a loop. She then walks back down the loop (Figure 2.2c), attaches a new thread, and continues to walk to the opposite extremity of the web (Figure 2.2d), pulling the loop of thread into a spoke as if raising the

mast of a ship in a bottle. This procedure is repeated until about twenty spokes have been spun. The spacing of the spokes is uniform at approximately 15°. New spokes are always added into the largest void in the web.

When the spokes of the web are completed the spider spirals out from the centre of the web, attaching a thread. When the outside is reached she spirals back towards the centre, spinning the catch spiral. The first spiral acts as a guide and is cut on the return journey.

Webs are usually asymmetrical, the spider spinning a switchback pattern at the bottom of the web before starting the catch spiral proper.

It has been shown that spiders can easily resolve angles of 15° by testing the tension on the strands of a web using their front legs. With this ability the rules required to make the web mainly comprise of spinning loops and pulling them out until the correct angle is produced. This behaviour is iterated until the spokes of the web are suitably spaced. Triggering the spiralling behaviours then finishes the web.

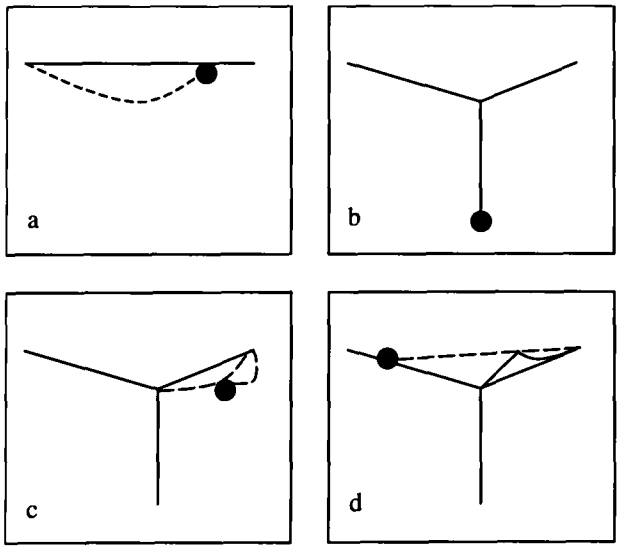


Figure 2.2 Orb web spinning procedure

The retrieval response of the greylag goose, described in section 2.6 'Behavioural Sub-units' is also, in fact a sequence of actions: Neck extension, followed by egg location and neck withdrawal with beak oscillation. Each of these actions is triggered by an individual event.

The response appears as a single behavioural unit as the conditions for starting each action is the successful completion of the previous. The final step, that of withdrawal, must continue until the neck is fully retracted, unless of course, a more urgent "survival" behaviour is triggered.

The power of sequencing has been postulated by Calvin (Calvin 1993) as a major reason for the huge power of the human mind. He argues that the desire to "hunt from a distance" produced mental adaptations suitable for the production of a timed sequencing of ballistic actions, as is required to produce an accurate throw. Development of this ability in turn enabled humans to accurately sequence other rapid actions, ultimately leading to the development of language. Although the sequences discussed operate at a lower level of abstraction than those involved in the spinning of an orb web, being concerned with direct control of muscular activity, they illustrate the importance of sequences and the complexity of behaviour that may be synthesised by sequencing simple behaviour primitives.

An animal may exploit the advantages of behavioural sequences with explicit knowledge of the sequence itself. This is especially true in simple cases and at low levels of abstraction. Connell in his book on the colony architecture (Connell 1990) uses as an illustration the case of a simple tidal snail. The food on which the snail depends occurs just above water level on rocks. A few simple behaviours are required to produce the series of actions that take the animal from below water level to its feeding ground. None of these behaviours require, as a triggering stimulus that the previous behaviour is successfully completed. All rely totally on the external environment for instantiation. As a behaviour is performed it has the effect of changing the environment in which the snail finds itself. The new environmental conditions then select the next behaviour. Effectively the behaviours communicate their successful completion through the animal's environment. Connell expands this concept to the creation of a machine "Herbert". Herbert roams offices collecting empty soft drink cans and depositing them at a location determined by the architecture of his controller. Herbert is more fully described in chapter 3

The emergent nature of the behavioural sequences in the colony architecture is very attractive, but suffers scaling problems. In situations where behaviour is more sensitive to environmental conditions the use of the environment as the only source of sequencing information becomes more difficult. This sensitivity is a product of there being more possible behaviours to pursue. Increased environmental information prevents this edging towards divergence, but causes a huge increase in the computational load. To increase the likelihood that a sequence is completed once commenced, a level of commitment to the component behaviours of that sequence is required. Again, commitment does not require that the sequence be known explicitly. A device or animal will show commitment to a task if each of the behaviours in a sequence support the triggering of those that follow it, while reducing the chance that those behaviours which conflict with the sequence are inhibited. Intuition suggests that a lowering of the activation threshold for the active behaviour pattern is likely, although inhibition between agents exists (Minsky 1988). A combination of the two is most likely to accurately reflect the natural world, behaviours that serve incompatible goals inhibiting each other while the activation threshold of the presently active behaviour and those that follow it in the sequence are lowered. A lowering of the activation threshold of a behaviour pattern is equivalent to an increase in the level of motivation directed towards that behaviour.

2.8.1 Back propagation of goals

In some cases an animal may be motivated to perform an action but is prevented by environmental conditions. If this task is important there is a requirement that the animal manipulates the environment such that the behaviour may be performed. Here the sequence must be worked through in reverse order. Such reversals are required when; for instance the animal needs to drink but is far from a water supply. The need for water will provide the required motivation to drink, but environmental conditions may prevent this behaviour being enacted. The lack of a water supply, but presence of the motivation to drink must activate those behaviours that cause the animal to arrive at a source of drinking

water. Maes has produced a simulation model of an artificial agent that operates using these chains of activation between behaviours (Maes 1990). Such a technique provides a powerful method of generating goal directed behaviour.

2.9 Quantitative controls on behaviour

2.9.1 Kinesis

In addition to simple threshold activation of behaviour, it may be necessary to provide quantitative controls to that behaviour. Some of these controls are incredibly simple. The woodlouse has a preference to damp conditions. To increase its level of comfort the woodlouse should linger in damp areas, but move rapidly in areas that are dry. This is a case of kinesis (undirected apparently random movement). Without additional motivational factors a woodlouse will move if dry, and remain stationary in the damp. The dampness of the environment modifies the effect of movement behaviour, such that the louse will move more quickly when the environment is too dry for its taste. Braitenberg has investigated the nature of these simple behavioural controls. The type of kinesis exhibited by the woodlouse is also exhibited by Braitenberg's "type 1 Vehicle" (Braitenberg 1984) as described in chapter 3

2.9.2 Taxis

Cricket mating requires that female cricket moves towards the male as he sings. The sound is produced by the male cricket moving his wings across his body, producing a short fixed duration bursts of sound, at a frequency dependent on species. The female achieves the male's location simply by turning towards the sound while walking. While the "find mate" behaviour is active, the cricket simply walks towards the loudest noise of the correct form (Webb 1994). This is an example of positive phono-taxis. The animal detects the source of a stimulus and responds appropriately, turning an angle dependent on the apparent source of the noise. In the case of the cricket this is achieved using a pair of ears to detect the

source, other animals may detect the source by repeated sampling of the environment. The same mechanism may be used to direct the movement of an animal dependent on other detectable properties of the environment, provided they exhibit an intensity gradient, for instance a diffuse chemical source (chemo-taxis). There are two major mechanisms for achieving taxis. An array of sensors may be used to detect relative intensity, as is the case for the cricket, or a multiple sampling behaviour may be employed, where the creature stops intermittently and checks the strength of the stimulus to each side. The creature then turns towards the direct in which the strongest measurement was made (Figure 2.3). An analogue of the multiple sampling technique forms the basis a gradient decent optimisation method.

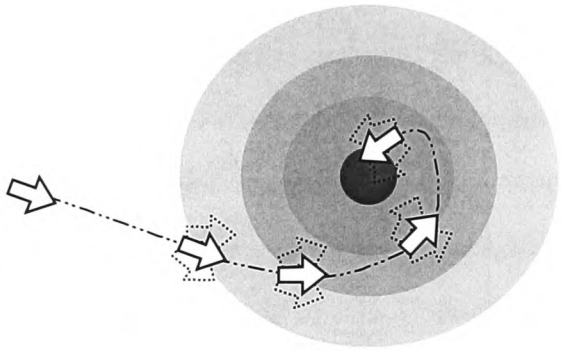


Figure 2.3 Multiple sampling chemo-taxis

2.10 The effect of morphology

The morphology of an animal has an effect on the behaviour that animal may exhibit. This is the result of the body and "brain" of the animal developing in tandem. It also has implications for the manner in which the controllers of machines are designed. The morphological affects on behaviour are more subtle than the ability of birds to fly and the differences between land and marine mammals. The phono-taxis exhibited by crickets while searching for mates is dependent on the phase cancelling effect at certain frequencies of their ears (Lund et al., 1997). If the ear of the cricket did not have that morphology the accurate resolution of the songs location would not be possible. The production of the male song is such that the frequency and repetition rate of the bursts fit the phase cancelling

in the female's ears. Different species must sing a different song in order to be heard due to the different sizes of ears exhibited by the different species.

The close coupling of the "mechanical" aspects of the animal and its behaviour may be further illustrated by the manner in which the sensor modes of an animal may become tuned to the tasks on which the animal depends. The eye of a frog is structured such that it contains a "bug detector", ideal for the identification of the animals food source (Lettvin et al., 1959).

As the absolute abilities of an animal are dependent on its morphology it is to be expected that the behaviour of an animal is also dependent on morphological factors, though it is almost impossible to prove many of these relationships by observation. Clearly if the morphological constraints of an animal prevent certain actions from being made then the animal will never be able to perform that action. The power of morphological effects however is in the exploitation of abilities that do not appear immediately to the human observer. The phono-taxis of the cricket is one example of this, there are many others.

2.11 Summary and conclusions

From the work above it may be seen that the major features of animal behaviour may be summarised as:

- Behaviour is distributed throughout the animal's nervous system. Much behaviour is produced using simple computational units close to the relevant muscles or sense organs.
- Behaviour is the product of the interaction of simple behaviour sub units. Each sub unit only uses the subset of the animal's sensorium that is necessary for the correct operation of that behaviour.
- Behavioural mechanisms are tightly coupled to the features of the environment in which the animal operates.

- Behaviour selection depends on both the environment conditions and the internal state of the animal.
- Sequencing simple action patterns can create complex behaviour and result in the achievement of abstract goals, without explicit definition of those goals.
- Behaviour is tightly coupled to the morphology of the animal.
- Short cuts, simplifications and novel solutions to the implementation of behaviours can be widely observed.
- The use of a modular distributed control system improves the robustness of animal behaviour both to physical trauma, and to information errors.

The success of animals is an illustration of the suitability of this approach to designing intelligent systems for real-world applications. Even the most primitive animals solve many of the problems associated with creating mobile autonomous systems. As the best example of existent intelligent systems, the use of the natural world as a model for the design of machines would appear to be appropriate. Adopting a biologically analogous design methodology for a machine will help to produce machines with improved capabilities, but it must be remembered that the morphology, sensor characteristics and technologies available in the construction of machines differ vastly from those found in nature. Rather than slavishly copying natural systems the designer should instead adopt the same general techniques, while considering the different properties of the materials available. This technique may generally be thought of as dividing the complexities of behaviour down into a collection of simple sub behaviours, the interaction of which synthesises the required system behaviour. Each schema should be designed so it takes maximum advantage of any features of the environment that simplify the computation required in achieving that behaviour. Communication between behaviours may be achieved simply by using the effect each behaviour has on the environment.

Many of the novel and interesting solutions obtained by evolution in the production of animal systems cannot be exploited by mechanical systems, due to the constraints of

technology. A balance must also be struck between maintaining a concept of elegant methodology and widespread dependence of arbitrary features of the environment.

Biological models of behaviour have been used as a template or inspiration for mechanical systems since the mid 1980s. The construction of these systems has been motivated by a desire to achieve rapid, reliable and robust control in difficult environments. There is also undoubtedly a cultural bias towards producing machines that operate in a zoomorphic manner. The next chapter (chapter 3: "Behavioural Control") investigates some of the major features of the attempt to create controllers after a biological model.

2.12 References

Agre P.E and Chapman D. 1990. What Are Plans For? Robotics and Autonomous Systems 6: 17-34.

Arkin R.C. 1998. Behaviour Based Robotics. Cambridge MA: MIT Press.

Arkin R. C. 1989. Motor Schema- Based Mobile Robot Navigation. International Journal of Robotics and Automation 81, no. 4: 92-112.

Bizzi E., Mussa-Ivaldi F.A., and Giszter S. 1991. Computations Underlying the Execution of Movement: a Biological Perspective. Science 253, pp 287-91.

Braitenberg, Valentino. 1984. Vehicles - Experiments in Synthetic Psychology. First ed. Bradford Books. Cambridge, Massachusetts: MIT Press.

Calvin W. H. 1993. "The Unitary Hypothesis: A Common Neural Cirsuitry for Novel Manipulations, Language, Plan-Ahead and Throwing? Tools, Language and Cognition in Human Evolution. ed., GIBSON K.R. and INGOLD T., pp 230-250. Cambridge University Press.

Chulder E. H. Untitled URL:<weber.washington.edu/~chulder/chreflex.html>. Undated

(Accessed August 1996).

Connell J.H. 1990. Minimalist Mobile Robotics: A Colony Architecture for an Artificial Creature. London: Academic Press Ltd.

Dawkins R. 1989. The Selfish Gene. New Edition. Oxford: Oxford university press.

Dock J untitled.

URL:<cogsci.soton.ac.uk/~harnad/hypermail/explaining.mind/0024.html> .28 Feb 1996 (Accessed28 March 1998).

Grier J.W., and Burk T. 1992. Biology of Animal Behaviour. St. Louis U.S.A.: Mosby Year Book.

Ison, J. Reflexive behaviour Research Lab

URL:<www.bcs.rochester.edu/bcs/research/ison_lab.html>. Oct 2nd 1995

Jeffrey, L. An Introduction to the Blink Reflex

URL:<www.psy.uwa.edu.au/user/linda/biglab/blink_1.htm>.12/12/95(Accessed June 1987).

Lambrinos D., Maris M., et al. 1998. Navigating With a Polarised Light Compass. Self-Learning Robots II: Bio-Robotics, pp 7-1, 7/4London: IEE.

Lettvin J.Y, Maturana H.R., et al. 1959. What the Frog's Eye Tells the Frogs Brain. Proceedings of the IRE , pp 1940-1951.

Lund H.H., Webb B., and Hallam J. 1997. A Robot Attracted to the Cricket Species GRYLLUS BIMACULATUS. Proc Ecal '97 Fourth European Conference on Artificial Life, Editors Husbands P., and Harvey I. MIT Press.

Maes P. 1990. Situated Agents Can Have Goals. Robotics and Autonomous Systems 6: pp 49-70.

Malcolm C. and Smithers T. 1990. Symbol Grounding via Hybrid Architecture in an Autonomous Assembly System. Robotics and Autonomous Systems 6: pp 123-44.

McNally J. 1998. -. Fortean Times , no. 109: pp 18-19.

Minsky M. 1988. The Society of Mind. New York: Touchstone (Simon and Schuster).

Mitchell R., Keating D.A , and Kambhampati C. 1994. Learning Strategies for a Simple Mobile Insect. Proc. IEE International Conference on Control.

Ridley M. 1995. Animal Behaviour: A Concise Introduction. Oxford: Blackwell Scientific.

Scutt T. 1994. The Five Neuron Trick: Using Classical Conditioning to Learn to Seek Light. Proc. 3rd Int Conf. on Simulation of Adaptive Behaviour, Editors Cliff D., Husbands P., Meyer J.A., and Wilson S.W., pp 364-70. "From Animals to Animats" , no. 3. Cambridge USA: MIT Press.

Webb B. 1996. A Cricket Robot. Scientific American : pp 62-67.

Webb B. 1994. Robotic Experiments in Cricket Phonotaxis. From Animals to Animats 3, Editors Cliff D., Husbands P., Meyer J.A., and Wilson S., pp 45-54. Proc 3rd International Conference on Simulation of Adaptive Behaviour, MIT Press.

3: BEHAVIOURAL CONTROL

In the previous chapter animal behaviour was investigated. Many of the properties described are desirable in machines, such as robustness and the ability to generalise and rapid response to important events. This chapter investigates some of the attempts that have been made to generate this sort of behaviour through the use biologically analogous mechanisms. The are grouped under the generic title "behavioural control".

Behavioural control has become an important technique for the production of intelligent systems. Even though the term behavioural control is widely used, a formal definition seems to be avoided. Below a review of texts that employ the technique and those related to it develops a definition of behavioural control. This definition is based on the major properties and features of the technique, which are contrasted to both more traditional methodologies and those that are more closely related.

In many respects the change of paradigm from traditional AI to behavioural control mirrors the development of object-oriented analysis, design and programming techniques. The object model forced a change in the method of decomposition of systems to a categorical decomposition that more closely matches a normal human point of view (Coad and Yourdon 1990).

The similarities and differences between behavioural control and animal behaviour are investigated, as are the difficulties associated with implementing this technique.

3.1 Introduction

Many techniques are referred to as behavioural, but bare little superficial commonality. The reason for this is most likely the variations in the meaning of the term behaviour itself (as discussed in Chapter 1, section 1.5.1). There are however similarities between all of the techniques. Fundamentally all behavioural or reactive control schemes adopt a decomposition of the problem space into more easily understood problems, to which solutions may be crafted individually. Each decomposite is a "real world", as opposed to abstract internal processing task. Each of these decomposite behaviours may be as simple as a proportional transfer from a sensor to an actuator, or as complex an abstraction as *"Search-for-Door"*. A behaviour would not normally be related to a task such as "*extract- edges*" or "*calculate...*"

As described in the previous chapter the exact nature of the processes involved in the production of behaviour may be hard to determine by observation of a machine or animal in action. Likewise it may be difficult to determine the nature of modules that should be combined to create behaviour capable of achieving a goal or task. Behavioural controllers synthesise action from a collection of simple competencies working in parallel. Such a system of interacting, potentially non-linear functions will produce complex behaviour (Lewin 1993). Further complexity will be added when interaction with the environment is considered.

As a simplification of behavioural control, reactive control addresses the complexity issue by reducing each element of the controller to a simple mapping between sensors and actuators. Even with such a simple system there is still scope for interesting behaviour. It is the simplicity of reactive controllers that makes them so attractive.

3.2 Traditional techniques

Traditional control techniques, often referred to as "deliberative" in the behavioural control literature, are based around a "stop, sense, think, act" cycle. The agent uses its sensors to take a snapshot of the world at a single moment; an impression of the state of the environment. This impression is then applied to a global world model. The model is used to determine the actions to be made. Finally the action is performed. A serial decomposition of the task of used, feeding information forward to the next process. Because of this deliberative systems tend to be though of as symbol processing, abstract systems. The first tasks are obtaining information from the agent's sensors. This is then used to develop a symbolic impression of the state of the environment. This impression is manipulated to determine action.

Throughout this work allusions are made to the analogy between behavioural control and Object Oriented programming techniques. Traditional, deliberative, control architectures are, in comparison, analogous to procedural programmes.

Problems occur when the data obtained from varied sensors is in conflict. "Sensor fusion" techniques are used to combine the data obtained from the sensors to try to obtain a coherent picture of the world. Data errors cause the view of the world to vary from reality, meaning that the wrong actions may be performed. The dependence on a number of sources of information and the act of combining possibly contradictory data exacerbate this. The number of parameters that may need to be considered can also lead to a proliferation in the number of specific cases that need to be considered, this can damage the generalisation capabilities of the controller.

3.3 Reactive versus behavioural control

It is both convenient and conventional to identify to main classes of controller that may be term behavioural in the broader sense. These are the reactive schemes, and behavioural architectures proper.

A reactive scheme represents a system where the motor action of the vehicle depends only on the condition of the sensed variables. There is a direct mapping between sensors and actuators. These schemes are often referred to in the literature as reflexive due to the similarity between such systems and reflex behaviours observed in animals.

The more flexible and complex behavioural techniques are a super-set of reactive control. A behavioural controller still represents a mapping between inputs and actuators, but has the ability to store state and representations of the environment (Mataric 1997). A behavioural controller may also use representations at a far higher level of abstraction.

In truth reactive and behavioural control strategies exist on a continuum, and may be mixed. A direct mapping may form the basis of a more complex hierarchical scheme, or be used alongside more complex schemas in a behavioural controller. In this way a behavioural control scheme may employ reactive elements.

As more complex tasks are attempted some state will inevitably need to be stored, for this behavioural controllers are used in preference. However, by reducing the amount of information stored to a minimum and not permitting the use of a central model a compromise may be made.

3.4 The subsumption architecture

Brooks is often considered to be the father of behavioural control. His widely cited 1986 paper (Brooks 1986) introduced the subsumption architecture.

A simple aim forms the basis of this architecture; the simplification of the design of a controller by division of a complex control problem into functional sections that may more easily be achieved. This may not appear to be a vast departure from the traditional deliberative approach where a task is divided into more easily managed stages. However in the subsumption architecture these simplifications are semiautonomous sub-controllers. They represent a subset of the behaviour of the machine, rather than a subset of the processing requirement of the controller. Brooks refers to these simplified controller elements as "competencies". Using this technique a controller may be constructed in an incremental manner. Implementation starts at the "lower" less abstracted competence levels. Each competence is considered in turn, with a simple control scheme being introduced such that the requirements of that competence are met. Design then proceeds by selecting the next competence, and implementing this. The previous elements of the controller are left in place, and may indeed be used to provide signals for the higher levels. Higher levels may "replace" signals produced lower with their own, inhibiting the action of the lower competence. Following the implementation of each competence the machine should be functional, exhibiting behaviour augmented by the additions that have been made. The behaviour of the system is the synthesis of the action of these individual competencies. The addition of layers continues until the machine exhibits the behaviour required in its specification. The diagram below (Figure 3.1) shows the structure of a subsumption controller. Illustrating the layer nature of the controller and the manner in which subsumption itself is achieved. The first competence to be implemented is "Avoid", preventing the machine from striking objects it detects within its environment. Once this behaviour is implemented satisfactorily a wander competence has been added, which can "steal" control of the machines motors to produce a random movement. Subsequently behaviours at higher levels of abstraction are added, however the original avoid behaviour is still operative and prevents the machine colliding with objects (unless forced otherwise by "Dock").

The adoption within the subsumption architecture by Brooks of an incremental design process is one of the most important properties of the technique. Although such design methodologies may appear to be rather unsatisfactory due to a rapid growth of complexity as more difficult tasks are attempted, incremental design has been enshrined in the procedure for systems analysis and design using the object model (Booch 1994)(Coad and Yourdon 1990).

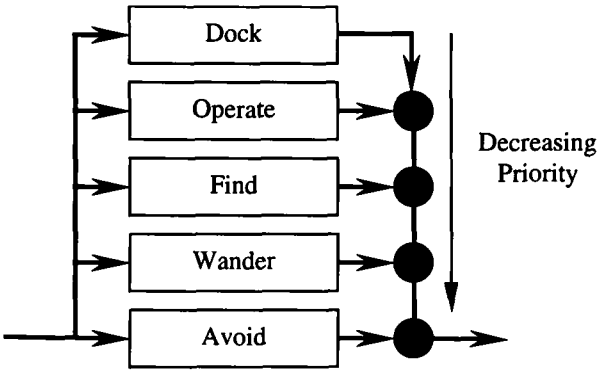


Figure 3.1 The subsumption architecture (after Brooks)

The subsumption architecture is often referred to as being a parallel combination of computational elements. The implication is that the elements are treated equally, this is not true. Elements have various levels of abstraction. At the bottom are the direct control competencies, which perform the most basic and general aspects of control. Above this are more specialised behavioural elements, which rely on the lower levels for integral parts of themselves. As such a subsumption controller is more hierarchical in structure than traditional deliberative techniques, although intuitively the opposite would appear to be the case. Deliberative controllers are perceived as having a highly hierarchical structure, the level of abstraction employed by individual elements increasing with distance from sensors or motors. However, the flow of information is very linear, passing through each of the elements in turn. There is no indication of a vertical structure, which is merely an edifice of the way that each element relies on the more highly abstracted output of those that proceed them. Indeed the purpose of many of the stages is to increase the degree of abstraction to a level that is suitable for the major planning and modelling sections of the controller, or for

the reduction of the high levels of abstraction required for symbolic processing to appropriate motor signals.

The subsumption architecture conforms to the definition of a behavioural controller as it does not rely on a large internal model of the environment. As each of the divisions of the control task addresses a small subset of the problem, the amount of sensor information required for that section is also reduced. Mappings between reduced sensor information and motor action are adequate. There is nowhere for the construction of a global model of the environment, from which decisions can be made. As with all behavioural control architectures small models may be stored in the competence implementations themselves, containing just enough information for the module to perform its allotted task.

By not relying heavily on a model of the environment, processing requirements are reduced drastically. Algorithms that manipulate large-scale environmental models, such as path planners are often computationally intensive. Maintenance of a large model is also a problem, sensor data must be fused to determine the environmental state, and any errors in these data will be propagated across the model. Any errors in the model will be passed to the controller, producing erroneous output.

In subsumption, and general behavioural architecture, the use of limited sensor data by each module helps reduce this problem, only those elements using the error ridden data will be effected, thus system robustness is maintained.

These advantages are also common to other behavioural control schemes.

3.4.1 Timing

Models using, deliberative control techniques are fundamentally discrete time systems due to the segmentation of the control task into a sequence of stages. Measurement of the environment is followed by construction of some type of global view of the world, taking a snapshot of the world at that particular time. This is applied to the decision-making

mechanism, which finally computes an action that is appropriate. In the subsumption architecture, and behavioural schemes in general, different elements of the controller may work at different rates, dependent on the speed at which the sensor used by that element produces meaningful output.

The atomic units from which the subsumption architecture is created are augmented finite state machines; state machines with an added timing element (Brooks 1986). The addition of timing elements allows each model to temporarily store state.

The departure between the subsumption architecture and those techniques employed more traditionally can be summarised simply:

- Division of task into parallel as opposed to sequentially active subtasks
- Minimal use of models of the environment.
- Sensor information available to each subtask limited to what is required.
- System created by task fusion.

3.4.2 Possibilities

Moving away from the traditional view of robots as large, complex, perambulatory calculating machines has lead to some intriguing possibilities. The lower cost and increase in robustness means that the type of mission for which a robot may be employed is also subject to change. There is also the possibility of using a collection of simple machines to achieve a complex task. This may be extended to the exploration of space (Brooks and Flynn, 1989)(Klarer 1997). Though it is initially hard to imagine that a large contingent of incredibly simple machines can achieve the type of science usually required from a space mission, the advantages of low-cost and robustness make the design of single experiment space missions seem more likely.

Sojourner, the mobile robot taken to Mars as part of the Mars Pathfinder Project was not heavily laden with experimental equipment, though proved to be very successful, and

popular. Undoubtedly the major success of the Sojourner was its ability to capture the imagination of a world population no longer as excited by space exploration as it had been when the race for the moon was at its height. However, it did illustrate that good science could be achieved using a small vehicle and (relatively) simple experimental apparatus providing that the goals chosen are appropriate (Matijevic and Shirley 1997).

3.5 Braitenberg's vehicles

Many of the properties that are present in the subsumption architecture can be traced to ideas that grew from ethology and neurological studies. Braitenberg's 1984 book "Vehicles" (Braitenberg 1984) proposes that systems possessing quite complex abilities may be generated by the incremental addition of functionality to a simple substrate. Each of the new functions that are added uses those that have gone before in order to realise the completion of its task.

One of the powerful concepts articulated in the book is the simplicity of each of the augmentations made between vehicles. None of the changes represent a large leap in the types of hardware or processing used; though each produces a significant increase in the complexity of behaviour that the system may achieve.

Systems, as outlined in "Vehicles", grow incrementally from a machine comprising a single sensor and actuator connected by a transfer function. This vehicle, "No. 1" exhibits kinesis. Behaviour of this type may be observed in simple animals such as the woodlouse (Chapter 2). connecting two of these systems, either in parallel, or cross-coupled creates Vehicle 2. Vehicle 2 may exhibit either "fear" or "aggression"; i.e., it is attracted to, or avoids sources that excite its sensors. It is worth noting that the terms "Fear" and "aggression" employed here are based on anthropomorphism of the type of response exhibited by the vehicle, not on any emotional responses that the words fear and aggression would signify when taken in a human context. The thought experiments in the book amply illustrate the emergent nature of the behaviour, when it is the product of the interaction of simple processing units.

Figure 3.2 shows the first three types of vehicles, and illustrates the behaviour expected from them. Vehicles of type 2 possess stimulatory links from sensors to motors. A wheel will spin faster when the sensor attached is activated more. The differences in behaviour between vehicles of type 2 and type 3 are due to the links between the sensors and motors of type three being inhibitory. As the activation of a sensor increases the speed of the attached wheel is reduced.

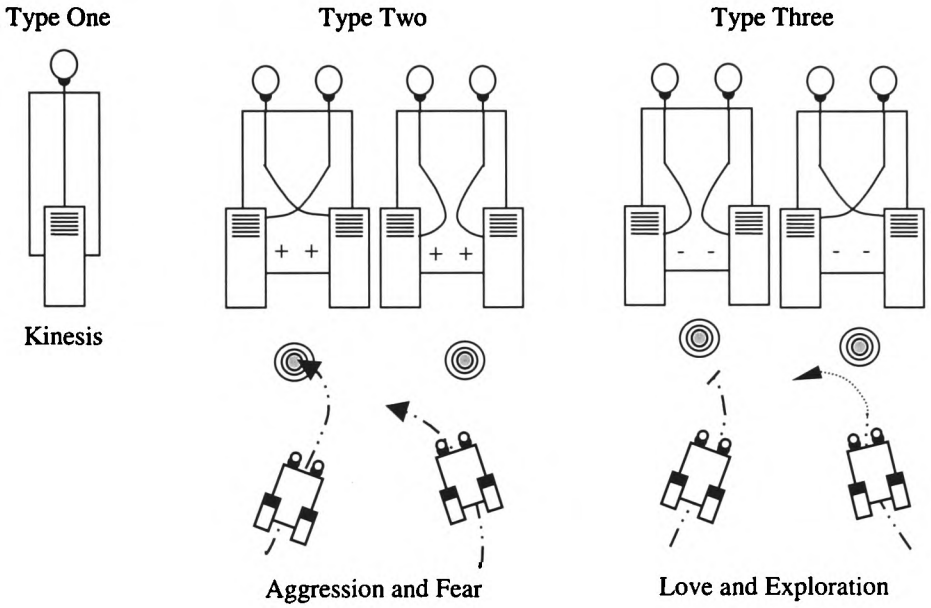


Figure 3.2. "Vehicles" (after Britenberg)

The addition of non-linear activation functions (vehicle No.4) to the basic type 2 and 3 vehicles allows far more complex behaviour to develop such as "decision making" and less violent responses to stimuli.

An important principle that is outlined in the early chapters of the book is that of "Uphill analysis and downhill invention". Braitenberg coins this term to illustrate how easy it is to produce interesting behaviour through the interaction of simple elements, but how hard it is to analyse behaviour produced in this way, a problem that still haunts the community of behaviourally controlled robot builders. A similar problem exists in colonies of insects such

as bees, where the community can often be thought of as a single organism. The collective behaviour of the hive is the product of the actions of every individual (Goodwin 1998).

The behaviour of the colony is grossly different from that observed in the individual members. It can be explained in terms of individual behaviour after careful observation has isolated both, but cannot be extrapolated from knowledge of the individual behaviour alone. This problem exists in any system where behaviour is determined by the interaction of a group of simple rules (Lewin 1993) "The Society of Mind" (Minsky 1988) uses this treatment of mental processes to attempt to explain the workings of human intelligence.

The next major step in the evolution of the vehicles is the addition of threshold transfer functions (vehicle No. 5), which has a dramatic effect. The thresholds used by Braitenberg have some special features. There is a propagation delay across the device. Inputs may or may not be inhibitory, and the designer may set the threshold value. That it is hard to imagine any real device not being subject to a propagation time of finite value helps reinforce the view that similar elements may exist in nature. It is no mistake that the description of the threshold as made by Braitenberg conforms to a simplified model of a (natural) neurone (Braitenberg 1984)(Patterson. 1996).

Using such threshold elements makes calculation, logic and simple memories possible. Though this is not a large jump in structure, it causes a large increase in the possible repertoire of the machine. Indeed, the only change in the design between vehicles 4 and 5 is the severance of the direct connections between the sensors and actuators by the insertion of additional simple computational elements. Chains and loops of transfer functions are allowed to exist in the vehicle controller. Simple extensions to a simple

machine have allowed a purely reactive system to store state, representation and perform symbolic processing[1].

By following an incremental design methodology Braitenberg's thought experiments finally create vehicles which he claims exhibits human traits such as egotism and optimism. It would be easy to discount the vehicles as a 'flight of fancy', but Braitenberg devotes the last fourteen chapters of his book to biological notes that support his opinions. Subsequent to the publication of his book, many authors have constructed the simpler vehicles, and observed the confirmation of Braitenberg's predictions.

"Vehicles" is intended as a simple, illustrative model of the processes that have generated human psychology; Natural systems that have been constructed and tuned by evolution. Adjustments may be made to an already existing element of the controller, as well as addition of further elements. This technique is partially supported in subsumption, though not in all behavioural control techniques. There are good engineering reasons for this disparity. In natural systems, evolution has vast quantities of time and materials to work with. Any mistakes may be easily absorbed into the large populations involved. Changes between generation are very small reducing the detrimental effects of errors on the population, which is only possible due to the large time scales involved.

When engineering systems in an incremental manner, as is implied by use of behavioural control techniques, time and materials tend to be less abundant. For this reason if a sub-section of the system works, it may be advantageous to leave it in place inviolate, additional competencies being achieved solely by the addition of new elements. The lack of interference with working elements to the system also implies that there is a far more

---

[1] The output of a threshold device is a classification of it inputs. The levels on the threshold output thus become symbolic representations of environmental classification. Storing symbols generated by the environment is directly equivalent to storing a representation of that environment.

structured approach to the assembly of behavioural controllers. Connections are possible only at inputs and outputs of each element. This is a more intellectually appealing methodology as it maintains the integrity of elements, and helps encapsulate behaviour in definite areas of the controller.

Encapsulation of behaviour as black boxes brings additional advantages; equivalent to those that are brought by the use of object oriented design methods to software engineering. These include code security, testability, reusability and ease of maintenance.

A variant of the subsumption architecture that exhibits these properties is the colony architecture.

3.6 The colony architecture

The subsumption architecture may be characterised as a layering of competencies, higher layers having the ability to influence the internal workings of layers that exist below, both of these aspects are seen as weaknesses by Connell (Connell 1987). His colony architecture, although very similar to the subsumption architecture, has no fixed layering of behaviours. Elements that represent more general behaviour, may be added to increase the behavioural repertoire "below" existing solutions which operate under more specific conditions. Each of the "modules" of the colony architecture is a "black box", no part of the control system may see, or affect the internal operation of a module.

3.6.1 Operation

Each of the behavioural modules within the colony architecture comprises a transfer function and an activation function. The activation function is a threshold device. When the threshold is exceeded the output of the transfer function is activated. The output of a module is thus a function of the input, switched in a manner analogous to motivation as described in chapter 2.

The modules also have the ability to temporarily record state using monostable multivibrators, which extend the output condition of the module in time. This is important as it allows a machine controlled by a colony architecture implementation to commit to a course of action. This ability also has a biological analogue, as was illustrated in chapter 2.

3.6.2 Control

With both the colony and subsumption architectures there are two methods by which more specific behaviours can rest control from those that are more general; inhibition and suppression. An inhibitory connection between two modules (or layers of the subsumption architecture) merely uses the controlling output to disconnect the other. Suppression is more complicated. Here the output of a behavioural element is used to replace the output of the behaviour with a lower priority. The use of continuous control signals between behaviours makes this possible.

When the environmental situation encountered does not conform to the activation criteria of any of the high-level elements, control will be achieved by those elements whose activation criteria are met. These will naturally be those of a more general nature. These properties give controllers of this type a high level of robustness and the ability to generalise. The controller can therefore respond to the more general features of a novel situation.

Although the colony architecture may appear to be less thoroughly regulated than subsumption, this is not the case. Connell uses a set of rules limiting the way that each of the modules may interact. These include preventing signals within each of the modules from being observed or replaced and making each of the modules an independent atomic structure. This imposes a higher degree of order than in subsumption, where convenience during design is given more weight. Adoption of a fixed interface to each of the elements of the colony also has the advantage that the nature of that element is hidden. There is no reason why member elements of the colony cannot have wildly different internal structures,

using a different set of representations and signals, providing the interface protocol is respected.

3.7 Motor schema

An alternative approach to the simple behavioural model provided by both the subsumption and colony architectures is motor schema theory (Arkin 1998). Schema theory has been used successfully for the study of behaviour in animals for many years. It is closely related to the dual concepts of innate releaser mechanisms (IRMs) and fixed action patterns (FAPs) used to explain behaviour by ethologists (Grier and Burk 1992). IRMs and FAPs form exclusive pairs, each IRM being closely related to its associated pattern of action. An IRM contains the conditions that are required to activate a FAP and together the two elements form a coherent whole.

A schema is an independent processing element. Schemas operate concurrently, each containing knowledge of how to react to a set of environmental conditions, and the mechanism by which this action may be performed. Like simple reflex actions in animals, the strength of the response is proportional to the strength of the stimulus that produces that response.

There are multiplex subtle differences between motor schema and other behavioural control techniques that influence the system design process. The most important difference is in the way outputs are produced by the schemas. All outputs are of the same format. In the case of Arkin's work on navigation, all schemas produce a vector field. (Arkin, 1992).

The truly parallel nature of motor schemas leads to a problem not encountered in the subsumption and colony architectures: what should the actuators do when confronted with conflicting signals received from motor schemas? Placing schemas at a higher level of abstraction than the competencies used in both the subsumption and colony architectures Arkin (Arkin 1989) solves this problem. An additional computational stage is used to

calculate actuator demands from all schema outputs. The fields of all the schemas are then combined by superposition. A control stage is then required to produce motor control signals, which is a matter of determining the required signals to move the vehicle along this vector.

3.8 Super-simple architectures

BEAM (Biology, Electronics Æsthetics, Mechanics) robotics, pioneered by Tilden (Tilden 1993) is a field of research where the motivating concept of machines as life analogues is taken to an extreme. The simple nature of the machines constructed in this way has lead to some intriguing insights into the system model of machines. Much of the interest in BEAM robotics stems from the emergent nature of the behaviour these systems exhibit.

The primary tenet of BEAM robotics are three rules of robotics proposed by Tilden (Tilden and Hasslacher 1995). These roughly state that the primary objective of the machine is survival:

1. A Robot must protect its existence at all costs.
2. A Robot must obtain and maintain access to a power source.
3. A Robot must continually search for better power sources.

Although these rules help to create interesting behaviour and enable the machines to exhibit a high level of robustness some issues must be addressed.

A machine that follows Tilden's three rules will not perform any useful task unless engineered carefully so that completion of the task becomes a side effect of another of the machine's actions. The emergent nature of the behaviour can make this hard to predict. A possible application of BEAM robots is the clearing of minefields which may be achieved by simply allowing machines to traverse an area in a random walk, triggering the mines as they are passed over. Very simple control strategies can be used on very simple machines to achieve this. BEAM robots have already been tested in this sort of application (Kliener

1998). Even this simple task can raise some issues for the design of BEAM robots. Tasks that are more structured and difficult will require a great deal of thought if they are to be solved in this manner.

3.8.1 Out of control

The rules of BEAM robotics are very survival-centric. This is in direct conflict with the three rules of robotics as stated by Asimov, (Appendix 2: Asimov's three laws) which are structured to protect humans. It may be that Tilden crafted his rules purposely to emphasise the difference between the requirements of artificial animals and a plot device used to prevent a series of novels descending into reworking of "Frankenstein". It is to be noted that stories of a human creation falling into conflict with either its creator, or the human race in general have shown remarkable longevity. The dangers of machines not sharing our goals may be exaggerated by the archetypal nature of the motif. The Golem, Frankenstein (Shelly) and the Terminator all exhibit the same basic plot. That the danger of a loss of control has been written of in works of fiction for so long does not mean that it can be ignored. It may be assumed that Asimov created his three laws of robotics so that his robot stories could investigate the robotic minds without slipping into a classic motif. However, many authors (Warwick, 1997)(Ashby 1948) have pointed out that machines with the ability to learn are likely to, if of sufficient complexity, develop their own objectives and values. It is unreasonable to assume that the values exhibited by machines would match those of a human, being due to the different experiences and requirement a machine is likely to have. The same authors have pointed out that conflict may be inevitable.

3.9 Action and intelligence

The third of Tilden's rules implies that a robot must be able to move freely through the environment. In contrast, a welding machine that is fixed firmly in place beside a production line has many advantages over a mobile welding platform for most industrial applications (including stability, easy resourcing and the availability of datum points against

which to control). Plants and some animals survive more than adequately without the ability to move. However, the level of intelligence required, or obtainable by a system is likely to be related to the complexities of un-tethered movement (Calvin 1993). An illustration of this is the sea squirt. During its larval stage the sea squirt swims freely through the oceans. When it reaches maturity it fixes itself firmly to a rock and filter feeds. It is a very successful animal that forsakes the ability to move. As it requires less computational power in its mature life, the sea squirt resorbs a large amount of it nervous system after tethering (Dennett 1993). This reduction in computational ability is presumably advantageous as it reduces the amount of "power" required by the animal. It also helps reinforce the view that the level of computation desirable is directly related to the action and morphology of the system. System design requires a balance between sensorium, mechanical capability and computational ability (Pfeifer and Verschure 1995).

3.9.1 Tilden's lessons

Two important ideas are embodied in Tilden's rules:

1. For a robot to perform a task, it must survive long enough for that task to be completed.
2. A machine that works without supervision is far more helpful than a machine that is merely an extension of a human operator. For this level of autonomy a machine must be capable of "looking after itself", this includes procurement at the appropriate time of a source of energy, and more subtly to schedule tasks, subtasks and internal requirements in a suitable manner.

An important aspect of the BEAM robots is the very tight couple between the physical machine and its controller. The mechanical transfer characteristics of the legs of a BEAM walking machine are as important as those of the electronics. In fact the electronics are minimal and function primarily as oscillators. This, however, is not unusual in the natural world where many animals have very simple control schemes that rely heavily on a few

neurones performing non-linear mappings or oscillatory functions (Scutt 1994)(Howlett 1998).

In Tilden and Hasslacher's 1995 paper (Tilden and Hasslacher 1995) the concept of survival in machines is taken further. A method for the classification of the survival characteristic of systems is developed, using a graphical representation technique.

Tilden's machines rely heavily on the concept of the "nervous net" (Figure 3.4) an oscillator made from an arbitrary number of "nervous neurones" (Figure 3.3), each of the neurones is a simple delay. The delay is partially fixed at design, though may be effected by external influences, such as the load on motors[2], the influence of sensors and temperature. These delay circuits are assembled into rings, producing multistage oscillators. Sensors may be added by injecting the signals they produce into the base of one of the Schmidt triggers. This is possible for switches, variable resistance sensors and voltage sources.

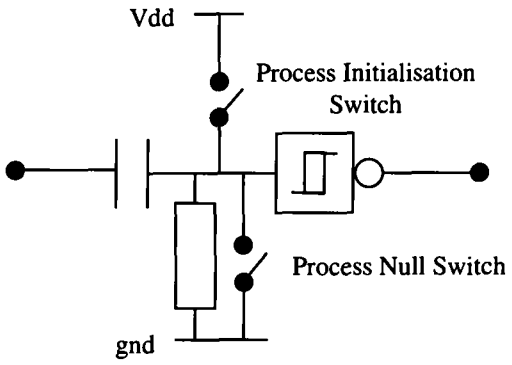


Figure 3.3. "Nervous Neurone"

[2] The load on a motor can effect the length of the delay if the nervous neurone and motor share a power supply. The current drawn by the motor increases with load, reducing the charge rate of the CR timing elements used.

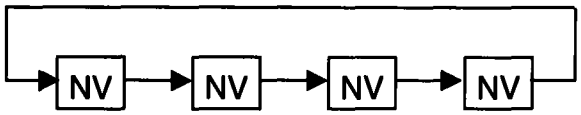


Figure 3.4 Four Stage "Nervous Network" Oscillator

3.10 Electrical life

Early neurological experiments which used Electro-mechanical systems, such as those created by Grey Walter in Bristol (Walter 1950), also used very simple circuits, with a mainly oscillatory action. Walter's "Turtles" Elmer and Elsie have all the hallmarks of simple reactive vehicles. Using simple electronic circuits, two types of "turtle" were made *Machina speculatrix* (Walter 1950) and *Machina docilis* (Walter 1951), which exhibited learning using just seven thermionic valves.

Walter seems to have been enthralled by the manner in which his creations behaved, especially when two machines were operated together. Their interaction producing complex patterns of behaviour. The turtles possessed a very simple sensorium, comprising a light detector and a contact switch, which operated on impact with an obstacle. Further interaction was produced by the presence of an indicator lamp that was extinguished when the photo detector was active. If two members of the species *Machina speculatrix* were placed in the same environment, they would approach, due to the inherent phototropic behaviour of the devices. When they become close enough, they will either touch, causing them to back off as the contact switch is made, or "dance", under the attractive influence of each other's indicator lamps.

The enthralling behaviour of these simples systems illustrates how complexity may emerge from the interaction of simple elements.

3.10.1 The Homeostat

Another, very successful, early attempt to produce behaviour analogous to that exhibited in the animal kingdom was the Homeostat (Ashby 1952). Though immobile and unable to produce actions that could be construed as animal like, the Homeostat has some interesting properties. Working on the principle that adaptation and learning in animals is the result of trying to maintain certain environmental conditions, the Homeostat attempts to keep environmental variables within certain, safe ranges (homeostasis). The variables that are under control are the displacement of moving coil meters, connected to a valve circuit. Measurement of pointer displacement is via the resistance between the tip of the meter pointer, and the ends of a fluid bath in which it is placed. The most impressive behaviour the Homeostat exhibits is the property of ultra-stability, the ability to reach a stable state, regardless of the initial configuration of the elements. Ultra-stability requires that the system may identify occasions when a variable appears to be moving out of its safe range, and use this information to instigate a change of system parameters.

The important property of all the above systems is emergent behaviour. The BEAM robots behaviour is dependent on the environment interacting with the machine, as are Elmer and Elsie, Grey Walter's turtles. The interaction in the Homeostat is between the identical coupled elements. It is impossible to describe the behaviour of these devices without considering the items with which they interact. Each element of the homeostat would perform rather mundane actions if uncoupled from the other elements, as would a BEAM robot or Grey Walter's turtles when placed in a homogenous environment.

3.10.2 Problems

The major problem with adoption of such simple architectures is in designing a device for a specific application. A machine created using these techniques must be planned with great care if it is to achieve any useful task. Connel (Connell 1990) created a machine (Herbert), which would tidy up empty soft drink cans from a suite of offices using the colony architecture. Relying heavily on the physical properties of the vehicle, the environment and

the drink cans themselves a level of success was obtained. There were however some problems with completion of the task, such as a failure to fully search the whole of the office environment, which would have been difficult to solve due to reliance of certain behaviours on certain patterns of trigger stimulus. The more complex the tasks on which they are to be engaged, the more difficult the design stage of the problems is.

Although Connell's vehicle, Herbert, successfully collected cans, expanding its role to collect other items would be problematic, requiring the addition of primitives for both the new movements required to grab different objects, and recognition of the objects themselves. Changing the navigational method of the device would be even more difficult.

Ultimately when relying on behaviour that emerges from the interaction of simple elements, the greatest challenge is to be able to determine the exact nature of the elements that will react in such a way. This problem is exacerbated when complexities of the environment in which the machine is to operate are also considered as part of the system.

3.11 Agent architectures

In contrast to the 'super-simple' architectures outlined above, controllers that follow a basically behavioural paradigm may be made more complex. When the level of abstraction between the behaviours and environment increases behaviours become complex enough to be considered individual agents. In this case each agent is a computational sub unit committed to the achievement of a specific task.

Unfortunately the term agent is also used to describe whole devices that operate independently. Many authors refer to embodied agents, or situated embodied agents, these are the whole machine or controller, rather than a single sub-element.

Agent architectures are control schemes based on the theory of mental process proposed by Marvin Minsky (Minsky 1988). In "The Society of the Mind" Minsky describes an agent to be any subset of the mind that is given a specific task. Agents may form conglomerates

(agencies) to produce higher levels of abstraction. An agent itself may be an agency of more simple behavioural units, as such agent architectures form hierarchically. These hierarchies are not heavily layered or structured. Some behaviours take the form of bureaucrats. Their function is to select and control the operation of agents at a lower level of abstraction. To an observer, the highest agents in the bureaucracy appears to achieve the desired behaviour. In reality it is scheduling the performance of those agents within it's agency, which may in turn be engaged in co-ordination tasks rather than actually manipulating the system actuators.

Figure 3.5 provides a schematic representation of the hierarchy that forms agent 'Builder', tasked with the construction of towers of toy building blocks. The 'Builder' agent schedules three sub-agents, Begin, Add Block and End. 'Begin' finds a suitable location for the tower to be started. Repeated applications of 'Add Block' then raise the tower, until it is considered finish, at which point, 'End' is activated and performs it's task (to strike down the tower).

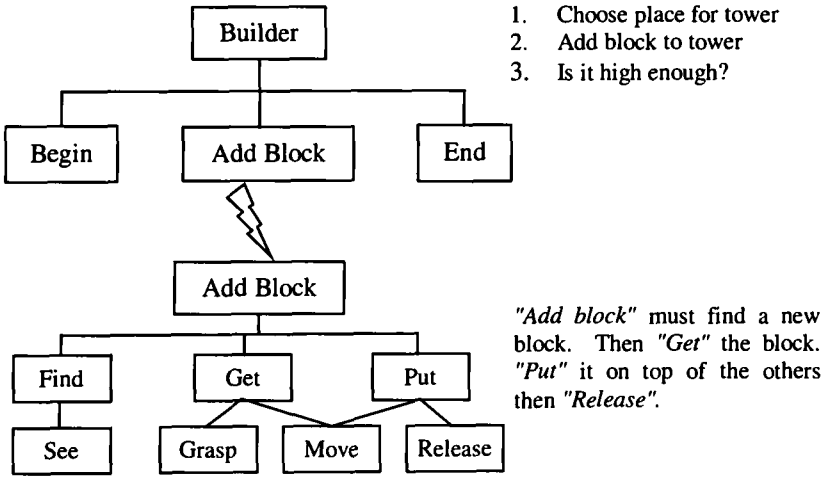


Figure 3.5 Agency for building with toy blocks (after Minsky)

'Add Block' itself is an agency, employing a range of sub-elements to locate a new block and place it on the burgeoning tower.

An agent may belong to more than one agency provided the task it performs is applicable to both. Viewed individually an agent would appear to follow a hierarchical structure, with the lowest levels directly interfaced to actuators. Viewed together a more webbed topology would be observed.

3.12 Selection or arbitration

A major concern in reactive and behavioural control schemes is resolution of conflict between the outputs of behavioural primitives. This is not a great problem when behavioural control techniques are applied to a single simple task. In such a case a large difference between the actuator signals produced by sub-behaviours is unlikely.

More complex systems may have a greater number of potential goals and sub-goals. In such cases the likelihood of a group of behaviours that are active simultaneously producing incompatible motor demands increases markedly.

In a controller concerned solely with the local navigation task of a land based vehicle (i.e. object avoidance) no severe conflict should occur. In any situation there will be a limited number of possible actions that the vehicle can take due to the reduced dimensionality of the environment. For such a device the simplest solution has a reaction which could most easily be implemented using a look-up table. Such as system is suitable only for the most primitive of tasks, such as moving randomly without collision.

As the vehicle is given more tasks or the environmental complexity increases, behaviours may come into conflict. An example of conflict introduced in this way is provided by a docking task, as identified in section 3.4. Here any behavioural acts that prevent the vehicle from colliding with objects within the environment will try to prevent the approach of the vehicle towards the docking position, this will involve motor commands such that the vehicle will stop, or even move away. A conflict exists between the requirements of these tasks, and thus the behaviours that are used to implement them. These conflicts are not

insurmountable, but require that these behaviours do not exist as peers, but form either a hierarchy or are competitors.

The case outlined above illustrates an important change in scale. The generality and abstractness increases to a point where a single element may contain a number of rule-type mappings between sensors and actuators, and hold a far more abstract goal. It can be seen that when there is only one behavioural act that needs to be achieved, it may be achieved without the need to resolve a conflict of motor signals. The conflict problem exists only between patterns of action. When the behaviour of the system comprises more than one behavioural pattern, conflict resolution becomes important.

Though it would be possible to create a "perfect" mapping between environmental conditions and motor actions, taking into consideration internal motivation of the vehicle, this task would be impracticably complex. The ability of the controller to cope with novel situations would be severely diminished if a hard mapping were used. Extensive knowledge of the environment is required; not just its physical structure but also how this is likely to change over time. All situations that can arise must be taken into consideration if there is to be no failure.

3.12.1 Conflict resolution

In the colony and subsumption architectures, resolution of conflict causes few problems as more specific competencies are given the ability to inhibit or suppress the activity of those that are more general nature. As each of the higher levels becomes active it will seize control of the machine.

It can be assumed that an increase in the complexity or abstraction of the task would be associated with an increase in the number of behavioural components. Increased numbers of behavioural components will require activation criteria that measure the environment at a higher level of precision, or use additional environmental or motivational factors (Figure 3.6). The hard suppression and inhibition used in simple architectures also represents a loss

of information from the system causing actuator function to become limited, and may cause rapid, sharp changes in actuator signal, where ideally there should be a smooth transition between actuator demands(Bisset 1995).

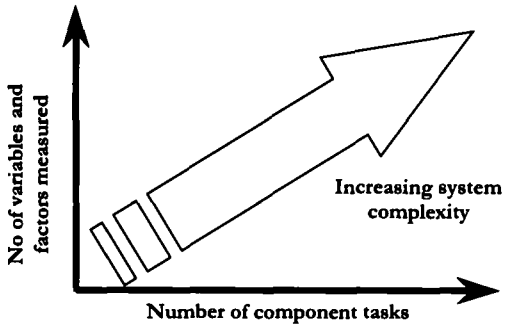


Figure 3.6. The relationship between increasing task complexity and the number of environmental factors that must be measured

An alternative to techniques where the motor signal is "selected" from the set of motor actions are schemes in which the motor actions are calculated as a compromise between the demands of behavioural elements. Motor schema systems fall into this category.

Concurrently running behaviours lead to prioritisation issues. In selection techniques, priority is encoded implicitly in the layering and connections between competencies, or the design of the selection mechanism. This is not the case where there is compromise where some mechanism is required that will assign a weight to each of the contributing behaviours.

If all the potential advantages of arbitration systems are to be exploited, some measure of the activation of behavioural elements is also required. By passing information about the extent to which the activation criteria of a behavioural pattern are met a greater range of actions is made possible. Even a simple threshold activation level for each of the behaviours will produce a larger range of possible behaviours than a selection technique, provided that there is an overlap of activation criteria.

3.13 Arbitration architectures

Many techniques have been used to achieve arbitration between the outputs of behaviours. These generally use an averaging function to determine a compromise solution.

In Arkin's motor schema navigation architecture an artificial potential field is generated for each of the active behaviours (Arkin 1989)(Arkin and Murphy 1990). Arbitration is achieved by placing all the fields generated in this way into superposition. The resultant field determines the direction of travel of the vehicle. Noise may be added to the fields to help prevent entrapment by local minima, giving a robust response. An alternative requires the addition of action vectors. Each of the behaviours produces a desired action vector. The output of the behavioural controller is then found by vector addition, the resultant representing the compromise action (Aylett et al. 1995). In both these systems a simple mechanism for indicating the present importance of each of the contributory fields or vectors exists implicitly in the medium chosen. The modulus of a vector and the strength of a potential field are ideal variables for holding this information.

DAMN (Rosenblatt 1997) is a voting mechanism for action selection. A set of possible output actions is provided. All behaviours vote for a favoured member of this set by producing action desirability values for each. In order that choices, which are unpopular with other behaviours are penalised, the voting is achieved by setting a desirability value between -1 and 1 for all possible actions for each behaviour. The action that the system then performs is calculated using a weighted average technique. A mask is applied so that the machine can adopt possible command signals that fall between the discrete options provided. The use of this mask ensures that using a fixed set from which actions must be chosen does not compromise the repertoire of the controller. No single behaviour has control of the actuator signals produced.

3.14 Action selection techniques

Action selection techniques choose an action from the control repertoire of the system. The exact manner by which this is achieved may vary but the result is usually a single control signal from a discrete set. Some techniques provide more flexibility, allowing the output to take on continuous values, helping to prevent vacillation between control actions.

The simplest of the action selection techniques are those that depend on the structure of the controller such as the subsumption architecture. The use of such a scheme implies that all behaviours have a definite, unique priority, which is known to the system designer. When behaviours have triggering conditions that are unlikely to be coincident, careful design to ensure correct prioritisation is not required.

Alternative, more complex action selection techniques exist including activation spreading, voting systems and selection by an external agent.

Action selection techniques are most useful when the behaviours are highly abstract. Indeed, as the level of abstraction increases some form of selection may be required, as the set of possible behaviours increases. Many behavioural control techniques use high level descriptions of the behaviour of the machine, rather than the primitive ethons employed by others. In these schemes, where the elements are behavioural patterns rather than individual sub-behaviours, each behaviour represents a complex goal. In order that a goal may be achieved it becomes important that there is a level of commitment to that behaviour. Figure 3.7 illustrates how the level of abstraction is related to the behaviour synthesis for systems that employ a single technique. The horizontal axis represents the level of abstraction found in the descriptions of the behavioural elements. The bars represent the applicable areas of the various techniques. Activation spreading (described in section 3.16) is a selection technique where the dominant behaviour is selected by the state of the agent and its sensors, rather than by the environmental conditions alone.

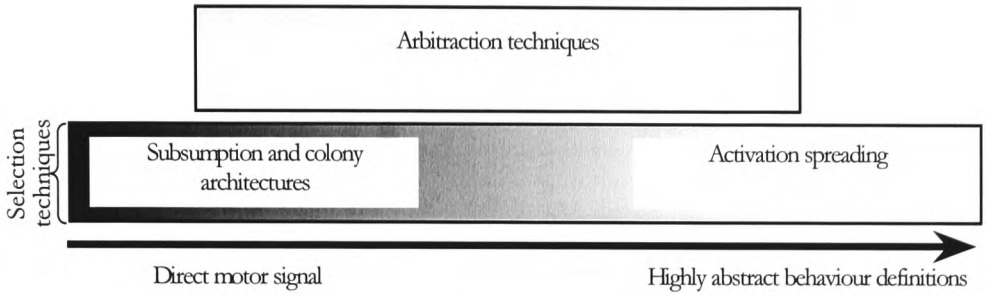


Figure 3.7. Selection of synthesis technique.

Where it is acceptable and useful for reflexes or simple behaviours to operate simultaneously, as the complexity of the behaviour patterns increases, the chances of two being compatible are reduced, requiring that some form of selection technique be used. A retrieval behaviour, which searches for an object in the environment before returning it to a fixed point will require incompatible components. The search stage requires that the machine explores the environment, identifies potential targets and collects them. The return to fixed-point behaviour requires that the machine perform directed navigation. Premature activation of this behaviour will prevent the machine from performing an adequate search.

3.15 External selection

A simple method for achieving action selection is to use some from of external agency to select the active behaviour. Murphy (Murphy 1996) uses a script to select which behaviours are active at any time. The vehicle is provided with a script, which contains the domain knowledge of the environment in which the system operates, this is used to schedule the activation of behaviours. The behaviours are of a high level of abstraction, with names such as "*NavigateHall*" and "*ThruDoor*"; each assembled from smaller behavioural elements.

3.16 Activation spreading

Activation spreading methods are more complex. They rely on each behaviour possessing a thresholded activation level variable. The environment and the internal state of the controller affect these activation levels. When the threshold is reached the behaviour becomes active. Active behaviours have control of the system actuators (or outputs in simulated instances). The level of behavioural activation spreads from the current situation and the present goals of the system, meaning that both have a direct influence on the selection of behaviour.

Maes (Maes 1991) gives an example of such a system. An artificial creature is furnished with a set of behavioural patterns. In addition to an activation level variable, each behavioural pattern possesses a set of triggering conditions. If all the trigger conditions of a behavioural pattern are met and the activation level exceeds its associated threshold the behaviour becomes active. The behaviours are networked via three types of connection. If a behaviour, by its action leads to the satisfaction of the trigger conditions of another behaviour, there is a *"Predecessor"* link between them in the network, *"Successor"* links provide the mirror of this service. A *"Conflicter"* link is used to illustrate that two behaviours are not compatible such that the second behaviour, by its operation prevents the achievement of the trigger conditions for the first. The artificial creature also has a series of motivations such as *"Hunger", "Thirst", "Fear"* and *"Curiosity"*. There are four mechanisms by which behaviours can be activated:

1. *Activation by situation.* The present perceptual conditions increase the activation level of those behaviours, for which they form part of the trigger conditions.
2. *Activation by motivation.* The motivations of a creature increase the activation of those behaviours with which they have been associated. For instance, the motivator *"Hunger"* increases the activation level of behaviour *"Eating"*.
3. *Activation of successors.* An executable behaviour increases the activation level of behaviours that succeed it. Those behaviours which have the potential to be

active when the executable behaviour is complete. As an example: if a behaviour *"Going towards food"* is executable the activation level of the behaviour *"Eating"* should be raised.

4. *Activation of Predecessors.* A behaviour that is not executable spreads activation to those that might contribute to it becoming executable. The rise in the activation of the presently achievable behaviours is proportional to the activation value of the un-executable behaviour. In this way a behaviour such as *"Eating"* can influence the behaviour of the creature such that it may become possible. If the motivation to perform an action is high, this will increase the activation level of those behaviours that must be met to trigger the behaviour that will satisfy that motivation. For example *"Hunger"* increases the activation of *"Eating"*, which will, in turn increase the activation of *"Going towards food"*.

Note that *activation of successors* and *activation of predecessors* have the potential to form self-supporting loops.

All behaviours will decrease, by a factor of their activation level, the activation level of those behaviours in which it not compatible via the *"Conflicter"* links of the network. The effect of these links is that the motivation of the creature influences those behaviours that are active, so as to achieve the goals of the creature. The environmental conditions make those patterns of action that are appropriate at any instant more attractive. The behaviour that best satisfies these two considerations will be selected. Activation of a behaviour will, by use of the *"Conflicter"* links, usually prevent the behaviour being prematurely terminated without a severe change in the environmental condition. In this way a level of commitment to a pattern of action, once embarked upon, can be implemented.

An electrical model provides a simple explanation of the behaviour triggering mechanism (Figure 3.8). The capacitor/discharge resistor network provide an element of time based

latency, ensuring the network exhibits hysterisis, in addition to that provided by the spreading of activation itself.

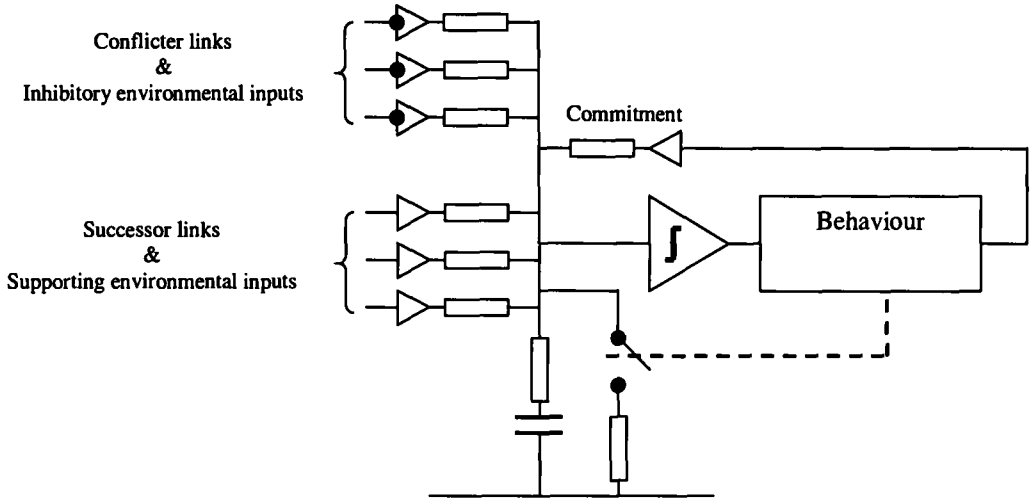


Figure 3.8. Electrically analogous illustration of the operation of the behaviour selection process in Meas' model of activation spreading.

3.17 Selection and arbitration joint schemes.

As an alternative to choosing either a selection or arbitration technique, it is possible to use both simultaneously within a single controller.

The humanoid simulation Adonis (Mataric 1998) is controlled using three behavioural primitives. Of these, two are mutually exclusive *"move-to-point"* and *"get-posture"*, while the third *"Avoid"* is always active. In the Adonis controller a selection technique (hard switching using a planner) is used to determine which subset of the behavioural repertoire will be used at each point in time. The active behaviour is then merged with the "avoid" primitive producing behaviour that defines the movement of Adonis' arms. Adonis dances the Macarena, the nature of the dance is such that 'his' arms must move freely, adopting either set postures, or fixed positions with a specific orientation. Posture may be described more easily in joint space, position and orientation in Cartesian space. The behaviour primitive *"get-posture"* is used to achieve the set postures of the dance, while *"move-to-point"* is more

suitable for the accurate positioning of the hands of Adonis in Cartesian space. At each 'step' of the dance, the planner will select which of the behaviours to use, dependent on the nature of that step; postural or positional.

Adonis shows that arbitration between a variable subset of behaviours makes for successful achievement of varied goals. Each of the tasks may use a different subset of the total behavioural repertoire as appropriate.

3.18 Integrating representations

The ability to integrate representations of the environment into a controller is an important property that helps draw the line of demarcation between the simple reactive schemes and more flexible behavioural architectures. This is the method of discrimination between the two techniques used by Mataric (Mataric et al., 1998).

An early attempt at the integration of representation into behavioural controllers is "Toto" the robot dog (Mataric and Brooks 1990). Toto learnt relational maps of the environment in which it was placed, using them for simple navigation tasks.

The maps employed by Toto are noteworthy as they are distributed, not centralised and learnt rather the imposed by the designer. The use of a relational, rather than the more traditional absolute maps makes this possible. Absolute maps are hard to create due to high sensory demands. Angles and distances must be measured accurately. A relational representation is created using topological features that may be recognised by the machine.

3.18.1 Toto's structure

Toto is programmed in behavioural language, a language written at MIT to implement layered, subsumption architectures. There are three main layers of competencies: *"collision free wandering", "landmark detection" and "map learning/path planning".*

The lowest level of the controller is concerned with wall following behaviour. The design of this competence illustrates how a subsumption controller is constructed. There are four interacting behavioural primitives: *stroll, avoid, align* and *correct.* These primitives rely on a classification of the distance of an object into three main sets. *"Danger zone"*, where the object is too close. *"Minimum safe distance"* and *"edging distance".*

The primitive *"stroll"* controls the velocity of the machine enabling it to travel forward, backwards or stop under the influence of the distance between Toto and any objects in the immediate environment.

"Avoid" changes the heading of the robot, to avoid objects depending upon information obtained from Toto's ultrasonic range finders. Together *"stroll"* and *"avoid"* produce a random collision free path (wandering).

"Align" maintains the heading of the robots parallel to the wall that is being followed. If Toto starts to head away from the surface, *"align"* will bring it back in. The interaction of "a*lign", "stroll"* and "*avoid"* creates a wall following behaviour that will trace the periphery of convex objects.

The addition of the "*correct"* sub-behaviour allows Toto to follow concave walls Figure 3.9 shows the structure of the object boundary tracing action pattern.

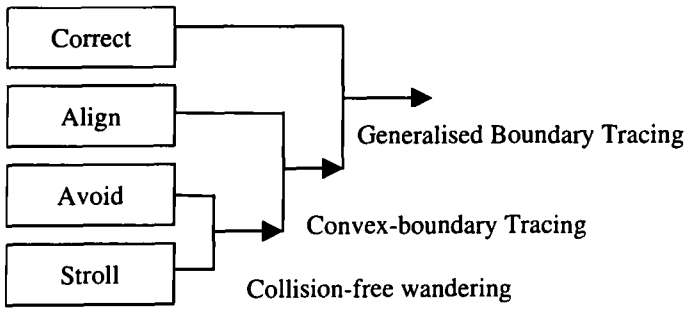


Figure 3.9 Toto's Basic Behaviour primitives (after Mataric)

3.18.2 Location

For a machine to make use of a map it must be able to locate itself within the map. This is often achieved by use of landmarks. Many schemes use static landmark recognition (Baptiste et al., 1994) to achieve this. The problem lies in ensuring that the landmarks can be reliably recognised and are not subject to change. Even an easily determined and particularly static landmark will appear to change with varied distance and approach direction.

As Toto is intended to explore the large-scale structure of a constrained environment the localisation task may be reduced to a general position. The robustness of the underlying behavioural controller helps with this simplified location task. Toto operates in a suite of offices, thus walls and corridors are common features, fortunately walls and corridors are easy to recognise, even with an impoverished sensorium. Landmarks are recognised dynamically. A magnetic compass records the heading, while the distances to objects are recorded. Short distances on one side of the robot and a consistent heading would be used to determine the presence of a wall. Taking an average over several sensor readings reduces misidentification of location due to the observation of transient objects and sensor noise.

Toto's map is a graph, with nodes distributed across a series of behaviour primitives, equivalent to any other in the machine. The nodes take recognised landmarks as input, outputs are connections to other neighbouring nodes showing spatial relationships. The map is used by comparing sensor data with that associated with the landmarks stored in the map. A close correlation causes a node to become active. This node is the map representation of the present position of the machine. Navigation is achieved by sending a call from the destination node through the graph. This call propagates through the graph until the active node is reached. The compass heading required to reach that node is also transmitted with the call. When the active node is reached the robot just adopts the correct heading and progresses, back along the path created by the spread of the call through the map.

Nodes in the map have the ability to inhibit each other. This is an important requirement of the learning algorithm. Inhibition between behaviours also has important implications for behavioural controllers in general. Inhibition and facilitation between behavioural elements help to create sequences of behaviour and commitment to patterns of behaviour such that successful performance of compound and long term tasks may be achieved.

3.19 Hybrid control strategies

As behavioural controllers have a dearth of explicit global symbols that can easily be manipulated by the designer, the implementation of controllers for a specific task becomes difficult when the complexity of the task increases. Any information held by the controller will be encoded across the behaviours, within the control structure. Manipulation of the information is thus rather difficult. In order to help confront this difficulty there have been many attempts to integrate deliberative controllers, which have simple to read properties and a definite goal oriented structure with the more robust and computationally less expensive behavioural techniques. This desire to 'hybridise' is also motivated by the belief that behavioural controllers are at best unprovable, while at worst merely the product of mindless tinkering. Many authors working in the field of mobile robotics seem to believe that hybridisation of the two control paradigms, (behavioural and deliberative) is the only way in which true objective design of autonomous vehicles is possible.

Adonis is effectively a hybrid system. An external planner is used to select which of the behaviours is active. Murphy's script technique for behavioural selection also follows this basic format. The scripts are external to the behavioural system and are used to schedule the behaviours with tasks.

Hybrid systems usually take a hierarchical form, where a behavioural system forms an executive, controlling the motors and actuators of the machine, while a deliberative section makes plans and models, performing a managerial function for the machine. The hybrid architectures are intended to have the best of both worlds, the direct close coupling of

behavioural and reactive schemes, with the associated properties of robustness and generality, but with the easily task oriented nature of deliberative systems.

3.20 Three layered architectures

Three layered techniques provide an alternative hybrid architecture that has been used with some success. Three layered architectures developed in response to problems experienced with the plain reactive systems where tasks are "hardwired" into the very structure of the controller, i.e. to address the question: How do you get your machine to do other things?

The three layered architectures are attempts to maintain the advantages of a purely reactive system, while manipulating it via two superior control layers. Layer two is a "sequencer", while the principal control element of the system is a "deliberator".

Combining the three layers allows the immediate, high-speed "house-keeping" actions to be performed by the reactive layer, while the long term, goal-oriented aspects of the agents task may be achieved by the deliberative layer.

Structure of the three layers (Gat 1992):

Reactive layer: The behavioural executive. Only ephemeral state should be stored at this level. If behaviours fail the failure should be detected by the system. Behaviours are generated by continuous functions.

Sequencer: Selects which behaviour is active at any time.

Deliberative: Concerned with time consuming tasks. Produces plans for the sequencer, or answers queries of the sequencer.

3.21 Hybrid architecture identification

Three general topologies for the interface between the reactive and deliberative sections of a hybrid controller can be identified. These are:

1. Pseudo-sensing
2. Abstract behaviour
3. Scheduling

The form of the interface is restricted by the need to communicate between the layers. The simplest technique is to use the output of the deliberative controller as additional inputs to the relevant behaviours. The behavioural controller is still autonomous, though falls under the guidance of the deliberative system. Examples include Tooth and Rocky 3 (Gat 1992).

Alternatively a deliberative controller may work independently from the behavioural section, producing an output suitable for consideration by the arbitration mechanism. The deliberative controller is an abstract behaviour.

Finally it is possible that the deliberative controller selects those behaviours that are suitable for use at any time. This is the foundation of three layered architectures (Murphy 1996)(Mataric et al., 1998)(Gat 1992).

Diagramatic representations of the three alternatives are shown in Figure 3.10.

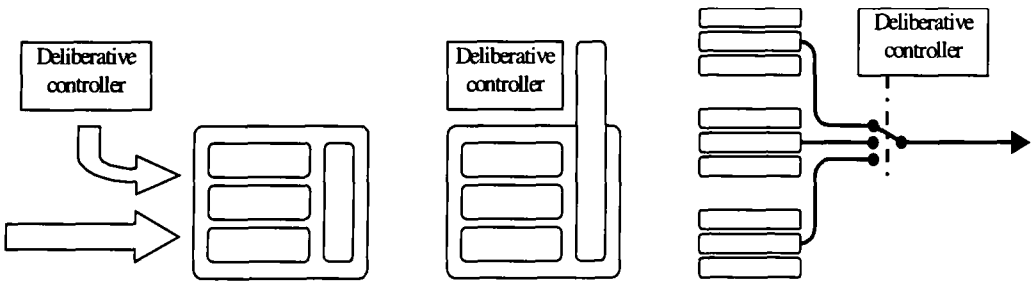


Figure 3.10. The three basic hybrid architectures

3.22 Sequences without planning

Regardless of claims that hybrid control strategies must be employed if useful goal directed behaviour is required (Thrun and Mitchell 1993) it can be shown that coherent sequences of behaviour may be produced simply by the interaction of a behavioural controller with its

environment. That designing systems to achieve this is difficult is not under doubt, however, claims that hybrids must be used denies the existence of machines such as Herbert (Connell 1990) which do perform complex, sequence driven tasks without the aid of deliberative planners. Much behaviour observed in natural systems is achieved by producing sequences of action patterns, which may be achieved without the use of explicitly stated plans. The production of sequences in nature depends on environment and task. The sequence of actions that results in the movement of a tidal snail from the sea to its feeding grounds is the product of the snail's fixed behavioural actions and the environment. The progressive completion of each act places the snail in a changed environment, which serves to trigger the next (Connell 1987). Brooks refers to such systems as "...using the world as its own model" (Brooks 1991). The plan, which defines the sequence of behaviours to be used, is coded in the design of the behaviours themselves. If this view is taken, then the sequence of actions as performed by the system is directly related to its behavioural selection mechanism.

An important issue for sequencing behaviour is measuring the completeness of a behavioural component of the sequence. In natural systems this is most often achieved by observation of the state of the environment, the environment becoming the scratch pad where the agent may identify the completeness of each behaviour as it is achieved. Connell's successful can collecting robot, Herbert, illustrates the use of the environment to sequence behaviour excellently (Connell 1990).

3.23 Summary

Behavioural control is an attempt to model the synthesis of behaviour in natural systems with the objective of producing truly autonomous intelligent behaviour with a high degree of robustness.

Behavioural controllers employ the interaction between a selection of primitive functions to generate complex activity. To build up behaviour in this manner has implications for the

robustness of the solution to sensor noise and other representational errors, due to distributed nature of the processing of the elements. It also serves to help the design process, by allowing controllers to be designed and implemented incrementally, the creation of more complex behaviour being postponed until the more simple action patterns have been successfully generated.

Behavioural control has been shown by many authors to be able to solve many everyday control problems that have been difficult to approach using more traditional techniques. This is most evident with those tasks which seem the hardest to define when presented in animal or human behaviour, but which are easily performed by natural systems.

The lack of global symbols or centralised representation in behavioural controllers does however pose some problems, which must be addressed. Setting a compound or complex task for a behavioural controller may be difficult, as can both measuring, and predicting the performance of such a system.

Like the animal systems they are intended to emulate, most behavioural controllers are implemented close to the hardware on which they run, and coupled tightly to the environment. Although this makes for good exploitation of environmental features that allow simplification of control and improves utilisation of mechanical abilities, it further increases the difficulty inherent in predicting the behaviour of these systems.

Several authors have proposed hybrid and hierarchical structures to help alleviate this problem, using other control strategies to complement a behavioural sensor-motor interface.

The interaction of the individual action patterns may be managed in many ways, which will either be as a compromise between the outputs of behaviour, or the selection of a single behavioural sub-unit to determine machine action, while the present conditions remain.

The selection of which of these techniques is appropriate will be determined primarily by the level of abstraction of the behaviours that are being managed.

It is essential that the designer of a behavioural controller for an autonomous vehicle considers all the properties of the environment and attempts to model the behaviour of the system. Not in great detail, but so as to understand the dominant interactions that will take place.

There are classes of task that would be can only be coded directly as behavioural controllers with a great deal of inefficiency. Tasks where a vehicle is expected to be able to navigate one of a collection of paths, as chosen by an operator, are a good example of this. In such a case it is desirable that an appropriate path may be communicated to the vehicle. To communicate and follow a path, representations of space are required. The next chapter investigates the nature of spatial representation to determine which representations may be appropriate.

3.24 References

Arkin R. C. 1989. Motor Schema- Based Mobile Robot Navigation. *International Journal of Robotics and Automation* 81, no. 4: 92-112.

Arkin R.C. 1992. Cooperation Without Communication: Multiagent Schema-Based Robot Navigation . *Journal of Robotic Systems* 9, no. 3: 351-64.

Arkin R.C. 1998. *Behaviour Based Robotics.* Cambridge MA: MIT Press.

Arkin R.C. and Murphy R.R. 1990. Autonomous Navigation in a Manufacturing Environment. *IEEE Transactions on Robotics and Automation* 6, no. 4: 445-54.

Ashby W.R. 1948. Design for a Brain. *Electronic Engineering* , pp379-383

Ashby W.R. 1952. *Design for a Brain.* London: Chapman Hall.

Aylett, R.S., Coddington, A.M., et al. 1995. Heterogeneous Agents for Multi-Robot Cooperation. *Design and Developement of Autonomous Agents* 95/211.

Baptiste, P., Bideaux, E., et al. 1994. Use of Perception Based Navigation for Autonomously Guided Vehicles. *1994 Joint Hungarian-British Mechatronics Conference* : pp279-284.

Bisset, D.L. 1995. Engineering Reactive Vehicles:From the Bottom Up. *Design and Developement of Autonomous Agents* 95/211, no. 95/211: 3.

Booch G. 1994. *Object-Oriented Analysis and Design.* Reading Mass.: Addison Wesley.

Braitenberg, V. 1984. *Vehicles - Experiments in Synthetic Psychology.* First ed. Bradford Books. Cambridge, Massachusetts: MIT Press.

Brooks R.A. 1991. "Intelligence Without Reason." *Intelligence Without Reason*, A.I. Memo No. 1293. MIT, also available in proc. IJACAI-91.

Brooks, R. A. 1986. A Robust Layered Control System For A Mobile Robot. *IEEE Journal of Robotics and Automation* RA-2, no. 1: 14-23.

Brooks R.A and Flynn A.M. 1989. Fast, Cheap and Out of Control: A Robot Invasion of the Solar System. *Journal of the British Interplanetary Society*, no. 42: pp 478-85.

Calvin W. H. 1993. "The Unitary Hypothesis: A Common Neural Circuitry for Novel Manipulations, Language, Plan-Ahead and Throwing? *Tools, Language and Cognition in Human Evolution.* Eds Gibson & Ingold, pp 230-250. Cambridge University Press.

Coad P., and Yourdon E. 1990. *Object-Oriented Analysis.* New Jersey: Yourdon Press.

Connell J.H. 1987. Creature Building with the Subsumption Architecture. *IJCAI 87*, pp1124-1126.

Connell J.H. 1990. *Minimalist Mobile Robotics: A Colony Architecture for an Artificial Creature.* London: Academic Press Ltd.

Dennett D. 1993. *Consciousness Explained.* Penguin.

Gat E. 1992. Integrating Planning and Reaction in a Hetrogeneous Asynchronous Architecture for Controlling Real-World Mobile Robots. Proceedings of the 10th National Conf. on Artificial Intelligence (AAAi-92)

Goodwin B. 1998. All for One and One for All. *New Scientist*, pp 32-35.

Grier J.W., and Burk T. 1992. *Biology of Animal Behaviour.* St. Louis U.S.A.: Mosby Year Book.

Howlett R. 1998. Simple Minds. *New Scientist*, pp29-33.

Klarer P. 1997. Small Scale Intelligence for Lunar Exploration. *Control Engineering Practice* vol. 5, no. 6: pp 859-63.

Kliener K. 1998. Brainless Wonders. *New Scientist* -, no. 2143: pp 40-44.

Lewin R. 1993. *Complexity.* Phoenix.

Maes P. 1991. A Bottom-Up Mechanism for Behaviour Selection in an Artificial Creature. *From Animals to Animats.*Editors J.A.MEYER and WILSON S.W, pp 238-46. MIT Press.

Mataric M. and Brooks R.A. 1990. Learning a Distributed Map Representation Based on Navigation Behaviours. - *USA- Japan Symposium on Flexible Automation.*

Mataric M.J. 1997. Behaviour Based Control: Examples From Navigation, Learning and Group Behaviour. *Journal of Experimental and Theoretical Artificial Intelligence* vol. 9, no. 2-3: pp 323-36.

Mataric M.J. 1998. Behaviour Based Robotics As a Tool for Synthesis of Artificial Behaviour and Analysis of Natural Behaviour. *Trends in Cognitive Science* 2, no. 3: pp 82-87.

Mataric M.J., Williamson M., et al. 1998. Behaviour-Based Primitives for Articulated Control. *From Animals to Animats 5- Proceedings 5th Int Conf Simulation of Adaptive Behaviour.*

Matijevic J. and Shirley D. 1997. The Mission and Operation of the Mars PathFinder Microrover. *Control Engineering Practice* vol. 5, no.6: pp827-835.

Minsky M. 1988. *The Society of Mind.* New York: Touchstone (Simon and Schuster).

Murphy R.R. 1996. Use of Scripts for Coordinating Perception and Action. *Proc. IROS 1996*, pp 156-61.

Patterson. D. W. 1996. *Artificial Neural Networks, Theory and Applications.* Prentice Hall.

Pfeifer, R. and Verschure, P. 1995. The Challenge of Autonomous Agents: Pitfalls and How to Avoid Them. *The Artificial Life Route To Artificial Intelligence: Building Embodied Situated Agents.* Editors Steels & Brooks, 237-63. Hove: Lawrence Earlbaum Associates.

Rosenblatt J.K. 1997. DAMN: A Distributed Architecture for Mobile Navigation. *Journal of Experimental and Theoretical Artificial Intelligence* 9, pp 339-60.

Scutt T. 1994. The Five Neuron Trick: Using Classical Conditioning to Learn to Seek Light. *Proc. 3rd Int Conf. on Simulation of Adaptive Behaviour*, Editors Cliff D., Husbands P., Meyer J.A., and Wilson S.W., pp 364-70. "From Animals to Animats", no. 3. Cambridge USA: MIT Press.

Shelly M. Wollstonecraft. -. *Frankenstein; or the Modern Prometheus.*

Thrun S.B. and Mitchell T.M. 1993. Integrating Inductive Neural Network Learning and Explanation-Based Learning. Proceedings IJCAI 93, pp 930-936.

Tilden M.W. 1993. The Evolution of Functional Robot Ecologies. *Proceedings ARS Electronica 1993*, Linz Press.

Tilden M.W. and Hasslacher B. 1995. Living Machines. Robotics and Autonomous Systems. 15, pp143-69.

Walter W.G. 1950. An Imitation of Life. *Scientific American* 182, no. 5: pp 42-45.

Walter W.G. 1951. A Machine That Learns. *Scientific American* 185, no. 2: pp 60-63.

Warwick K. 1997. *March of the Machines.* London: Century.

4: NAVIGATION AND SPATIAL REPRESENTATION

"Precisely," said Barbican: "Life without movement, and no life at all are equivalent expressions."

"Round the Moon", Jules Verne

4.1 Introduction

In "How Brains Think" William Calvin (Calvin 1993) suggests, controversially, that all intelligence is a development from the requirements of movement. This view is supported by the strange case of the sea squirt, which eats its own brain when it gives up it's mobility (Dennett 1993). The controversy is exacerbated by the various definitions of intelligence that are used. However it would appear that agents which exhibit directed movement do seem to require intelligence more than stationary objects.

Movement increases the chance of an agent finding itself in a dangerous situation to which it may respond. Movement is potentially damaging if not directed. Comparing the problems encountered by a plant it becomes obvious that a static object's survival requires less problem-solving ability than that of a mobile device. At the most base level, a mobile agent must guard against collisions and precipices.

The ability to move also increases the energy demands of a system. Animals must thus actively seek new sources of energy, and be able to return to them as appropriate. Directed movement is a must for the survival of any but the most simple of creatures. The importance of the energy

source of a mechanical agent also reflects this compromise between the demands of the system and the ability to move. A fixed machine may consume vast quantities of power, through fixed cables or pipe-work. Fixed lines are not suitable for most mobile agents; meaning that a sufficient stored energy must be supplied. A DC power supply is an appropriate form of energy for many embodied agents. The simplest way of providing such a supply is via batteries. Unfortunately batteries are heavy and bulky. When tasked with providing the power demanded by motors and actuators battery life is severely reduced. If the agent is intended to operate for any extended period of time, the ability to return to a charging station becomes imperative.

Animals and mobile agents both require that a set of locations may be identified, and suitable action generated to take the agent between these locations. Simultaneously with moving from location to location, the agent or animal must be on guard for potential collisions.

The task of moving from one location to another is not possible without some knowledge of how the two locations are related in space. A spatial representation is therefore required.

This chapter compares various spatial representation methods, considering the applicability of each to use by a mobile agent for navigation. In addition, navigation itself is considered, to determine the most appropriate methods by which it may be achieved, and the mechanisms required to do so.

4.2 The navigation problem

Navigation comprises of two main tasks, local navigation and way-finding:

4.2.1 Local navigation

Of maximum immediate importance is local navigation, which prevents an agent colliding with objects, driving off cliffs or down potholes. It is characterised by the need to respond quickly

and directly to the environment. Local navigation may be achieved by simple reaction to sensor information. Local navigation is the primary task of most reactive or behavioural controllers. Little or no representation of space is required for the successful completion of this task. What is required is the ability to determine the location of salient features of the environment with sufficient resolution for the requirements of the local navigation behaviour. Not only the complexity of an agent's response behaviour determines this resolution, the position of the feature relative to the agent, and the behaviour of any other mobile elements of the environment also help determine the amount of information required.

4.2.2 Way finding

An agent may also be required to find its way to an arbitrary location. Prescott refers to this task as "way finding" (Prescott 1994). Whereas local navigation is concerned only with those things that an agent may directly perceive at any time, "way finding" requires that the agent has some knowledge of the space beyond its sensor range.

To find its way an agent must have sufficient information to make its move.

Considering a goal location as a function of present location (equation 4.1):

$$location_{goal} = f(location_{initial}) \qquad (4.1$$

The most common view of navigation is the first case, where the present location and goal location are known. The act of navigation is then to determine the transform that will take the agent from the starting location to the goal.

To do this some knowledge of the space in which the two locations exist is required. In a simple abstract case, where a machine needs to move across a plane between two points, the controlling agent must have knowledge of the geometry of the space, and the relationship between the

executive machines actions and that geometry. If an agent is to calculate a route in this way, it can be assumed that both the target and start positions are known. For a driver to find a route to an unfamiliar city with no knowledge of its position, either in an absolute spatial frame, or relative to the point of departure would be very difficult. As would it be to find a route between two known cities if there were no knowledge of the world between. The difficulties in obtaining this information depend rather on the complexity of the environment, and to a lesser extent its scale.

Equation 4.1 can also be used to identify locations, if the route between the locations is known. Figure 4.1 shows the relationships between locations and movement. Locations A and B are both named explicitly. Provided the nature of the space in which they exist is known, then both $Activity_1$ and $Activity_1^{-1}$ can be calculated. The third location is not named explicitly, however, it is known that $Activity_2$ will take an agent to there from Location A. This location has a relativistic name based on the start location and the transform required to get there. Names of this type are often used to identify places that have few identifying features e.g "the landing site is 10km SSW of the town".

It should be noted that the relationship between $Activity_1$ and $Activity_1^{-1}$ might not be a simple inversion. Consider moving around a one-way road system, a return journey may require a totally new route.

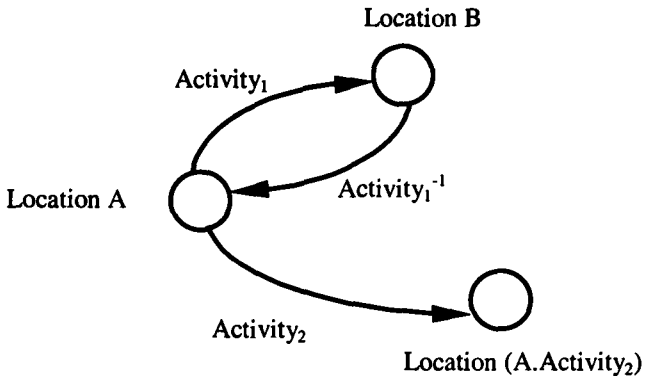


Figure 4.1 Relationship between location and action

If knowledge of space is required to calculate routes, then this information must be stored, the agent must posses some form of spatial representation suitable for the calculation of routes to specified locations. Even if explicit knowledge is not required as the route is to be communicated to the agent, then a representation shared by the agent and the author of the route is required.

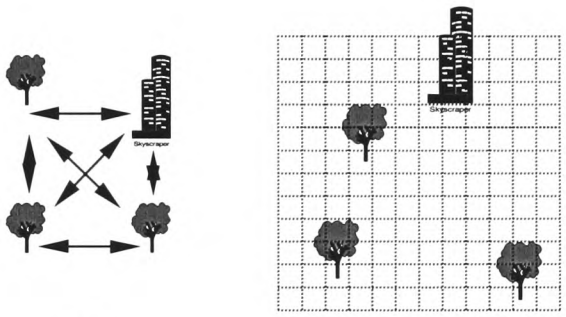
Way finding is complicated by the need to maintain the local navigation behaviour of the agent. It is tempting to believe that an agent must be able to create and store its own spatial representations. This is of course false, an agent, may in many cases be provided with this information. Indeed many tasks require that an agent navigates (autonomously) specified routes, or an environment in which the major features are know before commissioning. What is required is that the agent must be able to act upon the information provided, which demands access and understanding.

4.3 Spatial representations

For spatial knowledge to be stored by, or communicated to an agent requires that some representation is used. Spatial representations tend to fall into two main categories: absolute and relative.

Relative space (Figure 4.2a) demands that the concept of space is dependent totally on the objects it contains. Indeed there is no "space", just objects. The framework that might be perceived as space, in reality being the relationship between these objects.

Absolute space (Figure 4.2b) theories dictate that space exists as a fundamental truth. It comprises of places, which are independent of any objects they may contain. The relationships between locations are generated by the space itself, as a framework, regardless of the existence of any objects within it.



a b

Figure 4.2 Relative a absolute spatial representations

4.4 The Philosopher's Landranger

It would appear that most philosophers have at some point considered the nature of space and its representation in the minds of animals and men, as have many psychologists and ethologists. The available alternatives are simple:

- Space is absolute and is represented as such in the mind in this manner.
- Space is absolute, but is represented relatively in the mind.
- Space is relative, but is represented by an absolute framework in the mind.
- Space does not exist, but is merely a collection of objects, thus experience must be used to determine the correct motor transforms for directed movement.

More complete explanations do however suggest that a combination of representations of space is used in the minds of most animals.

Relative space is a counter intuitive situation, where space as a framework does not exist. Objects exist and define what is perceived as space by their relationships.

4.5 Philosophical and animal representations of space

As mentioned above, the representation of space is a field that has been highly discussed in philosophical circles. The following is an overview of some of the models of space that have been proposed:

4.5.1 Leibniz:

Leibniz proposed a relativistic argument, based on the assumption that space does not exist. His argument being that space, by definition, is nothing, and that "nothing" does not exist. His proposal for the mechanism of what is perceived by humans as space relied on the existence of primitive conscious entities, all of which could be uniquely identified, called Monads. Space being the full collection of these consciousnesses. Proximity in space is determined by similarity between these Monads. Objects positioned in space are collections of aggregated Monads that are sufficiently similar.

Each of the Monads must be unique in order that they may be individually identified. Without unique identification they cannot be independently referenced without recourse to spatial or temporal information. This would of course create a circular argument as neither of these things may exist without space.

4.5.2 Poincaire:

Poincaire adopted a more egocentric approach to the representation of spatial relationships; his opinion is also that space is relational, not absolute. Locations being identified in space by representing to oneself the sequence of actions that are required to arrive at the destination. There will be many possible paths (motor transformations, not actual paths in the traditional sense of the word) to the destination. To learn all of them an agent is required to make the journey several times. For each of these journeys the agent must start from the same place/object. This requires that all objects must be individually identifiable, without recourse to spatial or temporal information. The nature of objects is not as highly developed as the Monads of Leibniz. Poincaire does not specify how unique identification of each object is made.

The set of "paths" from origin to destination object builds a relational model of relative space in terms of the objects from which it is made. To fully 'know' the space, an agent must make all possible journeys.

Although Poincaire's model of space has not been used for navigation in mechanical systems, echoes can be seen in those navigational techniques where the behaviour of an agent is used to help determine position, including the script based navigation system described in Chapter 6.

4.5.3 Gibson:

Gibson's view of the mental representation of space uses textural information from the visual system to disambiguate between observer and object movement. This is effectively a relativist approach to the problem, as experience will be required to learn the relationships between visual information and movement, and between distance and movement. Without the inclusion of additional information, unobtainable visually, the absolute distance to observed objects cannot be found. There have been few uses of a Gibsonian spatial representation for the control of mobile agents, though they do exist (Duchon and Warren 1994).

4.5.4 Kant:

Kant adopted a position between relativism and absolute spatial representation. He assumed that learning and experience are used to determine the relationships between objects, taking input from movement processes. This relationship map then being used to determine the positions of the observed objects in an absolute three-dimensional spatial framework, which exists in some primitive form in the minds of animals.

Kant's model of space is the closest to those that are implicitly expressed by most works on agent navigation.

4.6 Absolute versus Relative Space

Most models of animal representation of space start from a relativistic position and use this to develop a framework for metric spaces, similar to Kant's view. These techniques fail on re-identification of points and movement. All positions in space must be immediately re-identifiable so that each time they are visited the correct node in the representation may be appended with new relationships that have been learned. In a representation where there is no concept of absolute metric space prior to the creation of the relative representation it is hard to believe that this may spontaneously be generated. There is thus a clear argument for unitary space, the concept of which cannot be created from experience, due to the relocalisation problem. The unitary space concept must be available early in an agent's development because location, identification and coherent organisation of knowledge and experience depend on it. The weakness in Gibson's view indicates that the unitary space must contain a metric. Mataric's attempt at relative mapping using experience was ultimately linked to an external framework by the inclusion of data from a magnetic compass (Mataric and Brooks 1990). A metric measure of distance is not explicitly identified, but is generated as a side effect of the use of the averaging function to ensure good sensor values (Mataric 1991).

An alternative argument states that space and framework are culturally defined (e.g. Western: Euclidean and Newtonian) because non-Euclidean geometry and relativistic physics exist (O'Keefe and Nadel 1978). This view is also supported by the Nativists who point out that the agent only ever experiences an "Ambient", a slice of reality, the features of which are totally dependent on the agent's sensory repertoire. O'Keefe provides an argument against this view, based on evidence that all peoples use similar map structures, including those that are unaware of Euclidean and Newtonian ideas. If a representation is assumed to be present, these geometries are good candidates as they describe the primate ambient very well. Experience may not be sufficient to generate such ideas within the agent, implying that suitable structures are contained within the primate (and rat) mental hardware. The studied animals must perceive the universe in these terms, even if it is possible to conceptualise it in others (O'Keefe and Nadel 1978)(O'Keefe and Burgess 1996).

4.7 Natural Euclidean representations

To test the opinion that animals have a basically Euclidean view of space a few simple questions can be asked, though these do not provide rigorous proof, they do imply strongly that some framework exists inherently within the structures of the human mind at least. The questions are:

- As the eye naturally produces non-Euclidean data, why think in this way, which requires complex transformations of visual information? Surely if no inherent representation existed, and the perception of space were developed from experience, spatial representations would take the form in which they are sensed. Representation would be expected to fit visual information, as this is the primary sense in humans.
- If the human brain has been using non-Euclidean geometries all along, why are they new (in mathematical terms)?
- Why is Euclidean geometry considered common sense?

It seems reasonable to assume that if a representation of space is included in the structure of the brain, all thought would have a natural affinity for that representation. The rhetorical questions posed above thus imply that a Euclidean representation of space is inherent in the minds of animals.

These frameworks may be referenced to an absolute framework also. Many works have identified ways in which animals may use the earth's magnetic field, the night sky, or the polarisation of solar light, to reference their direction of travel (Collett 1996)(Collett and Cartwright 1986) (Lambrinos et al., 1998)(Walker et al., 1997). Distal objects may also be employed for this task (Collett and Cartwright 1986)(O'Keefe and Burgess 1996).

4.7.1 Hippocampal maps

O'Keefe proposes a mixed representation in the mammalian mind: In most of the brain, space is represented relatively, referenced to the organism and built up by experience. This structure builds up egocentric "Taxon" space. However, there also exists a Kantian or "Locale" Euclidean space. This forms a framework for holding objects, independent of both the objects and the observer. Location is an indirect function of movements relative to the object. "Locale" space is allocentric.

Solutions to the identification and localisation problem in Nativist and Empiricist theories tend towards the existence of some innate knowledge of space, position and place. Even the use of reference set objects requires that objects are available that are suitable references and that they may be localised in some unitary spatial framework or identified spatially.

O'Keefe suggests that animals are able to identify places, using the environmental cues around them. When a place is recognised activity occurs in a certain area of the hippocampus[1], referred to as a "place field" (O'Keefe and Burgess 1996). These place fields form the basis of two major systems that form the animal's cognitive map:

1. **Place system:** Contains information about places and relations between them. It also illustrates which objects are at which places.
2. **Misplace System:** Produces a signal when a new object is discovered in a place, or an existing object disappears. The misplace system identifies changes in the environment which should be echoed in the map.

The place system allows an animal to locate itself in a familiar environment by identifying individual "Places" and allows for the automatic linking of places and concepts not previously linked conceptually. The place system is independent of senses and actions.

The misplace system is associated with exploratory behaviours. An animal will naturally try to complete its place map. To do this the animal moves around. Where the present location can not be identified with a place field, this will be identified by the misplace system, and a new place field added.

When a location is identified within the place system, the animal compares the expected contents with what is found. Any discrepancy is highlighted by the misplace system.

---

[1] A small area of the brain, shaped, in humans, like a sea-horse.

4.8 Navigation: how to move about

Navigation, the act of moving in a directed manner from one location to another requires more than a representation of space. This representation must be used to generate appropriate motor actions to take the navigating agent to the goal location.

Where a complete topological map of the environment is available, any location may be reached from any other, but routes or paths must be developed, and the position in the map and its orientation with respect to the environment must be accurate. Alternatively, routes may be learnt as a series of actions that must be made at specified locations. This technique has the advantage of simplicity, though it is less robust or flexible.

4.9 Three ways to reach a goal

The manner in which a spatial representation is used to generate an action within an agent falls into one of three categories:

1. **Piloting:** Steering by familiar landmarks.
2. **Compass Steering:** Heading in constant compass direction (sometimes referred to as using "goal vectors").
3. **True Navigation:** Heading towards a specific goal regardless of original starting position and orientation.

Piloting is the act of following a series of stimulus response pairs to follow a route through space. An example of this type of navigation is a driver following a route by use of road signs. As each sign is reached, an action is performed. An equivalent technique is employed when navigating via a series of known landmarks. This is a relative view of space; the value of the guide locations is the relationship between them, not absolute position. A system that implements this behaviour corresponds to O'Keefe's "Taxon system".

Compass steering is far more primitive, and not suitable for crowded or complex environments where a straight path cannot be followed. It is related to the blind navigation that is required when at sea. The present location is determined by dead reckoning, the heading to the goal is calculated, and followed. It is effectively a route with a single stimulus.

True navigation allows the spatial relationships between locations to be exploited. The recognition of a location enables the agent to fix its position on the map, which is then used as a model to determine the correct action to take to reach any desired position, provided it is represented on the map. The use of maps in this manner is the responsibility of "Locale systems".

4.10 On Maps and Routes

Locale and Taxon systems place vastly different demands on the representation of space employed, different representations are required for each. These two representations are routes and maps.

- **Route:** An elaborate set of instructions.

- **Map:** Illustrates the start and goal locations and features that lie in the vicinity

4.11 Routes

Routes are composed of a sequence of Stimulus-Response (S-R, or SR) pairs. These pairs may be divided into two main categories: guides and orientations. A guide is a Stimulus Response pair with emphasis on the cue, while an orientation has emphasis on the required response.

4.11.1 Guides

To use a guide, an agent moves in such a way that it maintains an egocentric relationship with the guide feature. Guides are often localised, stationary objects. This is not always the case however; a stream or a line drawn on a floor can be a very useful guide. The utility of these features as guides is increased as they are extended rather than localised. Indeed the best guide is neither localised or stationary, it is a fellow traveller that knows the way. No specific motor behaviours are associated with guides; any action that maintains the relationship is appropriate.

4.11.2 Orientations

Orientations are indications of the required alignment of the agent's body with some external axis. The major problem associated with the use of orientation is the maintenance of the orientation over time using the available senses. For this reason the use of external cues to maintain the orientation may be appropriate, suitable cues are landmarks and distal objects.

4.11.3 Using routes

The use of routes is highly goal oriented, where each of the stimuli is a sub-goal. The route is really a sequence of actions which must be performed to reach the goal point. No information is included, other than that directly concerned with the reaching of the goal location. The knowledge required to follow a route is simple. Guides and orientations must be recognised, and in some situations distances measured (the measurement may be relative).

Most importantly all the information and skills needed to follow a route are available before the journey commences, limiting the computation required during travel. Because of this, routes do not limit the speed of travel of an agent.

Unfortunately routes are inflexible. Each is applicable to one journey, in one direction, only. Reversal of route cannot be assumed, as in different directions the orientations used to plot the

course will no longer be valid and the relationship that must be made with a guide may not be possible. Extended guides may still be applicable, and some orientations may be reversible, but this can not be relied upon.

The failure mode of a route is catastrophic. Once the route is lost the agent no longer has a representation of space in which it may locate itself, and has insufficient information to generate a path back to the route, even if its location was known. Failure may be caused by degradation or a misunderstanding of instructions, the destruction or movement of guides or orientation marks, the imperfect translation of the instructions to behaviour or inability to recognise stimuli.

Most people seem to have some experience of the failure of directions to get them to their destination, and anecdotal evidence from visitors to such towns as Milton Keynes, which are laid out on a grid system suggests that the effectiveness of such a system of navigation is limited. Admittedly this would appear to be due to the difficulty inherent in describing a complex scene, to act as a stimulus point. The knowledge that routes have these problems is held by most travellers, this knowledge may lead to anxiety. The author has on many occasions suffered terribly when travelling to an unknown location, due to being unsure as to whether the route has been followed correctly. Routes rarely provide the cumulative feedback required to alleviate this stress.

4.12 Maps

Maps are a far more complete representation of space. A map is a representation of a set of places systematically related to each other by a group of spatial transformation rules. Places and spaces are logical primitives that are unreduceable.

As objects cannot define spaces, maps tend to rely on some form of matrix, each cell of which corresponds to a location. But those objects contained by the location may be used to describe

it, allowing areas in space to be referenced by the objects that they contain. This is the basis of relative mapping.

Maps fall into two categories, topographical and thematic:

4.12.1 Topological maps

Topological maps are used to represent the relationships between features of a space. The most common form is the topographical map.

Topographical maps present the topology of an area diagrammatically, symbolically showing both the features of the landscape and also the relationships between these features. Using such a map entities and features may be located in space. Topological maps usually employ Euclidean geometry, though there is often some scale distortion. This distortion is especially likely with small objects that may be given some exaggeration to prevent their loss in the map. Topological maps may be used for navigation.

4.12.2 Thematic maps

Thematic maps display or emphasise particular selected features or concepts such as distributions. These maps represent not just facts but also ideas and concepts and the results of analysis. A thematic map is not suitable for navigational purposes as it does not represent the relationship between locations.

4.13 Localisation

Maps require localisation and orientation within the environment which they represent. The agent must locate its present position, both on the map and within the space it represents. When the agent's location in both has be identified, it can be used to calculate courses (a multitude may

be possible) through the map to the goal point. These courses are then translated to the "real world".

The flexibility of maps comes from their general representation of the whole of the space covered. Further flexibility is created by the lack of dependence on specifics. If a route is blocked, one of the others may be chosen. If a landmark is destroyed others may be identified. If the agent becomes lost, the map contains information that can be used to determine present position. Unlike routes, maps are not goal oriented. A map defines no starting point or goal. It can be used to navigate between any locations represented.

Maps are also high in content; a single change by the addition of an object changes all possible routes around the vicinity of the change. Maps have the ability to reassure. While following a route the map may provide information and make predictions about what to expect. A map can be extensively degraded before it falls into uselessness.

4.14 Map Problems

Failures when using maps are likely to be less catastrophic than those experienced when following a route. However, there are several disadvantages associated with maps, especially cognitive or internalised maps. An early investigation of the use of "imaginary maps", highlights some of the failures that may occur (Trowbridge 1913). These may be summarised as:

- Coding and decoding from a map requires time and effort.
- The route needs to be chosen from the map.

An ideal situation would be to use a map to select the route, then obtain the route as a good set of directions which are used while travelling. This is effectively what happens when a traveller uses a road atlas to plan a route, or when a navigator gives instructions to a driver.

4.15 Animal navigation

Very few animals exhibit no navigational ability and though the nature of this ability may differ greatly across the animal kingdom there are some common recurring features between most. The directed movement of animals ranges from the chemo or photo-taxis of single cell organisms, to the complex behaviour demonstrated by mammals and birds.

It may be sufficient that an animal can move through an environment without "getting into trouble" by avoiding areas of danger and with a lack of resources. Many animals, however, have permanent nests, hives or homes and can reliably return to these after foraging. Resources are rarely distributed evenly throughout the environment and the ability to navigate from home towards sources of known food is thus as important. The mechanisms used to achieve this are dependent on the structure of the environment in which the animal acts and on the complexity of the navigational task on which the animal is engaged.

4.16 Capabilities

Before any discussion may be made on the capabilities of animals to navigate it is important to remember that the benchmark by which all achievement in the field of Artificial Intelligence is ultimately measured is human ability. This anthropocentric view of the universe influences human perception of the capabilities of animals and artificial systems. This colouring of the perceived usefulness of systems directly effects the type of machines that are designed and constructed.

4.17 Birds and mammals

Birds and mammals can, to some extent use all the navigation types listed above. The navigational repertoire of insects however is smaller, due to a reduced amount of computational

power resulting from the limited number of neurones (900,000 in the honeybee (Collett 1998), compared with the 10^{10} to 10^{12} neurones within the human brain (Patterson 1996)).

4.18 Use of maps and landmarks

Tolman provided evidence that rats had the capability to produce maps of the environment that contained geometric information (Tolman 1948). He also extended this work to speculate on the nature of mental maps employed in humans. He found that rats tended to use stereotype paths when possible. This implies that they prefer to use a simpler mode of navigation if possible. The mazes in which the rats were run ultimately also favour the use of routes by reducing the range of possible actions to a set of discrete choices. A closer examination of how the structure of a space influences suitable representations and navigational techniques is given in chapter 6.

4.18.1 Birds

"Map and compass" behaviour is the most complete way to explain true homing behaviour in birds, this requires true navigation. The compass may be formed in birds either by observation of the sun, or by sensitivity to the geomagnetic field. Evidence for this can be seen in the inability of young pigeons to home if they have magnets tied to their heads. This is so even if the sky is clear, when solar information is easily available. An interesting aside is that this implies that pigeons must learn the relationship between the position of the sun, their body clock and magnetic compass position (O'Keefe and Nadel 1978)(Ridley 1995).

Some birds (not pigeons, which will not fly at night) also use the stars as a navigational aid, by the construction of star maps. These are not innate, they need to have viewed the star field before the map can be generated and navigation commenced. The map is formed of a framework, which is based on the stationary pole star and easily identifiable starts that rotate

around it. The framework is referenced to the internal "body-clock" of the bird, to determine position and orientation.

Birds that produce caches of food have additional problems. They must be able to return routinely to a number of locations. A geometric map of course provides an ideal mechanism for such navigational tasks. Clark's Nutcracker, a seed storing member of the crow family, has been shown to store information on the geometric relationships between landmarks (Kamil and Jones 1997). This map is used to recover stored food.

4.18.2 Wolves

Wolves also exhibit complex map using behaviour. That a map like representations are used can be determined by observing that:

1. Wolves can take intentional shortcuts or detours
2. The pack can split up and regroup at some other point which lies beyond howling range.
3. Wolves can return to rendezvous point with pups from any direction.

These capabilities strongly imply the existence of some concept of space and maps in the wolf. These maps are likely to be multi-modal, and highly dependent on smell, as this is the primary sense employed by dog like animals.

4.18.3 Gerbils

Collet and Cartwright made extensive studies of gerbil behaviour and determined that they use landmarks to describe space in a map like manner, but that they also use simplified transforms to extract information from this map (Collett and Cartwright 1986).

A brief summary of their findings follows:

Gerbils use three types of landmarks:

1. beacons
2. compass
3. topographic

The beacons employed are close to the goal location. The gerbil will align with the beacon and move towards it, when within range of the goal, alternative navigation techniques are employed. This is a taxon system of navigation.

Compass landmarks are those that are at sufficient distance to appear stationary. These are used by a gerbil to locate itself within space and align its topographical map.

Topographic landmarks require the most information to be recorded, they are used to calculate the position of inconspicuous goals by the relationship between them and the goals (where the goal may be easily sensed it is used as a beacon, saving computational time). The use of topographic landmarks implies complete geometric knowledge of the environment. Such knowledge is required for the computation of all relationships between locations and objects. Gerbils exhibit Euclidean spatial knowledge. Evidence for this is provided by experiments where a gerbil may calculate a trajectory to a goal in a darkened room, providing it was given the opportunity to localise itself before the lights where extinguished.

Gerbils save computation by using landmarks independently (not to localise, for which they use the whole landmark array, but to calculate goal position). This is biased by proximity to the goal. Ambiguity may be resolved by a voting system between landmarks.

If a goal point lies centred between two landmarks, which are then moved so that there is a greater distance between them, the gerbil would continue to check at the original distance from either of the landmarks. There would be no averaging or calculation. A landmark that was

originally closer to the goal will be given a greater degree of trust by the gerbil. If there are several landmarks that suggest a similar location for the goal, they will have precedence.

4.18.4 Insects

Insect navigation has been highly studied. The small size of insects makes this simple, but also helps highlight the problems that are associated with mechanical navigation. As mentioned previously, a honeybee has some 900,000 neurones. This is significantly more that the average artificial neural network used for machine navigation. The limited processing associated with insects seems to preclude the use of complete geometric maps for navigation (Collett 1996). It has, however, been shown that compass like mechanisms, landmarks and optical flow may all be used by insects to aid in the navigational task. The complexity of the models required to explain insect navigation suggests that a multi-modal technique is employed (Collett and Cartwright 1986).

The basic mechanism of bee navigation appears to be retinal image matching of one type or another. Optical flow may also be used to determine movement or flight path error. A brief overview of the of insect navigation, particularly hymenoptera species follows:

Navigation to a goal point appears to require three stages (Collett and Rees 1997), employing four techniques (Collett 1996). During the navigation task, path integration is employed such that the bee, wasp or ant may return directly to the nest or hive (Collett and Cartwright 1986).

The three stages of navigation towards a goal are:

1. Aiming at landmark

   The bee selects a landmark that is easily visible, and in the direction of the goal location and flies towards it.

2. Motor trajectory image association.

   When the image of the landmark is central to the retina, presumably also at the correct scale, a learnt vector is flown, taking the bee from the vicinity of the landmark towards the area of the goal.

   This technique is made more powerful as the movement of the image across the retina may be used to calculate the error in the trajectory.

3. Scene matching

   The goal location is finally identified by matching the image of the scene with a stored image showing the relative positions of all nearby landmarks.

The four techniques that are employed by the ant, bee or wasp in navigation may be simply stated. They are directly related to the stages listed above, but are worth repeating as they clarify the mechanism employed through the stages:

1. Recognising scene

   The first part of an ant, wasp or bee's journey depends on matching of the scene in localisation. Once present position has been determined other techniques may be employed to reach the goal. Interestingly, when first leaving the nest or hive insects often follow stereotyped directions. A goal vector can thus be assumed to exist before the journey begins. Displacement studies have supported this view.

2. Biased detours

   Landmarks are used to correct errors in the goal vectors being followed. The visibility of landmarks encourages this. A sequence of such detours may be used in long journeys

forming a route. These detours are used to support the basic dead-reckoning technique which seems to form the basis of all insect navigation.

3. Aiming at beacons

   Biased detours may be further refined by aiming at a sequence of beacons and flying or travelling directly between them.

4. Image mapping

   At the end of the journey, the perhaps inconspicuous goal must be achieved. For this reason the relationship with nearby landmarks must be remembered. A panorama image is stored at the goal location. Regaining this location requires that the insect move until the image seen matches that which is stored.

The length of the journey causes problems that need more subtle techniques to solve. Scene matching algorithms deteriorate with distance. Landmarks that are close to the goal will appear to move more than those at greater distance with path errors. Cartwright and Collett (Collett and Cartwright 1986) suggest that a "distance filter" may be employed that accounts for this potential error, and removes close landmarks from the retinal matching process at distance, and returns them when the goal is close and a finer degree of navigation is required.

Note that all the stages and techniques of navigation listed above may be categorised as "Taxon system" procedures. All stages are concerned with the maintenance or achievement of egocentric relationships, or axis orientations.

4.19 Animal compasses

All of the navigation algorithms mentioned above require the use of some form of compass to align the map of the environment held by the animal with the "real world". Both true geometric

maps and routes require alignment between the animal's mental image and the real world, although in route following, initial location and orientation only, need to be enforced.

Several mechanisms suitable for this real world/ map alignment have been identified:

- Solar position (Dickinson and Dyer 1996)(Collett 1996)
- Light polarisation (Lambrinos et al., 1998)
- Magnetic (Ridley 1995)(Walker et al., 1997)(Kirschwink 1997)
- Distal objects (compass points) (Collett and Cartwright 1986) (O'Keefe and Burgess 1996).

The initial scene recognition seen in hymenoptera, is a form of compass point alignment.

Rats have also been shown to use allocentric information to align themselves in mazes (O'Keefe and Burgess 1996). It should be noted that there is likely to be a different mechanism employed by animals such as rats that tend to live in maze like environment, and those that live in more open conditions. Intuitively it would be supposed that such animals would employ a "maze space" view of the world. However, the information contained in the works referenced in this chapter suggests that this is not the case. All reasonably complex animals employ both taxon and locale systems of spatial representation, and it is the level of emphasis placed on each of these that make the difference.

4.20 Path Integration

Path integration is a common form of returning to a point of departure. It appears to be the fundamental navigation mechanism of insects (Collett 1996) (Lambrinos et al., 1998)(Prescott 1994). Path integration effectively creates a polar view of the environment. It is performed by integrating the headings taken on an outward journey over time.

Successful navigation by path integration depends on the ability of an animal to measure either the absolute heading using some form of "compass", or heading relative to the path previously followed. The usefulness of each of these two techniques depends on the features of the environment in which the animal moves. Another requirement of path integration is a measurement of distance, or of time and speed. Bees have been shown to calculate their speed of travel by measurement of the frequency at which objects seem to pass, this may not be the only mechanism however.

The basic technique of path integration requires that the animal records its heading, and how long it is maintained. Integration of this data reveals the direct heading from the start point. To return home the animal merely has to reverse this heading. (Collett and Cartwright 1986).

Such techniques obviously suffer from cumulative errors, and so are often shunned by the mobile robot and artificial intelligence communities. The success of insects, however, suggests that when used in conjunction with other simple route following techniques, path integration is a successful method of returning to a start point.

4.21 Landmark recognition

Perhaps the most important navigational task for the techniques described above is the recognition of landmarks. Landmarks form the guides and orientation points in taxon system. They allow animals to locate their present position in space and their spatial representations, allowing the two to be aligned.

Evidence for human reference to external networks may be obtained from experiments performed in darkened rooms with illuminated points and frames (Vernon 1962). In experiments where a seated subject is presented with a small illuminated dot, after some time the dot will appear to start to move randomly, although both the dot and subject are stationary. The

apparent movement of the dot may then be halted by surrounding it with an illuminated frame. Moving the frame will cause the dot to appear to move, while the frame is perceived as stationary. As the movement of the frame continues, the subject may feel discomfort, as they feel that they are moving.

The initial movement that is seen in the dot is a result of the very small natural movement in the eyes, causing its position on the retina to change. Evidence provided by other senses, which would identify movement in the joints or muscles, or measure the force of gravity, is sufficient for the body to be sure it is stationary. Thus the movement of the image is assumed to be cause by movement of the dot.

The provision of a frame provides sufficient information to cause the dot to be correctly identified as stationary during the second phase of this experiment. When the frame is moved however, the dominance of the visual sense and the "frame" archetype are illustrated. The conflict between the proprioceptive and gravitational information and the dominant visual sense means that discomfort may be experienced, or movement of the body may be incorrectly assumed.

Perhaps the most telling point of the experiments reported above is the discomfort felt when the subject was unsure of their relationship with the world. Trowbridge also reported that subjects developed a feeling of sickness during his map experiments, when they realised they were in error (Trowbridge 1913). The suggestion of these reports is that "knowing where you are" is of very high importance.

4.22 Machine navigation

Machine navigation has been achieved for both local and way finding subtasks, using a variety of techniques. Most of these are analogous to those found in nature. There is an additional class of machine navigation, however, that of path planning.

4.22.1 Path planning

Path planning uses total prior knowledge of the environment to calculate a trajectory through the world. The vehicle controller is then tasked with following this path, effectively a two (or three in the case of manipulators, underwater or flying vehicles) dimensional servo-controller problem.

Path planning algorithms may be used to calculate optimal paths through known spaces. If the space is known then the planning of such a path is the obvious solution to the navigation problem. The intelligence is taken from the vehicle.

Many techniques have been used to calculate this path, however, as the complexity of the space is increased, the computational time required to achieve a successful path increases dramatically.

4.22.2 Beacons and biased detours

Localisation is an important consideration. How does the machine know where it is? This is also a problem for path planning systems. A "perfect path" is useless if the machine has no way of following it. Beacons can be used to provide location information to the machine.

Beacons may also be used in a manner analogous to that of insects. A recognisable landmark may be used as a way-point in a defined path (Baptiste et al., 1994). Landmarks are then "snapshots" of the environment from a particular location. These methods approximate a "taxon" system.

Indeed the most popular and robust method used for navigation in commercial autonomous vehicles is in fact a taxon system. The coloured floor tape employed in the line following robots popular in many warehouses is a guide to which the vehicle maintains an egocentric relationship. The simplicity of this method of navigation is its major advantage for an industrial setting.

4.22.3 Maps

Toto (Mataric and Brooks 1990) is an illustration of O'Keefian mapping system. A relative map of the environment is constructed and is referenced to an external frame, provided by information from a magnetic compass. The same techniques may be employed to locate a machine within an absolute topographical representation of the environment. In this case a path-planning algorithm will be applied to the map to calculate a trajectory that may then be modified and updated as the vehicle moves. These techniques have the advantage of appearing to be secure and robust. If a path is discovered to be blocked a new path can be developed using the additional information in the planning algorithm.

Machines have been created using both relative (Mataric 1991) (Mataric 1997) (Nehmzow and Smithers 1991) and geometrically accurate (Thrun and Mitchell 1993) (Nourbakhsh 1998) (Kortenkamp et al., 1998) maps for navigational purposes.

4.23 Summary and conclusions

To briefly summarise, navigation, the act of directed movement, requires knowledge of goal location, its relationship to the present location of the agent and the relationship between the agents motor actions and space. Navigation cannot be attempted without prior knowledge of these relationships and the major occupation of the intervening space. Assuming that knowledge of the relationship between motor actions and space are implicit to the agent the requirements for navigation are:

- Representation
- Orientation
- Location

All three activities must be considered in the design and implementation of an agent which is required to move to a goal location.

4.23.1 Representation

The storage of this spatial knowledge requires the use of some form of representation of space. This is traditionally thought of as a map. There are, however, alternatives, such as routes. Routes are primitive maps, where the complex dimensionality of the space in which the machine or animal exists is reduced. The representation becomes a simple mapping between certain sensor triggers and motor actions. Ultimately the representation may be taken from the machine itself and painted on the floor of the workspace.

Where a map is used, it is important to remember that maps may be quite inaccurate metrically as long as the topographical relationships between the identified locations are maintained.

That the type of representation required depends on the capabilities of the agent may be inferred from the different techniques employed by different classes of animal. It would be easy to assume that this was a direct result of the different levels of complexity found in the brains of these animals. However, animal mental hardware did not evolve independently of the navigational techniques employed and neither developed independently of the animals environment. Remembering this fact allows the casual link to be reversed. A rat is capable of following stereotyped routes while running a maze, this is the observed fact. From this observation it is not unreasonable to assume that rats posses this ability as the usually inhabit 'maze-like' environments. Following the same reasoning the insect ability to perform path

integration and employ a solar compass for navigation allows navigation of open spaces in daylight. These abilities developed in insects as a direct result of the environment.

The same argument must be followed when the representation method employed in a mechanical agent is developed. The representation should exploit those features and properties of the environment through which the agent is to move. Application of this argument to the environment experienced within the built environment provides the motivation for the work outlined in chapter 6.

4.23.1.1Routes

As stated previously (section 4.11), routes may be defined as a series of Stimulus-Response pairs, or Stimulus-Response-Stimulus triplets, which describe the desired path as behaviour to be taken at each location. Route following like this greatly compresses the amount of information that must be stored and reduces the amount of processing that must be performed when following a path, compared to the extraction and following of paths from a topographical representation of space. These gains, however, are not without cost. Use of routes reduces flexibility and robustness.

When an agent is allowed little flexibility, either by the application task, or environment to which it has been applied, the use of routes for navigation purposes is suitable. That a rat running a maze tends to use stereotyped routes where possible supports this view.

4.23.2 Orientation

The use of any representation of space requires that the machine or animal may align its representation of space with the space in which it exists. For route following, landmarks may be sufficient once the initial orientation is achieved. Alternatively, or for true topological map use, some form of aligning compass is required.

Distant objects may be used to define directions as their apparent position moves less with movement of the vehicle. The objects may be as distant as the sun or stars, or may be far closer. The distance to useful compass objects is determined by the scale of the navigation task.

Alternatively a real compass may be employed to align the map with the earth's magnetic field. Active beacons may also be used to provide an artificial reference field, though this is unlikely to be magnetic.

4.23.3 Location

Following orientation of the representation with the "real world" the machine must determine its position in the world, and hence the representation employed.

Initial starting points may be provided and the position at all times after this extrapolated from movement (dead-reckoning), this system however has a tendency to cumulative errors. Alternatively beacons and landmarks may be used, the relationship of the agent to the beacon providing the required positional information. Recognition of the landmarks has problems of direction and scale. Even a simple object may look remarkably different when viewed from different directions or distances. A mechanism is required to account for this. Bees have been shown to approach landmarks from a fixed direction and low angle, thus negating the problem.

Finally, for machine navigation, it is possible to broadcast the position of the robot from a fixed observation point, which may make geometric measurements. Active beacons and Global Positioning systems may be used in this manner.

Several similarities exist between the localisation task and that of orientation. In many ways they can be seen as being different aspects of the same operation: relating the perceived environment to the internal representation. As with other elements of the navigation task, exactly how this is

performed depends somewhat on the structure of the environment and the capabilities of the navigating agent.

A solution for the navigation problem as experienced in a particular set of environmental conditions is given in chapter 6.

4.24 References

Baptiste, P., Bideaux, E., et al. 1994. Use of Perception Based Navigation for Autonomously Guided Vehicles. *1994 Joint Hungarian-British Mechatronics Conference* : 279-84.

Calvin W. H. 1993. "The Unitary Hypothesis: A Common Neural Circuitry for Novel Manipulations, Language, Plan-Ahead and Throwing? *Tools, Language and Cognition in Human Evolution.* Eds GIBSON K.R. and INGOLD T., pp 230-250. Cambridge University Press.

Collett T.S. 1996. Insect Navigation En Route to the Goal: Multiple Strategies. *Journal of Experimental Biology* 199: pp 227-35.

Collett T.S. 1998. How Are Studies of Insect Navigation Useful to Robotocists. *Self-Learning Robots II: Bio-Robotics* IEE.

Collett T.S. and Cartwright B.A. 1986. Landmark Learning and Visuo-Spatial Memories in Gerbils. *Journal Comp. Physiol.* 158: pp835-51.

Collett T.S. and Rees J.A. 1997. View Based Navigation in Hymentoptera. *Journal of Comparitive Physiology A - Sensory, Neural and Behavioural Physiology* 8, no. 1: pp 47-58.

Dennett D. 1993. *Consciousness Explained.* penguin.

Dickinson J. and Dyer F. 1996. How Insects Learn About the Sun's Course: Alternative Model

Approaches. *From Animals to Animats 4: Proc 4th Int Conf Simulation of Adaptive Behaviour*, Eds MATARIC et al, pp193-203MIT Press.

Duchon A.P and Warren W.H. 1994. Robot Navigation from a Gibsonian Viewpoint. *IEEE International Conference on Systems Man and Cybernetics.*

Kamil A.C and Jones J.E. 1997. The Seed Storing Corvid Clark's Nutcracker Learns Geometric Relationships Among Landmarks. *Nature* 390, pp 276-79.

Kirschwink J.L. 1997. Homing in on Vertebrates. *Nature* 390, pp339-340.

Kortenkamp D., Huber M., et al. 1998. Integrating High-Speed Obstacle Avoidance, Global Path Planing and Vision Sensing on a Mobile Robot. *Artificial Intelligence and Mobile Robots: Case Studies of Successful Systems.* Eds KORTENKAMP D., et al., pp53-71. AAAI/MIT Press.

Lambrinos D., Maris M., et al. 1998. Navigating With a Polarised Light Compass. *Self-Learning Robots II: Bio-Robotics*, pp 7-1 - 7/4 London: IEE.

Mataric M. and Brooks R.A. 1990. Learning a Distributed Map Representation Based on Navigation Behaviours. *USA- Japan Symposium on Flexible Automation*,

Mataric M.J. 1991. Navigating With a Rat Brain: A Neurobiologically-Insprired Model for Robot Spatial Representation. *From Animals to Animats.*, pp169-175.

Mataric M.J. 1997. Behaviour Based Control: Examples From Navigation, Learning and Group Behaviour. *Journal of Experimental and Theoretical Artificial Intelligence* vol. 9, no. no. 2-3: pp 323-36.

Nehmzow U. and Smithers T. 1991. Map Building Using Self-Organizing Networks. *From*

Animals to Animats. eds MEYER J.A. and WILSON, S., 152-59. Cambridge Mass. London: MIT Press.

Nourbakhsh I. 1998. Dervish: An Office-Navigating Robot. *Artificial Intelligence and Mobile Robots: Case Studies of Successful Systems.* Eds KORTENKAMP D., et al., pp73-90. Cambridge MA: AAAI/MIT press.

O'Keefe J. and Burgess N. 1996. Geometric Determinants of the Place Fields of Hippocampal Neurons. *Nature* 381, no. 6581: pp425-28.

O'Keefe J., and Nadel L. 1978. *The Hippocampus as Cognitive Map.* Oxford: Clarendon.

Patterson. D. W. 1996. *Artificial Neural Networks, Theory and Applications.* Prentice Hall.

Prescott T.J. 1994. Spatial Learning and Representations in Animats. *From Animals to Animats 3: 3rd Int. Conf. Simulation of Adaptive Behaviour,* Ed Cliff D., pp164-73 Cambridge MA.: MIT Press.

Ridley M. 1995. *Animal Behaviour: A Concise Introduction.* Oxford: Blackwell Scientific.

Thrun S.B. and Mitchell T.M. 1993. Integrating Inductive Neural Network Learning and Explanation-Based Learning. *Proceedings IJCAI 93,* pp 930-936

Tolman, E.C. 1948. Cognitive Maps in Mice and Men. *Psychological Review* 55: 198-208.

Trowbridge C.C. 1913. On the Fundamental Methods of Orientation and Imaginary Maps. *Science* , no. 38: pp 888-97.

Vernon. 1962. *The Psychology of Perception.* Harmondsworth: Penguin.

Walker M.W, Diebel C.E., et al. 1997. Structure and Function of the Vertebrate Magnetic Sense. *Nature* 390, pp 371-76.

5: FUZZY BEHAVIOURAL CONTROL

5.1 Introduction

Behavioural control and fuzzy logic exhibit commonality in both purpose and derivation. Both are intended to aid in the production of intelligence in artificial systems, both are inspired by natural systems. Behavioural control, (Chapter 3: "Behavioural Control") is a simplified model of the synthesis of animal behaviour. Fuzzy logic models the symbolic processing of diffuse classes seen in human reasoning (see Appendix 1: "Fuzzy sets and fuzzy logic").

This chapter investigates the assumption that this commonality forms a natural affinity between the two techniques, determining whether fuzzy logic forms a suitable medium for behavioural controller implementation for an automatically guided vehicle. Following this a new, motor schema like architecture is proposed and implemented using fuzzy logic. The advantages of doing this are discussed, along with the issues raised by such an implementation. Additional discussions of the architecture are given in the three previously publish papers included in Appendix 4, which chart the development of the architecture.

The proposed novel mechanism is then extended to include inhibition between behaviours, and the integration of non-fuzzy behaviours into the schema architecture. Suggestions for a suitable design methodology are also made.

5.2 Fuzzy logic and behaviour

Fuzzy sets (Zadeh 1965) represent categorisation into diffuse groups. Fuzzy logic is the set of operations and axioms that may be applied to fuzzy sets. Importantly fuzzy logic is modelled on conscious reasoning as auto-perceived in the human mind. Conscious reasoning is based on language and symbology. Although the 'jury is still out' on the purpose and nature of consciousness, it would appear that the conscious mind is a synthesis of the output of the agencies which perform processing within the mind (Minsky 1988)(Dennett 1993). In the role of mental commentator it is reasonable to assume that communication and language must be important aspects of consciousness.

Mamdani's controller exploited this language and communication ability to model the operator of a plant (Mamdani and Assilian 1975). The controller does not model the physical processes within the operator's mind, but the conscious action of interpreting instructions. This is a subtle, but important difference. The fuzzy controller is modelled on the phenomenology, or experience of operating a machine, the conscious experience of the operator, who then expresses this in terms of verbal communication. Fuzzy sets are then used to extract information on operating the machine from this message, allowing the controller and operator to share a set of symbols suitable for the control of the plant.

By using this shared symbology, the formalised rules of fuzzy logic provide a method of describing the behaviour of a controller in human (linguistic) terms. It is considered that the commercial success of fuzzy logic is directly due to this linguistic bias which allows controllers to be described quickly. The number of washing machines, microwave ovens and cameras with fuzzy logic controllers available is due to the economical advantage this provides. The low cost of embedded computing; coupled with a reduction in development cost makes fuzzy controlled items cost effective while achieving suitable levels of operationally effectiveness.

5.3 Advantages to the use of fuzzy logic

By using a symbol set close to human speech, problems that are easier described in linguistic terms rather than mathematically may be solved more simply. As human understanding of the universe would appear to not be inherently mathematical, this has obvious advantages.

In addition to the advantage of communicability, fuzzy logic has other properties that make it an appropriate tool for the development of behavioural controllers. Fuzzy logic is capable of reproducing any non-linear mapping (Kosko 1992), although this does not mean that fuzzy logic is suitable for the implementation of behavioural controllers, it does at least indicate that it is capable of doing so.

Fuzzy logic also has operational advantages over other rule-based techniques. Where an inference engine manipulates only crisp sets, sharp transitions exist between classes of input condition. These transitions will be passed to the outputs of the controller. By extending the transition between conditions, fuzzy logic can eliminate this problem.

5.4 Fuzzy Logic and reactive controllers

As stated in chapter 3: "Behavioural control", reactive controllers provide a direct mapping between an agent's sensors and motors. With the ability to model any function, fuzzy logic may therefore be used as an implementation strategy for any reactive controller. Several controllers of this type have previously been employed, for example (Li 1994) (Li and Feng 1994) (Liu and Lewis 1994).

All controllers are intended to describe the behaviour of a system under certain environmental conditions, reactive and behavioural controllers are no exception. It is necessary for the system designer to determine that behaviour which should be exhibited when certain conditions are met. The use of fuzzy sets means that the transitions between different behavioural aspects will be

smoother, this has been shown to be an advantage (Bisset and Webber 1994). More importantly fuzzy logic allows these environmental conditions and their associate actions to be described and communicated simply, using language that has meaning for the designer.

5.5 Development of the fuzzy behavioural architecture

To illustrate the use of fuzzy logic for the construction of reactive controllers a simulation has been used. This simulated agent has been placed in a number of "benchmark" environments and its behaviour observed. It should be noted that the simulation did not involve complex spaces in order to reduce the complexity of the emergent behaviour. This is required if the interaction of agent and environment are to be understood with clarity. Even reactive controllers, where the behaviour of the vehicle is dependent only on the present sensor conditions, exhibit some notional state; introduced through the coupling of the controller, simulation (or vehicle) and the environment. The behaviour at any point is dependent on the sensor inputs at that time, these are in turn dependent on the present location and orientation of the simulation, which is in turn dependent on previously taken actions. The introduction of non-linearity into this loop makes even a simple system complex and dynamic (Lewin 1993). By reducing the complexity of the environment into which the simulation is placed, the complexity of the interactions may be reduced, allowing a clearer insight into the generation of behaviour.

5.5.1 Simulation

The simulation employed is itself simple. It is modelled on a generic vehicle employing two forward facing range finders. These range finders return the distance to objects within the environment along a line at an angle to the axis of the vehicle. For the purposes of this experiment these angles are ± 45°. The distances greater than the operating range of the devices return a value equal to the maximum operating range. This is fixed at 1m, a figure appropriate for the simulation of modern infra-red sensor which work on trigonometric principles. It would

be simple to increase the complexity of the simulation, however, the same configuration is maintained throughout this chapter. Doing so maintains a constant set of capabilities, giving a better insight into the compromises that need to be made to create a successful controller. The addition of a forward facing range-finder could relieve the problems encountered when the agent directly approaches an obstacle (section 5.6.2), however, not all problems may be solved so simply. For this reason the configuration has not been changed between runs of the simulation (except for re-parameterisation) forcing controller solutions to problems encountered.

For the purposes of the simulation forward movement and heading change have been separated. The controller may determine each. By dividing the control signals in this way the simulation is equally applicable to differential drive vehicles and those which employ a wheel-geometry steering scheme. The simulation vehicle is considered as a single point. An illustration of the configuration of the simulation is given in Figure 5.1.

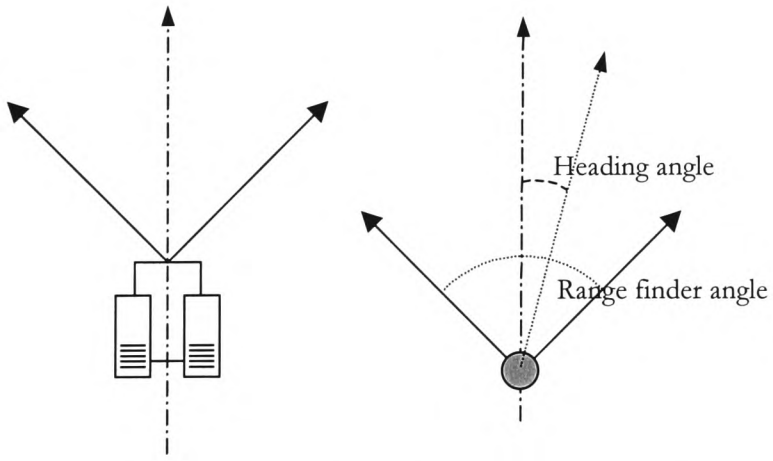


Figure 5.1 Configuration of the simulated vehicle

Such a simple simulation of a vehicle may at first appear to be of limited use, however more complex simulations are themselves plagued by inadequacies. It has been pointed out by many authors, including Brooks and Smithers (Brooks 1986)(Smithers 1994) that no simulation is ever complete. Both authors have used this argument to cast doubt on the validity of using

simulations for the design of vehicle controllers. However, if a real vehicle were employed, then the observed behaviour of the system would exhibit added complexity due to the inherent complexity of the vehicle and environment. By adopting a simulation, the primary interactions of the controller and environment may be closely observed independently of any specific properties of sensors or vehicles and parameters more closely controlled. The ease with which a simulation may be re-parameterised contributes to a deeper understanding of the functions that govern the interaction of the agent and environment, and helps highlight any critical values. Adopting a more realistic simulation of a vehicle lessens some of these advantages.

The purpose of the simulation ultimately dictates the required similarity to a physical vehicle. If the simulation is to prove a controller/vehicle combination then the simulation should be as accurate as possible, tests on the physical device being the ultimate expression of this. However, as a proof of concept, the simulation should be applicable to a general solution.

For the purposes of testing the fuzzy behavioural control architecture a limited turning circle is assumed. The radius of this turning circle may be changed as a simulation parameter. The controller can demand any heading angle that would not cause this limit to be exceeded. This may appear to be a harsh constraint to impose on a simulation that attempts to model a differential drive steered vehicle. This is not the case as although a vehicle that is steered by wheel geometry is inherently turning circle limited, so too are differential drive vehicles when under power. Experience with a range of differential drive models, including IEE micro-mouse competition entries and a MiRoSoT [Micro-Robot Soccer Tournament] robot football team has shown that the acceleration, both linear and rotational must be severely limited if a vehicle of this configuration is to exhibit predictable behaviour. Wheel-slip and the inertia contribute to a lack of control. The small scale of the vehicles mentioned, and their high relative speed undoubtedly exaggerates the problem. The much-vaunted ability to rotate around an axis placed

at the centre of a line between the driven wheels is unusual at anything but very low speeds and rotations in these small vehicles.

5.5.2 Environment

The environments in which the simulation has been operated are also very simple; again this is intended to help identify the interaction of the agent and the environment by reducing the complexity of the generated behaviour. The experimental environments used form "benchmark" tests by restricting the interactions to one or two major modes. Examples are interaction of a vehicle with a boundary object, a path constriction object, operation in constrained spaces and open spaces.

5.6 Simple controller

The most primitive component of the agent controller is a simple collision avoidance behaviour, employing a reactive control scheme. The scheme is, by the very nature of the fuzzy engine, a primitive motor-schema system, where the rules can be considered as simple behaviours or motor schemas, which are then marshalled by the defuzzification process, generating a single control signal from the varied outputs of the separate rules. There are two inputs, obtained from the range finders. In the simplest mode of simulation the output controls the heading angle of the simulated agent only. For a more complete simulation the speed of the simulation is also controlled. The collision avoidance task is easily solved using a zero order Sugeno fuzzy inference engine. To avoid collision the agent should turn away from objects. To do this the agent simply turns with an angle dependent on the measured range. The inputs are divided into three fuzzy sets, the output five singletons. The controller is represented by the FAM [fuzzy association matrix] (Kosko, 1992) shown in Figure 5.2.

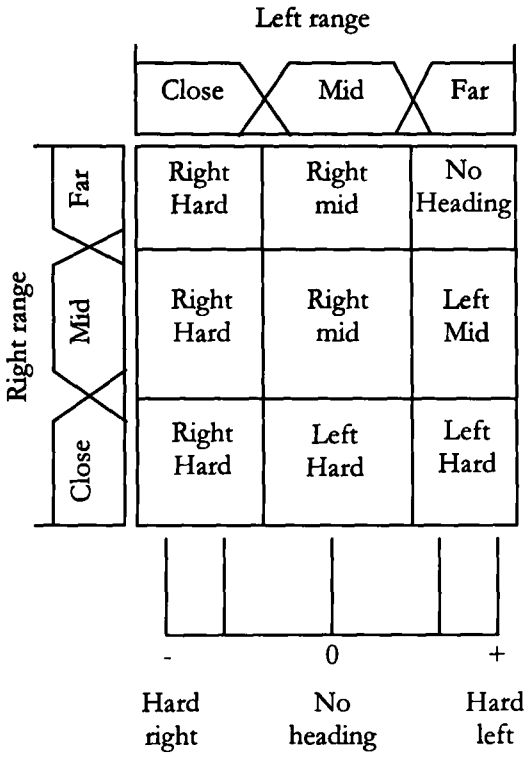


Figure 5.2 FAM of simple controller

5.6.1 Corridor experiment

This controller was initially tested with a simple experiment, equivalent to the agent being placed centrally within a corridor 170 cm wide, but headed towards one of the boundary walls at an angle of 60° to the centre line of the corridor (an illustration is given in Figure 5.3). This scale of environment is modelled on the corridors within the University of Wales College, Newport's Allt-Yr-Yn campus.

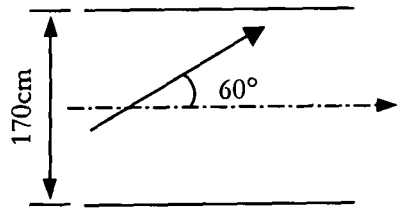


Figure 5.3 Simulation starting conditions

The inputs were fuzzified using a group of fuzzy sets described by the membership functions shown in Figure 5.4. The membership functions of both inputs were symmetrical.

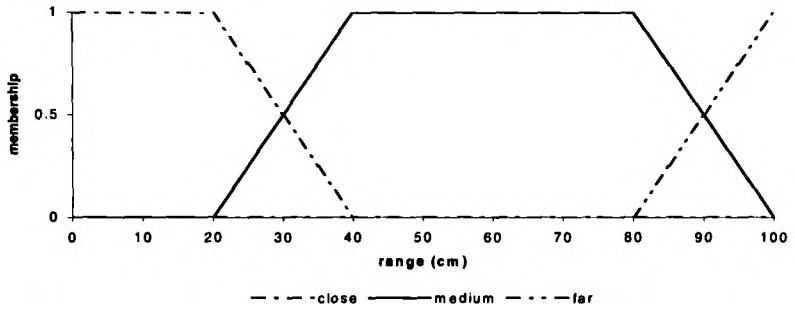


Figure 5.4 Membership functions for inputs

Output singletons were set to give left and right turn at the maximum turning circle, no change in heading (zero degrees) and two thirds of the maximum turning circle. The simulation was run for turning circles with a variety of radii.

The results for a selection of these runs are shown in Figure 5.5. The traces show the path of the simulated agent approaching the wall of a corridor, represented by the bold line across the top of the figure. The maximum turning radii for each trace is 60, 40, 20 and 10 cm from top to bottom respectively.

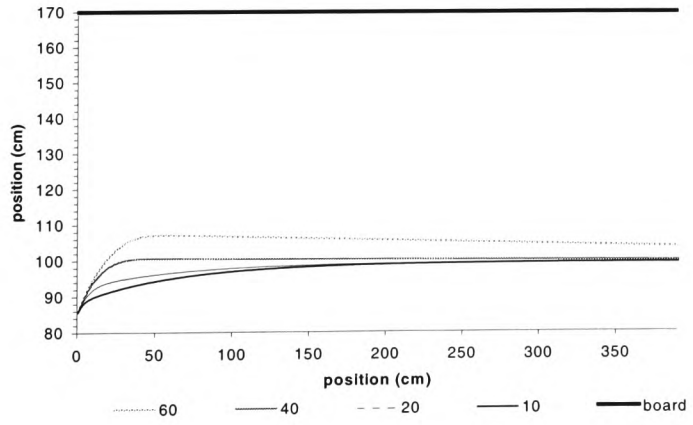


Figure 5.5 Approaching a wall, various turning radii (60, 40, 20 and 10 cm)

The traces of the diagram show some interesting effects. These may be seen more closely in Figure 5.6 where a portion of the diagram is repeated at larger scale.

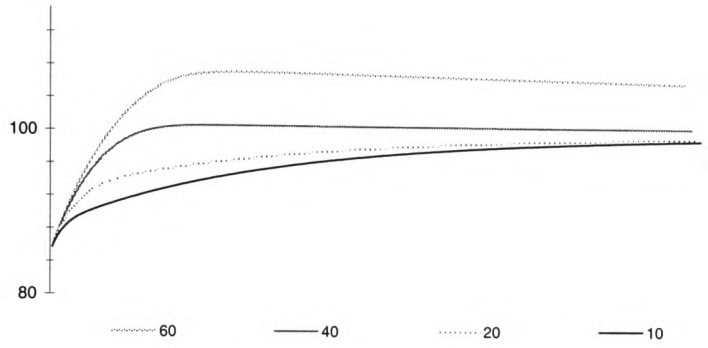


Figure 5.6 Enlarged scale diagram of agent paths

One easily observed effect is the smooth change in the output; there are no discontinuities as would be associated with the use of crisp set rule system. Smooth transitions are signified by the tightening in the radius of the agent's path, giving the ellipsoid appearance.

Another important observation is the apparent convergence. The two top traces, which show the runs where the simulated agent has the greatest turning radius, illustrate a 'bounce' effect, where the agent turns away from the boundary after a closest approach. The traces then appear to converge on a path parallel to the boundary at a distance of approximately (100cm from the ground plane). The bounce effect is caused by the agent coming too close to the boundary object. The agent continues to rotate, sweeping the contact point of the range finder, until the range is sufficient to remove the turn stimulus. The agent's heading is then away from the wall. The distance between this convergent point and the boundary is set by the operating range of the sensors and the angle at which they are set.

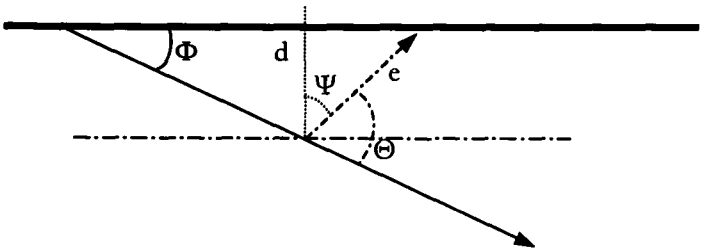


Figure 5.7 Agent at boundary

The angle of the path away from the boundary may be calculated:

$$\Phi = \cos^{-1}\left(\frac{d}{e}\right) + \Psi - 1.57^{c} \qquad (5.1$$

where:

d the distance between the agent and the boundary (cm).

e the operational range of the sensors (cm).

ψ the mounting angle of the sensors.

The appearance of convergence is only maintained on the approach to the boundary. If the agent is allowed to come closer to the boundary than the convergent distance, and 'bounces' the illusion is destroyed. Once the agent has adopted a heading away from the boundary there are no forces acting on it. The agent will continue on this course until falling under the influence of some other object. Fortunately the effect is reduced to a manageable level by the sensitivity of the range finders to the orientation of the vehicle. For the simulated agent a heading change of 5° from parallel reduces the distance from the boundary at which the agent experiences an effect in its range finders by 35 cm. If this were not the case, an oscillatory path between boundaries would be expected when traversing a corridor.

5.6.2 Constricted corridor

The interaction between a more complex environment and the simulated agent was tested with a constricted corridor. Figure 5.8 shows the effect of an encounter between the simulated agent and an object within a 'corridor'. The agent travels from left to right.

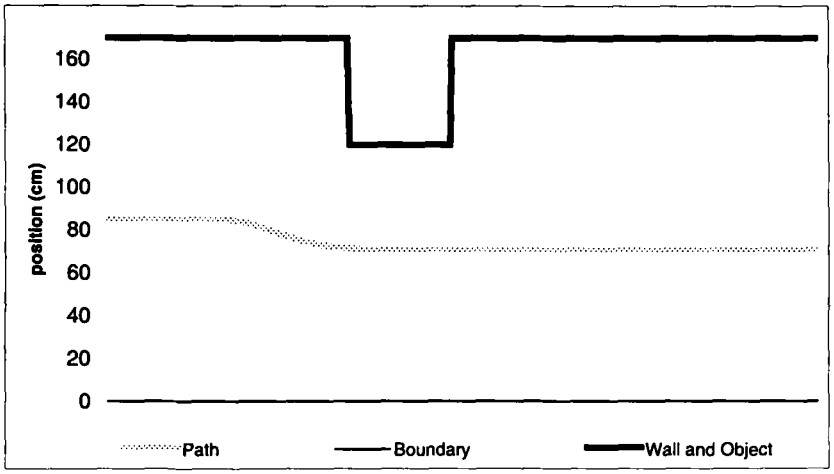


Figure 5.8 Object encounter

The corridor is 170cm wide, the object a square solid (of side 50cm) placed at the edge of the corridor. The agent turns on encountering the object and maintains its new course until the lower boundary wall is sensed. The final path of the agent is thus determined by the lower boundary.

Figure 5.9 shows, at greater scale the path taken by the agent when started at positions equivalent to 100, 110, and 120 cm from the datum, or lower boundary. The upper two traces (110 and 120cm) start under the influence of the upper boundary (at 170cm), preventing the agent from approaching the object straight on, an example of the complexity of the space masking some of the properties of the agent (as discussed in section 5.51).

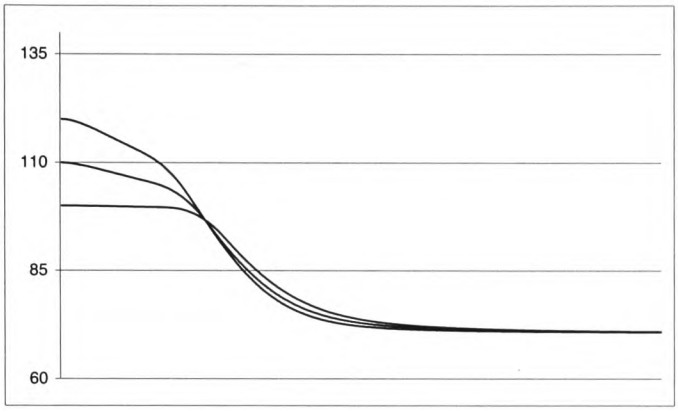


Figure 5.9 Object encounters from multiple starting points (increased scale)

For comparison Figure 5.10 shows the agent interacting with an object without the boundaries controlling its path. Once a path avoiding the influence of the object is achieved the agent continues on the new heading indefinitely. The complex multi-radius paths observed in Figure 5.9 are generated by the apparently simple system of objects and agent.

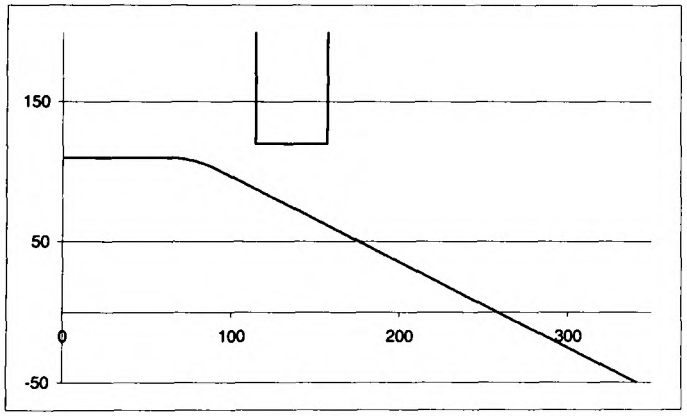


Figure 5.10 Encounter with object in free space

Not shown is the effect of an extended constriction within the environment. Such a situation may result in the boundaries of the space being close enough that the agent may sense both simultaneously, in which case it adopts a route equidistant from the two.

A direct approach to a boundary is shown in Figure 5.11. Here the vehicle was simulated approaching a boundary on a path perpendicular to it. The boundary is represented by the vertical line at a simulated range of 50cm from the starting position of the agent. The tightest turning radius of the agent in this experiment is 15cm. With just two range finders, mounted at 45° to the centre line of the simulation, the boundary is not detected until the vehicle is approximately 70cm from the boundary. When the boundary is detected the controller does not respond immediately. Indeed maximum turning circle is not employed until the range finders return a value of 20cm.

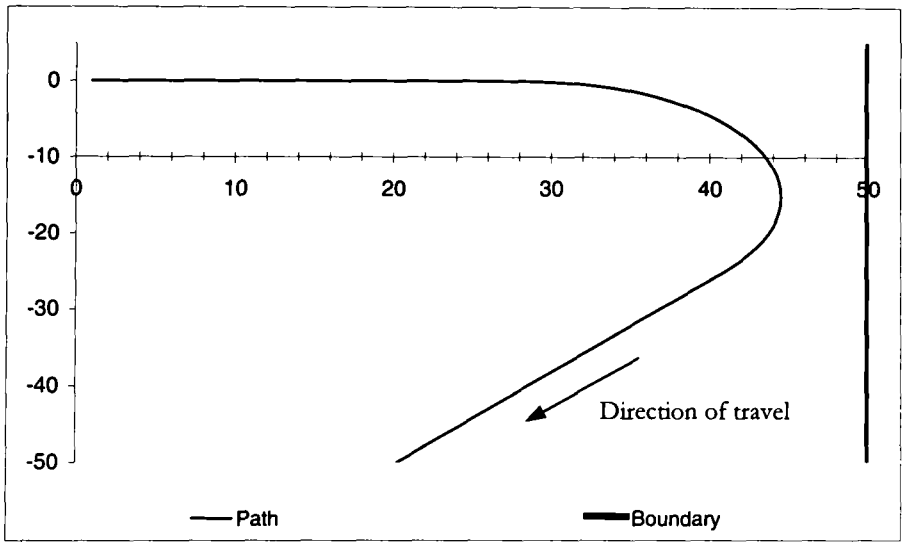


Figure 5.11 Agent directly approaching boundary

From the figure the danger of the bounce effect can clearly be seen. As the agent's path passes in the region of (45, -25) it leaves the influence of the object, its path has however been almost reversed. If the agent is engaged in some navigation task, such a response may be very damaging. The implications of this type of response to objects in the environment are further discussed in Chapter 6.

5.6.3 Path constraint

When placed into a corridor the agent behaves as shown in Figure 5.12. The effect of the 'bounce' effect may also be clearly seen here. After a period when the simulated agent travels away from the boundary on its new heading the lower boundary is detected. As soon as detected the boundary affects the path of the agent. The interaction of the agent and boundaries thus conspires to force the agent to follow a path parallel to a boundary at a distance dictated by the configuration of the vehicle sensors.

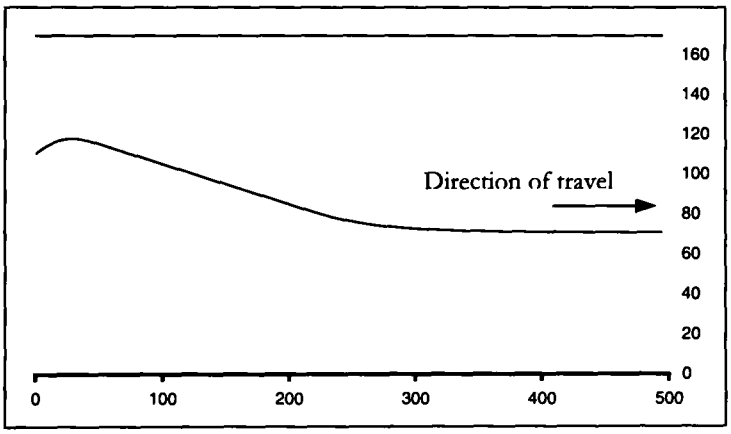


Figure 5.12 Agent in corridor

The convergence effect encourages the agent to travel in the direct of the corridor, while the proximity of additional parallel boundaries completes the effect by capturing the agent after a bounce. Provided the vehicle is capable of responding appropriately to the demands imposed by high boundary incidence angles the corridor is capable of effectively controlling the direction of the agent. As many mechanical agents find uses within human environments, where corridors and corridor-like spaces are common, this interaction between the agent and environment has implications for navigation (see chapter 6: "Navigation via Script in Constrained Environments").

5.7 Modifications to the controller architecture

The previous sections showed that a simple fuzzy system can be used to implement a reactive controller for single simple tasks. However there are problems that must be addressed if fuzzy logic is to be used to for more complex controllers, or to implement a behavioural controller. A major problem encountered when using a single FAM is describing the behaviours themselves. As mentioned in Chapter 3 (section 3.11), it is important to be able to prioritise the separate behaviours that combine to form the behavioural controller. Although it is conceivable that a controller could be carefully constructed so that a single set of rules describes all the possible behaviour of the agent without recourse to a separate prioritisation scheme, this would require consideration of the input space at a very fine grained level. One advantage of behavioural control is that the behaviour of a system may be described simply and in an incremental manner. The process of describing the behaviour as a single FAM diminishes this. To alleviate this problem a contribution factor is proposed (Tubb et al. 1997)(Tubb C.A. and Roberts G.N 1998a)(Tubb C.A. and Roberts G.N 1998b). This has the effect of prioritising each rule.

5.7.1 Contribution factor

Inspired by the desire to prioritise the atomic actions that make-up a reactive controller, the contribution factor allows the priority of each of the rules in the fuzzy system to be set. The mechanism acts like a firing weight modifier for each of the rules.

A zero order Sugeno system assigns weights to a set of constant values according to the firing weights of the individual rules (for a further explanation see appendix 1 section A1.9.4). The output of the engine is then calculated using a weighted beam function (equation 5.2).

$$output = \frac{\sum_{i=1}^{N} x_i \mu(x_i)}{\sum_{i=1}^{N} \mu(x_i)} \qquad (5.2$$

where:

x_i output constant

$\mu(x_i)$ firing weight of output constant x_i

This process may be easily extended to assign additional, permanent firing weights to each rule, allowing the contribution that each rule makes to the final output to be scaled. This "priority factor" is simply a second fractional multiplier manipulated in the same way:

$$output = \frac{\sum_{i=1}^{N} x_i \mu(x_i).p(x_i)}{\sum_{i=1}^{N} \mu(x_i).p(x_i)} \qquad (5.3$$

where:

$p(x_i)$ is the priority factor of the rule.

A priority value of 1 makes no difference to the operation of a rule. Any lower value will reduce the contribution that the rule makes to the output of the system. A value of 0 will prevent the rule making any contribution to the output.

The use of the priority factor allows subtle grades of meaning in the relationship between the rules and the control surface to be described. Rarely will a behaviour be fully described by a single rule in the fuzzy inference engine. In these cases each of the rules that make up the behaviour should be assigned the contribution factor that is required (Tubb et al. 1997)(Tubb C.A. and Roberts G.N. 1998a). For simple controllers this is sufficient. Unfortunately, the use of priority factors on each rule can make the design of the controller less intuitive where there are greater number of behaviours, or more complex behaviours to be considered. In these situation it also becomes more difficult to assign the contribution factor to each behaviour. In

these cases it would be advantageous to consider each of the behaviours as independent units, rather than a collection of rules within a single rule base.

In addition a problem is encountered when the number of inputs to the controller increases. When the number of inputs is low the behaviour of the controller may be described simply as a matrix. Actions are described for every possible input condition, especially when the number of fuzzy sets per input is limited. With an increase in the numer of inputs, or the number of sets these inputs are divided into, the number of rules explodes dramatically; determination of the correct behaviour at every cell of the matrix may become difficult. In addition, the ability of the designer to visualise the input space is reduced with increased dimension. For three-dimensional inputs a cubic representation may be used to illustrate the relationships, but convenient graphical representations become difficult for input spaces with greater dimensionality. For certain input conditions the behaviour of the controller may not be easily defined, a more generalised approach being appropriate. A solution to the problem of input dimensionality, the prioritisation of multiple behaviours and to the need to make generalisations about regions of the input space is required. The following sections propose a solution.

5.7.2 Simplification of the controller input space

To avoid the pitfalls of increased dimensionality in the input space, the space can be considered as smaller, manageable, independent chunks. This procedure also has the effect of increasing the functional similarity between the fuzzy motor-schema system and general behavioural controllers by ensuring that behaviours only receive sensor data that is required.

Conceptually the proposed structure is simple, however implementation does require that some issues are addressed.

Each of the behaviours is considered individually. The input data required by each is considered independently, and the rules required to generate the behaviour are developed. This is especially

attractive for behavioural controllers as it enables the individual behaviours to be separated in representation and implementation, as well as logically. In complex behaviours prioritisation may be required between the rules that form the behaviour.

If the behaviour of the system is divided between rule-bases like this, at some point a re-integration is required for control signals to be generated. The options for this re-integration are multiplex. Any of the basic schemes described in Chapter 3 may be adopted. However to preserve the advantages of the fuzzy controller an arbitration scheme is preferred. Again the basic form of the zero order Sugeno system suggests a simple solution. Each of the behaviours is treated as an independent fuzzy behavioural system, as described in the proceeding sections. The outputs must be compatible (the same variables being manipulated). A further aggregation and defuzzification stage is then appended to the controller, to combine these outputs.

A diagram of proposed controller architecture is shown in Figure 5.13b, where it is compared to a simple fuzzy reactive system (Figure 5.13a).

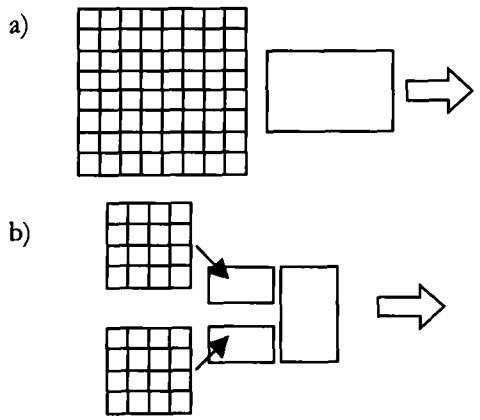


Figure 5.13 Comparing controller architectures: a) master input space b) many input sub-spaces

There are several ways in which this recombination may be made. To obtain an output equal to that which would be produced from a single system, the sum of the firing weights for each sub-element must be recorded and used to calculate the output as if each element were a single rule:

$$output = \frac{\sum_{i=1}^{M} x_i \mu(x_i).p(x_i) + \sum_{i=M}^{N} x_i \mu(x_i).p(x_i)}{\sum_{i=1}^{M} \mu(x_i).p(x_i) + \sum_{i=M}^{N} \mu(x_i).p(x_i)} \quad (5.4$$

where rules 1 to M represent the first functional break in the controller and M to N the second.

or

$$System\,Output = \frac{\sum_{i=1}^{i=j} B_i.W(B_i)}{\sum_{i=1}^{i=j} W(B_i)} \quad (5.5$$

where:

B_i output of a behaviour

$W(B_I)$ sum of firing weight of behaviour rules.

This is however inelegant. Alternatively, the outputs can be combined using the simple mean average. Most interestingly however the outputs may be calculated using priority values for each sub-element. Clearly, the total firing weight and a priority value may be used together, but this adds complexity to the design.

An example system would be the integration of "obstacle avoidance", and a "maintain heading" behaviours. Obstacle avoidance requires that range-finder values are recorded and used to determine the turning action of an agent. The maintain heading behaviour however requires only that the record of previous actions is used as an input (this record should be found by

integrating the previous output commands over time). Assuming the same agent structure as used elsewhere in this chapter, a flat architecture would require a three dimensional input space, which is hard to envisage, with a large number of cells in the FAM. Splitting the behaviours as suggested however results in simple two dimensional FAM for the "obstacle avoidance" behaviour, and a single dimensional FAM for the "maintain heading" behaviour. Reducing the number of cells that must be considered, and thus the number of rules. The trade off is in the computational complexity of the system.

Adoption of this fractional architecture allows the controller to be converted from a simple reactive, to a full-blown behavioural scheme quite simply. This is achieved by dynamically changing the value of the prioritisation factor for each sub-element. Although this is possible for the priority value of individual rules in a single flat FAM, the reduced number of factors that need to be manipulated is an obvious advantage, another being that each factor need only be applied once. When the behaviours are coded as a group of rules within a master FAM the same factor must be applied to each of the rules within that behaviour. Further complications are added when the rules that make a behaviour have relative priorities, in this case some method of combining the inter-rule and inter-behaviour priority factors would be required. Splitting the FAM, applying any inter-rule priority factors within the divisions, and inter-behaviour priorities between them alleviates this problem.

5.7.2.1 Effect of splitting the FAM

Figure 5.14 shows the path of an agent which has been augmented with a "sense of direction" (termed a "maintain heading" behaviour in the previous section). Such a behaviour is important if the agent is engaged on a way-finding navigational task (Chapter 6).

The behaviour implemented in the case illustrated by Figure 5.14 is very simple. If the heading is not that with which the agents journey started, a turn demand equal to a soft turn is output.

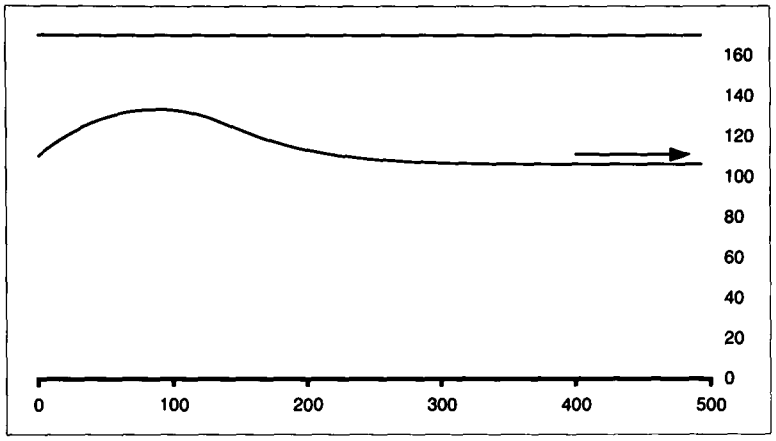


Figure 5.14 Agent in corridor with "sense of direction" (50% priority)

The effect of the additional behaviour on the path of the agent is dramatic (compared with Figure 5.12). The rate of turn as the agent first approaches the boundary is decreased, and the bounce effect limited. The limit to the bounce causes the path to converge to the upper convergence path, as dictated by the upper boundary. The interaction of the two behaviours has also moved this convergent point closer to the boundary. Were the boundary to be removed (as if the end of an obstacle was reached) the path would arc with a curve of radius dictated by the "sense of direction" behaviour until the original heading were achieved (Figure 5.15)

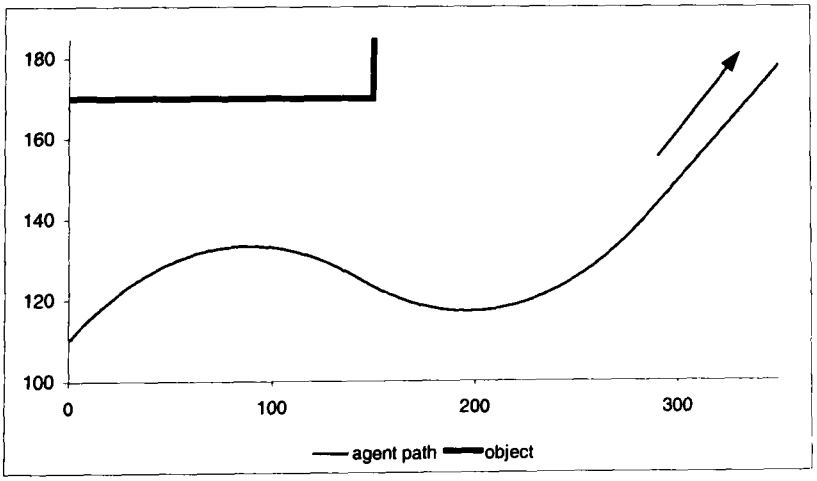


Figure 5.15 Effect of sense of direction at end of obstacle

5.7.2.2 Hybrid control techniques

By splitting the FAM, a method of integrating non-fuzzy elements into the controller is offered. The requirements of the method are that the outputs of each element share a common format, and that each element is assigned a priority value. The implementation of the behaviours is not important.

Figure 5.16 shows the effect of a "sense of direction" behaviour based on this system as it influences the behaviour of an agent approaching a boundary on a perpendicular path. For such a situation the simple behaviour of Figure 5.15 is inappropriate, as any expansion of the turning circle cannot be tolerated because of the low detection distance of the wall.

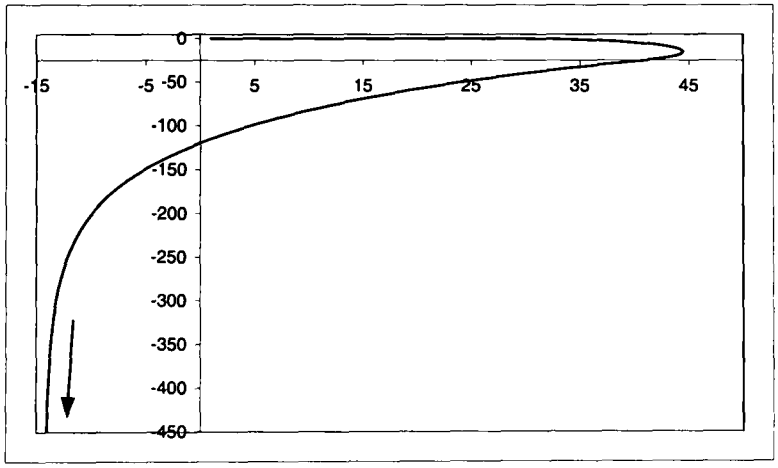


Figure 5.16 Direct heading to boundary, with "sense of direction"

The controller for this experiment comprised of the standard avoidance outlined above (section 5.6) this is given a priority value of 1 (100%). A simple conditional statement is then used to implement the "sense of direction" behaviour (Figure 5.17 shows an equivalent code fragment). The priority value of the behaviour is adjusted dynamically, dependent on the measured range. If the range is far, the priority is set to 25%, otherwise a priority of 0 disables the behaviour. Altering the value of the priority in this way prevents any interaction between the two existent behaviours, so the lowered priority is not strictly necessary, however it was maintained for future

compatibility. Three interesting possibilities inherent in the new architecture are raised by this controller, that of code re-use (of the avoidance behaviour), dynamic prioritisation, and of integrating behaviours not implemented as fuzzy inference engines.

```
priority = 0;
if((leftRange = far)&&(rightRange = far))priority = 0.25;
if(heading != originalHeading){
     if(fabs(originalHeading - heading) < bias)demand = -heading;
     elseif(heading > originalHeading)demand = - bias;
     else demand = bias;
}
```

Figure 5.17 Sense of direction function

Dynamically changing priority values such as this allow hybrid controllers that exhibit both arbitration and selection techniques to be implemented in the fuzzy behavioural architecture.

5.7.3 Code re-use

As behaviours are divided into independent sections it becomes possible to re-use them in the design of other controllers. This feature was used extensively in the development of the controllers used in the experiments discussed. Provided the output formats of the behaviours match, the individual controller elements that code the behaviour may be re-used wholesale.

5.7.4 Integration of dissimilar controller structures

A powerful capability of the proposed architecture is the ability to use other structures within behaviours. A common interface is maintained by the requirement for the motor schemas to share a common output form. Provided this requirement is observed any structure is possible within a behaviour.

5.7.5 Dynamic prioritisation

The priority value of the independent behavioural elements can be set by any method. This allows elements to support or inhibit each other.

5.8 Inter module inhibition

These capabilities have many possibilities, including the integration of deliberative and multi-layered control systems with the fuzzy behavioural architecture. No special method of achieving this inter-behavioural control is suggested, as the requirements of the inhibitory system are liable to be very dependent on the task on which the agent is engaged. The options range from discontinuous functions of other behaviour outputs, which achieves a switching action, to more complex, continuous functions that allow the contribution of a behaviour to be adjusted minutely.

To marshal these functions in systems with many behaviours, a structured implementation is required. A suitable construct would be to make the priority values the responsibility of the arbitration unit, where they are stored in a structure such as an array. The factors that affect the priority values, such as past firing weights and priority values, should also be made available in the arbitration unit. The unit then calculates the present priority value for each sub-controller. By grouping and processing the priority values together in this way, the integrity and readability of the controller code can be maintained.

5.9 Design methodology

Behavioural controllers, as stated in Chapter 3, can produce complex, dynamic behaviour even when simple. Predicting the behaviour of complex, dynamic systems is not trivial. Being unable to predict the behaviour of a system seriously impairs the design process.

Another difficulty is specifying the system. The expected behaviour may be known at a general level, though is unlikely to be mapped out for all situations. Indeed listing all the situations in which the agent may find itself may be difficult. For this reason a design methodology based on software engineering principles is suggested. The basis of the model is the Coad-Yourdon method of systems analysis (Coad and Yourdon 1990), though this is strictly an object oriented technique, the principles that inform it can be applied to this problem.

As with the Coad-Yourdon technique on which it is modelled, the design process is iterative and parallel. The method is based on an object-oriented technique because of the similarity in structure between a behaviour and an object. An object may almost be considered as the inverse of behaviour. An object may be described as an extended compound data type, storing state that has some relevance to a feature of the problem domain in the form of attributes. Also stored within the object are the methods that define the operations that may be performed by the object on its attributes. Access to data from outside the object is normally highly restricted.

In contrast behaviour describes an element of the problem domain in terms of its exhibited actions. Associated with these actions are any data that must be stored.

Within the object model, the "whole-part" (aggregation) and "is a" (generalisation-specialisation) relationships are very important. These relationships may also be extended to the field of behavioural control, where sub behaviours may be part of a larger pattern of action, or may be specialised cases that are substituted where environmental conditions dictate.

The Coad-Yourdon technique identifies 5 major activities in the analysis of a system:

1. Finding Classes and Objects
2. Identifying structures
3. Identifying subjects
4. Defining attributes

5. Defining services.

For the purposes of the behavioural control design method proposed these can be rewritten:

1. Specification
2. Decomposition
3. Implementation

Decomposition replaces the identification of structures and subjects in Coad-Yourdon, Implementation subsumes the definition of services (an alternative term for a method) and attributes.

5.9.1 Specification

The first task is to specify the behaviour of the system. The behaviour should be specified in terms of the problem, to identify as clearly what is required. Care at this stage should be exercised to avoid preconceptions about how this behaviour should be implemented or divided.

Once the behaviour is described as fully as possible, any information on the constraints and requirements should be gathered, the physical requirements or abilities of the vehicle included.

It may be important for many SEA [Situated Embodied Agent] projects to consider the design of the hardware controller together. Smithers (Smithers 1994) suggests that improved hardware makes the design of controllers more problematic, a reversal of conventional wisdom which dictates that inadequacies in one hardware may be rectified in the controller and vice versa. Either way it is unreasonable to assume that either aspect may be neglected.

If the hardware and software are to be designed in tandem the physical requirements are needed. If the hardware is already specified, then the abilities of the vehicles must be determined, and

compared to the physical requirements. Clearly if the requirements are not met, then the hardware cannot be used.

The specification process should be re-iterated, adding greater degrees of detail throughout the design process. Hard constraints and requirements should be added when discovered. These may include the nature of the signals received from the sensors, or sent to the actuators where the hardware has been specified.

5.9.2 Decomposition

The next task is to decompose the system into suitable behavioural components. The specifications and decomposition tasks are then repeated at the level of the behavioural units where appropriate. Different levels of decomposition should be identified where possible, corresponding to the subjects, structures and objects of OOA [object-oriented analysis]. By maintain these varied decompositions at different levels of abstraction, the marshalling of behaviour should be simplified.

Decomposition should be "bottom up" as this has the advantage of describing the more general behaviours first, with behaviours for more specific conditions described later. Approaching the design task in this manner helps simplify the process, by allowing behaviour to be "roughed-out", and then tuned. Another advantage the bottom up approach imbues on the design process that is drawn from its OO [object-oriented] roots is code reuse. Throughout the experiments, where differing behaviour were integrated, the basic obstacle avoidance behaviour remained unchanged, being reused in all controllers. The close interaction between embodiment and controller of the agent may prevent such re-use from being possible across hardware platforms, but in situations where the same physical configuration is to form the platform for agents engaged on a number of task, re-use of behavioural elements has great potential. Object orientation allows code reuse by inheritance and specifying the interface between classes and

objects, and only allowing access to the object through this interface. Provided the interface is maintained the code may be dropped into applications where appropriate, without need for re-implementation.

The manner in which the behavioural sub units interact, and the nature of the interaction between the rules or schemas of the individual behavioural elements should also be planned at this stage. To do this requires that the effects of individual rules are considered independently and in the context of the structure they occupy. The effect of individual elements should not be considered at a higher levels of abstraction, where the interaction is between the structures, not their elements. It may, however, be important to consider the effect of an individual rule, when operating individually, on the behaviour of the vehicle.

5.9.3 Implementation

Determining the manner in which the behavioural controller is implemented is the final stage of the design process. The interface between behaviours and rules is not the responsibility of the implementation. Interfaces are a fundamental feature of the decomposition of the problem as they dictate the information that is conveyed between elements. The types of information that may be transferred will affect the way the divisions between elements fall. During the implementation of the system certain decisions must be made regarding the parameters that make up the controller. If these can be found with any accuracy, for instance, by using limited sample point potential fields, this should be done. There is a temptation to design controllers using guesswork which may occasionally work, especially for simple systems. But system performance should be optimised where possible, suggesting that a more careful approach is required. A successful way of obtaining parameters is to consider the required action of behaviour when the vehicle is static in a worst case situation, calculating the effect of all defined behaviours, and substituting estimated or desired values for those that have yet to be

implemented. The process of applying the design methodology can be illustrated by example as described in the next section.

5.9.4 Example

The agent is required to move through a space avoiding objects while under control of another system that outputs turn commands.

5.9.4.1 Specification

The first activity is to identify the actual requirements of the agent from the specification and determine additional information.

For this example the agent has the same configuration as those used previously in this chapter, however the turning radius is limited such that a direct approach to the wall cannot be negotiated. The agent must back up. The form of the signals for the command system should be determined if previously specified. If not these may be specified during the design process. The same is true of the actuator output signals.

Further behavioural information should be drawn where possible. For instance for this example a behavioural requirement not previously stated is that between the application of command signals from the navigational system the heading of the vehicle remains constant where possible.

5.9.4.2 Decomposition

The task may now be decomposed into suitable behavioural sub-elements. In this case 4 major behaviours are suggested:

1 avoid obstacles

2 maintain heading

3 interpret commands

4 back up when required.

Once the major behavioural divisions have been made these should then be specified. Included in the specification for these behaviour should be the interaction and the inputs. The form of the outputs of the sub-elements should have previously been specified.

The specification for a behavioural element may be along the lines of:

> *If both inputs read a contact of the same magnitude, the other behaviours should be inhibited and the reversing behaviour should be activated. This is high priority.*

There are decisions that must be made here. The agent should not simply drive backwards and forwards while oscillating between this behaviour being active or not. Some heading change activity is required. This raises questions such as: should the back-up behaviour include this action, or should other behaviours be expected to do so. Is this expectation based on the specified behaviour of the other elements, or expected to be generated by the emergent behaviour. In this example it was decided that the just backing up will cause the agent to continue to follow the same arc, as in reverse the effective forces are reversed, so some action must be made. There are several options for this. The action may be directly made by the behaviour, for instance always turn to the right when in reverse or other heading influence behaviours may be inhibited or promoted by the behaviour. The exact behaviour of the vehicle is hard to predict as the exact configuration of the environment cannot be predicted. However, if the conditions for this behaviour are met, those for normal turning behaviours are not. Problems occur when the trigger conditions for other behaviours are experienced during the manoeuvre. For this example the inelegant option of always turning to the left while backing-up (on return to forward movement the heading of the agent will be to the right of the original path) is chosen. The curve will be of high radius.

Another consideration of this type of behaviour is how it should be maintained. If the "back-up" behaviour is only operative while the correct sensor conditions are met (as would be the case in a purely reactive controller) the agent will adopt a position at the transitional distance from the obstacle involved and either oscillate around this position or stop (depending on the exact configuration). The reverse turn of the behaviour will prevent this somewhat, but a commitment to the behaviour is still required. A simple timing element is all that is required, but that it is needed indicates that even the implementation of simple behaviours may require state and timing element.

5.9.4.3 Implementation

Finally the behaviour may be implemented. This may be achieved with the fuzzy expression:

```
If leftRange = medium and rightRange = medium then speed =
    medium, turn = leftMid

If leftRange = close and rightRange = close then speed =
    medium, turn = leftMid
```

Values for the input sets and outputs sets should be calculated if required at this point.

While implementation proceeds, the agent should be tested as each new behaviour is added allowing adjustments to be made. In this way "tuning" may be performed in a simplified environment, no changes being made to those already existing behaviours unless necessary. As the controller has a tendency to complex dynamical emergent behaviour, limiting the number of factors to be changed is desirable.

When the behaviour of the agent matches that of the specification the implementation and commissioning is complete.

5.10 Summary and conclusion

Fuzzy logic has features that make its application to reactive controllers very attractive. As reactive controllers are mappings from sensors to actuators, and fuzzy logic can map any mapping, fuzzy logic may be used to implement all reactive controllers. More importantly the design of reactive controllers is simplified by the communicability that is granted by fuzzy logic's use of linguistic variables.

In its native form fuzzy logic can only be used to implement reactive controllers, as there is no allowance for the storage or observance of state. As the example in section 5.9.4 illustrates, even simple behaviours can require state to be stored. For the implementation of behavioural controllers the basic fuzzy logic inference engine must be modified.

In addition to the state requirements further modifications may be made to the controller to allow the explicit statement of the priority of the behavioural elements, and to provide a logical decomposition of the system. This allows the design process to proceed in an incremental fashion.

An architecture is proposed in which these modifications are made by implementing a controller for each of the behavioural elements. The outputs of these are then combined using an arbitration stage. For successful combination the elements must produce outputs with a common form (the specified interface). For this reason the controller may be considered a motor-schema technique. Prioritisation of the behavioural elements is then achieved by the addition of a factor, applied in the same way as the firing weight of the individual rules of a Sugeno inference engine.

The logical extension of the decomposition and priority factor modifications is to make the priorities dynamic, allowing behaviours to be selected or inhibited. By allowing other behaviours to influence the priority value, inter-behaviour support or inhibition may be implemented.

Interaction between elements of the controller generates state within the behaviour of the agent, providing finite reaction time is assumed for each behaviour. This is implicit in the experiments where the outputs of the behavioural elements were updated in a cyclic manner. State may also be generated by storing appropriate values for use in the arbitration system.

Finally, the decomposition of the controller and dynamic control of contribution enables non-fuzzy elements to be integrated into the fuzzy behavioural controller. This has advantages at all levels of behaviour extraction. From the very simple cases where a single condition statement effects the behaviour of the agent, to the highest levels of abstraction where the deliberative controllers and command structures may be integrated with the behavioural elements, maintaining "housekeeping" competence, while directing towards goals. The navigational strategy outlined in Chapter 6 relies heavily on this property.

Behavioural controllers exhibit emergent behaviour when in operation. The fuzzy behavioural controller architecture outlined here is no exception. The interaction of boundaries and the simulated agent provides a powerful guidance method. This guidance, coupled with a simplified representation of space possible when a relativistic approach is used form the basis of an effective navigational strategy for use in constrained environments. This is discussed in the next chapter.

Even with the communicative advantages gained by the use of fuzzy logic and a decomposed controller, the design of controllers to meet task requirements is still involved. For this reason a design methodology is required. A methodology is suggested that evolved from the Coad-Yourdon object oriented technique during the design of the experimental controllers. The method concentrates on the specification of requirements and interfaces, allowing flexibility of implementation. The method also allows controllers to be implemented and tested in an incremental method, as is traditional for behavioural systems.

5.11 References

Bisset D.L. and Webber A.D. 1994. Building Reactive Vehicles: An Engineers Perspective. *Proc AISB'94 Workshop*,

Brooks, R.A. 1986. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation* RA-2, No. 1: pp14-23.

Coad P., and Yourdon E. 1990. *Object-Oriented Analysis.* New Jersey: Yourdon Press.

Dennett D. 1993. *Consciousness Explained.* Penguin.

Kosko, Bart. 1992. *Neural Networks and Fuzzy Systems : a Dynamical Systems Approach to Machine Intelligence.* 1st ed. Prentice Hall International Editions. Englewood Cliffs N.J U.S.A: Prentice-Hall International Inc.

Lewin R. 1993. *Complexity.* Phoenix.

Li Wei. 1994. Fuzzy Logic Based 'Perception Action' Behaviour Control of a Mobile Robot in Uncertain Environments. *IEEE Int Conf on Fuzzy Systems*, pp1626-1631. IEEE.

Li, W. and Feng, X. 1994. Behavior Fusion for Robot Navigation in Uncertain Environments Using Fuzzy Logic. 1790-1796.

Liu, K. and Lewis, F.L. 1994. Fuzzy Logic-Based Navigation Controller for an Autonomous Mobile Robot. *1994 IEEE International Conference on Systems Man and Cybernetics.*

Mamdani E.H. and Assilian S. 1975. An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *International Journal of Man-Machine Studies* 7: pp 1-13.

Minsky M. 1988. *The Society of Mind.* New York: Touchstone (Simon and Schuster).

Smithers T. 1994. On Why Better Robots Make It Harder. - *Proc 3rd International Conference on Simulation of Adaptive Behaviour*, Eds Cliff D., Husbands P., Meyer J.A., and Wilson S., MIT Press.

Tubb C.A., Roberts G.N. and Rowlands H. 1997. Behavioural control implementation using fuzzy logic. *ICSE '97, the 12th International Conference on Systems Engineering.*

Tubb C.A., Roberts G.N. 1998a. Behavioural Control Using Modified Sugeno Inferencing. *AMC '98, the 5th International Workshop on Advanced Motion Control.*

Tubb C.A., Roberts G.N. 1998b. Development Of A Fuzzy Behavioural Controller For An Autonomous Vehicle. UKACC International Conference; Control'98.

Zadeh L.A. 1965. Fuzzy Sets. *Information and Control*, No. 8: pp 338-53.

6: Navigation via Script in Constrained Environments

In Chapter 4 it was shown that directed movement – navigation – requires that the spatial relationships between locations be represented. In addition to the navigation task, spatial representations are required in the communication of a goal location to an agent. The appropriate form of this representation is determined by structure of the mission environment, the requirements of the navigating agent and the capabilities of the generating source.

It is possible to exploit environmental characteristics to restrict the information stored in a spatial representation. By reducing the representation so it contains only a smaller set of salient environmental features, communication between executive and author may also be simplified. Indeed the message and map can be merged to a common symbolic code. This is especially true of spaces where the path of a navigating agent is heavily constrained. It is not unreasonable to assume that most commercial autonomous vehicles will operate in such environments, for instance: corridors of buildings, the road and rail networks or gangways within warehouses. Such environments also contain many generic and easily recognised features that may be exploited to simplify the representation further. Clearly these simplifications depend on the ability of the navigating agent to recognise the constraints themselves. For a road-analogous system to aid the navigation of an agent, the agent must be able to determine what is road, and what is not.

This chapter discusses the nature of constricted environments and how the restrictions imposed by them effect the representation of space, communication of paths and the requirements of localisation for automatically guided vehicles. The use of routes to exploit these features is discussed. The use of scripts to implement a route based navigational system is then investigated and a suitable structure proposed. Using scripts also allows the representation of space and the path definition to be merged, precluding the need to interpret the communication. The utility of the simplified representations obtained is increased by the ease with which they can be communicated and exploited by a behavioural control implementation

A method of integrating this with the behavioural controller previously developed in Chapter 5 is investigated. Finally, methods for the extraction of novel routes from a set of scripts are shown.

6.1 Environmental constraints as navigational aids

Before a discussion of the effect of constraint within an environment on navigation and representation may commence, some thought is required on the nature of this constraint.

6.1.1 Constraint

For the purpose of this work constraint is considered to be when an agent does not have full access to all of the physical space due to either physical obstacles or some legal restriction such as tracks or a line on the floor that is followed. A good example of a legal constraint is the road system. Most road vehicles are capable of driving outside of the carriageway, though through the action of the driver they are (normally) constrained to it.

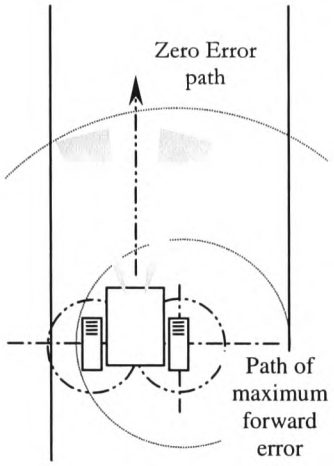


Figure 6.1 Agent within confined space

The amount of restriction experienced by an agent is a measure of the scale of the features in the environment with respect to the physical size on the agent complete with its sensed zone. To illustrate this the vehicle is considered to be in an enclosed corridor-like space. For the agent depicted in the diagram above (Figure 6.1) the maximum forward error is assumed to be when one of the wheels is stationary (100% slippage or stalled), while the other provides traction. In a differential drive steering scheme, such a situation represents the turning circle with the minimum erroneous radius (tighter turning would require that the vehicle was suffering some internal malfunction as reversal of one wheel is required, and is thus neglected). The maximum heading error for an agent in a confined environment is simply the maximum angle the agent may rotate through before the constraint is detected and acted on (Figure 6.2). An agent is constrained if it cannot rotate completely within the current space; that is, the diameter of the arc of maximum forward error is less than the distance between the constraining planes of the space. The agent can be considered fully constrained if the distance between confining planes is smaller than the combined agent and sensor field. In such a situation the path of the vehicle is always influenced by the constraining planes.

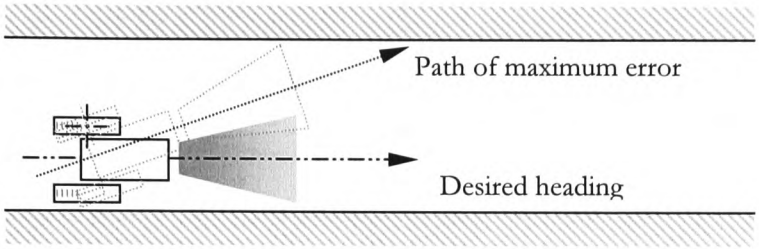


Figure 6.2. Maximum constrained heading error

6.2 Simplification in representation

Constrained environments reduce the information that must be stored in a representation of space, and change the nature of suitable navigational representations.

In his discussion of maps as models of the environment Board points out that it is the omission of detail that makes maps useful (Board 1967). Maps are valuable because they are simplified, imperfect records of the environment they represent, rather than in spite of this. For any given purpose, those properties of a space that must be identified are more obvious when distracting information is removed from the map. For navigation this means that the map is reduced to provide simply the detail that is used for location, orientation and route extraction.

Many authors try to produce accurate floor plan maps of the environment in which there vehicles are operating (this is the theme of many of the papers in (Kortenkamp et al. 1998)). These are then manipulated to extract the symbolic information that is required for path planning and task scheduling operations. The suggestion in these cases is that a good geometrical knowledge of the environment is required for navigation. Though such knowledge is believed to be used by, and is indeed considered inherent in, humans and higher animals, this is used mainly for the creation of large-scale topographical maps of the complex natural environment, used to calculate routes (O'Keefe and Nadel 1978). Toto (Mataric and Brooks 1990) showed that this heavily geometric approach is not required. Though such simple

representations may look far from complete and rather inelegant, restriction of concern to only the necessary information is considered good analysis when being used to design systems (Coad Yourdon 1990)(Booch 1994). The use of minimal representations is also more in line with the philosophy of behavioural control itself (Brooks 1995).

Care must however be taken to ensure that the simplification criterion corresponds to the environment and the navigating agent. It would be foolhardy to attempt to navigate London using the tube map in conjunction with any other transport system. However the 'tube' map is not only a design classic, but also perhaps the simplest and least ambiguous manner of showing the relationship between the stations of the underground railway network. If further information had been included, or angular and distance scale had been correctly maintained the map would be more difficult to use.

6.2.1 Simplification mechanism

Constrained environments reduce the amount of information that needs to be represented as there are areas were the agent cannot, or should not travel. Extrapolating from the "corridor" definition of space given above it can be seen that the topological relationship between items in the constraint field is forced into a linear arrangement. As the vehicle is restricted to a path along the 'corridor', features of the space may be identified by the order in which they will be reached. As the level of constraint increases space becomes a network of routes where individual nodes are the major, individually identifiable locations, connected by a very restricted set of paths. As the complexity of the space in which the agent moves decreases so does the complexity of the representation. An agent may navigate successfully using a spatial representation that is architecturally a network of diffuse locations. Indeed this was the representation employed on Toto.

6.2.2 Simple maps

How representations of this spatial structure are produced depends on the capabilities and requirements of the navigating agent (Chapter 4). A network may be represented either graphically, or as a "line list" or as one of matrix graph representations (e.g. an adjacency matrix). Along with selecting the structure of the representation used, it is also important to choose a suitable implementation that can hold the representation and be easily communicable.

The graphical representation of graphs has been extensively used for navigational purposes. The maps produced by the Romans were a poorer representation of the general geography of the world than those that had previously been produced by the ancient Greeks. As the Romans borrowed so heavily from the ancient Greek culture this may appear to be a quite strange, retrograde step. Roman cartographers certainly had the ability to produce full projections that reflected the topography of an area, but surviving Roman maps such as the "Tabula Peutingeriana" do not attempt to do this. Roman maps were made for more pragmatic reasons than those of ancient Greece. Primarily intended to aid in military and commercial travelling they are concerned with the road network. Geometric information is not shown, with the exception of the linear relationship between the road and those locations along it. Distances are given in figures written besides the route rather than through the scale of the map (Bagrow, 1964). Maps of this form (cartograms), where both distance and angular information is distorted are still used extensively for travel. Street plans and the London Underground "tube" map <http://www.londontransport.co.uk/info/lul_index.htm> are examples. Many features of the greater topography of the area are omitted from these maps, which show a network of possible paths through the represented space. These paths are created by heavy constriction of the space in which road vehicles and tube trains may move. The constraint is so high in the Underground map that a diagrammatic representation of the tube network is sufficient in most cases to successfully navigate London even though the map is correct in neither scale, area or orientation.

The route may not be optimal in terms of Euclidean distance travelled or time taken, but requires minimal calculation time. Navigation requires that a passenger selects suitable tube stations at start and goal location in the city, descends into the underground, navigates the network between these stations, and emerges into the city close to the goal location. The tube network thus represents an alternative expression of the city, one that has been spatially simplified and optimised for navigational simplicity, in addition to its role as a mass transit system.

6.2.3 List based representations

Compared to graphical methods, matrix representations of networks are encountered far less frequently in everyday life. List representations of space however, are regularly employed in navigation. They also have the advantage of being easy to communicate. The most common form of a list based spatial representation is a set of directions. These are further simplifications of the representation of space as they only describe the space relevant to the present navigation task. Arcs are described in terms of the topographical relationship of the nodes. The representation is further reduced in complexity by exploiting the sequential manner in which nodes will be reached.

6.3 Use of simplified representations for navigation

As stated in Chapter 4, the great advantage in using simplified representations is the ease with which they may be used. To navigate using a map, the following procedure must be followed:

1. Determine position of present location and orientation on the map
2. Find goal location on map
3. Plan a path between present location and goal point using the information represented in the map

4. Follow the path, feedback being provided by comparing the information provided by the map with that obtained from the environment.

For a fully featured topographical map, each stage of this procedure requires complex computation. The success of the path planning operation is also dependent on the relevance and completeness of the information stored in the map. As the complexity of the representation is reduced, then so is the task of extracting navigation information from it. The representation, must, however, contain sufficient information for the agent to navigate the environment that it describes. The level of path constraint dictates the required level of information. The higher the degree of constraint, the less information is required in the representation.

The level of constraint on the path also has a direct effect on the actions that must be performed by the traveller to correctly traverse the path. A train journey is amongst the simplest of journeys to make through unfamiliar territory. The traveller simply alights when the correct station is identified. As the complexity of the journey increases, for instance a journey with several changes, the complexity of the itinerary also increases, but remains a list of stimulus response pairs. Occasionally additional information is required to ensure correct use of this list (O'Keefe and Nadel 1978) but this is dependent on the constraint level of the environment in which the journey is made. As long as the correct train is boarded and the correct passengers alight at the correct station there is very little chance of getting lost on the train!

Network based maps allow for simplified calculation of a route, which may be simply expressed as a list of stimulus response pairs, employing common signals. These are ideally suited to use in constrained environments. Though such a "taxon" system is less flexible than a "locale" system which uses full topological representations of space, and is more prone to errors if a location is not identified correctly (O'Keefe and Nadel 1978) their simplicity makes them appealing for mechanical systems working in constrained environments.

How the navigation task is simplified by use of the types of representation suitable for constrained spaces is discussed below.

6.3.1 Location

Any navigating agent must have knowledge of its present location, before any sort of path may be constructed. The ability to localise is also required while following a path. Without the ability to continually localise the navigating agent exhibits only ballistic movement. Once a trajectory has been commenced, it will be followed to its conclusion, with no opportunity to correct for errors. In a taxon system it is very important that present location is identified correctly. Failure to identify a stimulus point will cause the navigating agent to become catastrophically lost. If lost within an area that has been mapped by a locale system, an agent may observed those features of the environment of the type that correspond to the maps symbol set, and observe the relationships between them. This arrangement may then be found on the map identifying location, and enabling new routes to be calculated. Within a taxon system no information additional to that which is required to follow the proscribed route is stored. There is no-opportunity to re-localise and extract new routes when lost.

A highly constrained environment, such as may be represented by a network, reduces the amount of information that is required for localisation, due to the sequential nature of a walk. The constraint of the agents path also directly contributes to a reduction in the difficulty of recognising a location, as there are fewer "points of views" from which it may be observed. The constraining features may also provide topological information themselves. Figure 6.3 illustrates this. Both locations A and B show the same major topological features, with the exception of the constraining wall. This type of location recognition is performed in the hippocampus of the rat (O'Keefe and Burgess 1996) and in Toto (Mataric 1991).

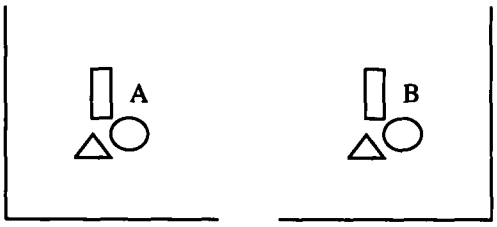


Figure 6.3 Constraining planes themselves provide identifying information.

The close couple between the environment and the behaviour of a vehicle may also be exploited to increase the reliability or simplicity of a location recognition system. Prescott draws attention to the fact that the use of wall following behaviours reduces the complexity of navigation tasks due to the imposition of spatial and behavioural constraint (Prescott 1994). An example of this effect may be observed in the work of Nehmzow (Nehmzow and Smithers 1991). Here locations are areas where a wall following behaviour forces a change of activity. The maze in which the machine performed consisted of a circuit comprised of straight walls and right angled turns. The length of the straight prior to the turn identified places within the maze. Other schemes have been suggested that also use learnt features of the environment to identify the way-points of a path for instance (Baptiste et al., 1994).

These schemes have the unfortunate requirement that the locations must be learnt or defined individually. Either impairs the ease with which routes and representations may be communicated to the vehicle. There is, however, an alternative, which relies on the sequential nature of a walk through a network and the simplicity of most constrained environments.

6.3.1.1 Generic features as locators

Directions given at the roadside to people unfamiliar with an area are a prime example of the use of generic features. Readily identifiable symbolic items, such as traffic lights, roundabouts and junctions are used to indicate the path to be followed, rather than specific, hard to convey information on the appearance of a location.

In built environments, recognisable generic features are common, though the homogenous nature of the décor may mean that being able to uniquely identify locations may be difficult. Recognising an open doorway inside a corridor may be relatively simple, but recognising a specific doorway is far more difficult, requiring a great deal of information.

A great advantage of using robustly detectable generic features of the environment for navigation is that they may easily be represented by symbols. The communication of the representation or route is thus greatly simplified.

Generic locations may be detected relatively easily by the impoverished senses associated with embodied agents. The junctions of corridors, walls, doorways etc. may all be recognised merely by range finder signatures, more specifically by changes in range, Figure 6.4 illustrates how an opening to the right of a vehicle effects range information. A sudden increase in range maybe used to identify the opening. This method of identifying such features is used by Stein in her paper "Imagination and Situated Cognition" (Stein 1991).

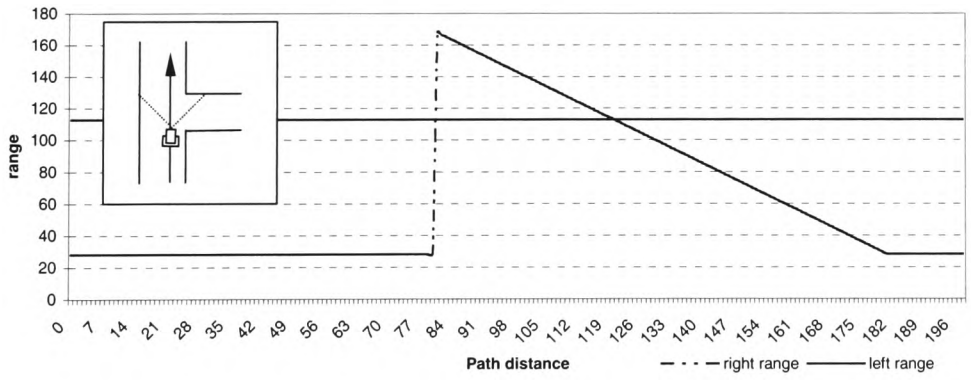


Figure 6.4 Identification of a opening to the right of a vehicle using range finder information (range in centimetres)

The behaviour of an agent may be used to directly form a constraint and so aid in the use of generic locators. In an otherwise open space a single constraining plane may be used to generate a valid network representation when specialised behaviour is introduced. Figure 6.5 shows a network representation of a space, rendered possible by implementing a wall following behaviour. There is a potential problem with forcing a representation in this way; for as soon as contact with the wall is lost so is the validity representation and any advantages this may bring.

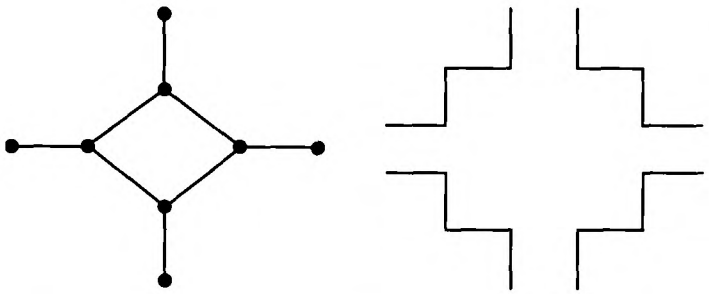


Figure 6.5 Network representation of an open space, behaviourally generated

6.3.1.2 Location by sequence

The greater topology of a space may be used to uniquely identify locations by the use of generic features. The relationship between features may thus be used to determine location. Figure 6.6 illustrates how two otherwise identical locations may be recognised by the sequence of arrival. Starting from the left of the diagram, the agent, if path constrained must reach Location A before Location B. No additional information, such as the distance between features is required for successful location.

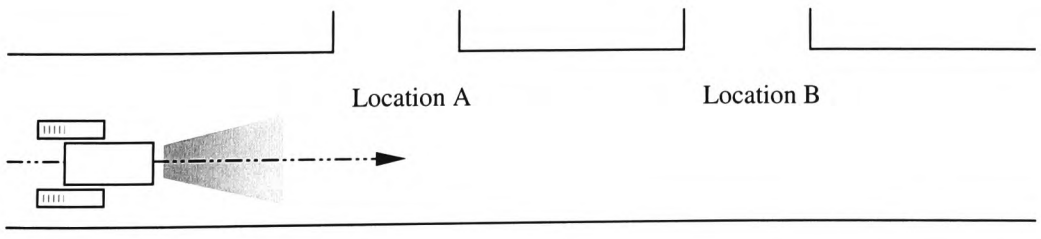


Figure 6.6 Locations identified by sequence

6.4 Scripts

Scripts have been used extensively throughout Artificial Intelligence research. They are generally easy to produce and use, but also very powerful. Ultimately a script provides a method for an author to communicate with an actor or executive.

Scripts derive from natural language processing. They provide an expectation driven method of determining action and chains of causality, procedures and sequences (Murphy 1996). The power of this type of script, which is based heavily on theories of language comprehension in humans (and so thus is has some similarities to fuzzy logic) is unmistakable. Scripts have been used for the selection, or creation of behaviour for some time. Similar techniques have also been proposed which employ different topologies and abilities, such as the agenda used by Mattellan and Borrajo (Mattellan and Borrajo 1998).

For the direct integration of scripts with behavioural control techniques, many simplifications can be made, yet still provide a powerful controlling structure. The adoption of simple Stimulus-Response lists, or, at higher levels of abstraction Stimulus–Reponse-Stimulus lists, (where the action is performed until some second environmental condition is met (Aylett et al., 1995)) provides the controlling capabilities of a script, but is simpler communication and interpretation. Such structures are ideal for the implementation of a navigation system based on routes.

6.5 Maps and scripts

Lists or matrices may be used to describe networks, and thus spatial representations that are based on graph like structures. For the purposes of navigation a script which embodies a list network representation provides an ideal representation of space; able to communicate directly with the agent controller, and describe a space and the paths through it. This is especially true when an external agency is attempting to communicate a route to an agent.

Unfortunately the information contained in a network is not sufficient for agent navigation in all but the most trivial of cases. The problem is caused by a lack of information to discriminate the correct arc to follow at a node. Simple representations of networks use adjacency of nodes to define arcs. An agent within the network will not be able to employ such a representation as it is unable to sense the whole network and so cannot measure adjacency. Also, if the agent is to use it's arrival sequence to determine the location of a node in the graph spatial representation, it can not then use the nodes to define the arc it has followed.

A method of discriminating arcs using only information available at each node is required. A reliable method of embedding this additional information into the communicated message is required. Fortunately at each node, only the arc to be followed need be identified. The proposed script based navigation system, described in the following sections solves this problem.

6.6 The script based navigation system

Exploiting the features of scripts and constrained environment together provides an analogue of giving directions, which may be used to communicate a path to an agent, allowing the agent to navigate a path. Such a system is very simple to implement with few issues to be resolved. Extensions that provide a network based spatial representation are also easily conceived. Most importantly the representations may be understood by both man and machine, they are small and require no complex coding or decoding mechanisms.

As stated above, scripts provide a simple mechanism for the conveyance of information between an operator and a machine. They also correspond very well to the stimulus response pairs that form the basis of a piloting or taxon navigation system. Scripts or script-like structures thus form the most obvious solution for the problem of navigation within a heavily constrained environment. By providing information on what action is to be performed at a location scripts

provide sufficient information for arcs of a network to be differentiated 'on the ground', exploiting the geometric relationship between arcs.

The system proposed may also be easily integrated with the behavioural control implementation previously suggested (Chapter 5).

6.6.1 Structure of the script-based navigation system

The basic requirements for script-based navigation technique are:

- The script must contain adequate information for navigation.
- The executive agent and author must both be able to employ the representation.
- The script must be communicable
- The symbol set employed must be understandable by both executive and author.

To successfully communicate and navigate a path, the simplicity of a taxon system may be exploited. Recognisable features of the environment are listed, along with an action that must be performed at that location. The list is then used as an action script for the executive agent.

6.6.2 Information content

The amount of information that must be conveyed is greatly reduced by exploiting the simplifications that a constrained environment allows to be made to the location task.

A location is identified by both a symbol that represents a generic feature of the environment and (implicitly) its position in the sequence of features experienced by the agent as it moves. The actions to be performed by the agent must also be communicated, so also require symbolic representation. The number of symbols required depends on the structure of the space under consideration, and the capabilities of the agent itself.

Within a large building, the agent will primarily navigate corridors. Assuming that these tend to be rectilinear spaces that intersect, locations are signified by doorways (openings) off of the corridor, or by the intersection of corridors. The geometric relationships between these openings also tend to be regular and predictable, perpendicularity being ubiquitous.

The actions the agent is expected to perform in such a situation are also constrained by the spatial morphology. Because of this the description employed to define the appropriate action may be highly abstracted. An agent may turn either left or right into an intersecting corridor or doorway, stop, pause etc. The major actions for the purpose of navigation are the left and right turns.

6.7 Script use

Use of a navigation script requires some thought. How is the script to influence the behaviour of the agent? How does the script interact with the other aspects of the agent's behaviour? The script should not interfere with the local navigation of the agent, yet still provide an effective control on the agent's movement.

It was shown in chapter 3: "Behavioural control", that all behavioural control architectures provide a mechanism for the integration of potentially conflicting actions. The integration of local and way-finding navigation presents exactly this type of situations. The fuzzy behavioural controller outlined in chapter 5 provides a framework for this integration provided the output of the way-finding navigation system matches the output of the behavioural sub-elements correctly, and is provided with an appropriate priority value.

6.8 Script structure and design

The basic script is formed from a collection of conditions, listed with associated actions. In use a condition is read from the script. When this condition is met the associated action is performed and the next condition loaded.

For route following in a constrained, corridor-like environment issues of communicability are thrown up by this basic script structure. As described above, to turn left at the intersection of two paths, the agent is instructed, through the script, to adopt a "left turn" behaviour when an opening to the left is discovered. When this action has been commenced the next condition should be loaded. This would be the 'in corridor' condition. The associated action would be to move directly forward. As this forward motion is activated, the next direction change trigger condition is loaded. This is the most simple of the possible script architectures, and has great potential for robustness. A state diagram representing the actions described is given in Figure 6.7a. A generalised state diagram is given in Figure 6.7b.

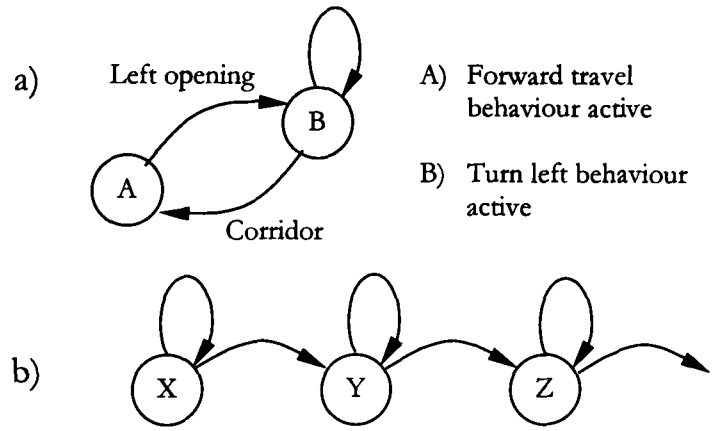


Figure 6.7 a) State diagram of action described in paragraph above. b) Generalised form

There are alternatives, however, that can be used to shorten the length of the script, or reduce the number of symbols used.

The first of these alternatives is to define free forward movement as the base state of the agent, while the route is described as triplets, showing trigger stimulus, response and end stimulus (S-R-S). A general state diagram is shown in Figure 6.8.

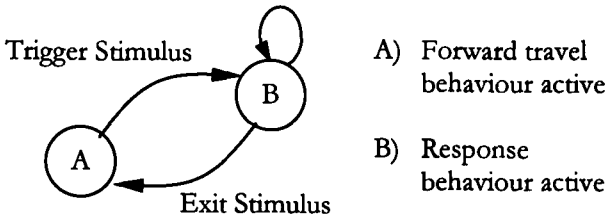


Figure 6.8 State transitions in triplet scripts

Once triggered, behaviour is maintained until the second, or goal, stimulus is reached. Although reducing the size of the script whilst ensuring that each action is completed before the next starts, problems are associated with the use of triplets. Primarily, each goal stimulus must be defined and assigned a script symbol. The goal stimulus must also be recognised robustly while the machine is working; this may be more problematic. Whereas the detection of the intersection of corridors may be achieved by measuring the rate of change of range-finder readings, corridors represent more the absence of high-frequency, or large transient changes in range information.

Considering free forward moment as the base state of a navigating agent may also reduce the size of the script where SR pairs are employed. The great advantage of doing this is not only a shortening of the script but also a reduction in the number of symbols required to communicate the route. This is only possible if the agent is capable of determining if an action has been successfully made]. To exploit the shortened script some way of determining the completion of a response task is required. Two main alternatives are possible, either the stimulus may be considered a trigger for a timed event, or the action may be continued for the duration of the stimulus (Figure 6.9 a and b respectively).

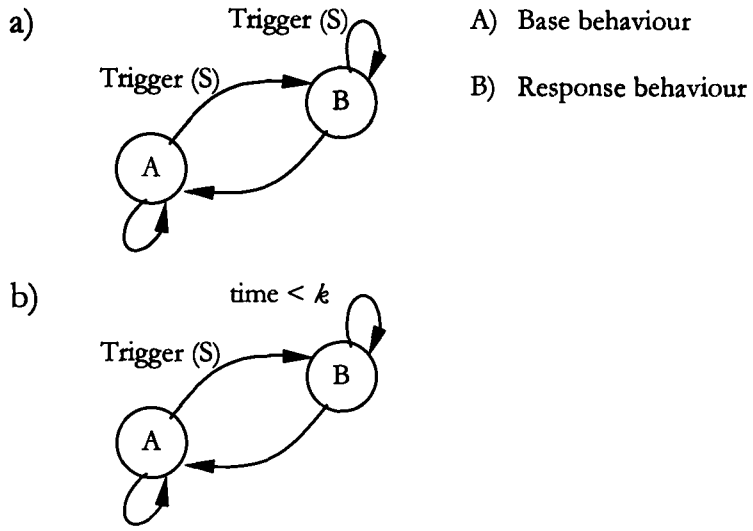


Figure 6.9 SR list operation with forward movement base behaviour. a) Response behaviour active while trigger stimulus present. b) Response behaviour timed

Both possibilities present potential problems.

6.8.1 Extended condition behaviour mediation

If continued presence of the triggering stimulus is required for the response behaviour to remain active (Figure 6.9a) then the navigating agent must be able to recognise the stimulus throughout the response manoeuvre. The sensor information received changes dramatically as the manoeuvre is performed (Figure 6.10). The mechanism that extracts trigger signals from the range data must thus be able to not only discriminate between similar sensor readings to produce the onset of the trigger signal, but also maintain that signal as the sensor data changes with time.

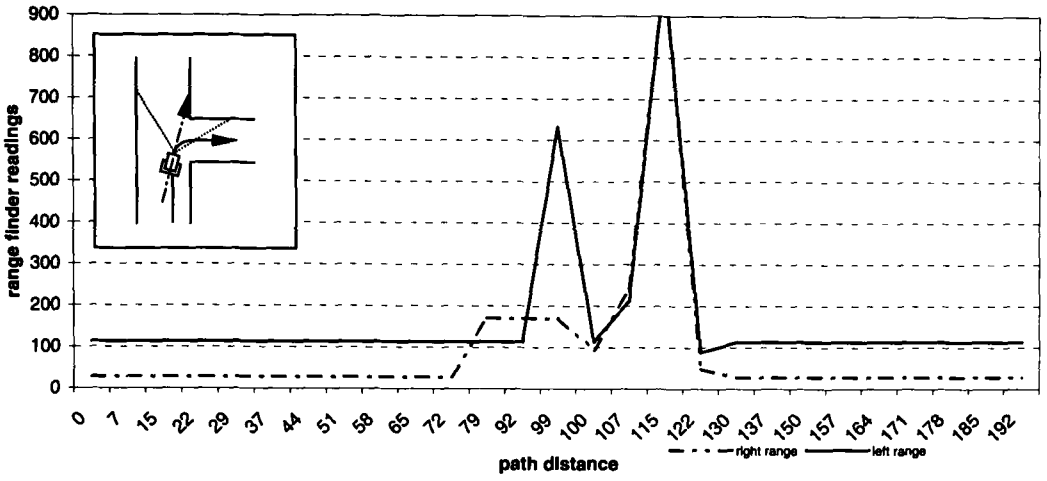


Figure 6.10 Range finder values as agent turns at corridor intersection (range in centimetres).

The constraint enforced on the agent by the environment does, however, provide some solace; while travelling in the enclosed corridors, and changes in range information are due to three causes:

1) Sensor 'noise'

2) Objects within the corridor

3) Openings and intersections

Noise will tend to cause a random variation in the range data. As noise must be taken into consideration while seeking trigger stimuli anyway, it can be discounted as an additional problem here, as schemes should already be in place to reduce its effect.

Objects within the corridor must cause an immediate decrease in the range value measurements. Trigger stimuli, which tend to represent openings in the constraining boundaries, may be recognised by an immediate increase in range values. As an agent manoeuvres as part of its object avoidance behaviour, range values may increase temporarily. It may be assumed,

however, that in the vicinity of an object the range, averaged over time, is generally shortened compared with free travel, while around trigger stimuli the measured range values will tend to be increased. Figure 6.11 shows the path, and range values obtained, during a 'drive-past' of a square object (of 50cm sides) placed at the edge of a corridor (170 cm wide). The vehicle starts the run 110cm from the right wall of the corridor (at the bottom of the diagram) and is facing right, parallel to the axis of the corridor. The obstacle has been omitted from the diagram for clarity. The diagram clearly illustrates how the net range readings diminish as a path around an object is negotiated. Though at any point during the manoeuvre there is no single signature range finder reading.

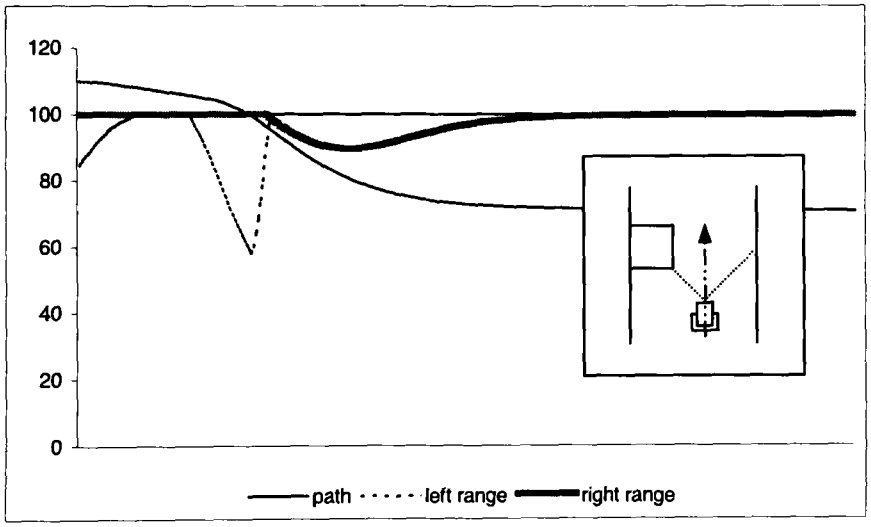


Figure 6.11 Effect on range values of passing a constriction

Travel in the vicinity of corridor intersections causes the average 'sum of range' value to increase.

6.8.2 Fixed duration behaviour selection

The alternative, fixed duration behaviour selection relies, on a timer being initialised as a trigger condition is met. The appropriate action behaviour is then run until the timer expires, at which point behaviour returns to its quiescent status. The immediately identifiable problem with this

technique is a dependence on predictions of the time a manoeuvre is likely to take. Again, constraint in the environment helps relieve this problem, provided a behavioural arbitration system is employed. Within a highly constrained space the boundary planes prevent the agent from deviating from its path. The response behaviours are thus only effective when there is an opportunity for the agent to make significant path adjustment. Near the constraint of the boundary walls, the local-navigation, or obstacle avoidance behaviours control the action of the agent, the demands of the way-finding behaviours being over-ruled.

Demands on the temporal resolution of the response action are thus dictated by the time required to travel between nodes. This should in most situations be sufficient for the time allowed to response behaviour to be non-critical. A case where this is not so would be when entering a node where multiple paths intersect.

6.8.3 Sequence counting

The sequence at which locations are encountered while following a path need not only be used to discriminate between various locations that are otherwise indistinguishable using the sensor information available to the agent. Such sequence information may also be used to reduce the amount of information that must be contained in the script. The achieved reduction in size of script depends on the nature of the space within which the agent travels and the complexity of the path to be followed, but real compressions are possible[1], and perhaps more importantly the resultant scripts are more easily communicated by a human author.

To make the reduction a trigger stimulus is identified not only by its generic form, but also by the number of similar triggers that proceed it in an unbroken series. These triggers may then be

---

[1] This is script compression by run length encoding.

excluded from the script. The script interpretation agency on completing a manoeuvre loads the next trigger stimulus type and sequence position. As each potential trigger of the correct type is encountered it is counted. When the count is equal to the loaded sequence behaviour the relevant action response is triggered, otherwise the quiescent, base behaviour continues. It should be remembered that the script, being an expectation driven representation of space is already highly compressed, many features of the environment being ignored as there is no associated response behaviour. Figure 6.12 shows example scripts, both are human readable, but the shortened script is not only significantly compressed, but also more simple for a human author to communicate.

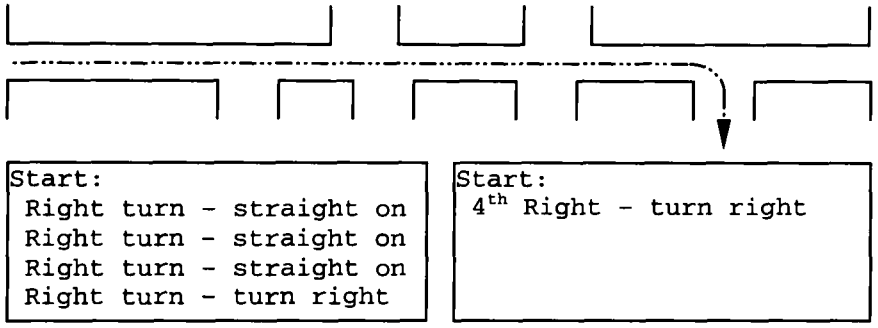


Figure 6.12 Simplification of script by sequence information

If sequence counting is to be used by the navigational system to simplify scripts, the structure of the command system must reflect this, additions must be made to the script interpreter to count the features as they are achieved.

6.8.4 Controller interface

The command script system must be structured to interface with the behavioural controller that forms the local navigational abilities of the agent. The manner in which this is achieved is highly dependent on the form of the behavioural controller. As the structure of the interface will be liable to great variation, only that required to interface the script system to a controller of the

type proposed in the previous chapter (Chapter 5: "Fuzzy Behavioural Control Architecture") will be considered. This is fully discussed in section 6.9 below.

6.8.5 Influence of way finding on local navigation

To correctly interface way-finding and local navigational systems requires not only that the script interpreter is structured in a way that allows the two to communicate, but also that the local navigation system provides a reliable base for way-finding. The sparseness of the script representation means that many assumptions and generalisations must be made. Behaviours are required in the controller to support these assumptions. These must be added to local navigation if script following is to be successful. An example would be a wall following behaviour that generates constraint in otherwise open spaces.

6.9 Implementation of the script based navigation system

The following sections describes the implementation of a command script based navigational system using the principles developed above. The design is intended to interact with a fuzzy behavioural controller as suggested in chapter 5: "Fuzzy behavioural control architecture". The command script interpreter produces speed (binary) and heading change information that matches that produced by the local navigation controller, so may be implemented directly into the fuzzy behavioural controller at the behaviour arbitration level. Local navigation is given a higher priority than the output of the command script. Recognition of the trigger conditions (stimuli) of the script is achieved using a separate pre-processor module. As in other motor schema systems an agency external to the two navigational controllers calculates motor signals. This not only allows the controller and script interpreter to be simplified by divorcing them from the mechanics of the agent, but also allows agents with different configurations to be coupled to the controller/interpreter combination.

The exact nature of the trigger recognition processor is not of concern here, provided it is capable of reliably extracting the stimulus signal set from available sensor information. For fixed duration behaviour selection, however, the timing of the behaviour may be achieved either in the recognition processor, by generating extended stimulus conditions from the trigger events, or within the script interpreter itself. The form the outputs take must also be considered. They may either be short trigger events, or extended conditions. The trigger module is independent of the fuzzy behavioural controller, allowing different pre-processors to be used if required. Indeed the trigger module need not use the same sensors as the local navigation system.

Along with the trigger processor and scripts themselves the script based navigation system requires a script interpreter module. This is interfaced with the behavioural controller. The nature of the implementation is thus dependent on both the structure of the script, trigger processor and the controller.

The architecture of the agent is shown in Figure 6.13.

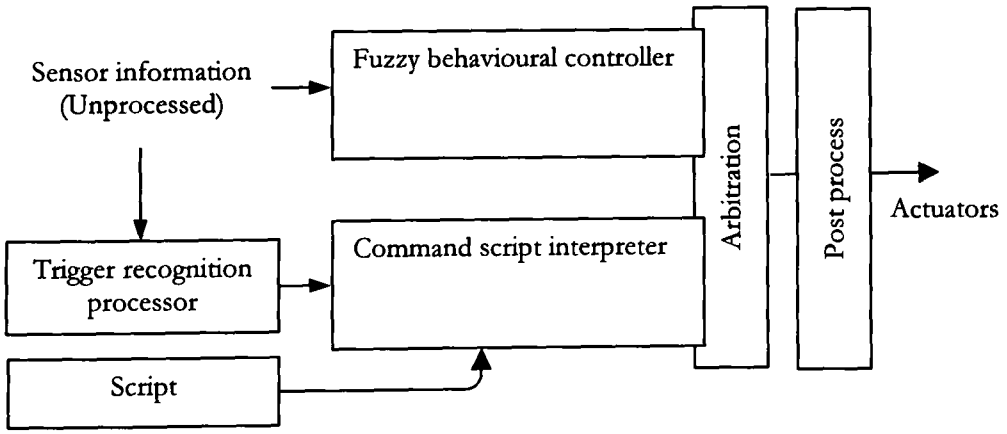


Figure 6.13 Controller/Script structure

The nature of the interaction between the module and the behavioural controller itself is determined by the nature of the controller.

For integration with the fuzzy behavioural controller as proposed in chapter 5 the command script interpreter generates heading signals identical to those generated by the controller itself. These are then integrated through a "weighted beam" averaging system, akin to a zero order Sugeno fuzzy system. As shown in the previous chapter this is identical mathematically to placing the command script interpreter within the behavioural controller, but has the advantage that it allows the controller and interpreter complex to be understood more clearly. Use of a more coarsely grained schema structure at the interface also simplifies the selection of utility values for the local navigation and way-finding schemas. A single utility function is required for each. Figure 6.16 shows the interaction of these two behaviours for a simple manoeuvre. Local navigation, being the most immediately important of the agent's behaviours is given a maximal utility value. The way-finding behaviour, as embodied by the command interpreter, has a lower priority (25%). As can be seen in Figure 6.14, where a range of utility values are used, the choice of this value is not critical but does make a difference[2]. Further robustness is inherent in a controller of this form as the contribution of the way-finding behaviours to the control of the vehicle is restricted. While the agent is away from trigger stimuli, control is totally dependent on the local-navigation system.

In the figure, the command script interpreter has been triggered by a transient stimulus, at the dashed line. The response has then been applied for a fixed duration. Due to the simple nature

---

[2] In the example paths shown above (Figure 6.15) the difference in path shapes are due to the fixed priority value assigned to the local navigation behaviour. Improved performance may be achieved by lowering the priority value while the agent is not engaged on avoidance behaviour. There are several ways to do this, the simplest being to remove the zero degree heading singleton, change the input membership functions and pass the sum of the firing weights through as the priority value. As the performance of the controller in the traces shown is deemed adequate, such a radical change of implementation was avoided.

of the simulation, programme flow within the software is linear, procedural; this allows the cycles of the controller to be used as the timing element responsible for maintaining the response behaviour. The duration of the response has been arbitrarily fixed at 250 polls of system sensors, as this is sufficient for performance of navigation actions.

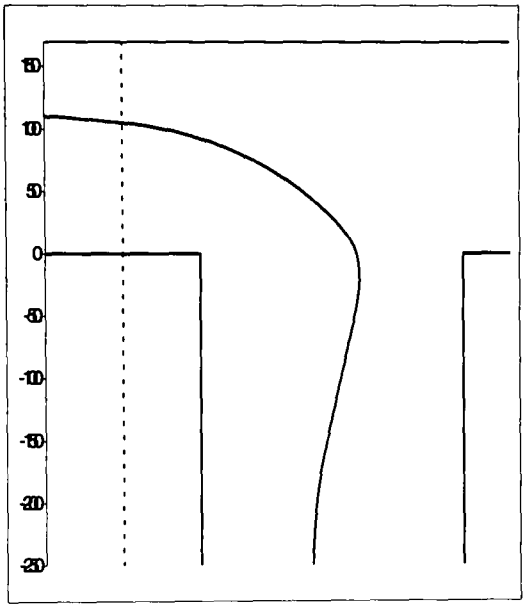


Figure 6.14 Path of agent while performing a turn.

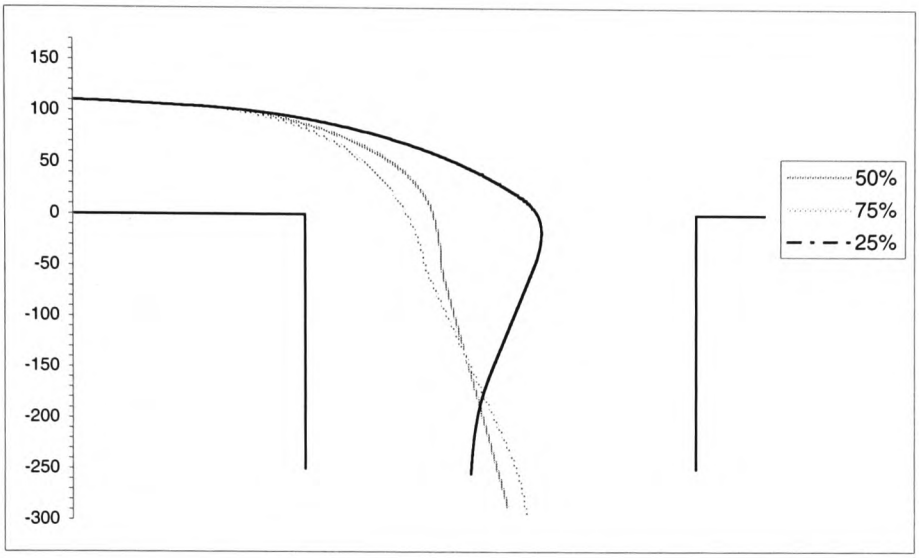


Figure 6.15 Effect of different utility values

With each system poll of the sensors the command interpreter reads the output of the recognition trigger. This output is an enumeration of the set of recognised features, with the addition of a "No Feature" state. The read condition is compared to that read from the current line of the script. If there is no match the condition is ignored. When no action is required, the utility value of the way-finding system output is set to zero, preventing interaction of the system with the agent. If the condition detected does match that read from the script the appropriate response is made. This response is either a decrement of the sequence counter or placement of a value at the heading and speed control outputs, while setting the utility factor of the output to some non-zero value.

Reading the script is rather simple. Scripts are created using enumerations to encode simple English expressions. Only very few options are possible, as the sets of possible actions and trigger conditions are limited. The script is stored in an ASCII flat-file, for simple generation, editing and interpretation. Because of this scripts may be generated within any text editor. The basic form of the script is of a collection of instructions, line separated of the form:

```
condition Sequence_number action
```

An short example is given in Figure 6.16. The simple nature of the scripts can be clearly seen.

```
leftOpening 3 left
leftOpening 1 left
rightOpening 2 right
rightOpening 4 stop
```

Figure 6.16 Example script

A simplified flow chart of the present, procedural system is given in Figure 6.17. As can be seen the command system itself is very simple. After the declaration of the appropriate symbol set (by enumeration), and assigning actual output values to these symbols the command script interpreter is rather straight forward.

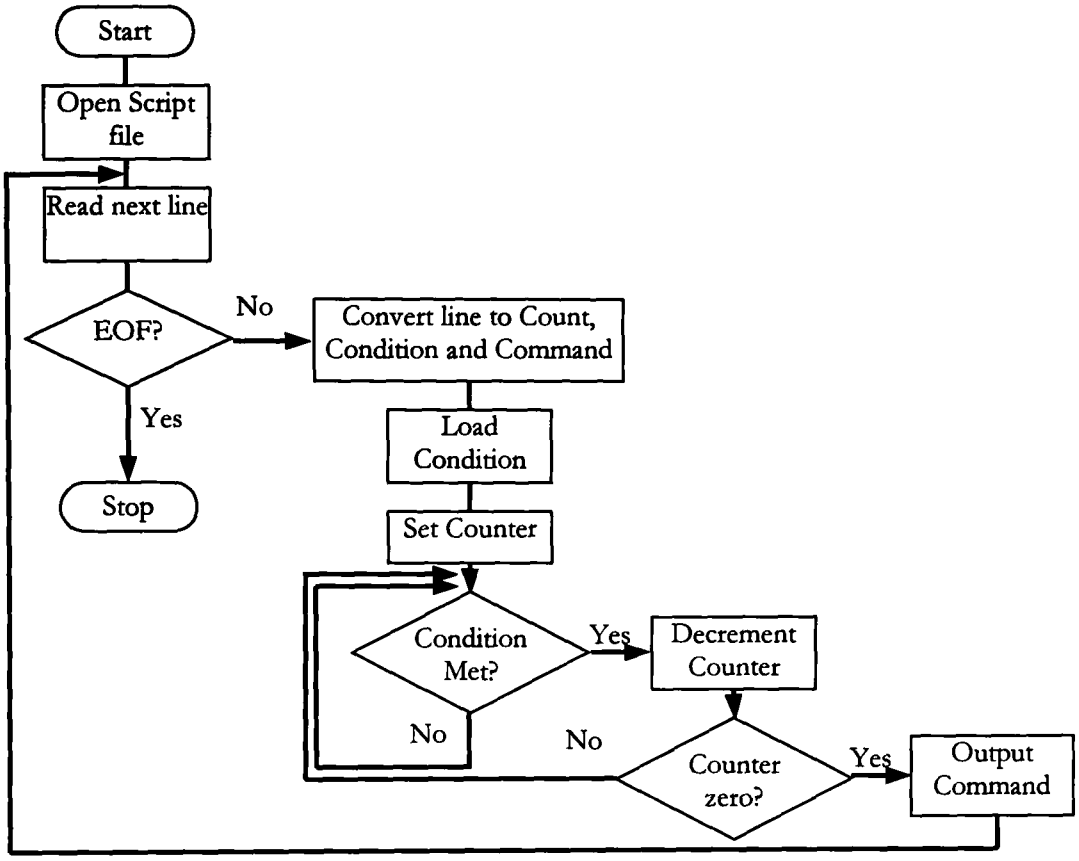


Figure 6.17 Simplified command system flowchart

6.10 Manipulation of scripts

In addition to providing a communications channel for navigation, scripts, like any other representation of space may be used to generate novel routes. To do this a set of scripts is required. A single script contains only that information required for the successful following of the route which it describes, when used in conjunction with other scripts however, a fuller representation of the environment is capable. To do this certain calculations must be made.

6.10.1 Calculations in direction space.

The reduction of spatial representation into a series of directions destroys the ability of the navigating agent to apply precise geometrical transforms to the representation. However, as the

script is a representation of a real topological landscape, and not just a record of a walk through a digraph, scripts may be manipulated to generate additional paths.

6.10.1.1 Concatenation

The simplest of the manipulations that yields novel routes is concatenation. By appending a route to the end of another, a script is generated that takes the agent from the start of the first to the end of the second. Care must be exercised to ensure that the terminal location in the first script is repeated in the initial location of that which follows (Figure 6.18). Using this method scripts may be generated of arbitrary length from any number of contributory scripts.

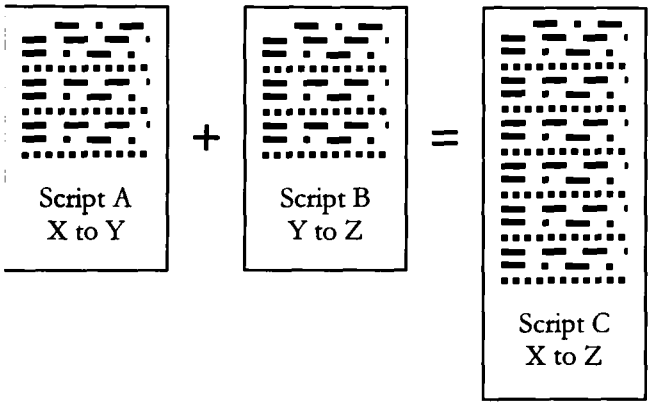


Figure 6.18 Concatenating scripts

There is a potential problem with the direct concatenation of two scripts to generate a third, caused by the loss of information experienced at the script boundaries. The beginning of the script does not define the action that should be made when the start location is reached, to achieve the second, as it is not known how the agent has achieved this point it can not. The end of a script can not define the actions that should be made in another script. This problem must be solved by intervention by the author, or by extension to the script structure (see section 6.11 below).

6.10.1.2Extraction

By comparing scripts, sub-scripts may be extracted, describing the route between locations contained within the scripts, but not explicitly defined. There are two issues here that must be considered. Firstly the child script must be an exact sub-script of the parent. Secondly, the child must be located at either the beginning or end of the parent-script. Without this concession it is impossible to ensure that the scripts are referring to the same area of space, exacerbated by the use of generic features. The only uniquely identifiable locations in the script are those named by their locations at the head and tail.

6.10.1.3Reversal

Whereas the concatenation of and extraction from scripts generate new routes by use of the representation of space provided by a database of scripts, reversal may be used to generate the route of a return journey, directly from the outgoing script. The basic procedure is to invert all the elements of the script, and then read the script backwards. An example out-bound journey and its reverse are shown in Table 6.1 (not in script form). The table refers to the network illustrated in Figure 6.19.

Outbound (A to F)	Return (F to A)
B: turn left	E: turn left
C: straight on	D: turn right
D: turn left	C: straight on
E: turn right	B: turn right

Table 6.1 Script and its reverse

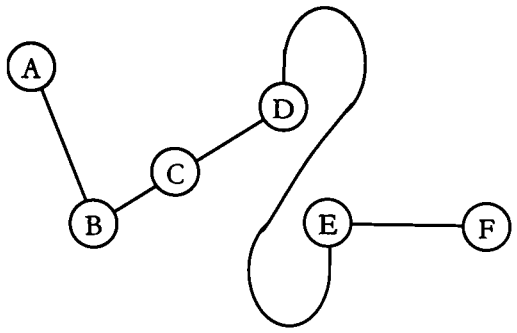


Figure 6.19 Simple path network

Even complex paths between nodes may be navigated in this way, as the turn information given in the script applies to relationship between arcs at a local scale, not to the large geometrical relationship between nodes.

As stated earlier, the reversal of routes does not always work. Issues to be addressed with reversal are mainly based on the reversal of the recognisable features of the environment used to determine the nodes of the taxological map. As stated above the constrained environment is preferred as it simplifies the recognition process, if, however, the location is approached from a different arc this advantage is destroyed. Recognition of location then becomes impossible.

Not all actions have a simple inverse. Consider the location illustrated in Figure 6.20.

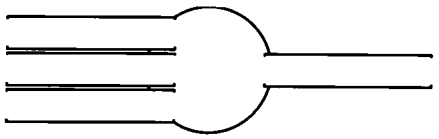


Figure 6.20 No simple reversal exists for this node

Travelling left to right through the node is simple; a straight-on command is sufficient. The return journey is, however, marred by there being insufficient information to continue if a straight-on command is given. Any of the three arcs leaving the left of the node may have been the origin of the "Straight on" command that formed the original route. Worse, the node may

have been omitted from the original route if no action were to be performed ("straight on" may be achieved by simply continuing to travel). The complexity of the recognition and command sets must thus be increased if reversal is to be achieved in environments that require such complex representations.

Each node must be considered as being approached by each arc to which it forms the terminus. The sets of commands and trigger features must then be expanded to include sufficient information to provide recognition of, and an appropriate action for each node from each direction. If reversal is to be employed an inverse locator and action must also be generated for every item in these sets. The task of reversal can be far from trivial.

Certain features of the environment, however, may be recognised from both directions easily, especially within built environments. The use of doorway openings as discussed above not only has the advantage of locating the stimulus directly at nodes on the network, but also provides a reversal feature. A doorway opening to the left of a corridor is the inverse of one to the right. This applies not only to recognition features based on a rapid change in range information, but also for more reliable and elaborate schemes such as those based on edge detection in images. This reversible nature is due entirely to the simple formal structure of the space in which the agent moves.

6.11 Extension to the script system

A simple and fruitful extension to the script system is to include location labels. A set of labels may be used to uniquely identify each location within the script. When added to the script representation these labels provide sufficient information to allow for the extraction of sub-scripts from anywhere within a parent, reduction in the concatenation problem by allowing over-lapping and simplified communication of scripts between generators by reducing the script to a list of labels (where information about the relationship between labels is available to the agent).

Labels need not be provided for all locations, a sensible compromise is to provide labels merely for salient features.

6.12 Summary and conclusions

Many navigation tasks are constrained to spaces that are path constricted. That is, the path of the navigating agent is heavily controlled by the environment, in many cases to the point that the space through which the agent travels may be represented by a graph; nodes of space connected by arcs (see Chapter 7).

All navigation tasks (more correctly all way-finding tasks) require that some representation of space is provided. From this representation routes must be extracted.

While engaged on directed tasks the route of an agent is rarely totally autonomous. The task often defines the start and goal location at least. If an agent is to be engaged on such a task, this information must be provided by the author. Thus navigation becomes the creation and use of a route dictated by the author. This throws up issues of communication.

The graph representation of space helps to alleviate the communication problem, as a graph may be represented by a list. For navigation purposes this list is a set of directions. Each entry in the list comprises of at least a trigger stimulus and an action to be performed, additions to the list may include a second goal stimulus. To use the list an entry is read, when the trigger stimulus is detected, the action performed. The next entry is then read and the procedure repeated. Where a goal stimulus is also included the action is performed until the goal stimulus is achieved.

Problems with the use of routes are the recognition of the trigger stimulus, and the maintenance of the route between trigger points. In a constrained environment, however, the constraint to the path of the agent forms a "guidance" to the agent. The actions to be performed are thus "orientations".

The recognition of location is improved by the constraint on the agent as this reduces the object constancy problem. The constraint ensures that the agent approaches a location from a restricted range of angles. The constraint imposed by the route, however, further improves the recognition of trigger stimuli by allowing general, stereotypical features to be used, with unique identifying features, as the route dictates which locations will be visited.

The representation of the route may thus be reduced to a simple script that lists generic features of the environment and actions to be performed at them. The constraint on the environment also controls the form of actions that may be performed, so making the definition of the route very simple.

By employing scripts the representation may be compressed. The route need only be described by those locations where an action is to be performed. Where the route passes many potential trigger features, but does not deviate from the path imposed by the spatial constrictions, these need not be recorded. However, as the script is expectation driven, features that match the trigger stimulus of an action, but exist between the location of the last action and the goal action, must be accounted for. This can either be by explicitly commanding the agent to continue past them, or to record the number of stimulus points that must be negotiated before the action is performed.

If these route scripts are recorded they may be used to generate novel routes through the space which they describe. This shows that they truly form a representation of the space, but that it is much reduced. For these novel routes to be extracted a collection of route scripts is required, these must then be matched-up with each other. In the standard script representation only the starting and finishing locations are named, thus only two locations in each script may be identified. Operations using scripts must thus be restricted to those that employ the start of end operand scripts.

The script representation may be augmented to relieve this need by adding labels to locations. Once a location is named in this way, the number and types of operation that can be performed on the scripts may be increased many-fold.

A script based navigational system may be easily integrated with the fuzzy behavioural architecture described in Chapter 6, and is in itself easy to implement.

The major problem with using such a representation is ensuring that sufficient symbols are provided to describe all necessary trigger stimuli, and all response behaviours.

Though behavioural control is normally not considered to be a symbol processing technique, the fuzzy behavioural architecture may be classified as a motor-schema method, as such it employs a common output format for each behavioural sub-unit. These output formats are themselves symbols. The symbol manipulation of the script is thus totally compatible with the local navigation system. The script based navigation system can be used to generate action demands that are of the same form as those generated by the individual behaviours of the controller. Thus the navigation system may be directly integrated into the controller structure.

Combining constrained environments with a script based route navigation system and the use of generic localisation features provides a powerful yet simple to implement way-finding technique that may integrated easily with local-navigation as performed by behavioural controllers.

6.13 References

Aylett, R.S., Coddington, A.M., et al. 1995. Heterogeneous Agents for Multi-Robot Cooperation. *Design and Development of Autonomous Agents* IEE digest no. 95/211.

Bagrow. 1964. *History of Cartography.* London: Watts.

Baptiste, P., Bideaux, E., et al. 1994. Use of Perception Based Navigation for Autonomously

Guided Vehicles. *1994 Joint Hungarian-British Mechatronics Conference* : 279-84.

Board C. 1967. Maps As Models. pp 671-725. Methuen & Co Ltd.

Booch G. 1994. *Object-Oriented Analysis and Design.* Reading Mass.: Addison Wesley.

Brooks, R.A. 1995. Intelligence Without Reason. *The Artificial Life Route To Artificial Intelligence, Building Situated Embodied Agents.* Eds. STEELS and BROOKS, pp25-81. Hove: Lawrence Earlbaum Associates.

Coad P., and Yourdon E. 1990. *Object-Oriented Analysis.* New Jersey: Yourdon Press.

Kortenkamp D., Bonasso R.P., and Murphy R. 1998. *Artificial Intelligence for Mobile Robots.*

Mataric M. and Brooks R.A. 1990. Learning a Distributed Map Representation Based on Navigation Behaviours. *USA- Japan Symposium on Flexible Automation,*

Mataric M.J. 1991. Navigating With a Rat Brain: A Neurobiologically-Insprired Model for Robot Spatial Representation. *From Animals to Animats.*, Eds MEYER and WILSON pp169-175. Cambridge Mass. London: MIT Press.

Matellan V. and Barrajo D. 1998. ABC^2 An Architecture for intelligent Autonomous Systems. *3rd IFAC Symposium on Intelligent Autonomous Vehicles.* pp 57-61.

Murphy R.R. 1996. Use of Scripts for Coordinating Perception and Action. *Proc. IROS 1996*, pp 156-61.

Nehmzow U. and Smithers T. 1991. Map Building Using Self-Organizing Networks. *From Animals to Animats.* Eds MEYER and WILSON, pp152-159. Cambridge Mass. London: MIT Press.

O'Keefe J. and Burgess N. 1996. Geometric Determinants of the Place Fields of Hippocampal Neurons. *Nature* 381, no. 6581: pp425-428.

O'Keefe J., and Nadel L. 1978. *The Hippocampus As Cognitive Map.* Oxford: Clarendon.

Prescott T.J. 1994. Spatial Learning and Representations in Animats. *From Animals to Animats 3: 3rd Int. Conf. Simulation of Adaptive Behaviour*, Ed CLIFF D, pp164-73Cambridge MA.: MIT Press.

Stein L.A. 1991. "Imagination and Situated Cognition." *Imagination and Situated Cognition*, A.I. Memo no. 1277. MIT.

7: IDENTIFICATION OF MAZE SPACES

7.1 Introduction

The navigation technique proposed in the Chapter 6 relies heavily on the constrained nature of the environments in which the agent travels. These spaces may be identified as being maze-like, or "maze spaces".

The dependence of the navigational technique on the nature of the spaces in which it is to be used requires that spaces of this form be identified reliably.

The following chapter outlines the properties of maze-like spaces, provides a definition, and suggests how these spaces may be identified.

7.2 Spatial phases: object matrix or maze space

The simplest way of considering a space is as a matrix of locations. Each location may be either occupied or empty. The areas not containing obstacles are considered free space and thus navigable. The position of these obstacles form the possible paths of an agent. Intuitively it appears that the size and number of obstacles will affect the ease with which an agent may navigate a space. However there are cases where the occupancy of space may be minuscule, while the navigable paths are tortuous. These are maze spaces. The maze for the IEEE micromouse competition has a maximum 12.8% occupancy, but is by no means simple to navigate <www.uml.edu/Dept/EE/IEEE/rules.htm>. Considering a space as a maze also has advantages for those case where the occupancy of a space is high.

The difference, conceptually, between space as an object matrix and a maze is one of viewpoint. A maze is characterised by its free space, while an object matrix is characterised by its occupied regions. At some threshold of occupancy it becomes more convenient to consider space as a maze. This threshold is not fixed across all spaces, however, and is more akin to the solidification of vitreous materials, where the size and shape of the components determines the temperature at which the phase change occurs. Also, like vitreous materials, the appearance, maze or object matrix, of a space depends on scale. For very large time scales what appears to be solid glass can be seen to be a highly viscous liquid. The corridors of the University of Wales College, Newport's Allt-Yr-Yn campus are approximately 178 cm wide, at human scales[1] they form a definite maze space. To a Khepera robot <http://diwww.epfl.ch/lami/robots/K-family/>, with its diminutive size (55mm diameter), an area of this size represents a vast open space.

7.2.1 Identification of maze spaces

Some way of identifying a space as either an object matrix or maze is required. As a starting point for a definition it can be assumed that an agent within a maze will have a restricted choice of path. These paths are determined by the structure of the maze. An agent is path restricted if the sum angles of the possible path cones is smaller than the cones produced by non-navigable space. Figure 7.1 provides an illustration. Here the range of possible paths are signified by the dashed lines. The sum of the angles over which the agent may travel is:

$$\Theta_{free} = \Theta_f + \Theta_r \qquad (7.1$$

The restricted areas however prevent the agent from travelling cones with an angle of:

---

[1] The scale of a human in 2 dimensions here is taken to be about 70cm, a measurement obtained from an average single doorway. It is assumed to be the smallest size that the majority of adults find comfortable.

$$\Theta_{restricted} = \Phi_l + \Phi_r \qquad (7.2$$

The agent is path restricted as:

$$\Theta_{free} < \Theta_{restricted} \qquad (7.3$$

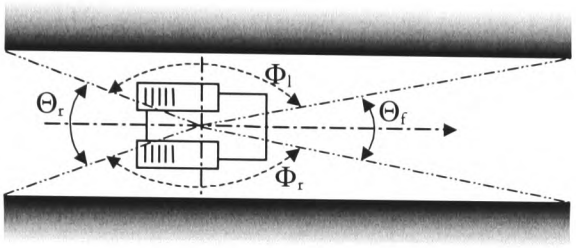


Figure 7.1 Identification of maze spaces

This can be rewritten simply:

$$\Theta_{free} < 180° \qquad (7.4$$

7.2.1.1 Terminal edges

At the edges of a space this measure collapses (Figure 7.2). Here the agent is path restricted, regardless of the major features of the space. A space should not be identified as a maze, simply because an agent suffers path restriction at any point.

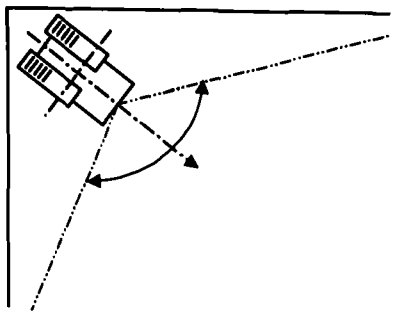


Figure 7.2 Effect of spatial termination on free paths

7.2.1.2 Path intersection

At the intersection of maze segments the sum of the traversable cone angles may exceed 180°. Figure 7.3 shows such a condition, to an agent placed at the centre of the intersection, the sum of the arcs of possible paths is nearly a complete circle, however the space is still a maze.

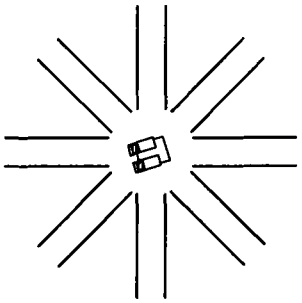


Figure 7.3 The angle criteria does not hold at maze segment intersections

The maze may be recognised in this case, as the agent is present with a set of discrete path options at this intersection. In Figure 7.2 where potential paths are constrained by the terminal edges of the environment, the agent has an infinite number of path options.

7.2.2 Maze recognition

Combining these criteria gives a definition of a maze space where:

- A maze comprises maze segments and intersections.

- A maze segment represents an area of space, where the path of the agent is restricted.

- An intersection is an area of space where maze segments terminate (and by implication may meet), such that an agent, when at an intersection must, by travelling, move into a maze segment.

The rather poor definition of a maze segment stated above may now be replaced by a recursive definition that relies on intersections:

- A maze segment is an area of space, which restricts the movement of a contained agent such that the agent may only travel to the segment's two terminal intersections.

Further, it is possible to envisage two possible maze segment forms: a "tight" maze segment and a "loose" maze segment. A tight maze segment is such that an agent entering at one terminal intersection must reach the other terminal intersection, assuming that travel continues, without reversal after the segment is entered. A loose maze segment restricts the travel of an agent to a lesser extent, an agent may reach either terminal, after entering the segment, while maintaining an arc of forward travel (Figure 7.4a).

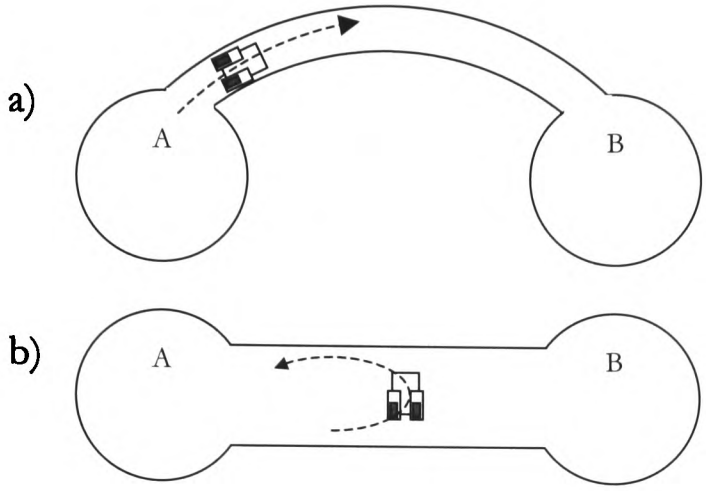


Figure 7.4 Tight (a) and loose maze segments (b)

7.2.3 Path restriction

A definition is now needed for what constitutes a restriction of path. Clearly hard physical constraint, such as a train feels from railway tracks is a restriction, but so are the lanes of a motorway, where a driver is constrained by behaviour. Thus a space is path restricted if the agent, following *all* normal behavioural patterns, is constrained. If, however, the agent is only restricted when employing a subset of its normal behaviours, then the path is *behaviourally restricted.*

7.3 Hybrid spaces

It is possible for the obstacles within an open space to be arranged so as to form a maze like sub-space. Also possible are maze segments that terminate in areas of open space, areas where the agent is not confined to a finite number of path options.

When considering the effect of such spaces on navigation, the behavioural repertoire of the contained agent is important. Any agent provided with only the means to travel either a maze, or open space will not be able to navigate easily. A potential behavioural safety net exists. Inclusion of behaviours such as wall-following that cause a maze travelling agent to remain close to the terminal

edges of an open spaces, create behavioural path restrictions. An agent optimised for use in an open space could be encouraged to avoid those areas where the environment in which it operates has maze like properties.

7.4 Representation of maze spaces.

A maze may be represented simply by a graph. A maze which is comprised exclusively of loose maze segments may be represented by a simple graph, intersections being the nodes, while the segments are represented by arcs. A maze which contains tight segments must be represented by a digraph.

The space should be categorised as to whether it may be considered a plane with objects, or a network of channels between objects connecting voids. The ultimate expression of this form of space would be a classical maze, where the space has very linear dimensions, but is highly convoluted.

7.5 Summary

Space may be considered not as void within which occupied regions (objects) are scattered, but as a network of intersecting paths. The validity of this view for a space is dependent on both the scale of the agent navigating within it and the morphology of the occupying regions. Identification of a space as a maze has implications for the navigation task, the communication of navigational goals and types of representation that may be used for navigation (see Chapter 6). The importance to navigation of using this classification has lead to the development of a definition of maze spaces:

- A maze comprises maze segments and intersections.

- A maze segment is an area of space which restricts the movement of an agent such that the agent may only travel to the segment's two terminal intersections.

- An intersection is an area of space where maze segments terminate (and by implication may meet), such that an agent, when at an intersection must, by travelling, move into a maze segment.

Effectively a maze space is the physical realisation of a graph.

Although 'real-world' environments are likely to fall on a spectrum somewhere between the extremes of maze and open space, the exploitation of the features of mazes spaces for the simplification of all aspects of the navigation task can be expedited by the use of behavioural constraints (such as wall following) that allow the open areas of a hybrid space to be represented and identified using the same tools as the more maze like segments.

8: CATEGORISATION OF SPACES

8.1 Introduction

Measurement of performance requires that systems be compared. For a comparison to be valid it must be between similar items. Thus the measurement of navigation systems performance requires that some method be provided such that the spaces in which a navigating agent operates may be compared. At present there are no methods of doing this.

The following chapter briefly discusses the requirements of a useful index of spatial complexity in relation to the navigational problem, specifically way-finding. Potential categorisation methods are also suggested based on assumptions and observations of those features of a space that influence navigation.

8.1.1 Why should spaces be categorised?

Measurement of performance is so fundamental that it is often taken for granted. The selection of a technique or product requires a comparison of performance between choices. For this comparison to have any value, measurements must be made under the same conditions. A good performance under one set of conditions does not guarantee similar performance under all others. Environments differ naturally. Maintaining, or reproducing the exact conditions previously used can be problematic. Where there is a range of potential usage modes, performance may be legitimately optimised for one situation, in which case the optimal performance conditions must be recorded. For these reasons, whenever navigational performance is to be measured and compared, the space in which the agent operates must be categorised, for identification purposes.

In Chapter 6, the importance of making spatial classifications is highlighted. Here the proposed navigational simplification depends on the space taking a special, constrained form. Before this technique may be applied to an agent the space in which it operates must be identified as being suitable.

Most work on the analysis and categorisation of spaces has been in support of the life sciences and to obtain geological data. In these cases the analysis and categorisation is concerned mainly with the distribution of some property, not with the form of the space itself (for examples see (Koetje 1987)(Haining 1990)). Application of techniques generated by these disciplines is thus inappropriate. A method of categorising space according to the requirements of navigation is needed.

8.2 Spatial categorisation requirements of navigation

When considering navigational strategies and implementation, space must be categorised according to the difficulties presented to the navigator. The issue is clouded, as navigation comprises two distinct tasks: moving through the immediate locality while avoiding obstacles; and moving from a present location to a target, which may be outside the immediate locality.

Like most other fields navigation may be achieved by varied techniques. These are suited to specific types of environment, and specific navigation task.

8.3 Navigational difficulty

Navigation depends not just on the ability of a vehicle to determine actions to be taken, but also to localise within a spatial representation. The ability to localise depends on the sensory ability of the machine, as well as the topography and properties of the space in which the task is being carried out. A perfectly empty space is an example of this. Navigation here has no meaning as all points within the space are alike. In addition, localisation is not possible within a featureless space.

There are other cases where problems may be experienced if the sensory abilities of the machine do not match the features of the environment or structure of the framework being used to represent space. A space which is empty but for some form of orientation framework may be difficult to navigate, depending on the form this framework takes. If the framework is egocentric to the agent, navigation is simple provided the machine has some form of proprioceptive system and goal locations are given in terms relative to the agent.

Where the machine is expected to move from its present, to a new location within an allocentric framework, navigation may be impossible if there is a lack of sufficient localisation information. If no suitable information can be drawn from the environment, the framework is useless.

Without proprioceptive feedback movement within a featureless space produces no experience. This precludes the determination of location as a function of movement.

8.4 Categorisation goals

The required features of spatial categorisation are:

1) Identify any different types of space that may exist, especially where these may suit the application of different navigation techniques.

2) Provide some method of comparing the "difficulty" of navigation within type similar spaces.

3) The categorisation technique needs to be simple to apply, and produce a simple measure of spatial type and navigational complexity.

8.5 Indices of complexity

Many authors refer to the environments into which they place their machines as complex, but few seem to have addressed the nature of this complexity. It is not sufficient to simply state that an environment is complex, some measure is required so different environments may be compared.

8.5.1 Occupancy

Intuitively a good measure of the complexity of space would appear to be related to the quantity and form of occupancy.

8.5.1.1 Measuring occupancy

For most navigational purposes space may be considered as a matrix of locations, each of which may, or may not, be occupied. To be considered occupied, the location must be in such a state that the navigating machine is precluded from entering. If an object occupies a position, that location should be considered occupied only if the object interferes with the free movement of the machine through that location. If the machine can pass under or over the object freely, that location is not considered occupied. All areas into which the agent cannot travel for some other reason, such as surface texture, are also considered to be occupied[1]. The simplest model of this type is a two-dimensional square grid, with binary occupancy (if the machine is assumed to be a land vehicle). If the matrix is formed by a squared grid, such a representation is referred to as a raster.

---

[1] A more realistic measure may be to assign a traversability index to each location that reflects the difficulty with which a machine may pass through that cell. Paths may then be calculated to be optimised for distance or ease. For simplicity, however, such measures are not considered here.

8.5.2 Spatial integrity

Spatial integrity is used as an index of spatial complexity when comparing landscapes. It is a techniques especially suited to investigations into biodiversity etc. It is also remarkably simple to apply. The number of "holes" and "fractions" are counted. A fraction is an independent area of some feature, while a hole is an area within a fraction where the feature does not apply. As an example consider the distribution of wood within parkland. Each wooded area is a fraction, while clearings are holes. A simple relationship is then used to determine the Euler number, a measure of spatial complexity (equation 8.1). A Euler number of zero shows low complexity, large absolute values indicate high spatial complexity. This is a very simple measure so can only give a coarse indication of the complexity of the spaces under consideration. Figure 8.1 shows the spatial integrity for two example spaces. Calculation of the Euler number for these spaces is developed in equations 2 and 3 respectively.

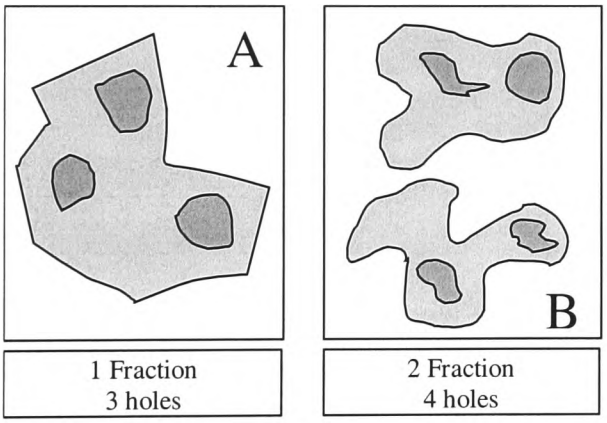


Figure 8.1 Spatial integrity

$$SpatialIntegrity(E) = holes - (fractions - 1) \quad (8.1$$

$$E_A = 3 - (1 - 1)$$
$$E_A = 3 \quad (8.2$$

$$E_B = 4-(2-1)$$
$$E_B = 3 \qquad (8.3$$

The example above clearly shows that the spatial integrity of a space cannot be used for navigation purposes since both spaces show the same spatial integrity value, but space B is a more difficult navigation problem. Spatial integrity is a measure of the completeness of the occupied spaces, not an indication of the total complexity of spatial occupancy. it can be assumed that navigation is concerned with those areas which are both unoccupied and joined.

Assuming that the agent is navigating the greater space, taking account of the number of fractions into which the occupied space can be divided does, however provide the type of information that is required. In the example above, assuming binary occupancy of approximately equal areas, a simple product provides a more useful index of navigational difficulty, with space B having an index double the magnitude of space A.

8.5.3 Complexity

An obvious method to compare similar spaces is to directly measure the complexity. Simple to apply measures of the occupational complexity of a space can be based on the information content of the space and the fractal dimensionality. This has the advantage of taking into account the distribution of the occupied spaces and may be achieved by assigning a higher navigational difficulty value to topographically interesting spaces than those whose more regular structures. There are several approaches to the measurement of this complexity.

8.5.3.1 Information content

Simple observation of model spaces that conform to this representation suggests that minimum complexity occurs when a space is either fully occupied or empty. Extrapolating from this assumption, maximum complexity is likely to exist at around 50% occupancy of the available space.

Obviously the distribution of the occupied cells across the space is also important. Where the objects form regular occupancy patterns, the complexity of the space is lower than when the space is filled randomly. A simple measure of the complexity of a space may be derived from these assumptions, by reading the occupancy grid as a continuous bit stream. The randomness of this stream gives an indication of the information contained in the grid, and thus the complexity of the space. Application of this technique supports the view that maximum complexity occurs when there is 50% occupancy of the space. However, this does not mean that a space with 50% occupancy is maximally complex. Figure 8.2 shows four samples spaces, exhibiting 0, 100, 50 and 50 % occupancy respectively. It may be noted that while the complexity at 0 and 100% is fixed, the complexity exhibited at 50% occupancy varies with distribution.

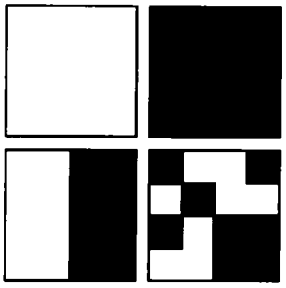


Figure 8.2. Zero and 100% spaces are very low complexity. 50% occupancy creates varied complexity of space

The bit streams produced by the four spaces illustrated in Figure 8.2 (when a quarter-side square cell grid is employed aligned with the sides of the space) are:

```
0000000000000000
1111111111111111
0011001100110011
1001010010110011
```

Determination of the information content of the bit streams may then be performed by auto-correlation of these streams.

8.5.3.2 Fractal dimensionality

An alternative measure of spatial complexity can be derived by obtaining a measure of the fractal dimensionality of the space. The fractal dimension is a gauge of the scale dependence of a measurement. With reference to spatial complexity the fractal dimension may be calculated based on the occupied area at different scales, or on the spectral power density of the size of occupying features.

8.5.3.2.1 Occupancy/scale ratio

The "Coast line approach" may be used to determine the fractal dimension of a shape, by measuring the scale dependency of it's perimeter (Green 1995). The length of a coastline measured on a map is dependent on the length of the scale used to measure it (Figure 8.3):

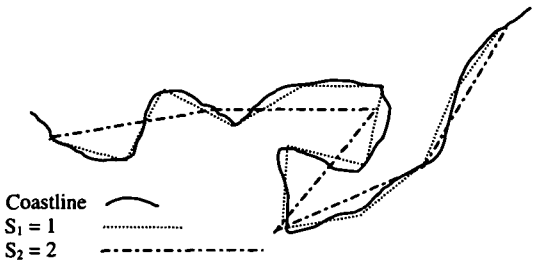


Figure 8.3 Measuring a coastline at varied scales

The length of the coastline is measured with two "sticks" one of greater length than the other. The number of "stick" lengths is recorded for each. The fractal dimension is then found by taking the ratio, of the log of the ratios of the length of the measure, and the recorded lengths (equation 8.4):

$S_1 = 1; L_1 = 12$
$S_2 = 2; L_2 = 5$

$$D = \frac{\log\left(L_2 / L_1\right)}{\log\left(S_1 / S_2\right)} \qquad (8.4$$

$$D = \frac{\log(0.461)}{\log(0.5)} = 1.265 \qquad (8.5$$

To apply a similar technique to find the occupational fractal dimension of a space, grids of different granular size should be applied to the space. The ratio of the grid sizes should be recorded, along with the number of occupied cells for each grid. The fractal dimension is then given by a simple relationship:

$$D = \frac{\log(O_2 / O_1)}{\log(G_1 / G_2)} \qquad (8.6$$

where:

D is the fractal dimension

O_1 and O_2 are the occupancy values measured with grid sizes G_1 and G_2 respectively.

For these calculations the ratio of the grid areas will be used.

Examples:

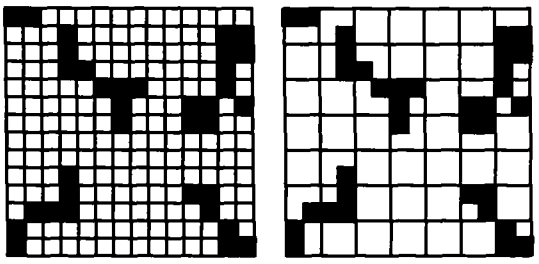


Figure 8.4 Measuring the occupancy of a space at different grid scales

When the occupancy of the space (Figure 8.4) is measured at the small scale there are 35 occupied locations. At the large scale there are 18. this gives:

$$D_{Area} = \frac{\log {}^{35}/_{18}}{\log {}^{4}/_{1}} = 0.4797 \qquad (8.7$$

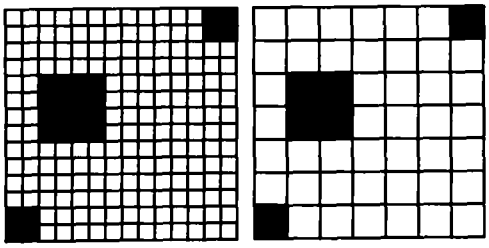


Figure 8.5 Example space

For the space illustrated in Figure 8.5. Large scale Occupancy is 6. At the small scale occupancy is 24.

$$D_{area} = \frac{\log(O_2 / O_1)}{\log(G_1 / G_2)} = 1 \qquad (8.8$$

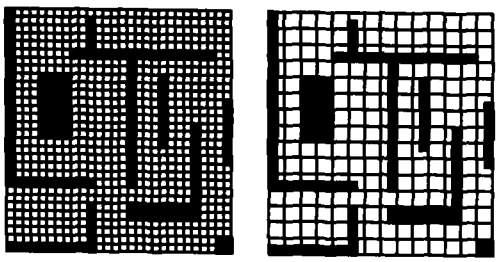


Figure 8.6 Example space

For the space shown in Figure 8.6, at the small grid scale there are 167 occupied cells. At the larger scale 72 cells are occupied giving $D_{area} = 0.6069$

This measure can be seen to be giving a reasonable indication of the morphology of the objects in a space. However there are problems, the space shown in Figure 8.5 has a fractal dimensionality of 1, this is the same value as a totally occupied, or empty space. Obviously this space is more complex than this. The figure in this obtained may also be varied by the placement of the grid. If the large scale grid is moved half a measure sideways, the occupancy at the large grid size increases to 10. In this case $D_{area} = 0.6315$

8.5.3.3 Spectrum

For cases where the fractal dimension of a surface needs to be determined, the height at various points along a cross section of the surface is measured, with reference to a base line. The "shore line" technique cannot be applied, but an approximation may be made from the power spectrum of the data series:

$$S(w) \approx k.w^5 - 2D \qquad (8.9$$

where:

D is the fractal dimension

w is the frequency, and

$S(w)$ is the spectral value of w

k is constant.

The use of a spectral content allows several gauges of spatial complexity. Either by measuring the distribution of fractions within a space, or by using it to determine the complexity of the shapes that occupy the space. If used to determine the complexity of fraction shapes, and used in conjunction with a measure of occupational distribution, the spectral content may provide a powerful metric. The length of the sides of the objects within the space may be recorded and used to provide the spectrum. Measuring the complexity of a space in this manner is dogged by the requirement to measure the number of sides of different lengths. This may be difficult for many shapes. When applied to spaces that have a form dependent on regular shapes and sizes (such as the built environment) the utility of this measure is also brought into question. The spectral measure of fractal dimension could be applied to the distances between objects, the distances recorded being the lengths of arcs in a complete graph, where the nodes are the centroids of the fractions that make up occupied space. Unfortunately, for such a measure to be valid, the space would need to contain a large number of fractions.

The complexity of applying these techniques makes them less useful, especially as there are more simple measures of the morphology of objects, and their distribution. The largest problem with applying these fractal measures of complexity to the analysis of navigational complexity is that the spaces under consideration will tend to be far from fractal in nature, meaning that although a fractal measure may be applied, it is not truly applicable. Fortunately alternative measures of occupied region morphology are available, more simple to apply and valid for all spaces.

8.5.4 Convexity

Convexity is a measure of the morphological complexity of a shape, effectively a gauge of the encroachment of the background into a shape. Convexity is found by a very simple relationship:

$$C = Perimeter/\sqrt{Area} \qquad (8.10$$

Minimal convexity is found on a circle, which has an index of:

$$C = \frac{2\pi}{\sqrt{\pi}} \qquad (8.11$$
$$C \approx 3.544$$

This value is often used as a datum, to which the convexities of other shapes are normalised.

Convexity provides a measure of the morphology of the occupied areas of a space, but not of the general complexity of that space. To do this, degree and distribution of occupancy must be taken into account. This may be achieved by simply finding the number of fractions (occupied areas), the general degree of occupancy and the mean fraction convexity. Additional information may be gained by also considering how the occupied space is distributed amongst the fractions, and how the fractions are distributed through open space. Generally, in this calculation the index should be normalised against the convexity of a circle. In rectilinear spaces, or where the representation is by raster, this normalisation may be replaced by the more simple normalisation against the convexity of the grid elements (for a square grid, $C_0 = 4$) as the smallest fundamental unit area. This simplified normalisation is especially applicable as the area of fractions in space will tend to be calculated by counting occupied cells, while the perimeter will be measured by the finding the sum of the sides that form the interface of the fraction with empty space (Figure 8.7).

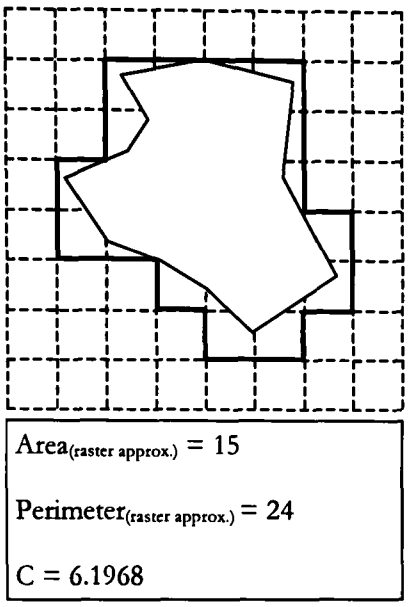


Figure 8.7 Raster approximation of convexity.

Using the convexity in this way does not take directly into account diminishing complexity of spaces as occupancy continues to increases. However, areas of free space smaller than the machine cannot be entered. They are effectively occupied. This limits the closeness that two fractions may exhibit and while still being considered disjoint. When the routes that join spaces are considered in terms of navigability this is further emphasised. A route should only be considered navigable if it has dimension suitable for the agent to traverse, not just occupy contiguous positions. The swept area of the machine while turning must also be taken into account. It is important to consider the configuration space (section 8.7) of the agent, not the physical space in which it operates. When those areas of space into which an agent may not enter are eliminated, being considered occupied, the number of fractions is reduced due to aggregation. If the fractions are sparse enough that the calculation of configuration space has no implication for the degree of navigability of the space, then the occupancy is low enough that these considerations may be ignored. As the fractions of a space converge with increased occupancy, a phase change may be observed in the space. The absolute occupational density when this occurs is dependent on the morphology of the fractions within a space, but the implications for navigation of this change are great (see Chapter 7).

8.6 Scale considerations and configuration space

The scale of the features within an environment is important for the calculation of navigational complexity. This is especially true where measures of fractal dimensionality are to be used. The spaces under consideration are unlikely to be truly fractal (self-similar over all scales). The features present will occur only in a limited range of sizes, meaning the "fractal dimension" determined by any technique would be dependent on the scales over which the measurements are made. As it is intended to use the fractal dimensionality index derived in this manner as a measure of the complexity of a space with respect to the navigation task, it becomes apparent that only those features that may affect navigation should be considered. The range of scales valid for the calculation is thus fixed by the properties of the navigating agent. The same criteria should be used with any technique used to determine the navigational complexity of a space; only those features that influence navigation should be considered.

As shown above, the concept of navigability also depends on the small scale (machine size and below) structure of the space. This type of detail may not be immediately available from raster representations of space as the position of the applied grid can make a large difference in apparent morphology of spatial features. Figure 8.8 shows the affect a small displacement of the grid may have. This problem could be alleviated by employing a check for navigability to constricted areas of the space that is independent of the master raster grid.

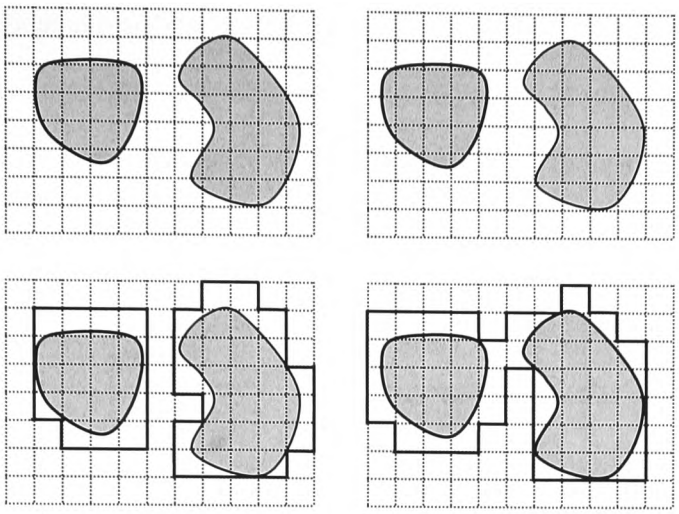


Figure 8.8 Position grid affects apparent navigability

8.7 Small scale obstacles

Features smaller than a vehicle may have an effect on its path if they form barriers to movement, while others that are larger do not pose a navigational impairment, and may be neglected. Features that form a barrier to movement may be very much smaller than the navigating agent, or the area through which the agent is to travel. While it would be possible to consider all features regardless of size within an occupancy calculation, such a measure would include much information (such as surface textures that do not influence navigability) not pertinent to the comparison of spaces for navigational complexity. How a feature is measured also depends on how the raster representation falls onto the occupied space, as shown in the previous section, this will have an impact on the perceived importance of a feature. A technique is required that will reduce the effect of the raster representation, and transform important features to a standard scale. In section 8.6 above, it was stated that the number of fractions will tend to decrease as they are aggregated if the space between is not navigable. These requirements may be met by calculating the configuration space of the agent prior to applying any categorisation technique.

Calculation of configuration space requires that the occupied fractions of a space are increased in size by an "erosion" algorithm so that any white space remaining within the configurations space would be achievable by the agent in environmental space (navigable). Obviously the morphology of the erosion and agent must be complimentary. After calculation of the configuration space the agent may be considered as a single point (de Berg et al., 1997).

The configuration space may be generated simply by selecting a single vertex and plotting its position while running the agent around the edge of the occupied region (Figure 8.9). The result is a non-uniformly expanded occupied region, such that the vehicle, if considered a single point, located at the selected vertex, can reach all post-transformation free space.

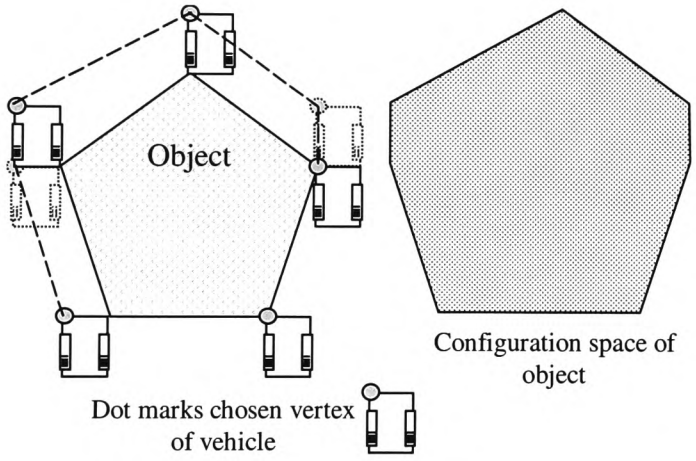


Figure 8.9 Derivation of configuration space by erosion

This derivation is fine for those path-planning algorithms that are intended to move around objects placed sparsely in a large open environment, or while ideal models of agents are employed. In those situations where the environment is considered cluttered there is of course the problem that a path found in this way may not be navigable, as the abilities of the vehicle have not be taken into account. Figure 8.10 shows this problem. An agent is placed in a maze like environment. The configuration space is derived as described, producing the supposedly navigable configuration space. Here the

turning circle of the agent is insufficient to navigate the maze, even though the configuration space would imply this is possible. For elongate agents, this problem is exacerbated. In the best possible case, where the agent is capable of rotating around its own centre, the area through which the body is swept may be significant, preventing the agent from traversing an apparently navigable configuration space. More complex methods of developing configuration space, which take into account the swept area of the vehicle, should thus be adopted.

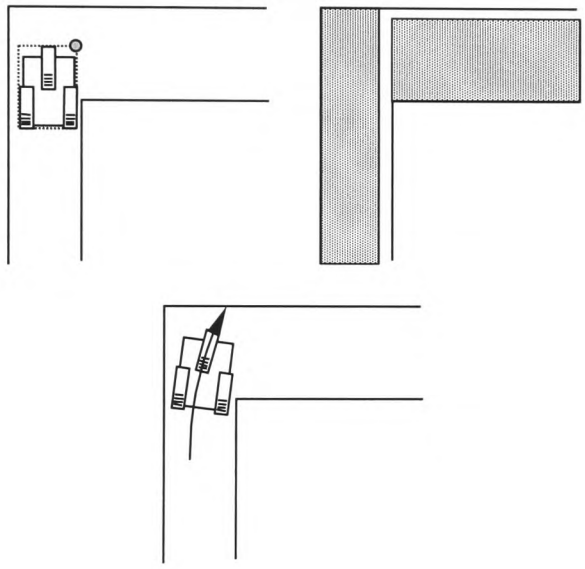


Figure 8.10 Capabilities and swept areas must be considered in cluttered spaces

Application of a configuration space transform will tend to reduce the number of fractions and increase the size of barrier features to a standard minimum, making complexity measures easier to apply.

8.8 Navigational difficulty in maze spaces

The way-finding navigational complexity of a maze space is determined by the difficulty an agent has in determining the correct maze segment to follow. Assuming that whatever cues are used are easily recognisable, it is reasonable to assume that the number of segments that meet at an intersection give

an indication of this. The complexity of a path will then be determined by the sum of the number of segments terminating at each of the intersections traversed. Where a measure of the navigational complexity of a maze space is required, it is suggested that the average number of intersections per journey (or average node density per square of journey length on mazes with an absolute spatial distribution) is multiplied by the average number of segments per intersection. As the graph representation of a maze contains no information on the intermediate topological features of a space, the local navigational complexity of a maze space must be considered independently. The difficulty an agent experiences travelling a maze segment depends on the topology of the maze segment. A measure of this can be found by taking a representative segment and determining the navigational difficulty it poses. This may then be extrapolated across all segments in the maze. The difficulty may be measured in terms of probability, speed, or cost (in terms of energy etc.) of traversal.

8.9 Representational alignment

Not considered in any of the techniques above is the difficulty of aligning a space with the representation used. In an open space this is the difficulty of recognising those features used to localise and orient the agent, in sufficient quantities. In a maze space, this is the difficulty of recognising those features used to discriminate maze segments at an intersection. This should be calculated or measured independently and specified along with the other measures of complexity suggested above.

8.10 Path optimisation

Measures of navigational difficulty may be used to select a single path when several are possible. The optimisation criteria must form the basis of the measure. Any attempt at path optimisation is liable to require consideration of both way-finding and local navigational aspects. To determine the time optimal path through a maze, the time required to select and enter the correct segment at each

intersection must be added to the sum of the transit times for each segment of a path. It is interesting to note that the importance and difficulty of selecting an optimal path are inversely related. As the number of possible paths increases, and the cost associated with each converge, the accuracy of cost measurement required selecting the best path increases, as they become lest distinguishable.

8.11 Summary

By categorisation, comparison between spaces can be performed. By ensuring that like spaces are used, categorisation allows the performance of different navigational strategies to be compared. Also, by allowing the differences between spaces to be stated, categorisation helps identify suitable navigational techniques for each particular space, not merely by implication (strategy A works in space X, so should work in other, similar spaces), but also by inference from the features the space possesses. To these goals categorisation needs to be informed by the correct measures of an environment, mainly the difficulty it poses to navigation. To achieve a suitable navigational difficulty index simple to apply measures are required that will provided information on the structure of the space.

Navigation comprises of several sub-tasks, each of which is important. The proceeding work has concentrated on the task of way-finding, while following a predetermined course. For this reason, the categorisation of space has been biased towards those features of an environment that influence this. The primary concern for any navigational task is where the agent may or may not travel. Travel is normally prevented by a location being occupied, so for convenience a binary measure of each location has been employed, each location being considered occupied (non-traversable) or empty.

By assuming that the locations within a space form a regular grid pattern and that each location has a Boolean occupancy value, several measures of complexity are suggested. These measure the degree of occupation and the way occupied space is distributed.

Both the degree and distribution may be measured directly by either determining the information content or the fractal dimension of the space directly. However, these measures may be difficult to apply and have flaws. The information content of the space depends on the randomness of the distribution of occupied areas at the scale of the grid, this is not directly related to navigational complexity, and may provide erroneous information. The measure of fractal dimension as an indication of complexity of space provides an indication of the morphological complexity of the occupied regions, but not of how the regions are distributed. An alternative is to measure the degree of occupancy and the manner in which it is distributed independently. When this method is pursued it becomes convenient to consider space as comprised of a number of 'clumps' of occupation (fractions) distributed through an open plane. The morphology of an occupied region may be indicated by its convexity, this in turn will give an indication of complexity of the free space in its region. A potential index of navigational ease can be determined by combining the convexity of the occupied region with the number of regions, and the overall level of occupation. Even more information may be added to the measure by considering how occupation is distributed amongst the regions, and how the regions are distributed in space.

Scale is an important consideration when measuring the complexity of a space for navigation. Not only the fractal dimension, which is a measure of the scale independence, is affected. Scale implications exist for all measures of complexity. This is exacerbated in cluttered environments where the spaces remaining between densely placed fractions may not be navigable. To relieve these problems the configuration space should be generated. Complexity measurements can then be made at an agent relevant scale.

Navigational difficulty in mazes is dependent on the number of intersections through which the agent needs to travel, and the difficulty inherent in travelling the paths.

8.12 References

de Berg M. van Kreveld M. Overmars O. and Schwartzkopf O. 1997. Robot Motion Planning *Computational Geometry.* pp 265-88. London: Springer.

Green D.G. *Fractals and Scale* URL: <life.csu.edu.au/complex/tutorials/tutorial3.htm>. 1995 (Accessed May 2000).

Haining R. 1990. *Spatial Data Analysis in the Social and Environmental Sciences.* Cambridge: Cambridge University Press.

Koetje T. A. 1987. Spatial Patterns in Magdelenian Open Air Sites From the Isle Valley, South Western France. Oxford: B.A.R.

9: Conclusions and Issues for Further Investigation

9.1 Summary of work and discussion

A suitable point of departure in the investigation of the generation of autonomous behaviour is the animal kingdom. Animals exhibit many of the properties which would be advantageous if endowed onto machines. To be able to grant these abilities animal behaviour should be understood. Chapter 2 investigated relevant literature to investigate the properties of animal behaviour, determine if it is a suitable model for intelligent behaviour in animals and to isolate those properties that make it so.

The mechanism by which behaviour is generated in animals cannot be investigated directly, but by observation of animals the following features have been identified:

- Behaviour is distributed throughout the animal's nervous system. Much behaviour is produced using simple computational units close to the relevant muscles or sense organs.
- Behaviour is the product of the interaction of simple behaviour sub units. Each sub unit only uses the subset of the animal's sensor space that is necessary for the correct operation of that behaviour.
- Behavioural mechanisms are tightly coupled to the features of the environment in which the animal operates.

- Behaviour selection depends on both the environment conditions and the internal state of the animal.
- Sequencing simple action patterns can create complex behaviour and result in the achievement of abstract goals, without explicit definition of those goals.
- Behaviour is tightly coupled to the morphology of the animal.
- Short cuts, simplifications and novel solutions to the implementation of behaviours can be widely observed.
- The use of a modular distributed control system improves the robustness of animal behaviour both to physical trauma, and to information errors.

The success of animals, and the ability to exhibit types of behaviour that are difficult to replicate in mechanical systems without using biological models, suggests that biologically inspired models of behaviour can achieve levels of performance beyond those of traditional system in some fields. The synthesis of behaviour from a set of sub-systems and the close couple between behaviour the environment and the morphology of the animal seem to be especially important. This perceived importance has formed the basis of reactive and behavioural control methodologies.

Chapter 3 discussed the various schemes that fall under the umbrella term "behavioural control". Several defining features of behavioural controllers were identified: like the natural systems on which they are based, behavioural controllers rely on the interaction between elements to generate complex, emergent behaviour and behavioural controllers are traditionally built in a bottom-up manner. To build in this manner has implications for the robustness of the solution, failure for any reason, of behavioural components is liable to leave more general behaviours on layers below still operative, the ideal being a general degradation of performance.

The bottom-up process was also identified as simplifying the design of behavioural controllers, by allowing them to be implemented incrementally. Simple action patterns can be created, before responses to more specific stimuli are considered.

Some problems with behavioural control methodologies were also observed. An ideal of behavioural control is the removal of global symbols from the controller. This, however, has been shown to have implications for the use of behaviourally controlled machines for complex tasks. Without symbols it may be difficult to direct the machine. In addition, the reliance on emergent behaviour makes predicting the response of a controller of this type problematic. It was shown that a behavioural controller has the potential to be a complex dynamic system. This was identified as a further reason for adopting an incremental design process. Incremental design allows more simple sub-systems to be considered independently and reduces the number of parameters to be considered at any time.

A solution to this problem proposed by several authors is a hybrid and hierarchical structures to help alleviate these problems, using other control strategies to complement a behavioural sensor-motor interface. These architectures employ a symbol processing deliberative controller, communicating with the behavioural controller, the intention being that the response to the environment of the behavioural controller will be maintained, while also allowing task directed behaviour to be implemented through the deliberative controller.

An issue that must be addressed in a behavioural controller is the method of marshalling the various elements to produce system behaviour. Two basic methods were identified, selection and arbitration, these may also be classed as competition and co-operation respectively. The distinction being used to form the primary method of differentiation for a taxonomy of behavioural control. The two techniques have their own advantages and disadvantages, there have been some attempts to exploit the advantages of both by adopting hybrid systems.

Chapter 5 suggested an implementation for behavioural controllers based on fuzzy logic. The motivation for development of this controller architecture was the observations that being biologically inspired behavioural control shares foundations with fuzzy logic.

Further attractions in using fuzzy logic as a behaviour implementation medium were determined as the ability of fuzzy logic to model any mapping, and the use of linguistic variables. The use of which makes the design of reactive controllers more intuitive, allowing the behaviour of the system to be defined in easily stated terms, which can then be implemented directly as fuzzy sets and rules.

Unfortunately in it native state, fuzzy logic cannot be used to implement behavioural control. Behavioural controllers need some method to store state. A fuzzy inference engine responds only to the state of its sensors at the present time. State could be introduced as a loop around the controller, but there are other issues that must be addressed before fuzzy logic may be used for the implementation of a behavioural controller for any but the most trivial of tasks.

The sub-behaviours of an agent are liable to have various priorities associated with them. Avoiding obstacles, or stopping on contact with a human occupant of the environment can be seen to be more important than maintaining a certain speed and direction of travel.

Employing the architecture suggested can solve both the state and prioritisation problems. The prioritisation of behaviour is achieved by assigning a factor to each rule in the inference engine. This factor is used like an additional fixed firing weight to adjust the contribution the rule makes to the output of the controller.

In addition, a decomposition of the controller into logical behavioural elements was suggested. Each component produced in this way operates as separate fuzzy controller, the outputs of which are then combined by a further "weighted beam" averaging stage to produce system

output. The decomposition provides a clear improvement in the readability of the controller, and allows priority values to be assigned to the groups of rules that implement an action pattern or ethon, simplifying the design process.

The controller architecture was then extended by providing dynamically variable priority values for behavioural components. Addition of dynamic prioritisation allows competition and co-operation between behavioural acts. This in turn allows the controller to exhibit commitment to activity and state influenced behaviour.

The decomposition proposed also allows for the integration of non-fuzzy elements into the behavioural controller. Provided outputs match the interface of the weighted beam system, any controller may be integrated. The proposed architecture can therefore be used as the basis of a hybrid system where deliberative and behavioural controllers are integrated.

The navigational strategy outlined in chapter 6 was integrated into the fuzzy behavioural architecture in this way. The strategy is based on the use of a script representation of space, exploiting the structure of highly constrained spaces of the types found in spaces that have been manipulated by humans.

In chapter 4 navigation and the representation of space were investigated. It was determined that navigation comprises of two distinct tasks: local navigation and way-finding. Local navigation is concerned with moving within the environment that may be directly sensed by the agent. Way-finding is the act of moving from one location to another which lies beyond the area that may be immediately sensed by the agent.

Way-finding was shown to require knowledge of the goal, its relationship to the agents present location and the relationship between the motor actions of the agents and space.

To store spatial knowledge some representation is required. This representation must contain information on the topological relationships between features of the space.

Factors that affect the structure of the representation required were determined to include the capabilities of the agent and the structure of the environment. Within highly constrained, "Map-like" environments a route identified as a suitable representation. Routes were defined as a series of Stimulus-Response pairs, or Stimulus-Response-Stimulus triplets, which describe the desired path as behaviour to be taken at each location. Route following was shown to greatly compress the amount of information that must be stored and reduce the amount of processing to be performed when following a path, when compared to using a map. These gains, however, are not without cost. Use of routes was shown to reduce flexibility and robustness.

Chapter 6 describes a new navigation methodology based on script-like representations of space that exploit certain features of the environment to ease the navigation task.

It was shown that human manipulated environments tend to contain features that constrain the path of an agent, reducing the navigable space to a collection of paths that intersect or terminate on open, or mainly open, spaces. Such an environment was shown to lend itself to representation by routes or list-like structures.

A range of easily identified features of this type of environment were also identified, these were suggested as guides and orientations for the navigational behaviour of an agent. It was also shown that the structure of the environment allows these generic features to be used to uniquely identify locations, in context of the route, by the sequence in which they are reached. A further improvement in location recognition was identified in a partial solution to the object constancy problem. This solution is a result of the constraint on the path reducing the range of directions form which the agent may approach a location identifier (landmark).

It was shown that the script based navigation system based on these observations may be easily integrated into the fuzzy behavioural architecture suggested in chapter 5. The simplicity of the mechanism needed to use the scripts was also demonstrated.

If collected and stored the route scripts proposed may be manipulated simply to generated novel routes. Several operations for the extraction of new routes from collections of script were demonstrated. Extensions to the script format were also suggested that would ease the manipulation process.

The close couple between the structure of the environment and the appropriate navigational strategy is exaggerated in the script based technique. To be successful, the script must be applied to those environments that exhibit maze-like features.

Chapter 7 investigates the categorisation of spaces for this reason, briefly identifying those features of a space that determine its maze like character. The identification of space, in more general terms is further investigated in Chapter 8, where factors that may influence the ease with which a space may be navigated are considered, and reasons why spaces should be categorised suggested. Several methods by which a raster representation of space can be gauged were proposed with the intention of providing a quantitative measure for the classification of spaces.

9.2 Contributions

The work has developed three major themes, namely: the development of a fuzzy logic implementation for behavioural control, a script based navigation system, and a definition of maze spaces that may be used to identify those environments which are particularly suited to the script based navigational system. It is considered that the contributions made while investigating these themes are:

- An extension to fuzzy logic based reactive schemes to allow the explicit prioritisation of behaviour at the rule level using a fixed weighting factor similar to and used in conjunction with the firing weight of the rule.

- An innovative restructuring of fuzzy reactive controllers to allow the logical separation of behaviours in separate rule-bases, re-integrated in a second layer of arbitration. This provides an improvement in the readability of the controller, and allows the integration of non-fuzzy behaviour generation elements in a novel manner. In conjunction with the use of dynamic priority values, fracturing of the rule base allows the introduction of time variance, which is not possible with previous fuzzy reactive architectures.

- Development of a new design methodology for behavioural controllers based on proven software engineering techniques and the identification of conceptual similarities between objects and behaviours.

- An analysis of constrained environments which identifies those features that may be used to simplify the representation of space and ease navigation. The insights gained were then used to development of a route-based representation of space and for the description of tools to extract novel paths from collections of route based representations.

- Development of a novel navigational method based on a routes, allowing the simple communication of paths and exploitation of generic spatial features as location identifiers. Integration of the method with the fuzzy behavioural system.

- An investigation of spaces in the context of navigation, identifying those features that provide a measure of navigational difficulty. Development of measures that may be applied to raster representations to determine navigational difficulty and form the basis of a spatial categorisation system.

- Development of a new definition of maze-spaces to identify those spaces for which the script based navigation system is particularly suitable.

9.3 Suggested further work

The work presented here has investigated to a small extent the interaction of situated agents and their environments. The investigation has been made possible by considering the interaction of the agent with simulations of small sub-environments that limit the complexity of the behaviour produced. The effect of more complex environments can then be extrapolated from this. The information found is used to design behavioural controllers. Increased knowledge of the interaction of agents with environments is liable to improve the quality of these controllers; further investigation is required to achieve this, ideally generating testable models of interaction, rather than cataloguing the behaviour observed in a fixed set of systems.

The navigation system proposed in Chapter 6 provides robust navigation in a reduced set of environments. Further investigation may be able to extend the number of spaces to which such a system is applicable. Creation of a "locale" system that may be integrated with the navigation system allowing the generation of routes from complete topographical representations (maps) and creating maps from a collection of routes would enable automatic scheduling of agents navigating using the route based system and allow learning of the environment. If the system were implemented locally to the agent the robustness of the agent to navigational errors could be increased. The simple nature of the representation used also provides opportunities for the integration of natural language processors. But using natural language to communicate the path and representation to the navigation system, the utility of agents may be greatly increased.

The design methodology proposed for the generation of behavioural controllers also has potential for further investigation. The method outlined here was developed from existing

object oriented techniques to aid the generation of the controller, however the careful development of a framework for the development of behaviour controllers will allow for more capable designs.

An important area of work that has only been briefly discussed in this thesis is the categorisation of space as suggested in Chapter 8. Ideally further investigations are required to develop a method of accurately comparing the navigational complexity of spaces so that the performance of navigation techniques may be compared. To develop a suitable measure of navigational complexity neither a fully mathematical or fully empirical approach is expected to work. Empirical tests can not exhaust all possibilities, while a mathematical model may not account for all parameters of the physical system. The categorisation of spaces for navigational complexity opens a new area of investigation. Initially this investigation should be reserved for simplified, binary occupation spaces, as discussed in Chapter 8, with raster representations, which may easily be generated for physical spaces. Those factors that affect the navigational ease should be determined by testing a variety of spaces, against a variety of navigational techniques. The design and performance of these experiments will be a major investment, but the insights that may be gained are potentially very important. Ideally a set of indices that can be used to identify the navigational complexity of spaces should be developed.

Appendix 1: Fuzzy Sets and Fuzzy Logic

The following is a brief overview of the major properties and features of fuzzy sets and fuzzy logic and an investigation of the major techniques for the implementation of fuzzy systems.

A1.1 Human Reasoning and Classification

Normal human reasoning relies heavily on the use of broad classifications of objects, processes and conditions. These frames (Minsky 1988) are not rigorously defined, but provide generalised rules that may be applied to thoughts and objects that have the same basic features as that being classified. It is this generalisation ability that makes the frames so powerful. Generalisation is an important tool as it allows the use of standard definitions for a multitude of situations and imperfect knowledge to be augmented with default values. Without the ability to generalise, each problem encountered would require a unique solution.

Generalisation is also applied when classifying the value of an environmental variable. Humans have a tendency to apply context dependent descriptive tags to a variable, rather than a precise value. These descriptions are not only context sensitive but normally poorly defined. Meaning is not only affected by, but also learnt from usage. These imprecise classifications are used both in language and the reasoning processes. In language they are characterised as adjectives. When using such widespread generalisation it is inevitable that some elements of a class will be, in the words of the pigs' epigram "...more equal than others." (Orwell 1994). The class name has a higher level of suitability to some members of the class than to others. The level of applicability of a class name to an item also has implications for the appropriateness of applying the generic rules for that class in a specific case.

A definition such as "the large blue ball" illustrates the use of general value tags. The two adjectives employed exhibit the properties of generalisation, context sensitivity and degree of appropriateness. Even the term ball is a generalisation to a class of objects that are usually spherical toys.

Blue refers to reflected light from an object. It does not however represent one wavelength, but an area of the visible spectrum. At the higher frequency end of this portion of the spectrum, the light changes from blue to violet. As the frequency is lowered blue becomes green. There is no firm boundary between the individual colour states.

As the frequency is lowered, the colour of the light reflected from the ball becomes *less blue* and *greener.* The applicability of the classification "blue" decreases. This decrease is reflected in the use of the term "less".

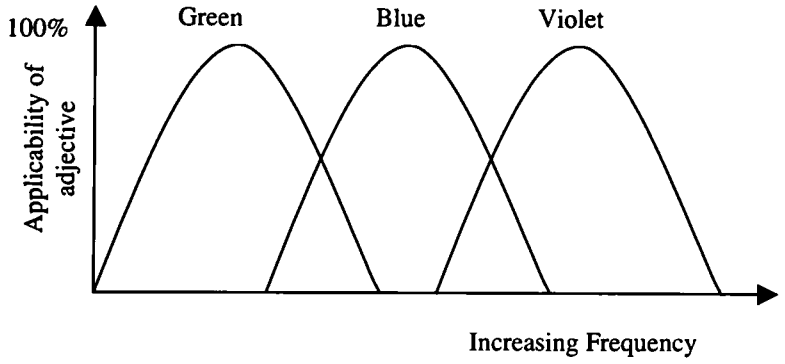


Figure A1.1 Applicability of Names to Colours with Respect to the Frequency of reflected light.

It is important to consider the nature of the tag classes. Large is a context dependent term that descibes the size of the sphere. A large ball would normally be significantly smaller than a small elephant. It is important to have a good prototype of the object being classified. The linguistic tags applied to a value are properties of the parent, generic class, not abstract external classes in themselves.

A1.2 Fuzzy Sets

Fuzzy sets (Zadeh 1965) are an attempt to create an mathematically analogous classification process, depending not on neural explanations of mental activity, but on language and perceived reasoning processes.

A fuzzy set classifies a fragment of the range of a variable and applies a linguistic tag. Unlike traditional (crisp) sets a value can have a non-Boolean membership to a fuzzy set.

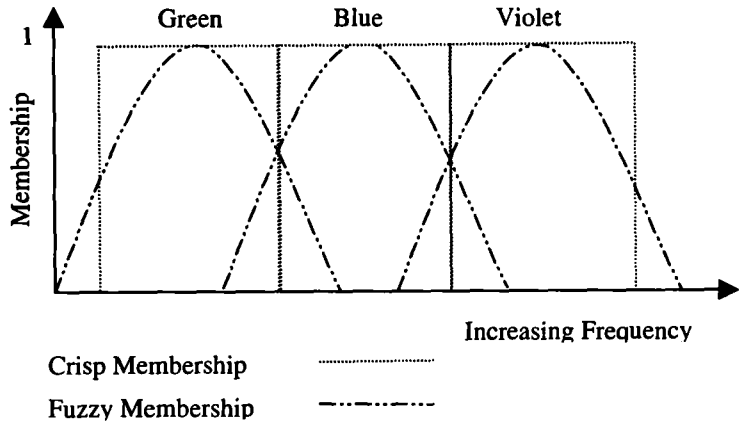


Figure A1.2. The Contrasting Effect of Fuzzy and Crisp set Membership.

The membership value of a point in the input space (universe of discourse) for a fuzzy set is given by a membership function:

$$membership = f(x)$$
$$\mu_{set} = f(x) \qquad \text{(A1.1}$$

Membership functions may be of any form, however there is a standard collection that is used most often. These are the S, Z, PI and Gaussian curves trapezoids, triangular and singleton functions (definitions for the functions are given in the following sections). Triangular and trapezoidal functions are often employed as piecewise approximations to the PI curve, as they are less computationally demanding. In the same way a piece-wise approximation may be made to the S and Z curves. A crisp set may be represented in a fuzzy system with a rectangular membership function.

A1.2.1 Gaussian membership functions

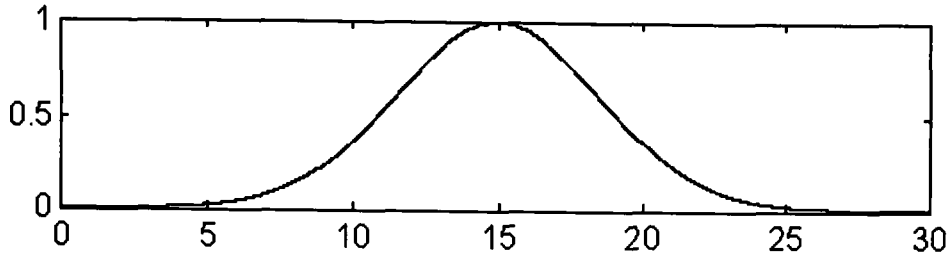


Figure A1.3. Gaussian curve G(x,15,5).

The Gaussian, or bell curve is a popular choice for fuzzy logic operation, though it does require a great deal of computation to calculate the membership value of the inputs. The Gaussian function follows the form:

$$G(x,\mu,\sigma) = e^{\frac{-(x-\mu)^2}{\sigma^2}} \qquad \text{(A1.2}$$

A1.2.2 S, Z and PI curves

The S curve is an open-ended function used to classify the extremes of the universe of discourse. There are no fixed S functions; they may have many definitions, however a popular form is the three-term function:

$$\begin{aligned} S(x,a,b,c) &= 0; x \le a \\ &= 2\left(\frac{x-a}{c-a}\right)^2; a \le x \le b \\ &= 1-2\left(\frac{x-c}{c-a}\right)^2; b \le x \le c \\ &= 1; x \ge c \end{aligned} \qquad \text{(A1.3}$$

The Z curve is simply the complement of an S curve. Placing an S curve and a Z curve back to back creates PI membership functions.

Implementation of fuzzy schemes reduces the usefulness of these continuous functions. Although ideal for mathematical analysis, and maintaining the principle that the membership function of a fuzzy set should have not discontinuities, computationally such functions are demanding. There are many branches and mathematical operations to perform in the calculation of membership of one of this family of curves. It is for this reason that the piecewise approximations have become popular, especially in engineering and embedded systems.

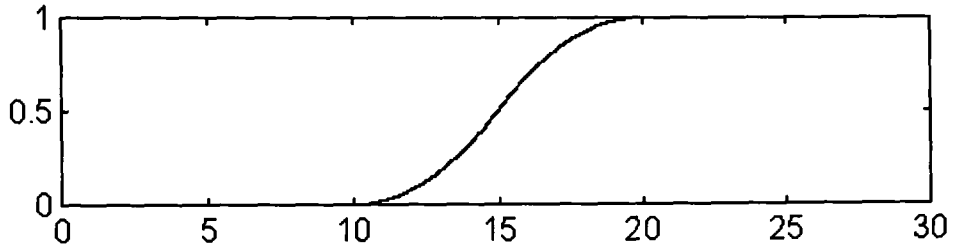


Figure A1.4. S Curve S(x,10,15,20).

A1.2.3 Piecewise approximations

Used because of their mathematical simplicity, the piecewise approximations are perhaps the most common form of membership function. Indeed they are the only functions supported by some software, such as that used to develop fuzzy systems on Siemens PLC's. There is some concern that there is too much information lost when trapezoidal membership functions are applied, causing an effect called fuzzy aliasing. This is mainly a problem however when fuzzy sets are used in signal processing, and appears to be of limited importance in most control applications, where the information loss can be considered an artefact associated with discretisation.

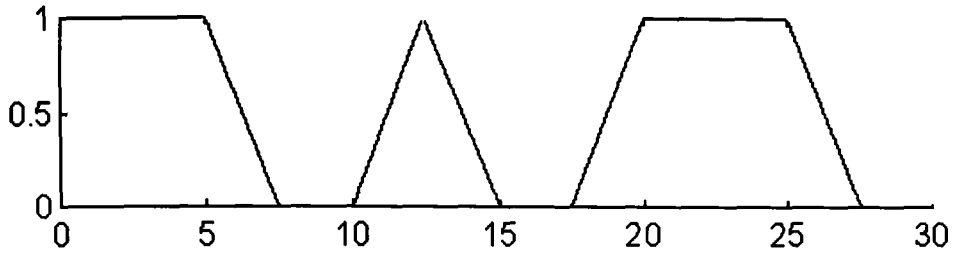


Figure A1.5. Piecewise Z, Triangular and Trapezoidal Membership Functions.

A1.3 Set operations

The entire collection of standard set operations may be performed on fuzzy sets, there may however be several methods to achieve each. All of the alternative operators will produce a different result. The most common of the operators are listed below:

A1.3.1 Complement:

The complement of a fuzzy set has the inverse membership to that set.

$$\mu_{set}(x)' = 1 - \mu_{set}(x)$$
$$\mu_{NOTset}(x) = 1 - \mu_{set}(x) \qquad \text{(A1.4}$$

A1.3.2 Union

The union of two fuzzy sets is equivalent to the maximum membership of the two sets.

$$\mu_{setA} \cup \mu_{setB} = \mu_{setA \cup setB}$$
$$\mu_{setA \cup setB} = MAX[\mu_{setA}, \mu_{setB}] \qquad \text{(A1.5}$$

A1.3.3 Intersection

The intersection of fuzzy sets is obtained by finding the minimum of the sets at all the points in the universe of discourse.

$$\mu_{setA} \cap \mu_{setB} = \mu_{setA \cap setB}$$
$$\mu_{setA \cap setB} = MIN[\mu_{setA}, \mu_{setB}] \quad \text{(A1.6}$$

A1.3.4 Product

The product of two fuzzy sets is the product of their membership functions.

$$A.B = \mu_{setA} \cdot \mu_{setB} \quad \text{(A1.7}$$

A1.4 Hedges

In addition to these standard mathematical operators there are also linguistic operators to be taken into account when models of human behaviour are being made. Commonly used hedges include very, much, slightly, more or less, plus and minus. They modify the membership function of a set in a deterministic manner. Though not required for many applications such as the production of fuzzy controllers, the use of hedges may be important when using information gathered from an expert. As the procedure definition provided by the expert is based on natural language it is important than any attempt to create a controller from this allows for the simple translation of the terms used into the formal definitions that are employed in fuzzy systems. The use of hedges also helps maintain the readability of the system.

The maintenance of readability is also the cause for the occasional use of changes of the tag names of the sets, as this may be required to preserve the grammar such that it may be understood as natural language. An example of this would be the use of the term *"inexact"* as a substitution for "*not exact*".

A1.4.1 Very

A simple illustration of a linguistic hedge is *very*. This has the effect of reducing the span of membership of a fuzzy set. *Very SetA* is a subset of *SetA*. This concentration of the meaning of a set may be achieved simply by finding the square of the membership. Thus membership of any

point in the universe of discourse to the *very set* is reduced, except where full membership is exhibited.

$$VeryA = \mu_{SetA}^2 \qquad \text{(A1.8}$$

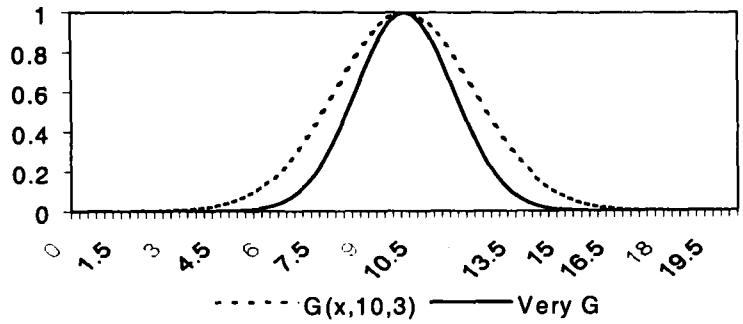


Figure A1.6. Hedge "Very "

A1.5 Precedence

The precedence rules for manipulation of fuzzy sets are given below.

Precedence	Operation
highest	Hedges, Not
	and
Lowest	or

Parentheses are used where required to alter the order of computation, the contents being evaluated first.

A1.6 Fuzzy Logic

The use of fuzzy sets for the design of systems has the advantage of intuitiveness. As the sets correspond to the types of classification made by people in everyday language and reasoning, it is relatively simple to transfer knowledge about a problem to the solution using fuzzy logic.

For reasoning to take place logical operations must be possible. As the sets themselves have membership values, it is important that the results obtained also have some measure of membership or truth. The result of a logical operation on fuzzy sets should produce not just a truth table, but a truth continuum, the solutions truth value being dependent on the membership values associated with the operand fuzzy sets. There are several common methods to achieve most of the standard logical operators with fuzzy sets. The most commonly used are often referred to as the Zadeh operators. Here an AND function is achieved using the simple intersection (A1.9). The OR function is formed using the union operator:

$$A\ AND\ B = \mu_A(x) \cap \mu_B(y)$$
$$A\ AND\ B = MIN[\mu_A(x), \mu_B(y)] \qquad \text{(A1.9}$$

$$A\ OR\ B = \mu_A \cup \mu_b$$
$$A\ OR\ B = MIN[\mu_A(x), \mu_B(y)] \qquad \text{(A1.10}$$

Shown (Figure A1.7) is the result of applying the MIN and MAX operators to various operand values.

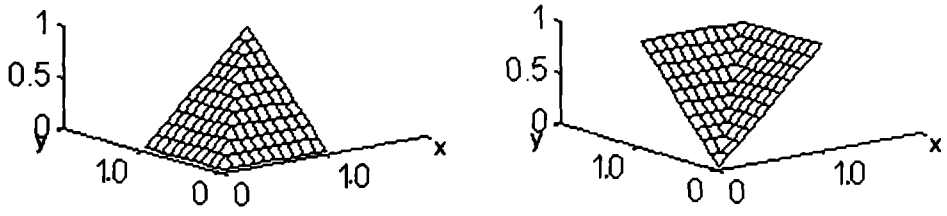


Figure A1.7. MAX(x, y) and MIN(x,y) Intersection and Union ooperators

A1.7 Compensatory Operators

The use of MIN and MAX as operators is perhaps the most extreme interpretation of AND and OR respectively. Much information is lost, as the result is dependent on merely one of the operands. There are other operators that can be employed for which this is not the case. These are termed compensatory operators. They are generally the bounded intersection, bounded union, product, probabilistic-sum and Yager functions.

A1.7.1 Bounded Operators:

The bounded intersection assigns low degrees of truth if the input memberships are low:

$$Bounded\ And(x, y) = MAX[0, x + y - 1] \quad \text{(A1.11}$$

If there are more that two input variables the bounded intersection must be chained:

$$Bounded\ And(x, y, z) = MAX[0, Bounded\ And(x, y) + z - 1] \quad \text{(A1.12}$$

The equivalent union operator is the bounded union, having a high level of sensitivity to low truth-values but exhibiting low sensitivity to high truth-value operands.

$$Bounded\ Or(x, y) = MIN[1, x + y] \quad \text{(A1.13}$$

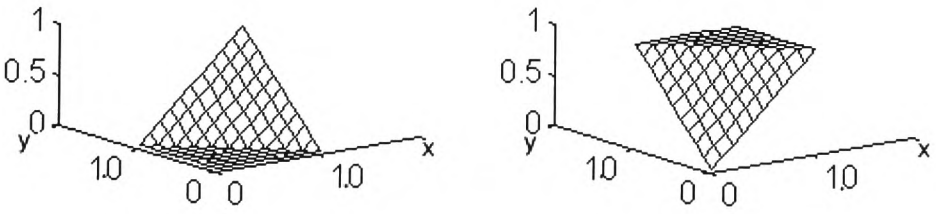


Figure A1.8. Bounded Operators. Bounded AND(x, y) and Bounded OR(x, y).

The product intersection operator returns a truth-value that is the product of the membership values of the operand fuzzy sets:

$$Product\ And(x, y) = [x.y] \tag{A1.14}$$

The DeMorgan conjugate of the product intersection is the probabilistic sum union operator:

$$Probabilistic\ Sum(x, y) = [x + y - x.y] \tag{A1.15}$$

If there are more than two operands, the probabilistic Sum must be chained:

$$Probabilistic\ Sum(x, y, z) = [prob.sum(x, y) + z - prob.sum(x, y).z] \tag{A1.16}$$

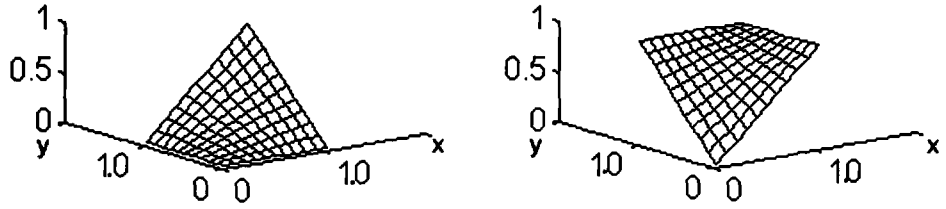


Figure A1.9. Intersection PROD(x, y) and Union PROB(x, y).

The Yager functions are more mathematically complex than those listed above. A control parameter, P influences the action of the function.

$$\begin{aligned} Yager\ And(x, y) &= 1 - MIN[1, [(1-x)^P + (1-y)^P]^{1/P}], P > 0 \\ Yager\ Or(x, y) &= MIN[1, [(x)^P + (y)^P]^{1/P}], P > 0 \end{aligned} \tag{A1.17}$$

A1.8 Application of Fuzzy Logical Operators

After classification of a variable into fuzzy sets a reasoning method is required. This is usually achieved using an IF THEN rule base. The rules take the form:

$$If\ (VariableX = SetA)\ AND\ (VariableY = SetB)\ THEN\ Output = SetC \tag{A1.18}$$

The most commonly used operator in the antecedent is the AND operator, as shown. This is not, however the only form that rules can take:

$$If\ (VariableX = SetA')\ OR\ (VariableY = SetB)\ THEN\ Output = SetC \qquad (A1.19$$

Each rule takes on a value (the firing weight) that is inferred from the values of the antecedents of the rules. It is important that as much of the information that is contained in the antecedent as possible is transferred to the firing weight of the rule. There is however a trade off between full transfer of information and computational requirements.

A1.9 Inference Methods

As with all aspects of fuzzy logic there are many techniques that may be used to infer the firing weight of a rule or degree of truth of an output set. The general rule is that the truth-value of the output sets should have a value that reflects the truth-value of the rule antecedent. A rule antecedent that has a zero truth-value should translate to a zero truth output set for that rule. This is also true for 100% truths. An antecedent that has 100% truth should be used to infer a 100% truth in the output set.

A1.9.1 Minimum

The correlation minimum technique simply truncates the output set to a level corresponding to the firing weight of the rule:

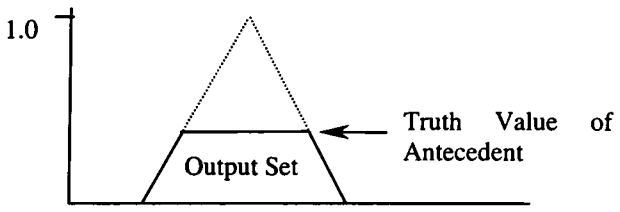


Figure A1.10 Correlation Minimum Inference

A1.9.2 Product

The output set is scaled to the truth-value of the rule:

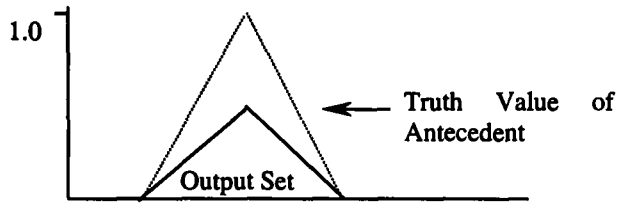


Figure A1.11 Correlation Product Inference

A1.9.3 MIN-MAX

This is the simplest of the inference methods, and includes the aggregation method of the system. In MIN-MAX inference the output fuzzy set is truncated, as in correlation minimum inference, the difference being, that for all rules that yield non zero truth levels for a particular set, only the maximum is passed forward for defuzzification.

A1.9.4 Sugeno Inferencing

In Sugeno inferencing, the firing weights of the rules are not applied to output sets, but rather as the coefficients of a linear function. For zero order Sugeno systems this is equivalent to applying product correlation to Singleton output sets:

$$If\ x = a\ AND\ y = b\ THEN\ output = z \qquad \text{(A1.20}$$

The more generalised form of a Sugeno inferencing system may be seen in the first order system:

$$If\ x = a\ AND\ y = b\ THEN\ output = p.x + q.y + r;\quad p,q,r \text{ constant} \qquad \text{(A1.21}$$

A1.10 Aggregation

When the fuzzy system is in operation there is unlikely to be only one rule giving a non-zero output at any time. Some method is required that will aggregate the outputs of the individual rule, so the defuzzification may be applied to extract a single crisp output value for each controlled variable. As shown above, this is most simply achieved in MIN-MAX inference, where the rule that has the largest contribution to the value of that set is selected for the final

aggregation process, this however doesn't account for the values of other sets, a further stage is required. This is achieved by summing the values yielded by the rules that had the highest truth-values for all the fuzzy output sets corresponding to one controlled variable. There is a large degree of information loss in such an aggregation system, but the speed at which the system may run is higher, as there is far less computation to be carried out.

A1.10.1 Additive Aggregation

An alternative to the simple MAX aggregation method shown above is that of additive aggregation. All the non-zero output sets corresponding to a controlled variable are simply added. This may yield values for some areas of the universe of discourse that are greater than one. This is acceptable, as the aggregated result is not a truth-value, but a mathematical attempt to combine the truth-values of several propositions, so that a decision may be made.

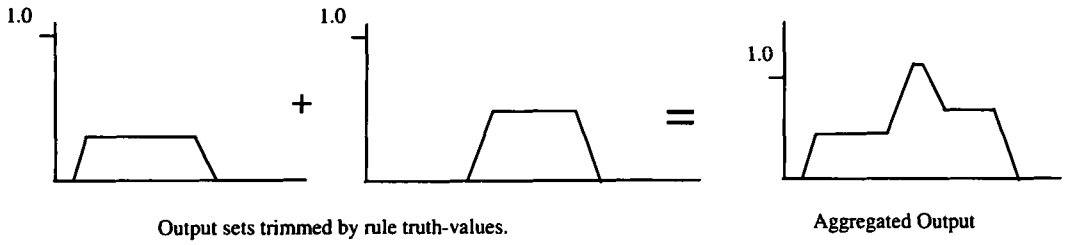


Figure A1.12 Additive aggregation

A1.10.2 Defuzzification

The final stage operation in the use of a fuzzy inferencing system is defuzzification. This takes the various aggregated truth-values for the output sets and calculates a crisp value for each of the controlled variables. The inevitable effect of defuzzification is a loss of some of the information present in the aggregated output sets, this is however unavoidable, and after all, the system is intended to produce single, good responses to the system inputs, not slavishly conserve information.

A1.10.3 Maximised

Maximised defuzzification takes perhaps the simplest stance for the calculation of output. The maximum value of the aggregated output sets is used as the crisp output value. If there is a plateau in the output space then the maximised output may be either the left most point of the plateau, the right most point, or their average.

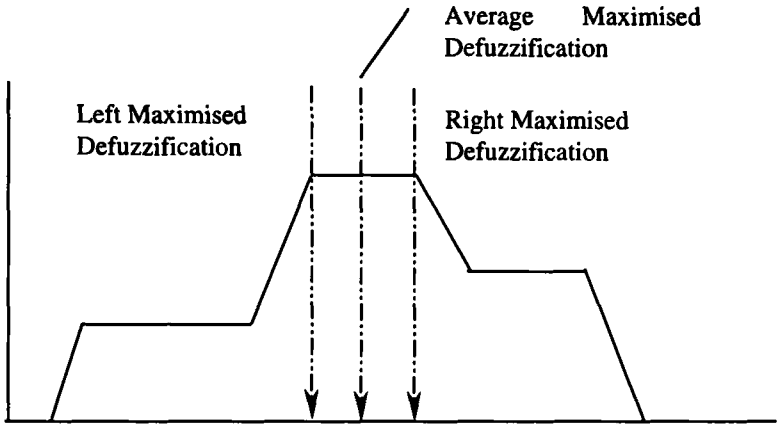


Figure A1.13 Maximised Defuzzification.

Probably the first use of a fuzzy logic controller was by Mamdani and Assilian in 1975 (Mamdani and Assilian 1975). The system was successfully applied to a steam engine with non-linear characteristics. This controller used an averaged maximised defuzzification procedure.

A1.10.4 Centroid

The centroid method is a very common technique for defuzzification. As the inputs change, there is a smooth change in output value, which is especially desirable for applications such as control systems. The centroid defuzzification process is based on centre of mass calculations. The output of the centroid defuzzification system is the average of the weighted input space, the limits of the integration are the range of the output variable x:

$$Output = \frac{\int x\mu(x)dx}{\int \mu(x)} \tag{A1.23}$$

An approximation to the centroid defuzzification technique may be made calculating the sum of a discrete sum of points (N):

$$output = \frac{\sum_{i=1}^{N} x_i \mu(x_i)}{\sum_{i=1}^{N} \mu(x_i)} \tag{A1.24}$$

A1.10.5 Sugeno

The use of zero order Sugeno inference system has many advantages for the calculation of outputs.

The singleton output sets are mathematically equivalent to symmetrical shaped output sets when subjected to centroid defuzzification, as the output value remains the same under both product and minimum correlation inference. The amount of computation required to calculate the crisp output is also reduced by using Sugeno inferencing, as the number of points required to calculate the centroid is reduced to one for each output set (after aggregation). Sugeno systems are often referred to as having weighted beam defuzzification as the calculations are like those required to calculate the pivot point of a balanced beam on which weights have been placed. The weights are the firing weights of the rules. Their positions are the crisp output sets.

Along with simpler defuzzification calculations Sugeno inferencing also provides better coverage of the output space. The singleton output sets may take any value throughout the range of the output variables. This is not so for shaped output sets where under centroid defuzzification the use of centre of mass calculations means that it is impossible for the system to take values at the extremes of the output range.

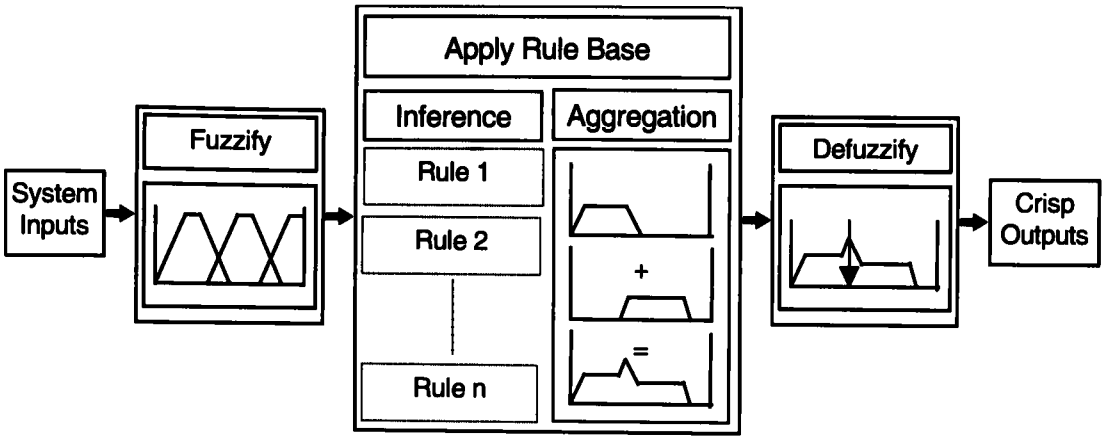


Figure A1.14. Generalised operation of a Fuzzy Logic System

A1.10.6 Fuzzy Algorithms

Early attempts to use fuzzy sets were not restricted to the model of fuzzy logic outlined above, a rule base of fuzzy conditional statements where the output is dependent on the input alone. Also investigated was the concept of the fuzzy algorithm (Zadeh 1968). A fuzzy algorithm is a ordered list of instructions like any other algorithm but with the exception that the instructions may operate on fuzzy sets, and produce a fuzzy solution. Fuzzy algorithms allow for the coding of natural instructions in a mathematically precise manner. In this way they may be used to model many processes that are difficult to reproduce mechanically in any other manner.

As with standard algorithms, the statements used to define the fuzzy algorithm fall into three main types, Assignment, conditional and unconditional.

At conditional statements, where execution of the algorithm may branch there may be a dilemma as to which of the alternatives to take. This can be resolved either by the defuzzification of the tested variable, or the parallel realisation of the branches. These alternatives may both have roles to play, depending on the nature of the problem to which the algorithm is applied.

Fuzzy logic systems such as those that are outlined above may be thought of a single "case" constructs from a fuzzy algorithm. They are not separate calculation paradigms, but a subset of fuzzy algorithm architecture.

A commonly occurring example of a fuzzy algorithm would be a recipe. A machine reading a cookbook would be confronted with a barrage of fuzzy variables and many conditional statements that depend on these values. The recipe for meringue, though very simple illustrates this point. There are only two ingredients and few instructions. Below is the recipe for a small quantity.

```
Ingredients:
     Whites of two eggs
     Two tablespoons of granulated sugar.
Procedure:
     Beat egg whites until stiff
     Fold in sugar
     Place in moderate oven until golden brown.
```

This can be represented in a flow chart:

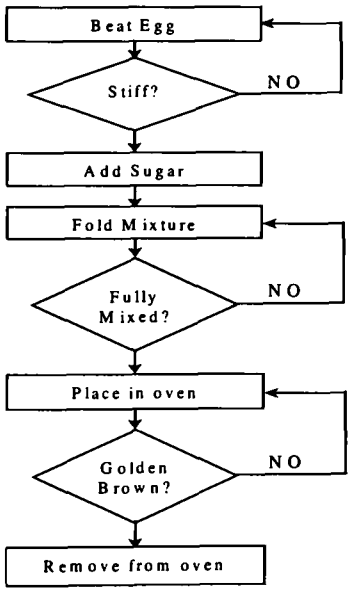


Figure A1.15 "Meringue" Fuzzy Algorithm

Though the chart above shows a very simple algorithm with no parallel branches, it does illustrate flow through a programme being controlled by conditional statements which have regard to fuzzy variables. It is also important to note that such a procedure would be hard to

implement in a standard fuzzy logic system where there is no state, and sequences are not possible.

A1.11 Summary

Fuzzy sets provide a simple mechanism for the coding of human knowledge in machines. Algorithms or logics that employ these sets may be used for the emulation of some thought processes. Fuzzy Logic has become popular in areas such as control engineering where mathematical definitions of a system may be highly complex, yet there is an intuitive solution that may easily be applied.

Fuzzy logic has been successfully used in large numbers of real applications, not least in high volume consumer electronic devices such as cameras and microwave ovens. It has also found applications in larger items such as automatic gearboxes for cars and lifts.

A major advantage is the ability to increase the MIQ [Machine Intelligence Quotient] of devices cheaply and relatively simply. This simplicity is the direct result of the use of easily understood language in the development of the systems.

Most commonly the system is organised as a rule base, with many simple parallel statements. The procedure for use of such a system is:

- Fuzzify input data.
- Apply truth-values to rule base in parallel.
- Inference
- Defuzzification to produce single, crisp output.

Though not always the best solution, and ease with which controllers may be understood is the greatest assets of fuzzy logic, allowing complex concepts to be encoded mathematically easily using the same type of coding that is used in human thought.

A1.12 References

Mamdani E.H. and Assilian S. 1975. An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *International Journal of Man-Machine Studies* 7: pp 1-13.

Minsky M. 1988. *The Society of Mind.* New York: Touchstone (Simon and Schuster).

Orwell, G. 1994. *Animal Farm.* Penguin.

Zadeh L.A. 1965. Fuzzy Sets. *Information and Control*, No. 8: pp 338-53.

Zadeh L.A. 1968. Fuzzy Algorithms. *Information and Control*, No. 12: pp 94-102.

Appendix 2: Asimov's Three Rules of Robotics

The following explanation of the three laws of robotics is reproduced in quotation from the "Isaac Asimov FAQs"
<http://www.clark.net/pub/edseiler/WWW/asimov_FAQ.html#series13>

The Three Laws of Robotics are:

1. A robot may not injure a human being, or, through inaction, allow a human being to come to harm.

2. A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.

3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

From Handbook of Robotics, 56th Edition, 2058 A.D., as quoted in "I, Robot."

In Robots and Empire (ch. 63), the "Zeroth Law" is extrapolated, and the other Three Laws modified accordingly:

0. A robot may not injure humanity or, through inaction, allow humanity to come to harm.

Unlike the Three Laws, however, the Zeroth Law is not a fundamental part of positronic robotic engineering, is not part of all positronic robots, and, in fact, requires a very sophisticated robot to even accept it.

Asimov claimed that the Three Laws were originated by John W. Campbell in a conversation they had on December 23, 1940. Campbell in turn maintained that he picked them out of Asimov's stories and discussions, and that his role was merely to state them explicitly.

The Three Laws did not appear in Asimov's first two robot stories, "Robbie" and "Reason", but the First Law was stated in Asimov's third robot story "Liar!" which also featured the first appearance of robopsychologist Susan Calvin. (When "Robbie" and "Reason" were included in I, Robot, they were updated to mention the existence of the first law and first two laws, respectively.) Yet there was a hint of the three laws in "Robbie", in which Robbie's owner states that "He can't help being faithful, loving, and kind. He's a machine - made so." The first story to explicitly state the Three Laws was "Runaround", which appeared in the March 1942 issue of Astounding Science Fiction.

Appendix 3: EXAMPLE DIRECTION SETS

The following appendix provides examples of directions given to help travel to a range of destinations. The directions have not been edited since receipt.

An analysis is provided in section A3.4, which identifies some properties of directions. as communicated spatial representations, direction sets have formed the motivation for the script based navigation system outline in chapter 6.

A3.1 Example 1

The first example describes the route to be taken from the University of Wales College, Newport to a home in St David's Crescent, Newport. This is not a vast distance but does involve negotiation of a large number of road intersections.

From the college, go out the main entrance, turn left up the hill, first road left (as you start encountering the prefabricated type places).
Straight to the T-junction (Ridgeway pub on the left) and then left. You'll immediately encounter another T-junction. Turn right and continue to roundabout. Left out of the roundabout (first exit) into the road that runs past the Handpost. Keep going to the Handpost traffic lights, straight through them and the next road right. It's a funny road that turns left as it splits into a one-way and then curves to the right past Mansion House, the Mayoral Residence. First road left (Caeperllan road) down the hill. Follow for a couple of hundred yards. It will bend to the right at the bottom of the hill and come to a T-junction. Left again down the hill. You should now be approaching a set of traffic lights at

a T-junction with Homebase directly in front of you. Right at the traffic lights into Cardiff road and keep going under the bridge past some shops towards Tredegar park.
After passing the shops and then a filling station, first road right, immediately left after that and then right again in to St David's Crescent.
Straight on to no 80 on the left, a semi.

A3.2 Example 2

Example 2 provides directions, again from the College, but this time to a home in Caerleon, approximately 3 miles distant. Even with the extended distance of travel the route is simpler, this is reflected in the directions.

Leave the M4 at Junction 26, take the town centre exit and at the next roundabout take the 6th junction signposted "Caerleon". At the next roundabout take the second exit and take the sliproad on the left to the next roundabout. Take the first exit and follow Caerleon Road for approx. 2 miles until you reach Caerleon. You enter the town over the Bridge of the river Usk. Follow along the main road, which takes you around a one-way system past the school on the left, Caerleon Common on the left until you reach a zebra crossing and mini roundabout at The Angel Hotel on the right. Take the third exit (signposted Newport) and then take the second exit on the left signposted Cambria Close. No 51 is on the right hand side two thirds of the way into the Close.

A3.3 Example 3

This example is a set of "second-hand" passed on by a third party. The route described takes the from the A35 as it approaches Bridport from Dorchester, however the directions were required to make the journey from Newport to Bridport. A route had to thus be generated to reach Bridport from Newport, and then the direction manipulated to interface with this. Finally the compounded route could be followed to achieve the goal.

Coming from Dorchester:
At 1st Island on A35, turn right towards Krewkerne
200yds take left turn, Just before traffic lights and Co-Op
Left into St Andrews Road
Follow bends to left
Pass Police Station
Grey Building on right, turn Right into Bedford Place
Continue till set of gates at fork
Go through Gates. Park

A3.3.1 Example 3a

Sufficient information is provided in this set of directions, when combined with the simplified map obtained from the map book to find the location. As there are a limited number of routes from Krewkerne to Bridport this enabled a simple transform to be enacted. As approaching Bridport from Krewkerne the road was not left until the Co-Op was seen. A turn could then be enacted to the right, immediately after the Co-Op. The performance of this turn is an illustration of the type of spatial operation outlined in chapter 6 (section 6.10.1.3). However whereas in chapter 6 it is stated that such operations require special information, the higher degree of abstraction and knowledge available to a human navigator makes such manipulations of the directions sets symbol set mundane. The Co-Op shop is a uniquely identifiable orientation marker, that is easy to communicate due to the vast array of common symbols the population of Britain share.

If the shop had not been seen then the route taken into Bridport would be assumed not to have been the correct path between Krewkerne and Bridport. In this case the road signs would be used to find the A35 to Dorchester. This would then in turn be used as a starting point for navigation using the un-transformed nodal map. Potential problems are with identification of the "Island" as little information enabling the location to be uniquely identified is available in the directions. The travel sequence has been used, if travel does not follow the expected route this is

void. The contingency was to drive towards Dorchester on the A35 after leaving Bridport, after the town is considered fully left, the route may be reversed, looking for the first island.

A3.4 Analysis

The first and most striking feature of these directions is that they are not given as lists of stimulus-response pairs.

Each set of directions is formed from short segments which list several trigger features with their respective response actions. Additional information is also added, such as the references to the road names and signposts, that provide "little checks" to enable the navigator to appraise their performance. If the additional identifier is not seen, the navigator may retrace their steps and attempt to find the location where error was made. These small pieces of information also have the advantage of reassuring the navigator.

The form of the directions (as short bursts of stimuli action pairs, separated, usually by some piece of additional information) is presumably intended to break the route into chunks that contain redundant information. The additional information provided usually easy to identify features of the environment, such as bridges, roundabouts and pubs. Small descriptions are also used to help support identification of individual locations. Nowhere within the descriptions is there any reference to stimulus points that are ignored. There are no references to "going past the left turn" etc. This is presumably to prevent confusion. By listing only those stimulus points (most of which have implied actions) and large identifying features as guides the script may be effectively followed, with reduced confusion. Some "Culling" of information is also required as long list of directions suffer from being hard to recall. It is interesting to note that the three sets of directions, when ordered by length exhibit a reverse relationship to the distances the routes they describe cover. Though not necessary for the use of the scripts by the navigating agent, which is provided with a text file, so is unlikely to be come confused, this culling of information

has been used as the motivation for the removal of goal stimuli, ignored potential stimuli and the adoption of the sequence counters. These changes to the script structure are an attempt to make the structure more simple to use by human authors.

A3.5 Identification of orientations and guides

Part of example 1 is repeated below, with some of the orientations and guides highlighted:

From the college, go out the main entrance(***ORIENTATION***), turn left up the hill(***ORIENTATION***), first road left (as you start encountering the prefabricated type places)(***ADDITIONAL IDENTIFICATION INFORMATION***).

Appendix 4: PAPERS

Reproduced here are papers:

Behavioural Control Implementation Using Fuzzy Logic,

Behavioural Control Using Modified Sugeno Inferencing

and

Development Of A Fuzzy Behavioural Controller For An Autonomous Vehicle

Presented at

ICSE '97, the 12th International Conference on Systems Engineering, Coventry 1997.

AMC '98, the 5th International Workshop on Advanced Motion Control. Coimbra, Portugal 1998.

and

UKACC International Conference; Control'98, Swansea 1998.

respectively

BEHAVIOURAL CONTROL IMPLEMENTATION USING FUZZY LOGIC.

C. A. J. Tubb, G. N. Roberts and H. Rowlands.

Mechatronics Research Centre,
University of Wales College, Newport,
P.O.Box 180, Newport.
South Wales.
NP9 5XR
Tel: (01633) 432442 Fax: (01633) 432430 E-mail: c.a.tubb@newport.ac.uk

Keywords: Behavioural control, Fuzzy logic, Autonomous vehicles.

Abstract

This paper describes the development of both a small test vehicle, and a behavioural control system, implemented in fuzzy logic, for low level control of the vehicle in an industrial equivalent environment.

1. Introduction

In an industrial setting Vehicles that can automatically perform simple fetch and carry operations are potentially very useful. Increasing the level of autonomy of these vehicles, enabling them to tackle more complex tasks and operate in less constrained environments improves their usefulness, especially the removal of human supervision. Autonomous vehicles, have already been used successfully in many industrial, commercial and service environments

The track following autonomous guided vehicle(AGV) is well known in many industrial applications. Though little more than a train, such systems are more than adequate for moving material along fixed routes in static environments. Unfortunately vehicles of this type requires that the workspace is adapted, by the addition of tracks and beacons, in order that the vehicle can locate itself. Problems can occur when such a system is to be integrated into an existing building space, if the environment is variable, or the vehicle's task is liable to change.

Alternatives to simple track following may involve the use of a "virtual track". The vehicle is provided with a series of points it is to visit. Beacons, or markers are used such that the vehicle may locate itself with respect to the next goal point. If the vehicle is given the ability to recognise features from its environment, these may be used, instead of markers for the localisation of the vehicle[1,2].

If the working environment of the vehicle is dynamic, following tracks, or fixed paths becomes problematic, the path may easily become blocked or obscured, in which case an appropriate contingency plan is required, in its simplest form, the vehicle may merely stop, and wait for the obstruction to move. This is satisfactory for highly static environments, such as a warehouse, where obstacles on the path and other vehicles or staff are likely to be rare. However, if there are likely to be semi-permanent obstructions to the path, or a quantity of mobile obstructions, the time taken to traverse the track may become extremely long. To avoid such delays a vehicle should have the ability to react to changes in the working environment.

2. Behavioural Control

In many systems, including autonomous vehicles, a lack of information about the operating environment and the complex nature of the goal preclude the use of traditional control methods. Symbolic AI approaches have also been hard to implement. In these traditional AI solutions to the problem of vehicle control, a cycle of operations is adopted, where environmental measurements are taken The information gained is used in conjunction with an internal world model to develop a plan for the subsequent actions of the vehicle. Finally the vehicle makes its moves. This has been referred to as the *Sense, model, plan act* cycle[3]. The quality of the environmental data, and the world model employed by the vehicle can adversely affect the success of this type of system. In situations like this behavioural control can simplify the design of the controller.

In behavioural control systems, the task to be performed is normally decomposed into various sub-tasks, or behaviours, which are more simple to implement. An

example behaviour for an autonomous vehicle may be to turn left when an object is detected to the front right of the vehicle. From the interaction of these behaviours emerges the operation of the system. Primitive behaviours may be used as components for the assembly of complex ones, for instance, a wall following behaviour could be assembled from two more primitive, taxis behaviour. If the vehicle turns away from objects detected at a range closer than some threshold, while objects further than the threshold are made attractive, the vehicle will remain at the threshold distance from objects. Unnecessary vacillation can be reduced by expanding the difference in the two threshold values.

A major concern in behavioural control systems is determining the system output at any given time. There are several techniques used to do this. Behaviours can be made concurrent, with the system output a function of the output of all behaviours[4], or the output of the system may be the output of a single behaviour that was selected through some mechanism[5]. In the subsumption architecture the operation of a behaviour can inhibit the operation of those bellow it[6,7]. Most behaviour selection techniques imply that there is some prioritisation, or utility value to each of the behaviours, that depends on the present state of the system, or the state of the environment, or both.

Behaviour selection method has been shown to affect system performance[8], supporting the intuitive concept of a smooth transition between behaviours being conducive to smooth changes in output.

3. Test Vehicle

A Vehicle has been designed as a test-bed for the fuzzy behavioural control strategy. This takes the form of a series of three clear, acrylic shelves, approximately 300 x 210mm, supported on a tricycle arrangement of two driven wheels, powered by DC Gearmotors and a castor for balance. Steering is by differential drive of the two powered wheels. The shelves form the substrate for the electrical, computational and sensor systems. The motors and power supply are mounted on the lowest of the shelves, between the wheels.

The vehicle sensorium comprises four ultrasonic (40kHz) range finders mounted at the front of the vehicle to give a field of view of approximately 120°. Each of the range finders is fired in turn, to prevent unwanted coupling of the range signals.

The DC gearmotors are each controlled by signed four bit words. The most significant bit of each word determines the direction of the motor, the other three are used to give eight speed levels via a simple pulse width modulation circuit. This level of control was chosen as it appeared to give a significant level of speed control, with each output having a definite, discernible value.

Computation is carried out on standard P.C. hardware

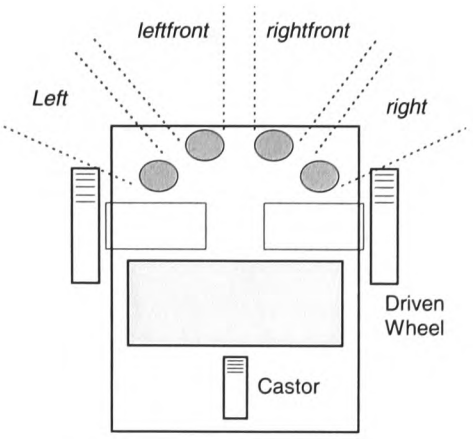


Figure 1 Vehicle Layout

4. Fuzzy Logic

Fuzzy logic, is an attractive tool for the construction of behavioural control system. Each of the rules may be thought of as a primitive behaviour. More complex behaviours can usually be defined as a series of IF, THEN rules of the type found in a fuzzy inference engine, or can be constructed using other behaviours as components. The use of linguistic variables helps to overcome the difficulties associated with the poor definitions of many of the tasks that are required of the vehicle, while the output of the fuzzy system is a combination of the output demanded by each of the rules, resembling some behaviour arbitration techniques[4]. The output of a fuzzy rule can change smoothly with changes of input, allowing for a smooth transition between output states. Fuzzy logic has been used to control vehicles successfully in the past[9].

The vehicle control system discussed is implemented as a zero order Sugeno system. In the Sugeno technique, the output of each of the rules is given as a function of the input variables. A zero order Sugeno system has rules which output a single constant value.

if x is A and y is B then $Output = C$ (1)

The output of the system is calculated as the weighted average of these, equivalent to finding the balance point of a weighted beam. For n rules with firing weights W:

$$Output = \frac{\sum_{i=0}^{n} WiCi}{\sum_{i=0}^{n} Wi} \quad (2)$$

The firing weight of each rule is a function of the membership values of each of the variables, calculated in the same way as for the Mamdani method. The firing weight for the example given above would be:

$$W = \min(A, B) \quad (3)$$

Although the outputs of the two techinques are very similar[10]. The Sugeno technique was chosen over the

more common Mamdani defuzzification method for computational simplicity.

As the number of behaviours and their complexity increases, problems can arise as all the rules have an equal contribution to the system output, allowing no prioritisation of the behaviours. This may be overcome by adding an extra weighting term to each of the rules. This term is adjusted to the perceived utility of the behaviour of which the rule is part. Using the weighted average scheme of the Sugeno system, this can be achieved simply by use of an additional weighting term for each rule, *Pi*.

$$Output = \frac{\sum_{i=0}^{n} WiCiPi}{\sum_{i=0}^{n} Wi} \qquad (4)$$

5. Behaviour Implementation

Initially the vehicle has a simple group of behaviours intended to provide simple reactive control of the vehicle in a simple, rather structured environment (laboratory and corridor).

The list of primitive behaviours is as follows:

1. Drive forward;
2. prevent collision;
3. turn left;
4. turn right;
5. avoid object left;
6. avoid object right;
7. maintain heading;

The turn and avoid behaviours are differentiated by scale. The return to course behaviour is a special action, intended to account for gross errors of heading, introduced as the vehicle is forced to turn by the other behaviours. It relies on an external (to the fuzzy system) calculation of the heading angle, based on the drive to each motor. It is intended that an additional behaviour, that accepts direction from another layer of the controller will be implemented in the future, as part of a hierarchical control strategy.

5.1 Input fuzzification

The inputs to the fuzzy system are the ranges detected by the four ultrasonic sensors and a heading error value. The ranges are fitted to three membership functions, *far*, *close* and *near*. All membership functions are trapezoidal, though the validity of using this type of membership function for controllers has been brought into question[11], the widespread use of trapezoidal membership functions in controllers, and the simplicity with which they can be implemented was considered to be sufficient justification for their use in the preliminary stages of controller design.

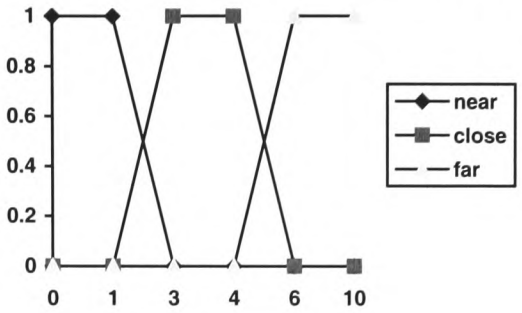


Figure 2 Range input membership

The heading input is also fitted to three membership functions, *okay*, *offACW* and *offCW*. Initial experiments are conducted with a minimum of membership functions, so that the interactions between rules can be more easily observed.

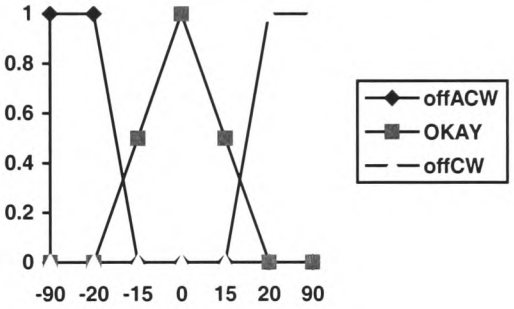


Figure 3 Heading input membership

5.2 Output sets

The fuzzy system has two output variables, *leftdrive* and *rightdrive*, coupled to the vehicle motor circuits. As the ouput of each rule these variables can take one of three crisp values, *fast*, *slow* and *stop*, representing 100%, 50% and 0% drive respectively.

5.3 Fuzzy rules

Each of the behaviours are implemented as a collection of rules in a single rule base. As such the architecture of the fuzzy inference engine does not reflect the architecture of the behavioural system. Each of the behaviours is an artificial subdivision of the rule base. Figure 4 gives an overview of architecture.

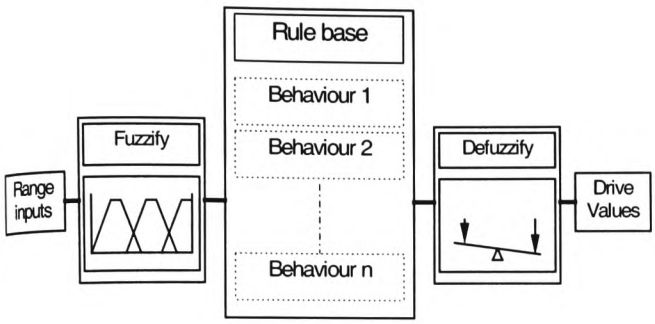


Figure 4 System Architecture

5.4 Example behaviours

5.4.1 Drive forward

The Drive forward behaviour is probably the most simple, taking into account only that the vehicle may move forward if there is no obstruction, and that while doing so an obstruction to the side is of less importance then one directly infront of the vehicle. The rules for the drive forward behaviour:

1. if (*left* is *far*) & (*right* is *far*) & (*leftfront* is *far*) & *rightfront* is *far*) then (*leftdrive* is *fast*)(*rightdrive* is *fast*) (utility1)
2. if (*left* is *close*) & (*right* is *close*) & (*leftfront* is *far*) & *rightfront* is *far*) then (*lefdtrive* is *fast*)(*rightdrive* is *fast*) (utility1)

The utility of these two rules is high as this behaviour is intended to keep the vehicle moving. The interaction of this behaviour with those involved in other vehicle actions is limited by its dependence on long ranges of free space to the front of the vehicle.

5.4.2 Maintain heading

The maintain heading behaviour is intended to lessen the divergence from the vehicles course when an object is encountered. At present this is a very simple behaviour, and several issues must be taken into consideration. Following obstacle avoidance the vehicle will regain the correct heading, if further obstacles don't prevent this, but there will be a displacement from the vehicles original path. Changes may be made to the external heading calculation routine to account for this, also this behaviour should be suppressed when course changes are desired. The rules for the maintain heading behaviour are:

1. if (*heading* is *offACW*) then (*leftrive* is *slow*)(*rightdrive* is *stop*) (utility 0.5)
2. if (*heading* is *offCW*) then (*lefdtrive* is *stop*)(*rightdrive* is *slow*) (utility 0.5)

The maintain heading behaviour has been given a low utility value, this is to ensure that the operation of this behaviour does not interfere with the correct operation of the more fundamental behaviours such as obstacle avoidance.

5.4.3 Turning behaviours

There are two types of turning behaviour, *turn left/right* and *avoid object left/right*. The avoid object behaviours ideally have only a minor influence on the action of the vehicle, as the turn behaviour should have previously reoriented the vehicle, should an obstacle appear in front of the vehicle. The turn behaviour, slows the vehicle on the opposing side to a detected object:

1. if (*right* is close) & (*rightfront* is *close*) then (*leftdrive* is *stop*)(*rightdrive* is *slow*) (utility1)
2. if (*right* is *close*) & (*rightfront* is *far*) then (*lefdtrive* is *stop*)(*rightdrive* is *slow*) (utility1)

The avoid behaviours are similar, but there action is more extreme. For this reason, the rules test the sensors on both sides of the vehicle. If the vehicle entered a passage like area, the turn behaviours, though trying to turn the vehicle in both directions at once, would only result in the vehicle slowing down. The avoid behaviour will try to rotate the vehicle, or if it is to close to the surrounding objects, make little contribution to the actions of the vehicle.

1. if (*right* is near) & (*rightfront* is *near*) & (*left* is *far*) & (*leftfront* is *far*) then (*leftdrive* is *stop*)(*rightdrive* is *slow*) (utility1)
2. if (*right* is *near*) & (*rightfront* is *close*) & (*left* is *far*) & (*leftfront* is *far*) then (*leftdrive* is *stop*)(*rightdrive* is *slow*)(utility1)
3. if (*right* is *near*) & (*rightfront* is *close*) & (*left* is *far*) & (*leftfront* is *close*) then (*leftdrive* is *stop*)(*rightdrive* is *slow*)(utility1)
4. if (*right* is *near*) & (*rightfront* is *near*) & (*left* is *close*) & (*leftfront* is *close*) then (*leftdrive* is *stop*)(*rightdrive* is *fast*) (utility 1)

5.4.4 Prevent collision

If all the sensors detect an object near the vehicle, the drive to both wheels is reduced to zero. This behaviour is primarily intended to as a safety feature. The input values that fall into the near membership function are too close to the vehicle for it to turn without a risk of collision.

6. Discussion

Several points of interest are apparent in the use of the fuzzy behavioural controller. The most important consideration being how the behaviours are designed. The use of fuzzy logic helps implement abstract ideas as behaviours, but

doesn't help form them. The behaviours implemented above, are firmly based in preconceived ideas of how the vehicle should react to sensor information, the validity of these behaviours can not be proved without experiment.

The parameters of the input membership functions are tuned manually after the operation of the system has been observed. Ideally these values would be calculated prior to implementation of the system. The selection of utility values for each of the behaviours is another area where a mathematical approach may be an advantage, especially as more behaviours are implemented on the vehicle. The interaction between the maintain heading behaviour and the behaviours that make up the vehicles obstacle avoidance capability is especially hard to judge.

Though the system can be tested in the fuzzy logic development environment, to ensure that the behaviours interact in the way intended by the designer, there is no information on how the controller will behave when implemented on a real vehicle, in a real environment. The use of computer simulations and models has been rejected by some authors as not sufficient proof of vehicle action[12,13].

7. Conclusion

The use of fuzzy logic, to develop behavioural controllers, allows for smooth transitions between controller output states, and in turn to smooth vehicle action. Fuzzy logic also lends itself to control of an autonomous vehicle as the desired behaviour of the vehicle may be more easily described, linguistically, as a series of fuzzy variables and actions, and the defuzzification process does away with the need for an explicit behaviour arbitration mechanism.

With the addition of an extra weighting, term, to each of the rules in the fuzzy system enables the different utilities of the behaviours to be set. The ability to set utility values for each of the behaviours enables more complex system behaviour to emerge from the interaction of the simple, fuzzy rules.

References

[1] Couroux, Wilfrid, Michel Dufaut, and Rene Husson. "Localisation Methodology by Image Processing for Mobile Robot Navigation in Outdoor Environment.,"*1994 IEEE International Conference on Systems Man and Cybernetics* , vol 3,1994.

[2] Baptiste, P, E Bideaux, C C B Day, and D J Harwood. "Use of Perception Based Navigation for Autonomously Guided Vehicles.," *1994 Joint Hungarian-British Mechatronics Conference* pp 279-84,1994.

[3] Brooks R.A. "Intelligence Without Reason." *Intelligence Without Reason*. A.I. Memo No. 1293. MIT, 1991. - also available in proc. IJACAI-91.

[4] Aylett, R. S, A. M Coddington, R. A Ghanea-Hercock, and D. P Barnes. "Heterogeneous Agents for Multi-Robot Cooperation.," *Design and Developement of Autonomous Agents* , vol. 95/211,1995.

[5] Maes P. "A Bottom-Up Mechanism for Behaviour Selection in an Artificial Creature." *From Animals to Animats*.Editors J.A.Meyer, and Wilson S.W, pp238-46. MIT Press, 1991.

[6] Brooks, R. A. "A Robust Layered Control System For A Mobile Robot.," *IEEE Journal of Robotics and Automation* , vol. RA-2, no. 1 pp 14-23,1986.

[7] Connell J.H. "Creature Building With the Subsumption Architecture."*IJCAI 87*, pp1124-26 1987.

[8] Bisset D.L., and Webber A.D. "Building Reactive Vehicles: An Engineers Perspective." *Proc AISB'94 Workshop*, 1994.

[9] Xu W.L., and Tso S.K. "Real-Time Self-Reaction of a Mobile Robot in Unstructured Environment Using Fuzzy Reasoning.," *Engineering Applications of Artificial Intelligence* , vol. 9, no. 5 pp 475-85,1996.

[10] Jang, J. S., and Gulley Ned. *Fuzzy Logic Toolbox User's Guide*. first ed.Math Works Inc., 1995.

[11] de Oliveira J. "Sampling, Fuzzy Discretization, and Signal Reconstruction.," *Fuzzy Sets and Systems*, no. 79 pp pp151-61,1996.

[12] Smithers, T. "On the Difference Between Plant-Controller Systems and Agent-Environment Interaction Systems.," *Design and Developement of Autonomous Agents*, vol. 95/211,1995.

[13] Smithers Tim. "On Why Better Robots Make It Harder." - *Proc 3rd International Conference on Simulation of Adaptive Behaviour*, editors Cliff D., Husbands P., Meyer J.A., and Wilson S. From Animals To Animats, MIT Press, 1994.

Behavioural Control Using Modified Sugeno Inferencing

C. A. J. Tubb and G. N. Roberts

Mechatronics Research Centre,
University of Wales College, Newport,
P.O.Box 180, Newport.
South Wales.
NP9 5XR
Tel: (01633) 432442 Fax: (01633) 432430 E-mail: c.a.tubb@newport.ac.uk

Abstract

This paper describes the design and construction of a behavioural controller for an autonomous vehicle using fuzzy logic.

Discussed are the desirable properties of a behavioural controller, and how they may be implemented using fuzzy logic. Issues regarding the suitability of fuzzy logic are considered and used to indicate the motivation for modification of the fuzzy system.

The nature of the fuzzy system used, and the modifications that have been made to the standard zero order Sugeno inference engine are illustrated.

1. Behavioural and Reactive Control Strategies

Behavioural and reactive control strategies are closely related but have fundamental differences. In effect behavioural control can be thought of as an extension to the simpler reactive technique. Both are formed from a series of parallel sub-units, each of which is a decomposite of the problem space. The overall behaviour pattern of the vehicle is a synthesis of the effects of all the active behavioural actions.

Each behavioural action (equivalent to an ethon in animal behaviour studies) can be thought of as a separate processing unit, connecting a subset of the system inputs or sensors to the outputs and actuators. Each of these processes corresponds to a single competence of the many that the system may require to operate effectively.

The decomposition of a complex task into a collection of parallel processes contrasts greatly with the more traditional method of breaking a problem into a series of processing sub tasks to be tackled in sequence. It primarily represents a different point of view, where tasks are simple but concurrent as opposed to being large and complex, requiring total control of the vehicle, and thus total knowledge of the environment. In traditional control architectures the flow of information is a single stream passing from each sub-task to the next. All information is generally passed through all processing modules. A partial failure causes failure of the whole system.

In behavioural or reactive strategies knowledge about the operating environment is stored implicitly in the rules and ethons that make up the controller, without need to call upon an explicit world model.

An important capacity of behavioural controllers is the ability to store both state and representations of the environment [5, 6]. This provides a contrast to the simpler reactive controllers that are effectively a series of mappings between sensors and actuators. State storage allows a behavioural controller to implement sequences of behavioural actions, and thus increase control repertoire.

The use of behavioural control allows the implementation of complex command strategies without the use of exhaustive descriptions of the

vehicle's operating environment. It is especially suited to the lower levels of control in autonomous vehicles. Breaking the control algorithm into concurrent processes enables the controller to respond more quickly to the environment, and increases the robustness of the system to partial failures and sensor noise.

An additional practical advantage of using a behavioural control strategy is the ability to implement the controller incrementally. More primitive executive competencies may be designed and tested, before the more complex behaviours are added.

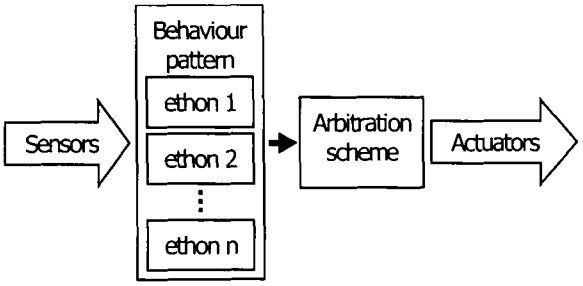


Figure 1. General Behavioural Control Architecture

2. Behavioural Control Issues

Certain issues exist that must be addressed in the design of a behavioural controller.

If several behaviours are active at any time arbitration is required. The two basic forms that this may take are compromise and selection (non-compromise). A selection technique has the advantage that there is always a single control output active, which has total control over the relevant actuator, with a system of compromise, if the two or more outputs are in direct competition an impasse state may be reached. Unfortunately a selection technique will however result in the loss of a great deal of information available to the controller, and though the output action will have a high confidence level it may be far from optimal.

Another potential cause of problems is vacillation between states. This can occur when two or more behaviours are at, or near their respective firing conditions. An example of this sort of problem would be an animal that is both thirsty and hungry, stood between a source of food and a source of water. If the animal drinks, the level of thirst falls, and the animal will go to food. If the desire levels are equal and the animal can conceivably die of thirst or hunger between the two sources, as the smallest sip or bite may cause a change in activation sufficient to allow the competing behaviour to become dominant. An example of the vacillation problem more likely to occur in a mobile robot would be when the vehicle is travelling between two obstacles. The proximity of the unit to one of the obstacles may cause a change in heading towards the other producing a zigzagging course. Though this may not be considered particularly problematic and thus is acceptable, ideally there should be few rapid changes in control action.

There are several approaches possible to counter these problems depends. In low level behaviours such as obstacle avoidance the solution may be multilevel or continuous output functions and compromise between conflicting behaviours. The use of compromise reduces the violence of the switch between output actions, and ideally will result in the vehicle adopting a stable middle position. For higher, goal seeking behaviours this solution may not be possible, there being little advantage to achieving a fraction of many goals. In such cases the solution may be either exploitation of additional behaviours, or derived from the ability of behavioural controllers to store state. In the example above this may be characterised by the addition of behaviours that cause the animal to eat or drink when at a source of food or water respectively, or a mechanism to maintain either the eating or drinking behaviours once they become active.

3. Fuzzy Logic in Reactive Control

Fuzzy logic has a history of use in reactive controllers [3, 4]. There are helpful architectural similarities between the two techniques.

Each of the fuzzy rules can be considered as a primitive element of a reactive controller, these being assembled into the systems overall behaviour. The antecedents of each rule define the trigger states for that element. Fuzzification performs classification of sensor information into a series of states and can be thought of as a perceptual system not dissimilar to the perceptual schemas of motor schema theory [1]. Although information is lost the simplification of the input space is great, and thus helpful for selection of appropriate action. The pre-processing which results from fuzzification simplifies the design of the reactive rules, while the use of linguistically valued variables makes them more open and frees the rules from containing overly detailed information on their required trigger states.

The intuitiveness of the fuzzy system design process also contributes to the usefulness of this technique. The use of linguistically tagged variables and simple rules enable complex relationships to be defined easily.

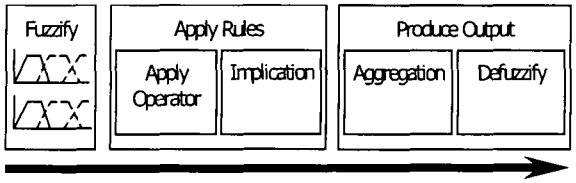


Figure 2. Operation of a General Fuzzy System

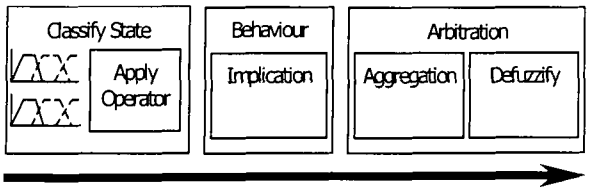


Figure 3. Fuzzy system as a reactive controller

As stated above, an important consideration of any behavioural or reactive control scheme is the arbitration between those behavioural actions that share actuators. The defuzzification process produces a single definite control action from contesting behaviour outputs. The nature of this output depends on the defuzzification technique used, but a good compromise, that changes smoothly with variation in input values is possible. The arbitration problem may be exasperated by the use of fuzzy logic, as there are potentially many rules exhibiting non-zero truth-values. A well-designed fuzzy controller should have few cases where this happens, but it may become inevitable in complex systems, as all possible input conditions are unlikely to have been taken into account.

4. Fuzzy Logic Behavioural Control Considerations

For proper operation of a behavioural controller several considerations must be taken into account.

There is a natural prioritisation inherent in most actions engaged upon by both machines and animals certain behaviours requiring immediate gratification while others, though they may be important to the goal achieving capabilities of the device or animal are not as urgent. An example from the animal kingdom may be avoiding predation having a higher immediate priority than eating. In the controller of an autonomous vehicle obstacle avoidance has a higher priority than route following. Behaviours that exhibit conflicting output actions may cause an impasse if their activation levels (fuzzy truth-values) are equal. Careful use of prioritisation can help to reduce this problem.

Storage of state is another important consideration. As mentioned previously a behavioural controller must be able to store some form of state information.

The standard Sugeno inferencing fuzzy system has some limitations when used to implement either of these requirements.

Each of the rules has the same potential effect on the output of the controller, meaning that there is no prioritisation between behaviours. The representation and recording of environmental state is not possible, as previous inputs are not considered.

5. Modified Sugeno Inferencing

The suggested modifications to the fuzzy engine are intended to address the problems outlined above. Firstly a utility factor is appended to each of the rules. This can be used to weight the effect that the rule has on the control signal output, and can thus significance of the behaviour that it represents. In addition to this lateral inhibition between rules is being investigated as a method of implementing a limited memory function facilitating the production behavioural sequences and reducing the vacillation problem. The inhibition of rules also has implications for the "grumbling rules" problem encountered in fuzzy filtering. This problem occurs where inappropriate rules have firing weights that are sufficient to cause degradation in performance of the filter.

6. Prioritisation

In order to achieve prioritisation of the various fuzzy rules that make up the controller the addition of a simple weighting factor to each rule may be used.

Each of the output sets in a zero order Sugeno inference system is a single constant value C_i. The output of the system is the sum of these weighted by their respective truth-values W_i. Calculation of the output is similar to taking turning moments.

$$Output = \frac{\sum C_i w_i}{\sum w_i} \qquad (1)$$

To achieve prioritisation of the rules an additional constant weighting factor may be added to each rule proportioning the firing weight of each rule, and thus changing the contribution the rule makes to the defuzzified output.

$$Output = \frac{\sum C_i w_i P_i}{\sum w_i P_i} \qquad (2)$$

This simple scheme allows rules to be assigned priorities while the rule base is under construction that remain fixed throughout the system's operation.

7. Lateral Inhibition

To achieve lateral inhibition between rules poses a greater problem, with more issues to be considered. Primarily how much inhibition should be allowed between rules?

Inhibition may be achieved by adding a further weighting factor to the output of each rule; the magnitude of this factor is under the influence of the inhibiting rules. A technique with which this may be achieved is to use the complement of the inhibiting rule's firing weight and scale it by an inhibition factor.

8. Implementation

Implementation of a behavioural system in the modified Sugeno inferencing engine is carried out in a hierarchical manner. Individual rules are a too highly detailed representation of the behaviour of the vehicle, direct use of them in the design process is thus awkward. To simplify the procedure the behaviour of the vehicle is broken down from a master behavioural scheme, through behaviour sub-patterns to individual rules (ethons). The master behaviour may be augmented easily by the addition of new behavioural sub-patterns.

An initial population of behaviour sub-patterns is chosen. Each behavioural sub-pattern is then independently defined as a series of fuzzy rules. The definition process may be any appropriate fuzzy design technique. The method used here is to define a FIS matrix [2] of input sets and link each cell in the matrix to a desired output state. Only those inputs that are thought necessary are used in the definition of a sub-behaviour. Using this technique a non-

exhaustive rule base may be produced that never the less defines the behaviour of the system fully and effectively. Inhibition is then applied between each of the behavioural sub-processes.

Originally a flat rule base was used, listing all the component rules of all the sub-behaviours equally. Inhibition is then a matter of choosing which rule in a behavioural action will produce the signal that is used to inhibit other behaviours. This is not an ideal situation. The choice of inhibiting rule can make a large difference to the level of the inhibition effect transferred to the other behaviours.

An alternative is required which allows sub-behaviours to be considered as atomic entities for the purpose of inhibition.

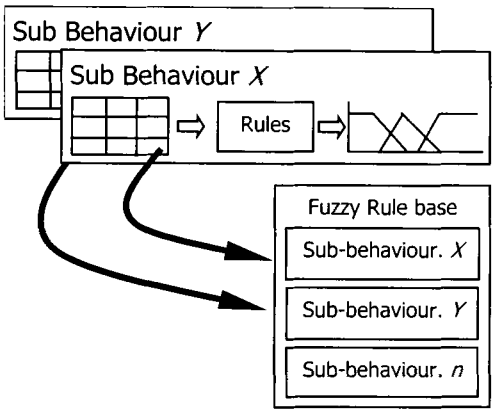


Figure 4. Creation of Rule base

Under consideration is an alternative architecture that provides a simple remedy for this problem. Here each of the sub-behaviours is treated as an individual fuzzy rule base, the truth-value outputs for each of the sub-behaviours being calculated using the standard zero order Sugeno technique.

The coherent output of the system is then calculated by applying the weighted-average operator, to the truth-values of sub behaviours. In this way the truth-values of the behavioural actions are treated identically to those of the rules that make up the action. In this manner the controller adopts a functionally hierarchical structure, though each of the behaviours can still be considered parallel.

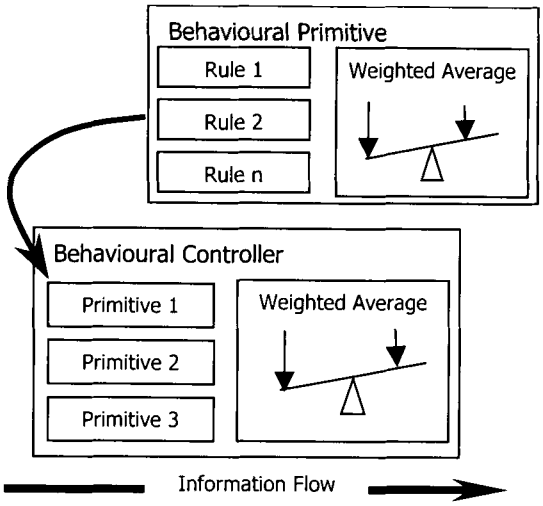


Figure 5. The Hierarchical Architecture of the Modified Inferencing Engine

9. Discussion

The use of the modified Sugeno inference engine produces a working behavioural controller. There are however some issues that must be addressed.

The prioritisation of behaviours is dependent on the chosen utility factor. This has been selected manually. For very small rule bases this is possible, however when the interaction between rules increases with higher levels of system complexity manual tuning becomes more problematic. In experiments used to prove this concept three values only were used, chosen by the expectancies of priority during design. These values were 0.5, 0.75 and 1.0. This was suitable for the small rule bases used, however if there is a requirement for greater subtlety of prioritisation manual tuning of utility factors becomes far more complex and time consuming. Further investigation in this area should include the use of a machine learning technique. Unfortunately this brings the requirement for performance measures for the system, and knowledge of the relationship between the performance of the vehicle and utility factors. Measurement of the performance of

autonomous vehicles and mobile robots is notoriously difficult [7, 8]

A similar problem exists in the selection of the inhibition factor. Initial experiments for proof of concept employed a scaling factor of 1. The complement of the inhibiting behaviour's previous firing weight being used unadulterated in the calculation of the firing weight of a inhibited behaviour.

With the addition of the utility and inhibition factors the number variables that may require tuning is increased greatly. Not only do these scaling factors have room for adjustment, but also do the membership functions of the system. Automation of the tuning process appears to be necessary.

An unexpected, though rather obvious advantage to using the tiered defuzzification technique required by lateral inhibition is the improvement in the ease of design and readability of the behavioural controller by separation into functional blocks.

10. Conclusion

A perceived similarity between the structures of fuzzy inference engines and reactive controllers has been exploited to create a methodology for the design of behavioural controllers.

The modifications required for full implementation of a behavioural controller using a fuzzy inference engine cause an increase in the number of coefficients that require tuning. Future work must include automatic tuning of these variables to reduce this task.

Both the increase in the readability of the controller, and associated ease of behaviour design are significant. Although the form that the system behaviour ultimately takes is due to the interaction of the individual behavioural actions and so is hard to predict, the ease with which the controller may be read helps in problem solving.

11. References

[1] Arkin R.C. 1989. Motor Schema- Based Mobile Robot Navigation. *International Journal of Robotics and Automation* 81, no. 4: pp 92-112.

[2] Kosko, B. 1992. *Neural Networks and Fuzzy Systems a Dynamical Systems Approach to Machine Intelligence.* 1 ed. Prentice-Hall International Editions.

[3] Li Wei. 1994. Fuzzy Logic Based 'Perception Action' Behaviour Control of an Mobile Robot in Uncertain Environments. *IEEE Int Conf on Fuzzy Systems*, 1626-31.

[4] Liu, K. and Lewis, F.L. 1994. Fuzzy Logic-Based Navigation Controller for an Autonomous Mobile Robot. *1994 IEEE International Conference on Systems Man and Cybernetics* Two.

[5] Mataric, M.J. 1992. Behaviour Based Control: Main Properties and Implications. *Proc. IEEE International Conference on Robotics and Automation. Workshop on Architectures for Intelligent Control Systems.* pp 46-54.

[6] Mataric, M.J. 1995. Integration of Representation Into Goal-Driven Behaviour-Based Robots. *The Artificial Life Route To Artificial Intelligence: Building Embodied Situated Agents.* Eds L. Steels and Brooks, R., pp 165-86. Lawrence Earlbaum Associates.

[7] Smithers T. 1995. On Qualitative Performance Measures of Robot Behaviour. *Robotics and Autonomous Systems.* vol. 15, no. Part 1-2: pp 107-33.

[8] Smithers T. 1994. On Why Better Robots Make It Harder. "From Animals To Animats" *Proc. 3rd International Conference on Simulation of Adaptive Behaviour*, Eds Cliff D. Husbands P. Meyer J.A., and Wilson S. MIT Press.

DEVELOPMENT OF A FUZZY BEHAVIOURAL CONTROLLER FOR AN AUTONOMOUS VEHICLE

C. A. J. Tubb and G. N. Roberts

University of Wales College, Newport, South Wales UK.

ABSTRACT

This paper considers the development of a fuzzy behavioural controller for an autonomous vehicle. The motivation for the work and the issues which need to be addressed are described. An overview of deliberative and reactive controllers is presented leading to the specification of a behavioural controller utilising fuzzy motor schemas and a fuzzy behavioural architecture. The paper concludes with a description of the definition and selection of inhibition schemes

INTRODUCTION

With applications ranging from planetary rovers to mail delivery, vehicles that can operate without human supervision are becoming essential. The intention of the work described in this paper is the development of a controller for a small vehicle that will allow autonomous navigation of an indoor environment. This controller is intended to give a high-level of automaticity for low-level operations, whilst the vehicle is engaged in useful human controlled tasks. These tasks are described loosely by an operator, in the form of directions, which define the path the vehicle is to travel. The directions are listed as location action pairs in a command script text file. The locations are described in generic terms, not identified uniquely using specific landmarks &c.

This paper discusses motivation for the choice of behavioural control and the use of fuzzy logic to implement the controller. Also illustrated are some modifications to standard zero-order Sugeno inferencing that are under investigation, intended to improve the performance of the fuzzy behavioural controller.

The test vehicle for the project is a simple platform, supported by two driven wheels and a castor. Steering is achieved by differential drive to the main wheels. Sensing is carried out using ultra sonic range finders.

CONTROL ARCHITECTURES

The use of behavioural controllers in mobile robots, though often contrasted to "traditional" techniques, has been in widespread use for many years. It is considered that both behavioural and reactive control techniques can be thought of as being based on the motor schema theory

of activity in biological systems. (Arkin (1) accepts a definition of "A pattern of action as well as a pattern for action" for motor schema). Advantages of such techniques over deliberative controllers are usually in the form of an increase in the reliability of the system and a decrease in the computational complexity. They represent a break with the symbolic processing associated with deliberative AI controllers.

DELIBERATIVE CONTROL

Deliberative control uses traditional AI techniques. Sensor information is encoded as symbols, which are used to calculate appropriate action through the utilisation of an internal model of the environment. System goals are defined explicitly. The environment is considered fixed, and highly separated from the controller.

The controller forms a servo system, where the error is the difference between present and goal states. Processing is carried out as a series of procedures, with information flowing in a linear manner through each in turn. The symbols used are independent of the control architecture. Their meaning is totally self-encapsulated, being defined during system design.

Deliberative systems have several weaknesses. For example, failure to maintain an accurate model of the environment, and a lack of robustness to sensor noise, can contribute to poor performance.

REACTIVE CONTROL

In a reactive controller, a series of task specific mappings are made from the machine's sensors to its motor circuits. Interaction of these simple mapping functions produces the overall action of the system. This emergent behaviour is not explicitly stated in the design of the controller. As the processing is carried out in parallel, a high level of robustness can be achieved, where failure of individual elements leads to a gradual degradation in performance. This is most important as a defence against sensor noise and uncertainty. With no global symbols present in the controller a failure to assign an accurate value is not reflected throughout the system.

As the behaviour of a reactive controller is emergent, design for specific tasks is difficult, the more complex the task, the less tractable the design problem becomes. To design a reactive controller capable of a variety of tasks would be very difficult.

BEHAVIOURAL CONTROL

The term behavioural control is often confused in the literature with simpler reactive schemes. There are however fundamental differences in the two techniques, as described in Mataric (2) .

Behavioural control can be regarded as the middle ground between reactive and deliberative approaches. Behaviours provide more than a mere mapping of sensor states to motor actions. Internal representations of the environment of different forms may be employed, and additional processing may be performed on these representations in order to support the decision making process. In contrast to deliberative systems the memory and decision-making processes are described implicitly, and distributed across behaviours rather than employing centralised models and knowledge bases, Mataric (3) .

Having the same distributed nature as reactive controllers, behavioural schemes show a level of robustness above that of a deliberative controller, whilst the ability to perform more involved computation allows more complex operations to be attempted than would be possible within a purely reactive system.

An important implementation issue for both behavioural and reactive controllers is arbitration between primitives. This problem is especially acute when several behaviour primitives control a single actuator resource, some mechanism is required that will produce a single control output from the conflicting signals produced by individual behaviours.
Arbitration schemes fall into two main categories; compromise and selection (non-compromise). Some systems such as the subsumption architecture employ prioritised inhibition, Brooks (4), and thus achieve a totally decentralised system. The method of prioritisation used in the subsumption and colony, Connell (5), architectures requires that priority is known at design. A behavioural primitive will be given the ability to inhibit or suppress the action of another. This web of inhibition grows with the incremental definition of the controller. Inhibition between behaviour primitives is total.

Other systems rely on techniques such as vector addition to calculate a compromise output, as described in Aylett *et al* (6). This approach produces a more flexible, continuous output space, which improves the repertoire of the controller. However, there is an associated increase in the difficulty of predicting how behaviours will interact.

FUZZY MOTOR SCHEMA

Fuzzy logic has been used in the past for the construction of reactive controllers for mobile robotic applications, Li Wei (7). The strengths of fuzzy logic; the ability to approximate any non-linear mapping and a high level of readability due to the use of linguistic variables, have made its use attractive. It is certainly true that the use of fuzzy logic enables the designer to describe the desired behaviour of the vehicle more simply than would be possible using alternative mathematically–based techniques.

In many respects fuzzy reactive systems conform to motor schema architecture, Arkin (8). The fuzzification of the sensor signals represents a classification of world state. The antecedents of the rules, by identifying these states, exhibit equivalent properties to perceptual schemas. Each of these classes or perceptual schemas is an "action triggers", used to pass activation to the behaviours that are encoded in the fuzzy rules.

The aggregation of the outputs of fuzzy rules naturally produces an arbitrated output based on the truth-value of each of the active rules. The truth-value of each of the rules is thus an indication of the perceived appropriateness of the behaviour encoded in that rule for the particular state of the environment.

Using fuzzy logic can produce a smooth change in output as inputs change. This has been shown to improve vehicle performance, Bisset and Webber (9).

FUZZY BEHAVIOURAL CONTROLLER ARCHITECTURE

The fuzzy system is based on a zero-order Sugeno system. An additional utility factor is appended to the

weight of each rule. In this way the contribution that is made by the individual rules that make up the behaviour can be adjusted. Each rule is a behaviour primitive, clusters of rules are built up into the motor schema which form the vehicle controller.

$$Output = \frac{\sum_{i=0}^{n} W_i C_i P_i}{\sum_{i=0}^{n} W_i P_i}$$

W_i Weight of rule

C_i Output value of rule

P_i Utility factor

(1)

This extra factor was considered necessary as it enabled of each of the behaviours to be described using only that subset of the inputs, and more specifically membership functions, which intuitively control a behaviour. Though it should be possible to build a controller with the required level of performance without this factor, by careful creation of membership functions and rules, the design process is problematic. All combinations of input membership function must be taken into account, and shaping of membership functions may become a difficult tuning process.

In the fuzzy controller, commands from the direction script may be integrated into the behavioural system by either using additional inputs and rules that define the required commands (figure 1), or by insertion of the command signals before arbitration (figure 2), in parallel with the rule base. This is the preferred option as it maintains the readability of the design, and helps keep the complexity of the rule base down.

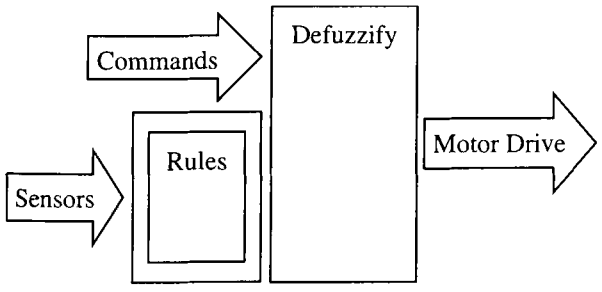


Figure 1, Direct insertion of Commands

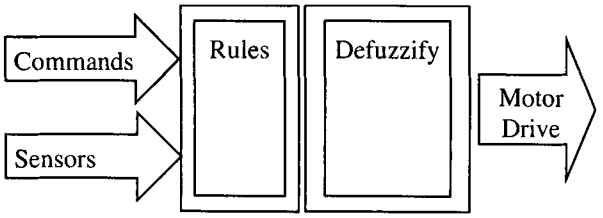


Figure 2, Command insertion as Input

EXAMPLE BEHAVIOURS

Below are listed a subset of the behaviours used, an indication of the interaction between then, illustrating the motivation for adding priority factor to the rules that make them up.

The most basic of the behaviours employed is *"move forward"*. This is an assertion that biases the controller towards movement. There is also a collection of obstacle avoidance behaviours that turn the vehicle away from objects detected by the ultra-sonic sensors. The range to the object determines the angle of turn. To correctly follow the command script it is important that the vehicle travels in a straight course between nodes. This problem is solved by the addition of a *"maintain heading"* behaviour. This behaviour returns the vehicle to the course it was following, after obstacle avoidance has caused divergence. The motor drive signals are integrated over time, and the present heading is calculated. This heading value is the input to this behaviour. Resetting the heading value after a turn has been performed, and inhibition of the behaviour during course-changes, are important aspects of this behaviour.

PRIORITISATION OF BEHAVIOUR

These behaviours are all given priority values. The utility factor of "move forward" is set at a low figure, the priority given to maintain heading is also low, however, as this behaviour may become more important, and a higher figure is used to reflect this. Finally, the obstacle avoidance behaviours are given full priority.

In some situations merely reducing the contribution made by a behaviour to the output may not be sufficient for adequate performance. This begins to becomes evident when the interaction between the *"maintain heading"* behaviour and the obstacle avoidance behaviours are observed.

If the heading error is such that the *"maintain heading"* behaviour becomes active, and an obstacle avoidance

behaviour is active such that there is conflict between the behaviours, the level of turn induced in the vehicle is decreased. In many cases this is not a great problem, as the resulting closer proximity to the obstacle produces a larger degree of turn from the avoidance behaviours. The amount of turn produced is however still smaller than the degree of turn that would be produced if there was no contribution from *"maintain heading"*, and may not be sufficient.

The use of the utility factor provides a mechanism to help manage the contribution rules make. When not in conflict, the contribution of the rules is undiminished. However when conflict arises the effect of the utility factor may clearly be seen.

INHIBITION

The widespread utilisation of lateral inhibition between activities observed in animals has prompted an investigation into the use of an inhibit input to each of the rules. This is justified by the belief that lateral inhibition aids the production of behaviour sequences, which are required to produce some more complex behaviour patterns and reduces the chances of vacillation between output states by providing a measure of state memory.

The introduction of lateral inhibition between behaviours also adds a small element of "non-compromise" to the arbitration system. Non-compromise helps prevent the controller becoming stuck in indecision when similar priority events are presented.

A method for the inhibition of rules in a fuzzy inference system has also been suggested as a solution in other fields, such as the "grumbling rules" problem in signal processing. An example of an application where this may be applied is filtering of images for depth map extraction Hughes *et al* (10) Here the fuzzy inference engine can be thought of as selecting filters from a bank, and applying them to the data. Rules with low firing weights can seriously affect the performance of such a system.

Inhibition Schemes

The inhibition schemes were considered. included:

- Best *X* rules
- Selecting rules by size of contribution
- Dynamically adjusted utility factor.

Of the techniques listed, the most applicable to behavioural control is the use of the dynamically variable utility function, as this adds the required element of memory to the system. This is also the only scheme which provides true lateral inhibition.

The use of *Best X rules*, and *selection by contribution*, may be appropriate for cases such as the grumbling rules problem, but in this application were found to be unsuitable.

Selection of this method raises some implementation issues. For full connectivity the number of possible inhibitory routes increases rapidly. This however may be countered by breaking the flat architecture of the rule base and creating groups of rules that correspond to the composite behaviours. Inhibition is then only applied between groups. Not only does this reduce the number of factors and connections to manageable levels, but it also helps maintain the clarity of the system. With full connectivity, trying to understand the system and predict its behaviour becomes vastly more complex.

The memory effect occurs because the fuzzy system is a synchronous system, the firing weights of all the rules are calculated, then the output calculated by aggregation and defuzzification. On calculation of the system output, the controller is free to return to measure the environment, and fuzzify the inputs. In this manner the fuzzy behavioural controller falls short of many authors' ideal for such systems, where asynchronous operation of behaviour elements is preferred. Inhibition values are only used in aggregation calculations in the following iteration.

Selection of inhibition scheme

In the work previously reported, Tubb *et al* (11), all inputs to the fuzzy behavioural controller were treated in the same manner. The rule base was a single homogenous unit, with all rules, regardless of their "home" behaviour shoulder to shoulder. In the system presently being developed the rule base is broken into a series of subsections, each of which is treated as an atomic behaviour for the purposes of the inhibitory system. Each of these may also be assigned a utility modifier, hence simplifying the prioritisation of behaviours.

Each of the behaviours is formed as a separate fuzzy system. The outputs of these are then aggregated and the weighted average is calculated. The controller is thus a fuzzy system, where each of the "rules" is also a Sugeno fuzzy system.
By adopting this approach the number of inter-behaviour connections is reduced, along with the total number of potentially "hard to tune" utility factors.

The inhibitory factor for each behaviour is calculated from the sum of the truth-values, of all the behaviours

that inhibit the behaviour. The inhibit factor is the complement of this sum.

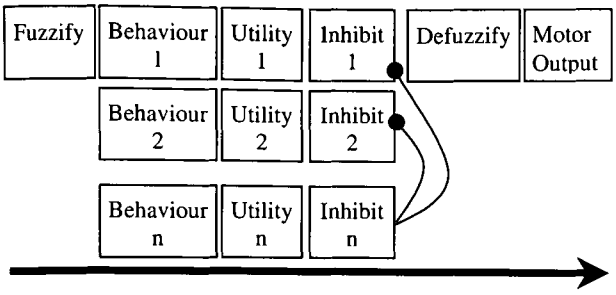


Figure 3, Inhibition scheme

The direct couple from the fuzzy system to the vehicle motors has also been severed. This also helps simplify construction of the rule base as it enables the designer to divorce vehicle action from individual motor action.

Breaking the direct connection to the motor also enables a greater degree of portability between test platforms and vehicle architecture. Some authors, for example, Bisset and Webber (9), have advocated this methodoligy in an attempt to produce a standard approach to behavioural controller design. However, as the morphology of the vehicle changes so do other control factors. Even if the structure of the rule base survives without change the fuzzification parameters will need adjustment.

The similarity between the motor schema control observed in biological systems and the fuzzy behavioural controller is to be further investigated. It is believed that this will suggest further changes in architecture that will produce more robust and flexible controllers.

CONCLUSIONS

A marked similarity observed between fuzzy logic and reactive control schemes has been observed and exploited in the past to create controllers for mobile robots.

Potential problems still exist, primarily concerned with the selection and tuning of input membership functions and rule parameters.

The problems associated with the use of reactive control schemes are not addressed by the use of fuzzy logic alone, these are the reflexive nature of the technique, its dependence only on the state of the environment at the particular point in time when the calculations are made.

An additional, non-resolved issue is the prioritisation of behaviours. Though the use of fuzzy logic does enable elements to exhibit differing levels of activation, due to varying levels of membership at the inputs in situations where several of the inputs show great similarity. In these cases the design of rules to cope with all situations becomes more difficult, and the degree of interaction between behaviours that may have quite different output requirements is increased, leading to hard to predict results.

The extra flexibility of behavioural control schemes over the simpler reactive techniques is due to the ability to store state and representations across behavioural elements.

To achieve this some changes must be made to standard fuzzy inference mechanisms. The use of scaling factors can create prioritisation effects between rules.

Lateral inhibition between rules can create a element of sequential state, were the condition of the controller is dependent not only on the environment now, but also its condition in the past. Inhibition between individual rules would cause an explosion in the complexity of the system, this is countered by breaking the flat rule architecture into manageable pieces. Each behaviour being encoded in a separate rule base.

REFERENCES

(1). Arkin R.C. Journal of Robotic Systems, **9**, 351 (1992).

(2). M. J. Mataric, *Proc. IEEE Interational Conferance on Robotics and Automation. Workshop on Architectures for Intelligent Control Systems.*, Nice France, pp. 46-54 (1992).

(3). Mataric M.J. Journal of Experimental and Theoretical Artificial Intelligence, **vol. 9**, pp 323 (1997).

(4). R. A. Brooks. IEEE Journal of Robotics and Automation, **RA-2**, 14 (1986).

(5). Connell J.H., *Minimalist Mobile Robotics: A Colony Architecture for an Artificial Creature* Academic Press Ltd., London, (1990).

(6). R. S. Aylett, A. M. Coddington, R. A. Ghanea-Hercock and D. P. Barnes. Design and Developement of Autonomous Agents, **95/211**, (1995).

(7). Li Wei, *IEEE Int Conf on Fuzzy Systems*, 1994 **3**, pp. 1626-1631 (IEEE 1994).

(8). Arkin R. C. International Journal of Robotics and

Automation, **81**, 92 (1989).

(9). Bisset D.L. and Webber A.D. - *Proc AISB'94 Workshop*, Leeds, 1994, (1994).

(10). Rothwell Hughs N., Roberts G.N. and Wilson G.R. *Application of fuzzy signal processing to three dimensional vision".* IEE 5th. International Conference: Factory 2000, pp 319-324, Cambridge. (1997).

(11). Tubb C.A.J., Roberts G.N. and Rowlands H. *Behavioural control implementation using fuzzy logic.* 12th. International Conference on Systems Engineering. Vol. 2. pp 695-699. Coventry (1997).