A Neural Endocrine Architecture for the Control of Multiple Robots

Lachlan Murray

12th May 2009

Submitted in part fulfilment for the degree of Meng

Supervisor – Professor Jon Timmis

Word count: 33,736 as counted by detex | wc -w
This includes the body of the report only
Abstract

The Neural Endocrine Control Architecture has been used on a number of occasions as an effective method of robot control, but never before has it been seen to control a group of multiple robots.

In this project an implementation of the neural endocrine control architecture is presented, that shows not only is the architecture capable of controlling the individuals of a multi-robot system, but with the inclusion of collaborative behaviours is able to do so in an extremely effective manner.

The performance of the system implemented is assessed at the task of foraging and is shown in some situations to produce a faster than linear speedup over a single robot attempting the same task.
## List of Figures

2.1 Diagram of a neuron, adapted from Hart Biology (2008) .................................................. 11
2.2 A multi-layer perceptron ........................................................................................................ 11
2.3 An artificial neuron ................................................................................................................ 12
2.4 \( y = \tanh(u) \) .................................................................................................................... 12
2.5 The classical horizontal decomposition of a robot control system is shown in (a). The vertical alternative, as proposed by Brooks (1986) is shown in (b) .................................................. 20
2.6 A single module of a subsumption architecture .................................................................... 21
2.7 A simple motor schema architecture .................................................................................... 23
2.8 Hierarchical control system. Adapted from Arkin (1998) .................................................... 26
2.9 AuRA architecture. Adapted from Arkin (1998) ................................................................... 28

3.1 World 1 .................................................................................................................................... 32
4.1 The simulated Pioneer robot used by this project ................................................................. 39
4.2 Two networks, the neurons of which are all sensitive to the same hormone ......................... 40
4.3 A single network with different neurons sensitive to different hormones ......................... 40
4.4 Overview of the system .......................................................................................................... 42
4.5 The division of the cameras field of view into eight segments ............................................. 44
4.6 Inheritance diagram for Behaviours ...................................................................................... 47

6.1 Graphs showing the number of pieces of rubbish collected over periods of 300 (a) and 1200 (b) seconds, with varying numbers of robots between one and seven ..................................................... 58
6.2 Graphs showing the number of pieces of rubbish collected per robot over periods of 300 (a) and 1200 (b) seconds, with varying numbers of robots between one and seven ..................................................... 59
6.3 Graphs showing the average performance speedup (a) and the average efficiency (b) of robots at collecting the first six pieces of rubbish .................................................................................... 59
6.4 Graphs showing the number of pieces of rubbish collected (a) per robot (b) over a period of 300 seconds, with varying numbers of robots between one and seven ............................................................. 61
6.5 Graphs showing the number of pieces of rubbish collected (a) per robot (b) over a period of 300 seconds, with varying numbers of robots between one and seven ............................................................. 61

A.1 Overview of the system .......................................................................................................... 67
B.1 The distance covered by seven robots in world 1 after 160 seconds ..................................... 68
B.2 The distance covered by seven robots in world 2 after 160 seconds ..................................... 68
B.3 Graph showing the levels of a single robot’s bin, rubbish and wander hormones; along with their desires to find rubbish and the bin. Over 450 seconds the robot picks up and drops off three pieces of rubbish. ................................................................................................................. 69
B.4 Graph showing all the hormone and desire levels of a robot over a period of 450 seconds. ............................................................................................................................................ 70
1 Introduction

There are several real-world tasks which at present are either carried out ineffectively by humans or inefficiently by single robots. Tasks such as the cleanup of hazardous waste or search and rescue operations at disaster sites are often too dangerous for humans to respond effectively to and are simply too large for a single robot to solve alone. Less critical tasks such as cleaning rooms or mowing lawns are considered undesirable by some people and where speed is of the essence would take too long for a single robot to complete. The use of multiple robots provides an alternative approach to solving these tasks; multiple robots are capable of reaching places that are either too dangerous or impossible for a person to reach and because of the inherent parallelism are able to tackle very large scale problems in relatively short periods of time. Environments within which multi-robot systems are expected to function are often highly dynamic and unpredictable, and so the method by which the robots are controlled must take this into consideration. One area of robotics that is particularly well suited to the control of robots in dynamically changing environments is known as behaviour-based robotics, the neural endocrine control architecture designed in this project falls into this category.

The neural endocrine control architecture is a novel, biologically inspired, design model for a robotic control system. It has successfully been used to control a single robot on many previous occasions (Lord, 2007; Neal and Timmis, 2003, 2005; Vargas et al., 2005) but until now its effectiveness at controlling a collection of robots has not been investigated. The addition of more robots brings added complexity to the system, it is necessary that a multi-robot control system not only encompasses the ability to control individual robots, but is also capable of appropriately handling the interactions with other robots. As well as this being the first attempt at controlling multiple robots using the neural endocrine control architecture, this project presents one of the most complex system designs to date. In order to assess the effectiveness of the system it was necessary to design a task for the robots performance to be measured on, the task chosen was a variant of foraging and was one of the most complicated tasks that the neural endocrine control architecture has been applied to.

Project aims

1. To investigate whether the neural endocrine control architecture is capable of controlling a multi-robot system
2. To investigate how effective the architecture is at controlling a multi-robot system
3. To investigate the capabilities of the architecture at a new and complex task

Report structure

2. Literature Review This section covers all the necessary background information and looks at some of the previous work done with the neural endocrine control architecture as well as reviewing other behaviour-based and multi-robot systems.

3. Problem Analysis This section develops the problem, describing the task in detail and introducing the behaviours necessary to complete it. As well as presenting some functional and non-functional requirements of the system.

4. Design and Implementation This section present the full system design, looking at it from a high-level at first before describing specific details of some of the low-level implementation.

5. Experiments This section outlines the experiments that were carried out, including the environments in which they took place and the information that was collected.

6. Results and Evaluation This section presents the results of the experimentation and evaluates the performance of the system, as well as describing how the requirements were met.

7. Conclusions This section concludes and summaries the projects results and findings.
Statement of ethics

This project was carried out entirely in simulation with no human interaction and hence no direct link to causing harm. Some of the positive applications of behaviour-based robotics were mentioned earlier but equally the general principles of the architecture presented here could be applied to more ethically dubious situations, such as for the control of military robots. As mentioned there are no human participants in this project and so consent is not relevant and since no data was collected from participants, confidentiality is not an issue.
2 Literature Review

The neural endocrine control architecture presented in this project can be described as what Arkin (1998) calls a "behaviour-based reactive system". The first half of this chapter, sections 2.1 and 2.2 are concerned with providing all the necessary background information to the key topics of this project, starting with an introduction to behaviour-based robotics and followed by an introduction to the neural endocrine architecture. Once the background information has been covered, section 2.3 is dedicated to discussing multi-robot systems. As well as describing some of the key components, a short review is provided that covers one of the most important features that must be addressed when designing a multi-robot system. The final three sections of this chapter, 2.4, 2.5 and 2.6 are concerned with particular types of robot control architecture, the first focuses on reactive control systems, of which the neural endocrine architecture is said to be. Following on from reactive systems are deliberative reasoning systems, often thought of as the opposite of reactive systems. The final section of this chapter covers hybrid deliberative-reactive systems which combine the advantages of both the previous types of system. In the final three sections the advantages and disadvantages of each type control system are discussed and specific implementations from each of them are reviewed. At least one multi-robot system from each architecture is described and any relevant issues from previous work are detailed and analysed. Since the neural endocrine control architecture is a behaviour-based reactive system, the second half of this chapter is heavily weighted towards reactive control systems.

2.1 Behaviour-based robotics

The field of behaviour-based robotics first arose in the 1980s, due largely to many people’s frustration with the poor performance of robots in real world environments (Brooks, 1991). In 1986 Rodney Brooks drastically changed the field of robot control with the introduction of his behaviour based subsumption architecture (Brooks, 1986), the details of which are discussed in section 2.4.4. Many different behaviour based architectures have been created since Brooks (1986), the neural endocrine control architecture of this project being one of them. This section introduces the field of behaviour based robotics generally, specific examples are saved for review in section 2.4.

2.1.1 Basic principles

Behaviour-based systems are made up of several individual behaviours, when combined in the appropriate way these behaviours allow robots to carry out complex tasks. Individual behaviours are defined by stimulus-response pairs: when a robot receives a known stimulus, the response of the associated behaviour is executed. There is no centralised control in a behaviour based system, behaviours interact with each other through the environment rather than internally through the system, for example, one behaviour may initiate a change in the environment which may in turn stimulate a new behaviour (Matarić, 1992a). Robots do not maintain a persistent model of their environment, they simple observe their neighbourhood and react according to their behaviours. As summarised by Arkin (1998), the main features common to all behaviour-based systems are:

- tight coupling of sensing and action
- lack of symbolic knowledge or world model
- decomposition into separate behaviours.

Behaviour-based systems can be differentiated in a number of ways. A system may be made up of different types of behaviour and the mechanisms for coordinating behaviours may vary, the mapping between stimuli and responses can also be very different, the remainder of this section focuses on these three points.

2.1.2 Types of behaviour

Behaviour-based systems have an ethological background, many different types of animal behaviour can be observed that a robotocist may wish to replicate or draw inspiration from. Beer et al. (1990) support the use of ethology as a source of inspiration and discuss three behaviours in particular that Arkin (1998) summaries as:
**Reflexes**: involuntary, spontaneous responses to certain stimuli. The size of the response is often directly related to the strength of the stimulus and the response only persists as long as it is still being stimulated. Examples in humans include sneezing or literally knee-jerk responses, more useful reflexes include the withdrawal of ones hand from a hot object or the escape reflexes found in some snails and worms.

**Taxes**: behaviours that cause an animal to move towards, or away from a certain stimulus. Taxes can be classified according to the stimuli that instigate them, chemotaxis, phototaxis and thermotaxis for example correspond to chemical, light and heat stimuli respectively. Trail following ants are one of the most obvious example of a taxis in action, in this case an attractive chemotaxis.

**Fixed Action Patterns (FAP)**: responses that last longer than the stimulus that triggered them, FAPs normally run uninterrupted until completion. They are not effected by the strength of the stimuli as reflexes are, so generally occur with the same magnitude each time they are stimulated. Examples from nature include the songs of crickets, or the mating dances of birds.

### 2.1.3 Encoding of responses

A key component of behaviour-based robotics is the mapping between stimulus and response or perception and action, this can be achieved in a number of ways. Often the strength of the stimulus will effect the strength of the response (reflexes, taxes) but not always (fixed action patterns). In their simplest form responses can be encoded in a lookup table, or a series of if-then statements between perceptions and actions. Responses can be triggered when the strength of their stimulus reaches a certain threshold, or simply due to its presence or absence. In more complex encodings the strength of a response can relate to the strength of the stimulus, for example the two may be directly \((2.1)\) or inversely \((2.2)\) proportional to one another. The strength of response can also be effected by factors other than the stimulus that triggered it, for example the internal state of the robot, if a robot were running low on power then its response to sensing a charging station ought to be more urgent than if it were not.

\[
R \propto S \quad (2.1)
\]

\[
R \propto \frac{1}{S} \quad (2.2)
\]

where \(R\) is the strength of the response and \(S\) is the strength of the stimulus

### 2.1.4 Coordination of behaviours

Conflicts can emerge if two or more behaviours that have competing responses are triggered at the same time, for example one behaviour may wish to turn a robot right whilst another may wish to turn it left. If two conflicting behaviours are allowed to proceed they may result in undesirable effects such as the robot stalling. In order to achieve a more suitable result the behaviours must be coordinated effectively, this does not necessarily mean choosing one behaviour exclusively over another although that is one possibility. Arkin (1998) describes two of the main behavioural coordination mechanisms:

**Competitive**: this method is based on a winner-takes-all strategy, from a list of competing behaviours only one will be allowed to execute. The winning behaviour can be selected in a variety of ways. One possibility is to create a hierarchy over all the behaviours in the system, whereby at each level behaviours suppress all those of a lower priority. Another alternative is to allow individual behaviours to vote on a set of specific actions such as {turn-left, turn-right, straight-ahead}, the action which receives the most votes is then executed.

**Cooperative**: this method allows for more than one behaviour to have an influence on the robot at any one time. The responses of different behaviours are fused together to produce the overall global behaviour. The usual method of fusion is to assign every behaviour a relative strength or gain, to determine the overall response of the system, the output of each behaviour is multiplied by its gain and the results summed.

### 2.1.5 Designing behaviour-based systems

The types of response encodings, coordination methods and behaviours used in a behaviour-based system are decisions for the designer, and it is perfectly legitimate to combine different behaviour types, response encodings and coordination methods in the same architecture. In section 4 when discussing the design of the system implemented for this project, all of the design choices are explicitly stated.
2.1.6 Emergent behaviours

It is not uncommon for behaviour based systems to be described as having some emergent behaviours. Emergent behaviours in this sense are defined to be: observable, distinguishable behaviours that are not programmed into a system, rather they result from the robots interaction with the environment and the combination of other simpler behaviours.

2.1.7 Summary

The key features of behaviour based systems are: a tight coupling between perception and action, the lack of a symbolic world model and the decomposition of control into separate behaviours. There are three characteristics that can help define behaviour based systems: the types of behaviours they involve, i.e. reflexes, taxes or fixed action patterns; the way in which responses are encoded, i.e. whether they are directly or inversely proportional to the strength of the stimulus; and how the responses of different behaviours are coordinated, i.e. competitive vs. cooperative approaches.

2.2 Neural Endocrine Control Architecture

The neural endocrine control architecture of Neal and Timmis (2003) grew from the desire to create a system whereby emotion was more than simply an observed property of behaviour. An agent that moves away from an object at speed, may be described as fearful by an observer, regardless of how the movement was instigated. Neal and Timmis devised a system where emotions, such as fear, are actually responsible for stimulating responses in the first place, rather than just labelling them.

The approach taken by Neal and Timmis is heavily inspired by biology, hormones are used to describe the emotional state of robots and neural networks are used to control their actions. In this section the basic neural endocrine architecture is introduced, a review of some of the previous implementations of neural endocrine control systems is saved for section 2.4 where comparisons will be made to other types of robotic control system. Before describing the architecture it is necessary to very briefly introduce the biological components that served as inspiration. More detailed information on the biological systems described below can be found in Widmaier et al. (2004).

2.2.1 Biological background

**Homeostasis**  The eventual goal of the neural endocrine architecture is to create robots capable of surviving for long periods of time without human intervention, the concept of homeostasis is key to ensuring this. An organism can be described as homeostatic if it is capable of maintaining a stable internal state, regardless of any external environmental changes. Homeostasis, to some degree, can be achieved through the interactions of the nervous system and the endocrine system, both of which are described shortly. The immune system also plays a big part in maintaining homeostasis, no artificial alternative is implemented in this project and so it is not described here, however it is discussed in section 7 as a possible future extension.

**The nervous system**  The nervous system is made up largely from a network of cells known as neurons, a diagram of a single neuron is shown in figure 2.1. Signals are passed between neurons as electrical charge, emanating from the terminals of one neuron and travelling across synapses into the dendrites of other neurons, when this happens the first neuron is said to have “fired”. From the dendrites the signals proceed through the cell body and towards the axon, if the strength of the signal is above a certain threshold the signal is able to propagate up the axon and fire again at the terminals. Signals can have either an excitatory or inhibitory effect on the receiving neuron, increasing or decreasing its chances of firing.

The general purpose of the nervous system is to process sensory input and produce effector output. Multiple neurons are connected together to create very complex neural networks, input is provided to the network through afferent neurons from senses such as sight and touch and after propagating through the network signals eventually end up at efferent neurons which stimulate other cells such as muscles and glands.

**The endocrine system**  The two main components of the endocrine system are the endocrine glands and the hormones which they secret. Hormones are simply chemical messengers that travel around the body in the bloodstream. The secretion of hormones can be caused by a number of factors, including: changes in the concentration of organic nutrients, changes in the concentration of other hormones and stimulation by the nervous system. Not all cells in the body respond to all types of hormone, the cells which do respond to a specific hormone are known as the target cells of that hormone. Target cells contain a number of specific
receptors for each type of hormone they are sensitive to, the quantity of a particular receptor determines the cells responsiveness to the corresponding hormone. When a hormone comes across a target cell, it binds to it via the cell receptors and initiates signals within the cell, which in turn alter the cell’s activity. Hormones are not persistent and eventually they will decay, at a rate specific to each type of hormone.

### 2.2.2 Artificial counterparts

**Artificial neural networks** Put simply, Artificial Neural Networks (ANNs) are massively parallel distributed processors, inspired by the functionality of the human nervous system (Haykin, 1999). Since they were first introduced by McCulloch and Pitts (1943) ANNs have been used for a variety of different problems from pattern recognition to robot control.

There are many different types of ANN, one of the simplest, and that which is used in this project is the Multi-Layer Perceptron (MLP). Multi-layer perceptrons consist of one input layer, one output layer and one or more hidden layers, each of which is constructed from a number of artificial neurons. Figure 2.2 shows a typical configuration of an MLP with one input layer, one output layer and one hidden layer. Each circle in figure 2.2 represents an artificial neuron, also referred to as a node, the arrows leading into a node are the inputs of that neuron and those leading away are its outputs. ANNs can also contain bias nodes which take no inputs and always output the same value, in figure 2.2 there is a single bias node that always outputs ‘1’.

Each neuron in an ANN is a mini computational unit, through the interactions of individual neurons different values at the input layer lead to different values at in the output layer. Figure 2.3 shows a diagram of an artificial
Figure 2.3: An artificial neuron

Figure 2.4: $y = \tanh(u)$
neuron from a hidden or output layer. Neurons in the input layer differ from those in the other layers as they have only one input and contain no activation function, they simply pass any input directly to their output. The strength with which an individual neuron “fires” in the hidden and output layers, i.e. the value of its output, is determined by three factors: its inputs \((x_i)\), the weights of its inputs \((w_i)\) and its activation function \((f(u))\). In order to determine the output of a neuron all of its inputs are first multiplied by their weights and summed together \((2.3)\).

\[
u = \sum_{i=0}^{nx} x_i \cdot w_i
\]  

(2.3)

The result \((u)\) is then passed through an activation function \((2.4 & 2.5)\), this function can take a number of forms but is usually chosen such that the majority of inputs will lead to one of two different outputs, representing whether or not the neuron fires. A commonly used function is \(tanh\) which outputs values between 1 and \(-1\), as shown by the graph in figure 2.4

\[
y = f(u)
\]  

(2.4)

\[
f(u) = tanh(u)
\]  

(2.5)

Perhaps the most important factor in the design of an ANN are the values of its weights, these can either be chosen directly by the designer or produced automatically through the use of a learning algorithm. The most common learning algorithm for deciding the weights of an MLP is the back-propagation algorithm. Before the back-propagation algorithm can be run it is necessary to create a set of training data, this consists of a list of inputs to the network and the corresponding desired outputs. The back-propagation algorithm starts by randomly initialising the weights of the network and then running the network on some of the training data. A gradient descent approach is used that incrementally adjusts the weights of the network until the error between the desired outputs of the training data and those actually given by the network is below some threshold. The steps as described by Haykin (1999) can be summarised as follows:

1. Initialisation - randomly initialise the weights of the network
2. Presentation of training examples - present some subset of the training data to the network
3. Forward computation - pass the input values of the training data through the network in the normal fashion and then compute the error signal between the resulting outputs and those suggested by the training data
4. Backward computation - propagate the computed error signal back through the network and update the weights according to how much they contributed to the error
5. Iteration - repeat 3 and 4 for different training examples until the error is sufficiently small

Artificial Endocrine Systems  The Artificial Endocrine System (AES) described here is based on the original design proposed by Neal and Timmis (2003, 2005) as well as subsequent modifications made by Timmis et al. (2006) and Lord (2007).

As is the case in the biological endocrine system, the two main components of an AES are glands and hormones. Artificial glands \((g)\) release artificial hormones when they are stimulated. Stimulation can be caused by both the internal state of the system and external stimuli. In Timmis et al. (2006) signal values \(A_i\) were obtained by summing sensor inputs and in Lord (2007) similar gland activation values were calculated from the combination of sensor values and the internal state of the robot. The stimulation of a gland \((R_g)\) as given by Timmis et al. is shown in \((2.6)\) where \(a_g\) is the stimulation rate, that is the rate at which a hormone is released from a gland \(g\).

\[
R_g(t) = a_g \sum A_i(t)
\]  

(2.6)

Lord (2007) investigated a second method of stimulation that also takes into account the current concentration of hormone \(c_g(t)\) this is given by \((2.7)\). As can be seen in \((2.7)\) the amount of hormone released in this method is subject to a negative feedback mechanism, the reason for including this is to prevent the system from becoming over saturated with a particular type of hormone.

\[
R_g(t) = \frac{a_g}{1 + c_g(t-1)} \sum A_i(t)
\]  

(2.7)
Every hormone has an associated decay rate \((\beta_g)\) which takes a value from \([0,1]\), this means that without stimulation the concentration of a hormone will eventually be reduced to an insignificant amount. The concentration of a particular hormone \(c_g\) at time \(t + 1\) is given by equation (2.8).

\[
c_g(t) = \beta_g c_g(t - 1) + R_g(t)
\]

**Neural Endocrine Systems** In this project the only things that hormones can effect are artificial neurons. In line with the biological endocrine system not all of the neurons in a system will be sensitive to all hormones, the sensitivity of a neuron \(i\) to the hormone released by a particular gland \(g\) is given by \(s_{ig}\). The effect that hormones have on neurons can be calculated by modifying (2.3) so that it takes into account the sensitivity of inputs to particular hormones and the concentration of those hormones. (2.9) is a modified version of (2.3) that integrates an artificial endocrine system with \(ng\) glands.

\[
u = \sum_{i=0}^{nx} x_i \sum_{g=0}^{ng} c_g \cdot s_{ig}
\]

An artificial neural network that is combined with an artificial endocrine system is described here as a neural endocrine network, when a system contains multiple neural endocrine networks their outputs must be combined in a suitable way. The most common form of coordination is a cooperative approach whereby the outputs of each network are simply summed together. The resulting behaviour of a multi-network neural endocrine system is dependent on the current hormone levels of the system. High levels of a particular hormone will effect some networks more than others, giving these networks more or less influence over the global result when the network outputs are summed together.

### 2.2.3 Summary

The neural endocrine control architecture was inspired by the biological nervous and endocrine systems and the desire to create a homeostatic robotic control system. The release of hormones in an AES is controlled by the stimulation and decay rate of hormones as well as the current signal or activation values of the glands. Artificial hormones can effect the overall output of an artificial neural network by influencing the network at the neuron level. In systems that contain multiple neural endocrine networks the overall response can be determined by summing the outputs of the individual networks.

### 2.3 Collective multi-robot systems

This section begins with a brief introduction to some of the types of task that collective multi-robot systems can be applied to, following which, some of the advantages of using multiple robots are described. The specific problem of interference in multi-robot systems is introduced and one example of how it can be reduced is described in detail, other alternatives are mentioned briefly. Finally some of the main design considerations that must be taken into account when implementing multi-robot systems are discussed.

#### 2.3.1 Multi-robot tasks

The decision to use a collection of robots instead of a single robot is often influenced by the task at hand. Cao et al. (1997) and Dudek et al. (1996) identify three different types of task for which a designer may choose to use a multi-robot system. The first type of task are those which are too complex, or impossible, for a single robot to complete on its own, for example when it is necessary for two spatially separate actions to be carried out simultaneously, in this case the use of multiple robots in a necessity rather than a choice. The second type of task are those which are traditionally solveable using multiple agents, examples given by Dudek et al. include transportation and industrial tasks. In traditionally solveable tasks there is often a clear division of labour into separate subtasks, if each of these can be completed in parallel, then the task is particularly well suited to a multi-robot system. The third and final type of task is that which could be completed by a single robot but would benefit from the use of multiple robots, tasks such as tidying a room fall into this category. A single robot would be capable of tidying a room on its own but by adding more robots it may be possible to complete the task quicker or more reliably.
2.3.2 Advantages

Sycara (1998), Cao et al. (1997), Parker (1994) and Dudek et al. (1996) identify some of the main advantages of using multi-robot systems:

**Speed** Perhaps the most obvious advantage of a multi-robot system is the potential to reduce the time taken to complete a particular task. Multiple robots can often complete a task in a shorter period of time than a single robot. It is intuitive to think that the more robots that are added to a system the greater the speed-up will be, and whilst increasing the number of robots certainly can improve factors such as speed, this performance gain cannot be achieved without careful planning, as will be seen shortly.

**Cost** Often it may be much cheaper to use several simple robots than it is to use a single complex one. Directly linked to cost is expendability, if the robots are cheap to make and plenty are available then as long as the task is successfully completed it does not matter if some robots are damaged or destroyed along the way. For this reason multiple robots lend themselves particularly well to tasks involving dangerous or high-risk situations.

**Robustness** Having multiple agents, each with identical goals and behaviours, provides a level of redundancy that can lead to a highly fault tolerant and robust system. In the face of unexpected changes, such as if one robot from a group fails, the remainder of the system should be able to carry on without it, albeit perhaps with at a lower level of functionality or quality.

**Efficiency** Multi-agent systems can be described as time, energy and computationally efficient. Speed has already been identified as an advantage of multi-robot systems. By adding more robots to a system it is possible to reduce the time taken to complete a task and if designed correctly this speed-up can be super-linear, that is to say, if the number of robots was doubled then the time taken to complete a particular task would be reduce by more than half. The power requirements of the robots in a multi-agent system will be much less than that of a single complex robot; due to the small and simple components used it is likely that the robots will also be more energy efficient. In terms of the system as a whole computational efficiency is increased as a result of the inherent concurrency of multi-agent systems.

**Flexibility** Having multiple agents, some of which have different goals and behaviours, can provide an amount of flexibility making it possible for a group of robots to dynamically reorganise itself to solve different problems. If carefully considered it is possible to have flexibility without loss of robustness. If the robots are suitably intelligent and able to spot a drop in the performance or the failure of other robots (a fairly big assumption), then in the best case they may be able to take responsibility of the broken robot’s goals and in the worst case they may be able to ensure the system fails safely.

**Reliability** A reliable system can be defined as a system that completes its task to a high standard a large proportion of the times it is run. Thanks to properties such as robustness, redundancy and flexibility, multi-robot systems can be seen to be highly reliable, however this does not come for free and in order to guarantee reliability the system must be sensibly designed.

2.3.3 Interference

The first goal in the construction of a multi-robot system is to overcome the problem of interference between robots (Matarić, 1992b). Robots can interfere with each other in a variety of ways depending on the task at hand. Robots with identical goals may conflict whilst competing for resources and robots with different goals may hinder the performance of others by undoing their work or causing deadlock and looping situations (Matarić, 1993). To even reach a linear speedup with regard to the number of agents in a system is a non-trivial task, to go beyond a linear speedup requires that not only is interference minimised but also that some form of inter-robot cooperation is incorporated.

Shell and Matarić (2006) studied the effects of interference for large-scale multi-robot systems in simple open arena environments. Two different strategies were investigated, a traditional homogeneous approach and an alternative known as bucket-brigading. In the homogeneous approach all robots are given the exact same set of goals and behaviours. In tasks such as foraging, where robots are required to deposit objects at some centralised location, the traditional approach can lead to a large amount of interference as robots strive to reach the goal and avoid crashing into each other at the same time. The bucket-brigading strategy aims to reduce interference by confining robots to overlapping sub-regions. Rather than attempt to deposit objects at a single location, in the bucket-brigading approach robots simply deposit them outside of their own sub-region, in the general direction of the goal. The resulting behaviour observed with the bucket-brigading approach is that robots pass objects to their neighbours in regions that are closer to the goal. Eventually objects reach a region that includes the goal,
from which point it is easy for a robot who occupies that region to deposit them at the goal. The results of Shell and Matarić’s experiments showed that on the whole that the bucket-brigading strategy performed better than the homogeneous approach, although only when the number of robots was greater than one hundred.

The most obvious problem with the bucket-brigading approach is a lack of redundancy, if a robot fails then any objects exclusively within its sub-region cannot be reached by the other robots. This problem is reduced in Shell and Matarić’s experiments due to the way in which robots monitored their position in the world. To determine whether they were still within their sub-region, rather than have a map or a global description of their location, robots simply monitored where they had travelled relative to their starting position. The robots motors were subject to a certain amount of random noise which meant that the sub-region they believed to be occupying slowly drifted over time. If one robot did fail, at some point the region in which it broke, would eventually be covered by others as their regions shifted location. Another potential problem with the bridge-brigading approach is that it involves an extremely large number of pickup-object and drop-object actions, especially when compared to the homogeneous approach. If pickup and drop operations consume a lot of power it could make the system very inefficient.

An issue that is not raised by Shell and Matarić and a general problem for multi-robot systems is how to suitably deploy the group of robots. In their experiments Shell and Matarić positioned robots randomly, however for a real world task this approach may not be appropriate or even possible. Stoeter et al. (2002) describe an interesting robot team whereby a single large “ranger” robot is able to deploy small very mobile “scout” robots by firing them from a cannon. A simpler method of deployment may be possible for the bucket-brigading task whereby robots randomly wander and avoid other robots until as a group they have covered a large proportion of the environment, at this point they could proceed with the strategy described by Shell and Matarić.

Zuluaga and Vaughan (2005) propose an alternative approach to reducing the problem of interference, theirs is based on the aggression of conflicting robots. Zuluaga and Vaughan’s method was designed to reduce the interference that robots experience in situations such as the navigation of tight corridors or doorways. When navigating small spaces if two robots meet it is often necessary for one of them retreat and allow the other to pass. In nature some species fight or display aggression to assert their dominance, similarly Zuluaga and Vaughan’s robots fight to determine which robot should retreat. Rather than physically fight, Zuluaga and Vaughan’s robots simply display their levels of aggression, the robot with the highest level of aggression wins, and the loser is forced to back down and retreat.

In the experiments of Zuluaga and Vaughan (2005) the aggression of a robot is determined by the amount of effort it has recently put into navigating areas where the likelihood of interference is high (areas such as corridor or doorways), the more effort they have put in, the higher the level of aggression. The best resolution to a conflict is the one in which the retreating robot is hindered the least. The hindrance of a robot can be measured by how far they have to travel in order to let the other robot pass, this distance is greater if the robot has come from an area where there is a high probability of interference occurring, therefore if the robot with the lowest level of aggression retreats it will have to travel a shorter distance than the other robot would have to, which has a higher level of aggression.

A simpler, but less efficient version of what Zuluaga and Vaughan (2005) implemented would be to impose a hierarchy over all the robots in the system, such that robots always retreat for those which are above them in the hierarchy. Matarić (1993) investigated the effectiveness of such a dominance hierarchy, the findings of which are reviewed later in this chapter. Another solution to minimising interference is what Matarić (1992b) calls intelligent-coexistence, this is also reviewed in a later section of this chapter.

2.3.4 Designing multi-robot systems

There are many different approaches that can be taken when implementing multi-robot systems. The designer must consider the type of hardware and control methods to be used, as well as the way in which the robots will communicate (if at all). The actual task that the system is being designed for will also play a part in the design process. A lot of work has been done towards the classification of multi-robot systems. Dudek et al. (1996) present a taxonomy for multi-agent robotics and Cao et al. (1997) produce a broad review of cooperative robotics, Matarić (1993, 1992b) also adds insight into some of the design features of multi-robot systems. Based on the observations of Cao et al., Dudek et al. and Matarić the main properties of multi-robot systems are summarised below, the remainder of this section describes each of them in more detail.

- **Control** - the methods used to control the individual robots
- **Population** - the hardware of the robots and the number of robots in a collection
- **Interaction** - whether the robots interact with each other and the methods they use
- **Cooperation** - whether the robots cooperate with each other and the methods they use
2.3.5 Control

The control of the robots in a multi-agent system can be described as centralised (global control) or decentralised (local control) (Cao et al., 1997; Matarić, 1992b). In a centralised architecture a single agent controls the actions of all of the individual robots, this allows the behaviours of the robots to be extremely well coordinated. The status of each robot in a centralised system needs to be recorded in order to represent the global state of the system, as the number of agents increases, maintaining the global state becomes increasingly difficult. In a decentralised architecture each robot is controlled by a separate system. Cao et al. (1997) states some of the advantages of a decentralised system to be: increased fault tolerance, parallelism, reliability and scalability. Both Cao et al. and Matarić suggest that commonly multi-robot systems incorporate ideas from both centralised and decentralised control.

2.3.6 Population

A group of robots, commonly referred to as a swarm, can be described as being homogeneous or heterogeneous. A homogeneous swarm, as defined by Matarić (1993), is one in which all the individual robots are similar, both in terms of their abilities and their goals. A heterogeneous swarm, according to Cao et al. (1997), is a group of robots that cannot be described as homogeneous. A useful property of a homogeneous swarm, pointed out by Matarić, is that because all the agents have innate knowledge of their peers, provided they are able to detect fellow robots then they are able to predict (perhaps implicitly) each others behaviours. Homogeneity also introduces redundancy and hence reliability to the system. A heterogeneous swarm, due to its wider range of abilities, can potentially be used to complete more complex tasks, however they are also more difficult to implement.

The size of a swarm is also important, in the taxonomy of Dudek et al. (1996), four group sizes are defined: SIZE-ALONE, a single robot; SIZE-PAIR, two robots, the simplest possible group; SIZE-LIM, more than two robots but limited by the size of the task; and SIZE-INF any number of robots, not limited by the size of the task.

2.3.7 Interaction

The way in which robots interact and communicate has a strong influence on the types of behaviour and cooperation they are capable of. Three major types of interaction are characterised by Cao et al. (1997): interaction via the environment, interaction via sensing and interaction via communication.

Interaction via the environment: the simplest form of interaction. Robots do not directly interact with each other but their actions may cause changes in the environment that affect the behaviour of other robots. Since robots are not able to identify other members of the swarm, the cooperation that can arise from this type of interaction is very limited.

Interaction via sensing: a step up from interaction via the environment. Robots do not send messages to each other but they are able to distinguish between fellow robots and the other objects in the environment.

Interaction via communication: this form of interaction involves the explicit passing of messages between robots. Allowing robots to share information about the world with other robots, this can lead to some of the most complex forms of cooperation.

Dudek et al. (1996) provide an extensive classification of communication, the range over which robots can communicate, the different types of communication and the communication bandwidth are all considered. The range of communication can be global i.e. encompassing the whole system, or local, covering only some predetermined area around a robot. Various types of communication are possible including private messages, sent only to a specific robot, or broadcasts, sent non-discriminantly to all robots within range. Communication can be expensive or inexpensive in terms of processing time.

2.3.8 Cooperation

The type of cooperation possible between the robots in a swarm is highly dependent on the previous three properties. If the method of control is fully centralised it is hard to describe the cooperation between individual robots. There is only one agent in control of a centralised system and a single agent cannot, by definition cooperate with itself, any cooperation observed between the robots of a centralised system is purely superficial. If a system is decentralised, several forms of cooperation can take place depending on the population and the interactions between individual robots.

Heterogeneous swarms can have a notion of hierarchy amongst the robots, this can lead to forms of cooperation whereby single or multiple “leaders” can coordinate the behaviours of other robots. The cooperation
of homogeneous swarms depends more on the types of interaction between robots. The size of a swarm can also affect what form of cooperation can take place.

The type of interaction between individual robots is perhaps the biggest factor in the cooperation of multiple robots. Interaction via the environment can only provide very simple forms of cooperation since robots cannot differentiate between other robots and the rest of the environment. Interaction via sensing is more useful, if a robot knows the locations of other robots then, as was shown by Reynolds (1999), it is able to cooperate with them to produce emergent behaviours such as flocking or dispersion. Interaction via communication is the most powerful form of interaction due to the fact that robots can share knowledge with their kin.

### 2.3.9 Summary

Multi-robot systems can provide several advantages over single robot systems in terms of efficiency, reliability, extensibility, flexibility and cost. To reap the benefits of multi-robot systems they must be carefully designed so as to minimise inter-robot interference and to maximise inter-robot cooperation. Shell and Matarić (2006) showed one way of reducing interference between robots through the use of a bucket-brigading strategy, which performed best with a large group of robots. Some of the properties that may be useful to consider during the design of a multi-robot system include: control, population, interaction and cooperation. The control of a multi-robot system can be centralised or decentralised. A population of multiple robots can be heterogeneous or homogeneous and contain many or few individual agents. There are various ways in which robots can interact, which in combination with the control and population of a system can lead to various different forms of cooperation.

### 2.4 Reactive Control Systems

The number of different types of system that can be described as “reactive control systems” is fairly large, it is possible to categorise this group further, within this project two different categories are defined: purely reactive systems and behaviour based reactive systems. A reactive control system is considered purely reactive if all of its actions are directly linked to specific conditions (often by a simple look-up table) and it retains no sensory information from past observations (Arkin, 1998). Purely reactive systems are also known as reflexive systems. Behaviour-based systems are often confused with reflexive systems but the two are fundamentally different (Matarić, 1992a). Neither behaviour-based nor reflexive systems hold a persistent world model, but importantly, within behaviour-based systems past observations can have an influence on future actions and the mapping between perception and action is not so restricted. As the system implemented for this project is behaviour-based, reflexive systems are considered no further, from this point on the term “reactive system” will be used synonymously with the term “behaviour-based reactive system”.

The principles of behaviour-based robotics were introduced in section 2.1, shortly some of the general advantages and disadvantages of such systems are discussed, following which some specific behaviour based architectures are reviewed, this section then closes with a comparison of these specific architectures.

#### 2.4.1 Advantages

One of the main driving forces behind the creation of behaviour-based robotics was the desire to create robots that could function well in the real world, consequently one of the main advantages of behaviour-based systems is their ability to cope with real dynamic environments (Brooks, 1991).

Environments involving multiple robots become increasingly hard to describe as the number of robots and interactions between them increases. Though the environment in which they reside is often complex and highly dynamic, robots controlled by reactive systems do not need to maintain a world model and only observe their local environment, because of this, as Matarić (1997) points out, behaviour-based systems scale very well as the number of agents increases. The absence of a world model and the fact that robots need to carry out very little computation between perception and action also means that they are extremely computationally efficient (Matarić, 1997).

In some people’s eyes an advantage, in others simply a feature, is the inspiration that reactive-systems have taken from nature. As mentioned by Arkin (1998) the study of neurology, psychology and ethology have all led to interesting discoveries that have helped advance the field of behaviour-based robotics. The study of neurology has led to the construction of computer models such as Artificial Neural Networks (ANN). ANNs have been used successfully on a number of occasions as a method of robot control, this project itself is strongly reliant on them. The study of psychology has led to better understanding of behaviours, in particular the connections between perception, action and the environment. The study of ethology, that is the study of animal’s behaviour in their natural environment, has led to better classification of behaviours.
2.4.2 Disadvantages

Some early criticisms of behaviour-based approaches were that they were not general enough and there existed a wide range of problems for which they were not applicable. Brooks answered some of the early criticisms of behaviour-based systems the paper “Elephants Don’t Play Chess” (Brooks, 1990b). Brooks argues that it is unfair to judge a system outside of its normal mode of operation. You would not expect a chess computer to assemble parts in a factory and neither would you expect an elephant to play chess, this does not signify lack of intelligence on the part of the elephant or the computer. Although behaviour-based systems are often billed as general models of intelligence Brooks also argues that like classical approaches they should be allowed time to reach their eventual goals. That said, Brooks’ paper was published nearly twenty years ago and some critics may still harbour similar criticisms to date.

With regard to reactive multi-robot systems Mataric (1997) warns of the global consequences of the local interactions between agents and whether such consequences are always predictable or desirable. This point is backed up by Sycara (1998) who adds the suggestion that because of the complex interactions between robots and the environment it is hard to engineer them to fulfil specific tasks. Both the problems mentioned by Mataric and Sycara can be reduced to a certain extent if the system is carefully designed. Mataric (1997) mentions some further criticisms of behaviour-based systems, firstly that due to their limited representation of the world they lack run-time flexibility, and second that they lack structure and rigorous definitions.

2.4.3 Neural Endocrine Control Architecture

Since the neural endocrine control architecture was first devised by Neal and Timmis (2003) it has been applied to various different situations and has spawned many variants on its original basic concepts, this subsection looks at some of the previous work in order to determine the best approach for this and future work.

Discussion & review

The initial work (Neal and Timmis, 2003) involved a few simple behaviours, namely: wandering and obstacle avoidance, soon work moved on to include seeking behaviour (Neal and Timmis, 2005), the same work also started to show the neural endocrine control architectures ability at action selection, with experiments involving a robot switching between the desire to move towards a white placard and a black placard.

Vargas et al. (2005) introduced the concept of feedback mechanisms to the artificial endocrine system, increasing the control over hormone release, as well as making the architecture more biologically plausible. Vargas et al. also points out the adaptability of the architecture; for a collective robotic system with multiple agents constantly effecting their surroundings, the ability to adapt to a dynamic environment is clearly a desirable property. Timmis et al. (2009) take adaptability a step further and add to the architecture the capacity for “on-line” learning. On-line learning provides robots with the ability to learn how to react to the environment throughout their normal operation, rather than exclusively before being deployed thus increasing the adaptability of the architecture over long periods of time. Although on-line learning would be extremely useful within a multi-robot system, its implementation goes beyond the scope of this project; furthermore it was expected that the robots in this project would be deployed for short periods of time and therefore might not be able to take full advantage of such a system.

Bott (2006) implemented a neural endocrine architecture to carry out a simple foraging task, this involved: collecting rubbish, disposing of rubbish and recharging. Four behaviours were implemented in total using a complicated combination of neural networks and glands. One of Bott’s main concerns was “task selection”, for example when should the robot stop looking for rubbish and focus on finding the bin. Based on the premise that the more rubbish a robot has picked up, the more likely it should be to seek the bin, Bott designed the bin-seeking gland so that it was influenced directly by the output of the rubbish-seeking gland. Whilst the basis of Bott’s approach is sound, the design is seemingly flawed. When in the presence of rubbish the bin-seeking gland would be effected as desired, increasing the chance of the robot finding the bin, however if there was no rubbish in sight then there would be no extra incentive for the robot to seek the bin, even if it was carrying a full load of rubbish, clearly not the desired effect in this case. A simpler approach would simply be to keep track of the amount of rubbish the robot had picked up and use this value to influence the bin-seeking gland.

Perhaps one of the most fully-featured applications of the neural endocrine control architecture to date is that of Lord (2007). Lord assessed the performace of the architecture at completing a non-trivial task. Like Bott the task chosen was foraging, though Lord’s task and implementation were slightly more complex. Seven behaviours were implemented in total, five of which were controlled by neural endocrine networks, the other two were fixed action patterns and reacted purely to specific input conditions. Lord implemented his system in Java and consequently was able to design a multi-threaded implementation where all the behaviours executed in parallel. Lord tested the effectiveness of including a negative feedback mechanism within the hormone glands, which was shown to have a noticeable improvement on performance. The architecture was developed
Figure 2.5: The classical horizontal decomposition of a robot control system is shown in (a). The vertical alternative, as proposed by Brooks (1986) is shown in (b).

for a relatively simple environment, in order to investigate its adaptability it was tested in both the simple environment and a more challenging obstacle filled one. Unfortunately the robot did not cope as well as expected with the switch to the new environment, indicating that for a non-trivial task, the architecture (without on-line learning) is not as adaptable as originally thought. If the system had been developed with the more complicated environment in mind from the start it would no doubt have found the switch to a new (simpler) environment easier. Sometimes in the tougher obstacle filled environment the robot would crash into walls, when this occurred the robot was “nudged” to allow it to complete the task, in a multi-robot system this would not be necessary as the failure of one robot does not represent the failure of the swarm as a whole. Lord concluded that whilst more work still needed to be done, the architecture showed great promise as an action selector in a behaviour based robot control system.

Summary

The neural endocrine control architecture has been shown on a number of occasions to be an effective method of control for single robots carrying out both trivial (Neal and Timmis, 2003; Vargas et al., 2005) and non-trivial (Bott, 2006; Lord, 2007) tasks, there is little to suggest that the same will not be true for multiple robots. The introduction of more robots will produce a more dynamic environment than has previously been used, which raises the concerns highlighted by Lord (2007) over the adaptability of the system, especially without on-line learning, however new behaviours specifically designed for the interaction between robots will also be introduced which should help nullify any new problems. The use of feedback glands was proved effective by two authors (Lord, 2007; Vargas et al., 2005). A collaborative robot system is anticipated to include more behaviours than have previously been used due to the interaction between robots, this makes it even more important for the glands and neural networks to be designed carefully without overcomplicating them as was perhaps the case with Bott (2006).

2.4.4 Subsumption Architecture

When it was first proposed by Brooks (1986) the subsumption architecture was a drastic change from the popular methods of robot control at the time; instead of using a horizontal decomposition of control like that shown in figure 2.5a, Brooks proposed a vertical decomposition, similar to figure 2.5b.
Each of the layers in the decomposition represents a different level of competence and describes a desired class of behaviours. A complete system is the combination of multiple layers of behaviours, each of which is linked directly to both sensor inputs and actuator outputs. There is no stored world model, with each layer acting only on the information they receive from sensors.

Generally higher level behaviours are more abstract and goal orientated, whereas lower level behaviours are more concerned with the immediate survival of the robot. The vertical decomposition serves as a method of prioritising behaviours, higher layers have higher priorities and the ability to subsume lower layers via suppression and inhibition mechanisms. Each layer operates without regard for the layers above it and is able to subsume lower levels by feeding data into their interfaces, however they do not know when they themselves are being subsumed.

In Brooks (1986) each individual layer is made up of a collection of finite state machines (FSMs), referred to as modules, an example of such a module is depicted in figure 2.6. Each module has input, output and reset lines which are used to string modules together in a variety of ways, however usually there is always at least one module that receives inputs from sensors and at least one module that sends outputs to actuators. Output lines can be inhibited and input lines can be suppressed for set periods of time. Whilst an output line is being inhibited any signals leaving the associated module are lost. Whilst an input line is being suppressed, as well as inhibiting the input signals the suppression mechanism is able to replace them with new signals.

In order to make subsumption more accessible to those without experience in the specification of FSMs, Brooks (1990a) developed The Behaviour Language. The behaviour language abstracted the problem away from the specification of FSMs and allowed users to instead specify rule sets to encode behaviours (Arkin, 1998).

Discussion & review

Brooks (1986) showed a three layer subsumption architecture, encompassing: obstacle avoidance, wandering and exploration, to be an effective method of robot control in both simulation and the real world. Since the original work by Brooks many more subsumption architectures have been implemented, some of the early work is summarised in Brooks (1990b). The works most relevant to this projet are the multi-robot implementations of Matarić (1997, 1993, 1992b). Some of Matarić’s implementations are reviewed later in this subsection, before which proceeds a discussion on the subsumption architecture in general.

Because of the hierarchical nature of the subsumption architecture, implementations must be designed in a bottom-up fashion. Design starts with the lowest layer, and only after this layer has been fully implemented, tested and debugged can more layers be added (Brooks, 1986). If done properly this design process can lead to robust, extensible and well built systems, however the approach is not without its critics. Hartley and Pipitone (1991) implemented a subsumption architecture for the control of a simulated autonomous aircraft, their implementation was successful but outlined some problems with the architecture when applied to a “reasonably complex” problem. The main problem that Hartley and Pipitone mention is the lack of modularity, because of the interaction between layers Hartley and Pipitone found that the results of combined behaviours can be hard to predict, especially as the number of behaviours and the complexity of the problem increases. Hartley and Pipitone suggest that improvements could be made by formalising the interfaces between behaviours. Another disadvantage of the incremental design approach is that once a layer is finished it cannot be changed without affecting the rest of the system (Brooks, 1986).

In his original paper on subsumption Brooks (1986) identifies four requirements for the control system of an intelligent autonomous robot. After introducing the subsumption architecture Brooks goes on to explain how the four requirements are satisfied, his explanations are summarised below:

**Multiple goals**: a key feature of the subsumption architecture is its support for parallelism, by allowing all the layers to work concurrently it is effectively possible to pursue all goals at the same time. The overall global
was reduced.

The subsumption architecture was one of the first architectures to use a behaviour-based approach to robot control. Rather than using a horizontal decomposition of control, as was popular at the time, Brooks's subsumption architecture used a new vertical approach, characterised by its lack of a world model and the fact that all layers were closely linked to both perception and action.

Subsumption has been shown to be a successful form of robot control on a number of occasions. Subsumption receives some criticism for lack of modularity, particularly when it comes to making changes to existing implementations, or designing complex systems with many interacting behaviours. Despite its lack of modularity, because of the parallel nature of the architecture it is always possible to extend a system by adding new layers (provided they are at higher levels) with little or no loss in performance. Subsumption is also praised for its robustness and its ability to pursue multiple goals at the same time.

As well as performing well with single agent systems, subsumption has been shown to be a viable method of control for multi-agent systems. Various methods of cooperation have been shown for multi-agent systems. The experiments of Matarić (1992b) showed interaction via sensing to be the most appropriate form of interaction for a simple multi-robot system. Matarić (1993) investigated the performance of homogeneous and heterogeneous populations but could not show either to definitively outperform the other at a simple task.

In Matarić (1992b) a variation on Brooks's original subsumption architecture was used to control a population of twenty homogeneous robots, each of which was independently controlled in a decentralised manner. The task of the robots was to head towards and remain in a specified area, a behaviour also known as “homing”. Three different cooperation strategies were investigated, differing in terms of the way the robots interacted with each other and the behaviours they possessed. The first and simplest cooperation strategy was referred to as ignorant coexistence, in this strategy the robots interacted via the environment, viewing each other simply as obstacles to avoid. The second strategy was known as informed coexistence, in which the robots interacted by sensing each other. If a robot detected another member of the swarm it would stop and wait for a set period of time, if after this period of time the detected robot had moved it would continue moving forward, otherwise it would change direction and move away. The third and final strategy was known as intelligent coexistence, the robots interacted in the same way as they did with informed coexistence, but their behaviours were more intelligent. Rather than simply stopping and waiting for other robots, individual behaviours were chosen so that global flocking behaviour would emerge between members of the swarm.

The results of Matarić's experiments showed that the robots performed best when using the strategies that involved interaction via sensing. The intelligent coexistence strategy showed the best performance speedup relative to the number of agents, followed by informed and lastly ignorant coexistence. The increased performance was due to the fact that as the methods of cooperation improved the interference between robots was reduced.

Matarić (1993) used similar robots and a similar control architecture to that seen in Matarić (1992b), however the more recent work was done entirely in simulation. The experiments of Matarić (1993) involved a homogeneous and a heterogeneous group of robots each capable of the same set of emergent behaviours, including: aggregation, dispersion, flocking and homing. The heterogeneous group were characterised by the fact that there was a dominance hierarchy amongst the robots.

The performance of the two groups at carrying out an aggregation and a dispersion task were measured. The homogeneous group was shown to perform better at the dispersion task whereas the heterogeneous group was shown to perform better at the aggregation task. Matarić states that although the differences in the performance of the two groups were consistent with repeated runs, they were not very large and on physical robots may be negligible. Matarić concludes that a dominance hierarchy for dispersion and aggregation based tasks is neither necessary nor helpful.

Summary

The subsumption architecture was one of the first architectures to use a behaviour-based approach to robot control. Rather than using a horizontal decomposition of control, as was popular at the time, Brooks's subsumption architecture used a new vertical approach, characterised by its lack of a world model and the fact that all layers were closely linked to both perception and action.

Robustness: the use of multiple sensors can provide a certain amount of robustness to a system. The layered structure of the subsumption architecture provides a further source of robustness. Higher levels only interfere with the inputs and outputs of lower levels and not the modules themselves. If a high level behaviour was to fail then the lower level behaviours would still continue to work and the robot would continue to function, though at a lower level of competence.

Extensibility: Each new layer can be run on its own processor or set of processors, as the communication between different layers is cheap, the addition of a new layer should have little effect on the performance of the other parts of the of the system.

Multiple sensors: Different layers may use different sensors and do so independently of how other layers are using them.

Robustness:

Extensibility:

Multiple sensors:

Summary
2.4.5 Motor schema

Motor schema take their name from the components for which they are designed to control (robot motors) and the principles of schema-theory upon which they are based. Motor schema and schema-theory have links to the fields of both philosophy and psychology. As far back as the eighteenth century philosophers have used schemas to model behaviour (Arkin, 1998) and more recently psychologists have used schemas to help link perception and action (Neisser, 1976).

There are several variations on motor schema, this project focuses on those used by Arkin and Balch. Arkin (1998) defines a schema to be a unit of behaviour that can be combined with others to produce complex actions. Contained within a single schema is all the information on how it should perceive or act, along with the computational process by which it can do so. A complete motor schema system consists of several individual schemas that execute concurrently, each schema represents a single primitive behaviour and the combination of multiple schemas leads to the desired global behaviour of the system.

Like other behaviour-based architectures, motor schema rely on a tight coupling between perception and action. Each schema receives stimuli through a subset of the sensors available to the complete system. When the strength of a stimulus is above some predetermined activation level a response is initiated, the strength of this response is usually related to the strength of the stimulus. The response of an individual motor schema is represented by a vector, which corresponds to the movement of the robot suggested by that particular schema. Based on the individual vector responses of each schema, a cooperative form of coordination is used in order to determine the global behaviour of the system. Each schema has a gain \( g \) associated with it which corresponds to the amount of influence it has over the system. The response vectors of each schema \( r_i \) are multiplied by their gain and summed in order to determine the robot’s overall response \( R \), given formally in (2.10) where \( s \) is the total number of motor schemas in the system.

\[
R = \sum_{i=0}^{s} r_i \cdot g_i
\]  

(2.10)

A very simple motor schema architecture, adapted from Arkin (1998) is shown in figure 2.7. This example has two motor schemas, each of which contains a number of perceptual schemas. Perceptual schemas are used to map the input from the environmental sensors into stimuli which the motor schemas can respond to, as is seen in figure 2.7 it is possible for perceptual schema to be recursively defined.

Discussion and review

Some of the main advantages of motor schema are outlined in Arkin (1987, 1998), the first of which is support for modularity; in most cases there is little communication or dependence between different schemas, meaning that they can be designed, implemented, tested and debugged in isolation, although at some stage schemas must be combined and tested together in order to ensure they produce the desired behaviour. The fact that different schemas can be processed concurrently is also advantageous, this leads to improved real-time performance, ensuring that perceptions are always up to date and adding support to the close mapping between perception
and action. Because there is no real concept of hierarchy amongst different behaviours it is easy to add in new
behaviours or change existing ones without regard for the order in which they are implemented.

Using motor schema, Balch and Arkin (1995) implemented a behaviour based control system for multi-robot
teams. The work of Balch and Arkin in the simulation, however, the eventual anticipated use was as for real
life military scout robots. Much of the later work of Balch and Arkin uses a hybrid motor schema architecture, it
is expected that this was always the intended purpose of Balch and Arkin (1995) however at the time the system
contained no deliberative features and so is suitable for discussion in this section. Some of Balch and Arkin’s
hybrid work is discussed in section 2.6.

The task of Balch and Arkin’s robots was to simultaneously move to a goal location, avoid obstacles, avoid
colliding with each other and maintain a set formation. The use of formations was partially inspired by their
existence in nature, the flocking of birds is one such example. Four different formations of four robots were
investigated, namely: line, column, diamond and wedge (a regular trapezium shaped formation).

A decentralised form of control was used, each robot running its own individual program. There was a
central system that when requested, relayed information about the world to individual robots, however this
was only used to simulate what could be perceived by a real robot with sensors, that is to say robots were
only told about objects that were within sensible sensor ranges. Each of the robots had a unique ID which
determined their specific location within each formation, in this sense they were a heterogeneous population.
Three types of interaction were investigated, all of which can be classified as interaction via sensing. The first
type of interaction was unit-centre-referenced where each robot determined where it should be from the average
position of the whole group, the second type of interaction was leader-reference in which each robot determined
where it should be in relation to a single “leader” robot and the final type of interaction was neighbour-referenced
which followed the same principle as leader-referenced except that each robot had a different leader.

All combinations of formation and cooperation successfully completed a simple obstacle avoidance and goal
seeking task. The results showed that on the whole the robots performed best when using the unit-centre-
referred method of interaction, however there were some situations, such as environments densely populated
with obstacles that benefited from having one leader that all robots followed.

**Summary**

A motor schema system is made up from a number of individual schemas, perceptual and motors schemas
combine to produce individual behavioural units, which in turn can be combined using a cooperative form of
coordination, to produce more complex behaviours. Support for modularity is one of the major advantages
of motor schemas, allowing for schemas to be designed and added to the system in any order, the fact that
schemas can be processed concurrently is also advantageous. Balch and Arkin (1995) showed that multi-robot
teams, controlled using motor schema, could successfully complete a simple task whilst remaining in formation.

Three types of interaction were investigated, the most successful being the case in which all robots were treated
as equal.

**2.4.6 Comparisons**

In this subsection the three types of reactive architectures that have been introduced: neural endocrine,
subsumption and motor schema, are compared and contrasted with each other. The main points for comparison
are how the behaviours are encapsulated in each architecture, how responses are encoded and how multiple
behaviours are coordinated.

All three architectures include the notion of a behaviour, though they are represented in three very different
ways. The subsumption architecture represents behaviours as individual layers made up of connected finite
state machines, whereas the neural endocrine architecture uses artificial neural networks and artificial endocrine
systems to represent behaviours. Motor schema use another method, whereby each behaviour is constructed
from a collection of perceptual and motor schemas.

The way in which behaviours are coordinated differs between the three architectures. Subsumption takes a
competitive approach to coordination allowing higher level behaviours to suppress or inhibit the inputs and
outputs of the lower layers. Motor schema and the neural endocrine architecture take a similar cooperative
approach, in motor schema each behaviour has an associated gain and in neural endocrine each of the neurons
in a behaviours network can be associated with a number of hormones. The value of a schemas gain and
the strength of a neurons hormone are both used to effect the eventual outcome of a behaviour. Multiple
behaviours in both the neural endocrine and motor schema architectures are coordinated by simply summing
their individual responses.

The encoding of responses is not architecture specific, all three have the ability to encode them in a number of
ways. The most popular method for encoding responses is that which allows the strength of a stimulus to have
an effect on the strength of the response.
The design of systems differs between the three architectures, once again motor schema and neural endocrine take a similar approach, allowing behaviours to be designed and implemented in a modular fashion. Behaviours in the neural endocrine and motor schema architectures can be designed in any order at the discretion of the designer, without having to worry too much about how other behaviours will be effected. Motor schema and neural endocrine both have a similar notion of strength, it is these values and only these values may need to be adjusted when new behaviours are added to a system. Subsumption takes a different approach from the other two architectures requiring that systems are designed in a bottom-up fashion, one layer at a time.

Finally, as Lord (2007) showed for neural endocrine, Brooks (1986) showed for subsumption and Arkin (1987) showed for motor schema, all three architectures all well suited to concurrent execution.

2.5 Deliberative Reasoning Systems

Deliberative systems are often described as the “classical” approach to robot control. In the mid-1980s deliberative reasoning systems dominated the field of robotics (Arkin, 1998), however since the criticisms of Brooks (1986) and the introduction of the subsumption architecture they have been forced to share the limelight with reactive and hybrid systems. A major feature of deliberative reasoning systems is their heavy reliance on symbolic knowledge, which is usually encompassed in some form of world model. Within deliberative systems perception is not tied directly to action as it is within reactive systems, instead actions are the result of extensive reasoning over the world model. This section begins by discussing some of the main advantages and disadvantages of deliberative systems, before introducing a specific architecture and describing an example of how this architecture has been used to control a multi-robot system.

2.5.1 Advantages

In situations where it is possible to accurately model the world over an extended period of time, such as in largely static environments, deliberative reasoning systems can be highly effective (Arkin, 1998). The reason for the success of deliberative systems in static environments can be attributed to their high-level of artificial intelligence and their capacity to accurately predict and plan for the future.

2.5.2 Disadvantages

As already mentioned, reactive control systems first arose from the desire to create systems that worked well in real dynamic environments (Brooks, 1991), the fact that deliberative reasoning systems cannot always achieve this is considered their the first major disadvantage. The reason that deliberative systems find it hard to function in dynamic environments arises from their necessity to maintain a world model. In order for a deliberative system to successfully reason over and draw the right conclusions from a world model, the model must be reliable and consistent with the real world (Arkin, 1998). Maintaining a reliable world model is especially challenging in a dynamic environment as it requires that the model is updated more frequently. The effects of errors from sensors are also more prominent in a dynamic environment since multiple readings cannot be fused together over long periods of time, as would be possible in a static environment.

Because of the amount of reasoning performed by deliberative systems they are computationally quite complex Arkin (1998) and hence they require powerful hardware in order to perform well in real-time, they also require high-quality sensors to ensure reliability in the world model. The need for powerful processing hardware and high-quality sensors means that deliberative reasoning systems are generally quite expensive (Brooks, 1987).

Deliberative systems are not particularly well suited to multi-robot systems. As pointed out by Matarić (1992a) the state space of a multi-robot system increases exponentially, due to the fact that all of the robots in a system interact with each other over extended periods of time, for a planning approach this would be impractical.

Another disadvantage, pointed out by Brooks (1986) is that almost the entire system needs to be constructed before any sort of testing can be carried out. As described shortly deliberative reasoning systems are often arranged in hierarchical levels, which communicate with each other in a predetermined manner, consequently all of the levels need to be developed before a system can be used. To add a new level in at later stage would be extremely difficult, even to make changes to an existing one would be challenging, requiring the designer to do so in such a way that the interfaces to other levels are not effected, or that any changes are also taken into account in other levels.

2.5.3 Hierarchical Control System

Many types of deliberative reasoning system can also be described as hierarchical, the general system proposed by Albus (1991) is one example. In Albus’s system, control is split into a series of levels, each level can interact
with the levels above and below it and only the bottom layer communicates with the environment (figure 2.8). Tasks are decomposed across the different levels, with the higher levels providing subgoals for the lower levels, right down until the bottom layer where the actuator commands are issued. Further down the hierarchy as the scope of the goals decreases, so too does the size of the plans and the duration of the timing requirements.

The four main components, or “modules”, of figure 2.8, from Albus (1991): sensory processing, world modelling, behaviour generation and value judgement can be summarised as follows:

**Behaviour generation:** selects goals and creates plans that are sent to the world modelling module for analysis. Receives recommendations about previously created plans and from them decides which plans should be executed.

**World modelling:** contains a symbolic representation of the robots environment, also known as a world model, using this representation the world modelling module predicts the outcomes of specific plans provided by the behaviour generation module. It is extremely important that the world model is reliable and complete, this is ensured by its interaction with the sensory processing module.

**Sensory processing:** compares sensor observations with predictions generated by the world modelling module. Over time the sensory processing module helps to build up a reliable world model through the fusion of multiple sensor readings and the expectations of the world modelling module.

**Value judgement** determines the quality of the current world model through communication with the world modelling and sensory processing module. The value judgement module also evaluates the results of the plan analysis and suggests the best course of action to the behaviour generation module.

**Discussion and review**

The Multiple Autonomous Underwater Vehicles (MAUV) project (Albus and Blidberg, 1987; Herman and Albus, 1988) is one example of where a hierarchical system has been used for robot control. The project used the NASREM control architecture (Albus et al., 1987), a similar system to that proposed by Albus (1991), except that the “value judgement” and “behaviour generation” modules are replaced with a single component known as “task decomposition”.

The MAUV project investigated the control of two underwater vehicles operating within two simple scenarios. The first of the scenarios was a cooperative search and approach task, requiring the robots to exhibit behaviours such as fly-in-formation and follow-the-leader. The second scenario was a cooperative search and map task, the area to be mapped could potentially be considered hostile and so as well as map the environment the robots were expected to use features of the environment such as ridges and gullies to avoid detection.

The control of the robots was decentralised, that is to say each robot was independently controlled. The deliberative control system that was ran on each robot used a six level hierarchy, consisting of, from the highest to the lowest: a mission level, a group level, an vehicle level, an elementary move level, a primitive level and a servo level. The mission level decomposes a mission, such as “search and map”, into a sequence of subtasks that are passed to the group level. Group tasks define the actions of a group of multiple robots, these are decomposed at the group level into vehicle commands which are then passed to the vehicle level. The vehicle level splits their commands into a sequences of elementary moves and passes them to the level below. The process of
decomposition continues through the remaining three levels until at the servo level commands are sent to the actuators of the robots.

Herman and Albus (1988) discuss a couple of ways in which the robots from the MAUV project could cooperate their behaviours. By maintaining identical world models with respect to the significant properties of the world such as the positions of landmarks and other robots, Herman and Albus’s robots were able to achieve cooperative behaviour. Maintaining identical world models though, is an expensive process, requiring robots to frequently communicate with each other when new information is discovered, this approach would clearly not be scalable. To address the issues of scalability Herman and Albus proposed an alternative technique where one robot is assigned as mission leader, responsible for performing mission level planning, and each group is designated a group leader who receives plans from the mission leader, and communicates further plans to all the members of their group. This new method is flawed though as increased scalability comes at the cost of redundancy and robustness. Since the control is distributed amongst robots single points of failure are introduced at both the mission and group levels, this problem could be alleviated somewhat if robots were able to spot when leader had stopped functioning and assume their responsibilities, whether this is possible though is not discussed by Herman and Albus (1988).

Summary

Deliberative reasoning systems are often described as hierarchical, in Albus’s system each layer was made up from a combination of different components, namely: behaviour generation, world modelling, sensory processing and value judgement. Tasks are decomposed across levels, the further down the hierarchy, the smaller the tasks, until at the bottom level commands are issued to actuators. Deliberative systems have several disadvantages including the computational complexity, monetary cost, the fact that they do not work well in dynamic environments and the fact that almost the entire system needs to be constructed before testing can be done. With regard to multi-robot control Matarić (1992a) pointed out some of the problems with the scaling of the deliberative approach and the work of Albus and Blidberg (1987) confirmed these problems.

2.6 Hybrid Deliberative-Reactive Systems

Deliberative systems can perform well in static environments, for such conditions, due to their ability to predict and plan for the future, they are often preferred to reactive systems (Arkin, 1998), however they do not cope well in dynamic environments. Take for instance a navigational path planning problem, as pointed out by Arkin (1989) to successfully complete the task a robot must be able to respond rapidly and correctly when faced with changes in the world. In order to respond appropriately a deliberative system might attempt to plan for every possible situation, but this would take far too long especially in a highly dynamic environment. Reactive systems on the other hand, are computationally efficient and function very well within real dynamic environments, they would have little trouble rapidly responding to a changes in the world, however they have been criticised for lack of run-time flexibility, and according to Arkin (1989) lack sufficient cognitive ability. Hybrid architectures provide the ability to combine the advantages of both reactive and deliberative systems, as well as hopefully losing some of the disadvantages.

Arkin is one of the main proponents of hybrid systems, in this section Arkin’s hybrid AuRA architecture is introduced and briefly discussed, an example is also given of it in use as a multi-robot control system.

2.6.1 AuRA

The Autonomous Robot Architecture (AuRA) (Arkin and Balch, 1997) is a hybrid reactive-deliberative robot control architecture, originally created by Arkin and Hanson (1987). The architecture can be split into two parts as shown by figure 2.9: a hierarchical planner, based on the classical deliberative approach to robot control; and a reactive controller, based on motor schema control systems like those discussed in section 2.4.5. The hierarchical planner consists of three layers, the purposes of each are summarised below:

**Mission planner** Creates high level goals for the robots

**Spatial reasoner** Uses map data to create sequences of path segments

**Plan sequencer** Creates a sequence of motor schemas and sends them to the schema controller

After the sequencer has sent the schema to the controller, the hierarchical part of the system ceases to function and the reactive part takes full control. Using the schema provided the schema controller attempts to complete the mission. If something goes wrong and a problem is detected, the hierarchical layers are re-initiated one-by-one, starting with the plan sequencer, once the problem has been resolved the hierarchical component hands control back to the reactive component.
Discussion and review

A particular feature of AuRA that bears extra relevance to this project is the inclusion of a homeostatic control system (Arkin, 1992). Much like external and internal properties of the neural endocrine system can effect the concentration of particular hormones; by monitoring internal properties of a robot, such as temperature and fuel level, Arkin’s homeostatic system can affect the strength (gain) and properties of different motors schemas.

Arkin and Balch (1997) state some of the major advantages of AuRA to be: modularity, flexibility and generalisability. Modularity comes from the ease with which different components, such as the layers of the hierarchical planner, can be swapped in and out of the system, flexibility is added through adaptation and learning methods which were not discussed here and generalisability, according to Arkin and Balch, is evident with the wide range of applications and situations AuRA has been applied to.

There has been much research into the control of robots using AuRA, however most applications have not fully integrated the planning capabilities of the system (Arkin and Balch, 1997), the authors instead having chosen to develop simpler reactive systems, similar to the motor schema implementation described in section 2.4.5. Rather than have robots generate plans themselves, some authors have generate them “by hand”, which was the approach taken by Arkin and Balch when programming a group of three homogeneous robots to complete a rubbish-collecting task, the system is described in Arkin and Balch (1997). The implementation included no reasoning or on-line planning, one plan was created by hand and provided to the plan sequencer at the start. The robots were very successful at the task even though they had no form of cooperation and interacted purely by the environment.

Summary

Hybrid deliberative-reactive systems were created to take advantage of some of the best properties of reactive and deliberative systems, and at the same time attempt to remove some of the worst properties. AuRA is one example of a hybrid deliberative-reactive control system, it consists of two parts, a high level hierarchical planner and a low level reactive controller. Some work has also been done into integrating a homeostatic control system into the architecture. Despite having both a hierarchical and reactive component most authors to date have focused almost exclusively on the reactive part.
3 Problem Analysis

This chapter starts by introducing the task of foraging and describing the specific variant used by this project. The simulation software used in this project is then introduced, following which there is a discussion on the different types of hardware components that a group of mobile robots could utilise, mentioning how these different components would be useful for a foraging task. As was noted in section 2.3 there are a four characteristics that can be used to describe multi-robot systems: the method of control, the type of population, the interaction between robots and the cooperation of the group, each of these properties is analysed in turn with relation to the foraging task and the neural endocrine control architecture. Towards the end of this chapter the different behaviours that a swarm of robots requires to complete the foraging task are listed and analysed, and finally a list of requirements for this project is presented.

3.1 Foraging Task

Foraging is a popular task for mobile autonomous robots, both individual robots and swarms have been shown to successfully complete various types of foraging problem. The basic principles of foraging involve an agent collecting objects that are spread throughout the environment and returning them to some specified location. The task is completed once all of the objects in the environment have been collected.

There are several reasons for using a foraging task to investigate multi-robot systems. From a research point of view foraging is an appropriate choice because it is capable of showing the potential of a system at complex collaborative behaviour, without tying it down to one specific application. The fact that the performance of a system is easy to quantify is also advantageous and several possible measures exist for this purpose. The time taken to complete a task, the amount of energy used, the efficiency of the robots or simply whether or not the task was completed can all be used to measure the performance of a system. Further support for foraging comes from the fact that its principles can be applied to a number of real-world situations, including for example: the cleanup of hazardous waste, the clearing of minefields and the rescue or assistance of survivors at disaster sites.

The variant of foraging that was chosen for this project is known as rubbish or garbage collection. The task of rubbish collection used here involves a group of robots collecting pieces of rubbish that are randomly distributed throughout the environment and returning them to a bin. In order to make the task slightly more complex and to model the real world closer robots are required to monitor their power levels and when they are running low find a charging station at which to recharge.

Using neural endocrine control systems, foraging tasks have already been completed by single robots (Bott, 2006; Lord, 2007), as well as determining whether multiple robots can complete the task, this project focuses on whether they can provide a performance increase and how such an increase can be measured.

3.2 Simulation

This project was carried out entirely in simulation using the tools provided by the Player/Stage Project (Player, 2009). The development of robotic systems in simulation has several advantages over the development of similar systems in the real world, including: lower costs, capacity for rapid prototyping, faster than real-time execution, automation of execution and ease of data collection.

Brooks (1987, 1990b, 1991) has frequently warned about the problems of developing robotic systems in simulation. Brooks’s main criticisms stem from the fact that without feedback from real experiments it is possible to waste time solving problems that are simply not applicable in the real world. As Brooks states further problems can arise from the temptation to use idealised sensors, or the accidental use of unrealistic sensors, caused by insufficient knowledge of the amount of noise that exists in the real world. The use of idealised or unrealistic sensors, whatever the cause, can lead to the development of robotic systems that may work well in simulation but could fail completely when deployed in the real world.

Brooks does not disregard the use of simulation entirely, in fact there is one situation for which he states simulation to be the most obvious choice, this being the case where a large number of trials needs to be performed, such as when genetically evolving control programs (Brooks, 1992).

Torrance (1992) advocates the use of simulation but stresses the importance of well designed simulators and the responsibility of the user not to misuse them. Torrance acknowledges the problems pointed out by Brooks...
as some of the major inspirations for well designed simulators. One of the advantages pointed out by Torrance that has not yet been mentioned is the ease with which users can share code and robot designs, to a much higher degree than is possible with real robots. Simulation also allows experimental results to be replicated and verified by others easier, this was one of the driving forces behind the creation of the Player/Stage Project (Gerkey et al., 2003).

Since the criticisms were made by Brooks there has been much improvement in the quality of simulation software, whilst the Player/Stage Project does not provide the most realistic simulation software to date, it is sufficient for the task at hand. Due to the time constraints of this project the advantages of using simulation are deemed to outweigh any disadvantages.

3.2.1 The Player/Stage Project

The Player Project has two main parts, the Player robot server and the Stage simulation backend, both of which are described below. All of the software from the Player Project is free and released under the GNU Public License (GNU, 2007).

Player

Player is a robot server that can run on both simulated and real hardware. Player is designed so that robots (simulated or real) provide an abstract interface to their controls. It is possible for simulated robots to provide the same interface as real robots and so in theory a client program written for simulated hardware should have no problems running a real version of the hardware.

Stage

Stage simulates populations of mobile robots and the environments in which they reside. Environments are simulated and visualised in two-dimensions, figure 3.1 shows one of the environments created for this project. Stage can be used as a plugin to Player, which is the approach taken in this project. Stage was designed with research into multi-agent systems in mind, thus adding further support to the decision to use it in this project.

3.2.2 Hardware

When designing robots in simulation it is easy to be seduced by the abundance of different components and configurations available. Although the ability to design systems that would otherwise be too expensive is one of the major advantages of implementing robots in simulation, care must be taken so that the designs are still viable in the real world.

The design of systems that are reproducible in the real world becomes even more important when dealing with multi-robot systems since any costs will need to be multiplied by the number of robots used. Multi-robot systems commonly involve many small simple robots with little processing power and limited energy capacity, consequently they require both low-powered and computationally inexpensive components. Some people would argue it is OK to design expensive or unfeasible systems in simulation, as in the future the cost of components will decrease and quality of technology will improve, this is not the approach taken by this project and the robots are designed with potential real-world deployment in mind.

The type of the robots simulated in this project are MobileRobots’ Pioneer (MobileRobots, 2007), these were chosen as they are the most widely supported robot in the Player/Stage Project. The hardware components of robots can be split into two categories, sensors and actuators, some of the possible components that are available to Pioneer robots are categorised below, as well as being described in general, the ways in which they would be useful in a rubbish collection task are analysed.

Sensors

Laser  A laser device would be desirable to accurately determine the position of obstacles within the environment, however lasers are computationally expensive and require a large amount of power, not to mention their cost from a monetary point of view.

Sonar  Sonar devices have the same function as lasers, allowing robots to determine the positions of obstacles, they are less accurate than lasers and a single robot would require multiple devices to ensure a sufficient coverage of the area, however they are much cheaper and consume less power than lasers.
### Table 3.1: comparing various properties of three different range-finding devices

<table>
<thead>
<tr>
<th>Device</th>
<th>Power consumption (W)</th>
<th>Range (cm)</th>
<th>Field of view (rad)</th>
<th>Cost ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SICK LMS-200</td>
<td>20</td>
<td>0 – 8000</td>
<td>$\pi$</td>
<td>&gt; 5000</td>
</tr>
<tr>
<td>SensComp 600</td>
<td></td>
<td>15 – 1000</td>
<td>$\frac{\pi}{2}$</td>
<td>&lt; 30</td>
</tr>
<tr>
<td>SHARP GP2Y0A710KoF</td>
<td>0.15</td>
<td>100 – 550</td>
<td>n/a</td>
<td>&lt; 15</td>
</tr>
</tbody>
</table>

Infrared  Infrared, like sonar and laser can be used as a form of range-finder, they are not as accurate as sonar or laser and work over much smaller distances, however they are very cheap, consume little power and are not computationally intensive. Like sonar, multiple devices are required to ensure sufficient coverage.

Camera  A camera would be desirable in a foraging task so that robots are able to tell the difference between the different objects in the environment. The power consumption and processing needs of camera devices are determined by factors such as: their range, their field of view, the quality of their images and the image processing software they use.

Light sensors  Cameras are generally both quite expensive and quite large, if space or monetary constraints refutes the possibility of a camera device, robots would at least require a component such as a light sensor to allow them to distinguish between their fellow robots and other objects.

Global Positioning System  A Global Positioning System (GPS) device may be useful in helping the robots find there way back to the bin, however they are very expensive and require a large amount of power.

Bumpers  Bumpers would be desirable on a robot, in the event that a robot crashed into an obstacle, any damage would be minimised by the presence of bumpers. Bumpers would also help a robot to determine what actions to take to get away from any obstacles it crashed into.

As the component for which there are the most options, much thought must be put into deciding which range-finding device is the most appropriate for the task at hand. Table 3.1 shows a side-by-side comparison of three specific range-finding devices, the SHARP GP2Y0A710KoF (SHARP Cooperation, 2006), an infrared sensor; the SensComp 600 Transducer (SensComp Inc, 2004), an ultrasonic sensor similar to sonar and the SICK Laser Measurement Sensor (LMS) 200 (SICK AG, 2009). The values in table 3.1 can differ considerably between different devices, even those of the same component type. Despite variations in the properties, the trend between infrared, sonar and laser, in terms of price, power consumption and accuracy, generally remains the same, the three specific components shown in table 3.1 were chosen to reflect this.

Actuators

Beacon  Some form of beacon or light is necessary to allow the robots communicate. At its simplest and cheapest this would be a single beacon signifying two different states when on or off. Multiple lights with various colours would allow the robots to be more expressive but would also be more expensive.

Gripper  Some form of gripper is necessary to allow robots to pick up and drop off rubbish. The most complex grippers may allow robots to store rubbish as well as pick it up, once again however this extra functionality comes at a higher cost.

Motors  Motors are necessary for the simple purpose of moving robots around the environment.

3.2.3 Environment

To show that a robotic system is adaptable it is important to test it in a variety of environments. The previous work of Lord (2007) involved two different environments, one with obstacles and one without, Lord found that having tested and debugged the robots in the simpler environment, it was necessary to tweak the implementation when switching to the new more difficult environment. In order to reduce the problems Lord had with switching between environments, it was deemed sensible in this project to test and debug robots in the most complicated environment first. By ensuring that the system does not need to be tweaked before being deployed it is possible
to make viable comparisons about the performance of the robots across different environments. The proposed approach does not fully test the adaptability of the system since it is unknown how the robots would perform when deployed in a more complex environment, unfortunately time constraints prevented this from being fully investigated.

Environments in the Player/Stage Project are made up of walls and objects. Walls are static and immovable, whereas objects are fully configurable by changing factors such as: their colour, what sensors they are sensitive too and whether or not they can be picked up and moved by robots. Figure 3.1 shows one of the environments used in this project, this environment is discussed in detail in section 5 when introducing the experiments.

3.3 Multi-robot system

The literature review of chapter 2 identified four properties of multi-robot systems that can be used to describe them: control, population, interaction and cooperation. In this section the four properties are each introduced in turn and analysed with relation to the neural endocrine control architecture and the foraging task of this project.

3.3.1 Control

There are two main types of control for multi-robot systems: centralised and decentralised, the literature review consistently showed decentralised methods of control to be the most fruitful for behaviour-based multi-robot systems, for this reason a decentralised method of control was chosen for this project, whereby each robot is separately controlled by its own neural endocrine system.

3.3.2 Population

The population of a swarm of robots, in this context, is defined by two characteristics: the number of robots in the group, and whether or not these robots are heterogeneous or homogeneous. The number of robots used in simulation is limited by the hardware upon which they are simulated (see section 5.1.1 for this project’s hardware specification). Preliminary experiments showed that on the hardware available, the largest number of robots that could be simulated at the same time, without risk of crashing or loss of performance, was seven.

In section 2, four previous behaviour-based multi-robot implementations were reviewed, two investigated solely homogeneous populations, one solely heterogeneous populations and the other investigated both heterogeneous and homogeneous populations. Matarić (1993) who looked at both types of population, concluded that a heterogeneous approach, with a hierarchy imposed over the robots was neither necessary or helpful for a simple task, Balch and Arkin (1995) back up these conclusions with their experiments into robot formations. Although, due to the unique ID assigned to each robot, Balch and Arkin’s experiments technically only involved
heterogeneous populations, the *least* heterogeneous experiments whereby their IDs were not used to define a hierarchical leader produced the best performance for the majority of situations. For the reasons stated above and the inherent simplicity of the approach, a homogeneous population of robots is used in this project.

### 3.3.3 Interaction

There are three main types of interaction in multi-robot systems: interaction via the environment, interaction via sensing and interaction via communication. Each type of interaction has different hardware and software requirements and effects the ways in which robots can cooperate differently.

The simplest form of interaction is via the environment, this has little in the way of hardware requirements; since robots do not need to sense each other, there is no need for a camera or light sensing device (although they may require these sensors for other purposes, such as finding rubbish in the foraging task). The level of cooperation that can arise from interaction via the environment is generally considered to be lower than for the other forms of interaction, however the experiments of Shell and Matarić (2006) contradict this. In their investigations Shell and Matarić use a bucket-brigading approach to solving a foraging problem, without the robots directly interacting with each other a large amount of cooperation is still observed as robots pass objects between each other’s sub-regions.

From the literature review, the commonest form of interaction for behaviour-based systems was found to be interaction via sensing. In Matarić (1992b) three types of cooperation were investigated, one based on interaction via the environment and two based on interaction via sensing. In Matarić’s experiments the interaction via sensing approaches were consistently seen to perform the best. Interaction via sensing requires hardware that allows robots to distinguish between fellow robots and objects, it also requires extra behaviours that determine what a robot should do when it senses a neighbour. Behaviours that can be stimulated by the sensing of other robots include for example, cohesion and separation. Interaction via sensing can also be used to relay basic information between robots, one example from the foraging task is that of a robot signalling the presence of a large amount of rubbish, perhaps by flashing a light. After sensing that a robot is signalling the other robots should act accordingly i.e. by moving towards the area where the signalling robot was located.

The final type of interaction, interaction via communication, was not reviewed in any behaviour based systems. The fact that literature on interaction via communication for behaviour based systems was sparse could signify in itself that it is not appropriate for this project, however it is still worth considering. Interaction via communication requires the most advance hardware yet, robots need some form of channel over which to communicate and potentially the use of some form of protocol to do so (Cao et al., 1997). The need for a communication protocol is especially important when dealing with systems that involve a large amount of robots that communicate with each other frequently, unfortunately it adds further complexity to the system. An advantage of interaction via communication is that robots do not need to be in line-of-sight of each other to interact, depending on their method of communication they can potentially send messages to any other robot in the system at any moment in time. In the foraging task, assuming that they had knowledge of their location somehow, perhaps by GPS, robots would be able to broadcast the location of areas containing large amounts of rubbish. Because of the extra hardware required and associated costs it is expected that interaction via communication would be best suited to smaller groups of larger sized robots.

The type of interaction is not strictly selected by the designer, rather it is a product of the behaviours chosen and the hardware available, however, from the analysis provided above is possible to suggest that interaction via sensing, in delivering a tradeoff between the hardware required and the resulting functionality, would be the most desirable form of interaction for this project.

### 3.3.4 Cooperation

Shell and Matarić (2006) investigated two different methods of cooperation that differed in only in terms of the behaviours of the robots, a bucket-brigading approach and a traditional homogeneous approach. The bucket-brigading approach was shown to out-perform the homogeneous approach the majority of the time, however it did not fair so well for small numbers of robots (less than one hundred), or more correctly for situations where population density of the robots was too small.

The size of the arena used by Shell and Matarić was $64m \times 64m = 4096m^2$, assuming the minimum number of robots required for the method to be effective is one hundred then the minimum population density for the method is given by (3.1).

$$\frac{100}{4096} = 0.02 \text{ per } m^2 \tag{3.1}$$

From the result of (3.1) it is possible to calculate what the maximum arena sizes would need to be for bucket-brigading to be effective on the sort of scale used by this project. Table 3.2 shows the maximum arena size...
Table 3.2: The necessary arena sizes to satisfy the minimum population density for the bucket-brigading approach to be effective, based on the results of Shell and Matarić (2006).

<table>
<thead>
<tr>
<th>No. robots</th>
<th>Max arena size ($m^2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>350</td>
</tr>
<tr>
<td>6</td>
<td>300</td>
</tr>
<tr>
<td>5</td>
<td>250</td>
</tr>
<tr>
<td>4</td>
<td>200</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>50</td>
</tr>
</tbody>
</table>

sizes for various numbers of robots up to seven.

For comparison, the size of the world in figure 3.1 is approximately $35m \times 35m = 1225m^2$, meaning that to satisfy seven robots the environment would need to be roughly a quarter of the size it is in figure 3.1 and contain no obstacles. In order to make fair comparisons between the performance of the system as the number of agents changes, the size of the arena would need satisfy the minimum number of robots to be investigated, clearly the arena would be far too cramped if this number was small. In Shell and Matarić (2006) robots were not able to sense pellets until they were very close to them, if this restriction was lifted then the minimum population density would no doubt decrease, hence increasing the size of the arena and alleviating the cramping problem to some extent.

Shell and Matarić only tested the method in an environment with no obstacles and suggest that the addition of obstacles could cause problems, for this reason and the potential problem of cramping, it would perhaps be better to take a more conventional approach to cooperation.

Heterogeneous approaches such as the hierarchies of Balch and Arkin (1995) and Matarić (1993) have already been disregarded for not being effectual enough (section 3.3.2), which leaves only the more traditional homogeneous approaches.

Matarić (1992b) investigated three homogeneous methods: ignorant coexistence, informed coexistence and intelligent coexistence. The intelligent coexistence method proved to be the most effective in which, inspired by the methods of Reynolds (1999) robots were programmed to exhibit robust flocking behaviour. Flocking is an emergent behaviour that arises from the local interactions of robots and their environment, it is discussed in more detail in section 3.4.4 along with other emergent behaviours. Matarić states that the use the use of behaviours such as flocking can not only reduce interference, but can go beyond that and produce more efficient systems. When applied to the foraging task behaviours such as flocking would be useful to allow robots to converge upon the same location, for example the bin, without interfering with each others progress.

### 3.4 Behaviours

The minimum number of behaviours required to complete the task described in section 3.1 is between six and eight, depending on how individual behaviours are defined. In order to reduce the problem of interference in multi-robot systems and to ensure the robots exhibit some form of cooperation, this project proposes the use of eleven behaviours to accomplish the task. The majority of the behaviours can be categorised into the three different groups: taxes, reflexes and fixed-action patterns (FAP). One of the behaviours, wander, can not easily be classified by type, this behaviour is analysed on its own shortly, following which the remaining behaviours are grouped and analysed according to type. At the end of this section some desirable emergent behaviours are also discussed.

**Wander** A wander behaviour is necessary to ensure that robots keep exploring the environment even if none of their other behaviours are currently being stimulated, without a wander behaviour an unstimulated robot would just remain stationary.

The motor commands of wander behaviours are required to produce both forward and angular speed to ensure that robots do not get stuck in cycles. In the world shown in figure 3.1, a robot positioned at location A, who’s wander behaviour contained no angular speed, would struggle to reach location B, as to do so would require them to travel through small gaps which their obstacle avoidance behaviour might normally discourage them from doing. The likely result of a wander behaviour which only applies forward speed, would be for the robot to continually travel around the outskirts of the environment. A wander behaviour that did include an angular component would be more adventurous, overcoming some of the repellent force of the obstacle avoid behaviour and allowing the robot to travel through the small gaps.
There are several ways in which to implement a wander behaviour, Lord (2007) took the approach whereby without stimulation a robot simply attempts to move in a large circle. Another popular approach is to allow a robot to move forward for a random but bounded period of time, before turning by a random angle and repeating. Reynolds (1999) proposes a more realistic less “twitchy” form of wandering in which changes in direction are more gradual and based on the current heading of the robot.

An alternative is suggested here that takes into account the current hormone levels of the system. If a robot has gone for a long period time without picking up a piece of rubbish it could mean they are stuck inside a cycle or uninteresting area, to break free from this they must be more adventurous. It is proposed, that since there could be a hormone associated with a robots desire to find rubbish, the strength of this hormone could be used to effect a wander behaviour. If the basic wander behaviour, that is when unaffected by hormones, is for the robot to trace a sinusoidal wave, then an increase in the level of a desire hormone could be used to increase the amplitude of the wave, hence making the robot more adventurous.

3.4.1 Reflexes

To recap, reflexes are involuntary, spontaneous responses to stimuli, which last only as long as the stimulus that initiates them. The foraging task of this project requires only a single reflex behaviour. Because of their spontaneous and sporadic nature reflex behaviours do not require a neural endocrine control network, their response is simply tied directly to their stimulus.

**Signal bin**  As robots will have no awareness of the location of the bin, in order to improve their chances of finding it a signal bin behaviour is required, allowing robots to communicate the approximate location of the bin to others. As mentioned, interaction via sensing is probably the most suitable form of communication for this project, so rather than explicitly broadcast the bins location, robots should signal that the bin is in their vicinity by the use of a light or beacon. The strength of the response should always be the same, i.e. the brightness of the light should not effected by the closeness of the bin, it should either be on if the bin is in-sight, or off otherwise.

3.4.2 Taxes

Taxes are behavioural responses that cause agents to move towards, or away from certain stimuli. This project involves six taxes behaviours, two of which are repellent and four of which are attractive. Taxes behaviours are well suited to control using neural endocrine networks because both their inputs and outputs are continuous and should vary according to current state of the system, i.e. the hormone levels.

**Obstacle avoid**  An obstacle avoidance behaviour is necessary to prevent robots from crashing into the walls of the environment, or obstacles within the environment. The response of an obstacle avoid behaviour should be proportional to the distance between a robot and its nearest obstacle, such that a robot responds more urgently to obstacles that are nearer. The inputs to the network of an obstacle avoid behaviour should come from a range finding sensor, for example a sonar or laser device.

**Separation**  A separation behaviour is required primarily to prevent robots from crashing into each other, however when combined with other behaviours it can result in the emergence of interesting global behaviours, these emergent behaviours are discussed at the end of this section. The stimuli of a separation behaviour, also the inputs to the behaviour’s network, are the locations of other robots, these can be determine using a camera device with image processing software. In a similar manner to obstacle avoidance, the strength of a response should be proportional to the distance between a robot and its neighbours, such that the closer a fellow robot is, the faster the robot should retreat.

**Cohesion**  Sometimes it is beneficial for robots to follow the movement of others, this requires a cohesion behaviour that attracts a robot its neighbours. As with separation a cohesion behaviour is useful in the development of emergent global behaviours. The strength of the stimulus should have an effect on the strength of the response so that robots are less attracted to neighbours that are closer, reducing the chance of collisions. The inputs to a cohesion behaviour’s network are similar to those of a separation behaviour and come from the positions of their neighbours via a camera device.

**Seek rubbish**  In order for robots to collect rubbish they must first approach it, for this a seek rubbish behaviour is required. Robots should be stimulated by the presence of a piece of rubbish, which can be detected using a camera. Robots should be attracted to the location of the rubbish with a strength of response that is relative to the how far away the rubbish is, the further away, the stronger the attraction.
Seek power  A seek power behaviour is very similar to a seek rubbish behaviour, though instead of attracting robots to pieces of rubbish, it should attract them to charging stations. Inputs are provided in the same manner as the seek rubbish behaviour, using a camera device, and the strength of the response is once again relative to the distance of the stimulus.

Seek bin  A seek bin behaviour is very similar to both the seek power and seek rubbish behaviours, however in this case robots should be attracted to the bin. The stimulus is the presence of the bin, and the strength of response is relative to the distance between the robot and the bin.

3.4.3 Fixed-action-patterns

Fixed Action Patterns (FAP) are responses that continue even if the stimulus that triggered them is not present, usually they run uninterrupted until completion. Their response is always identical and so they are not suitable for control using neural endocrine networks, like reflexes they can be implemented by directly tying stimulus to response.

Pickup rubbish  A pickup rubbish behaviour should be stimulated when a robot is close enough to a piece of rubbish and is not already carrying some. The behaviour should involve the robot moving towards the piece of rubbish and either successfully or unsuccessfully picking it up, both of which should result in the end of the pattern, however if the pickup is unsuccessful it is possible that the behaviour will be re-stimulated immediately.

Drop Rubbish  If a robot is carrying a piece of rubbish and is close enough to the bin, the drop rubbish pattern should be stimulated. The pattern starts with the robot approaching the bin and continues until the robot has either successfully or unsuccessfully dropped the rubbish into the bin.

Recharge  A recharge behaviour should be stimulated when a robot is close enough to a charging station and its internal state dictates that it needs to recharge. The behaviour should begin with the robot moving towards the charging station and attempting to dock with it, if the robot fails to dock, the pattern should end, if the robot successfully docks the pattern should continue until the robot is fully charged.

3.4.4 Emergent behaviours

Emergent behaviours are those which are not explicitly programmed into the system, but emerge from the combination of the other behaviours and the robots interaction with the environment. They are not completely spontaneous though and do require some input from the designer to ensure that the perform as expected, in the case of the neural endocrine architecture the hormone need to big configured so that each individual behaviour contributes appropriately to the global behaviour. There are two anticipated emergent behaviours in this system, flocking or more correctly following and dispersion.

Flocking or following  Flocking emerges from the combination of obstacle avoidance, seek bin, signal bin, separation and cohesion. It is expected that flocking in its usual sense would not be effective for the task at hand as it could result in many robot converging on the bin at the same time, as the bin is relatively small this could cause problems, a sensible alternative is following where robots travel in a line formation, rather than as a group. Following can be achieved in exactly the same way as flocking except that robots cohesion behaviours are only effected by robots that are in front of them, rather than all around them.

Dispersion  Dispersion emerges from the combination of obstacle avoidance and separation, put simply it is the spreading out of robots over the environment to ensure the greatest amount of coverage.
### 3.5 Requirements

The requirements of the system developed for this project are listed below, the requirements were derived from the problem analysis and the literature review.

#### 3.5.1 Functional requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>Robots should be able to exhibit all nine of the individual behaviours.</td>
</tr>
<tr>
<td>F2</td>
<td>Robots should be able to complete the task of rubbish collection as an individual.</td>
</tr>
<tr>
<td>F3</td>
<td>In groups of up to seven, robots should be able to exhibit each of the two collaborative behaviours.</td>
</tr>
<tr>
<td>F4</td>
<td>In groups of up to seven, robots should be able to exhibit each of the two emergent behaviours.</td>
</tr>
<tr>
<td>F5</td>
<td>In groups of up to seven, robots should be able to complete the task of rubbish collection.</td>
</tr>
<tr>
<td>F6</td>
<td>If one or more robots breaks, the remaining robots should still be able to complete the task of rubbish collection.</td>
</tr>
</tbody>
</table>

#### 3.5.2 Non-functional requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1</td>
<td>Multiple robots should provide a speed up over a single robot at the task of rubbish collection.</td>
</tr>
<tr>
<td>N2</td>
<td>Multiple robots should provide a linear speedup over a single robot at the task of rubbish collection.</td>
</tr>
<tr>
<td>N3</td>
<td>Multiple robots should provide a faster than linear speedup over a single robot at the task of rubbish collection.</td>
</tr>
</tbody>
</table>
4 Design and Implementation

This chapter begins by introducing the technical specifications of the simulated robots that were used. Some of the factors that need to be considered during the construction of artificial neural endocrine systems are then discussed, before introducing the design itself. As well as describing the whole system from a high level, the design section includes specific details of each of the individual behaviours. When discussing the neural endocrine behaviours, important factors such as which hormones they respond to and how they were trained shall also discussed. Alternative design choices and the problems encountered in this project, as well as the chosen solutions, are mentioned throughout.

4.1 Simulated robots

As already mentioned in chapter 3, the robots chosen for this project were MobileRobots’ Pioneers. When designing the robots, to ensure that they were feasible in the real world, both in terms of their capabilities and their cost, the points raised in section 3.2 were all taken into account. Figure 4.1 shows one of the robots. Each robot was equipped with the following devices:

**Sonar** A sonar device was chosen above the other range-finding devices because it provides a good compromise between cost and functionality. Robots were given two sonars, the bare minimum necessary to achieve effective obstacle avoidance. Each sonar device had a field of view of $\frac{\pi}{6}$, a range of 500 cm and was orientated at an angle of either $\frac{\pi}{4}$ or $-\frac{\pi}{4}$ relative to the centre line of the robot. Sonar devices return a single value that is the distance between themselves and any obstacles within their range and field of view.

**Camera** A camera device was primarily chosen because of the ease with which it allows objects in the environment to be differentiated, however the fact that the alternatives provided by the Player/Stage Project were rather limited in their abilities also influenced the decision. The device was configured to have a wide field of view of $\frac{4\pi}{3}$ and a range of 500 cm. The wide field of view was chosen to allow the robot to interact with not only those in directly front of them, but also those to either side and behind them. The processing software of the camera represents individual objects in the world as ‘blobs’ of colour, mapped onto a 2D image. The colour of a blob is determined by the predominant colour of the object it represents. The device returns three pieces of information about each blob: its colour, its position within the 2D image and how far away it is.

**Gripper** A simple gripper device was chosen that provided robots with the ability to pick-up and drop off objects, but not to store them in any way. The gripper device takes two input commands, open and close and is also able to return information about its current state, for example, whether the gripper is currently open or closed and whether it is holding an object.

**Beacon** A beacon was necessary to allow robots to signal to each other, but unfortunately no such device currently exists in the Player/Stage Project. This problem was overcome by using a component of the Player/Stage Project that allows client programs to directly manipulate objects in the simulation at runtime. By changing the colour of a robot it was possible to simulate a beacon turning on and off. Two input commands for controlling the beacon were created and tied to the necessary components for manipulating the simulation, these commands are briefly discussed later in this chapter. The normal colour of the robots is yellow and a signalling robot is cyan.

**Motors** Motors are an obvious necessity to move the robots. The motor devices used in this project had two inputs, one for controlling the forward speed of the robot and one for controlling the angular speed.

4.2 Neural endocrine design

There are several factors to consider when designing a neural endocrine control system, the most important of which is how the artificial neural networks (ANNs) and artificial endocrine systems (AESs) are integrated. Once the details of the neuro-endocrine interaction have been determined, there are some behaviour specific
ANN-AES integration  In section 2.2, it was noted that not every hormone in a system must effect every neuron. In all previous work the approach has been to make all the neurons of a single network sensitive to the same hormones, for example in a system with two hormones $h_a$ and $h_b$ and two networks $N_a$ and $N_b$, a possible configuration would be that all the neurons of $N_a$ are sensitive to $h_a$ and all the neurons of $N_b$ are sensitive to $h_b$, this is shown in figure 4.2. The alternative is to make different neurons of the same network sensitive to different hormones, for example in a system with two hormones $h_a$ and $h_b$ and a single network of seven nodes $\{n_1, n_2, ..., n_7\}$, nodes $n_1 - n_4$ might be sensitive to $h_a$ and nodes $n_5 - n_7$ might be sensitive to $h_b$, this is shown in figure 4.3. Since each network in a system corresponds to a single behaviour, it seems sensible that as is the case in the first approach, each network should be effected by the same hormones. Whilst the second method sounds more flexible, it has not been previously investigated and because of the additional complexity it could prove difficult to engineer. This project employs the first method, with all of the neurons of a single network sensitive to the same hormones.

Another factor that needs to be considered is whether a network is sensitive to a single, or multiple hormones. If a network is associated with more than one hormone, it is likely that it will be more flexible, but in a similar manner to that where different neurons in a network are sensitive to different hormones, the system may be harder to engineer and furthermore if the hormones conflict, it could make the behaviour indecisive. To keep things simple, in this project each network is only associated with a single gland-hormone pair. The sensitivity of a neuron $i$ to a particular gland $g$ is denoted $s_{ig}$, in theory $s_{ig}$ can take any value, however in this project $s_{ig}$ only takes the value 1 or 0, representing full or no sensitivity of $i$ to $g$.

Having decided that all neurons in a network will be sensitive to the same hormones, that each network will only respond to one hormone and that sensitivity is only ever 1 or 0 it is possible to refine equation (2.8) from section 2.2 in order to suit the needs of this project. Removing the sensitivity and multiplicity of (2.9) leaves (4.1)
Figure 4.2: Two networks, the neurons of which are all sensitive to the same hormone

Figure 4.3: A single network with different neurons sensitive to different hormones

where $c_g$ is the concentration of the networks only associated hormone.

\[
\begin{align*}
\mathbf{u} &= \sum_{i=0}^{n_x} x_i \cdot w_i \cdot c_g \\
\end{align*}
\]

The activation of a gland, can be calculated from a combination of both internal and external properties of the system. In Lord (2007) each gland’s activation was calculated using a function specific to that gland, a similar approach is taken by this project. Each gland is associated with a single activation value which changes over time according to a dedicated function and is represented here as $a_g$. The stimulation of a gland, as was seen in section 2.2 can be calculated in one of two ways. Based on the success Lord had with using a negative feedback mechanism, this project takes the same approach. The stimulation of a gland is calculated using (4.2), which is a slightly adjusted version of (2.7) in order to take into account the new representation of activation.

\[
\begin{align*}
R_g(t) &= \frac{\alpha_g \cdot a_g(t)}{1 + c_g(t - 1)} \\
\end{align*}
\]

The final consideration with ANN-AES integration, is what values of stimulation ($\alpha_g$) and decay ($\beta_g$) rate are used by the networks. The stimulation rate helps determine the amount of hormone released by a gland at a particular timestep and the decay rate determines how long the hormone remains in the system, hence they both have a big influence on the behavioural response. Values of $\alpha_g$ and $\beta_g$ can vary widely between different networks, in section 4.3.2 when describing how the behaviours were implemented the stimulation and decay rate values are both mentioned.

**Network size** ANNs can be defined by four properties: the number of hidden layers, the number of nodes in each of the hidden layers, the number of nodes in the input layer and the number of nodes in the output layer. The number of nodes in the input layer of a network are determined by the number of sensor values needed to define the stimulus of that behaviour, for example in the case of obstacle avoidance which is stimulated by the presence of nearby objects, the number of sonar devices (two in this project) determines the number of input nodes. The number of output nodes is determined by the actuator that the response effects, in most cases, where the response effects the locomotion of the robot, it is the number of inputs to the motors that decides the number of output nodes (which again in this project is two).

The number of hidden layers and the number of hidden layer nodes is less dependent on the behaviour, and puts more pressure on the designer to choose sensible values. Networks that are too large will not only take
longer to train and perform slower when executing, but have a tendency to over-fit the training data, reducing the adaptability of the network (Walczak and Cerpa, 1999). Small networks will be more apt at dealing with unexpected inputs (that is, those not found in the training data) but without enough processing elements may struggle to appropriately process inputs, clearly a compromise is needed. As stated by Walczak and Cerpa (1999) views differ when it comes to choosing the number of hidden layers and hidden layer nodes in a network, some authors maintain that a single layer with an arbitrarily large number of nodes is enough to model any categorisation problem and others suggests that two hidden layers will always outperform one for certain problems. It was known from the work of others that the networks required by this project would be relatively simple, consequently it was decided early on that all of the networks would only include one hidden layer. Originally a cautious approach of trial and improvement was planned in order to find out how many hidden nodes would be needed, however in the end, two nodes were sufficient for all of the neural endocrine behaviours encompassed in this project.

**Network weights** The weights of a network can either be determined “by hand” or through the use of a learning algorithm such a back-propagation. If the function of the network is simple enough and the network does not contain too many nodes or layers then it may be easiest to simply work out the weights by hand, however if the function of the network is more complex and involves mapping many different inputs to many different outputs, then the best course of action is normally to use a learning algorithm. Learning algorithms, such as back-propagation require a set of training data which must be constructed in some way by the designer. In this project a mixture of both manual assignment and the back-propagation learning algorithm were used.

**Coordination of different behaviour types** Behaviours that are encapsulated as neural endocrine networks, as was discussed in section 2.2, are coordinated by summing their outputs, what has not yet been discussed is how these behaviours are coordinated with the other types of behaviour, such as the fixed-action-patterns and reflexes. The singal bin behaviour is a reflex, it does not effect anything other than the state of the robots beacon and so it does not need to be coordinated with the other behaviours.

In terms of the FAPs, when stimulated, these will always take complete control of the robots motors, inhibiting any of the suggested commands from the other behaviours. One potential problem with this approach is that if two robots are executing FAPs and are on a collision course with each other, their separation behaviours will be inhibited, meaning that they will not be able to avoid each other. The collision-course problem was overcome by only initiating FAP behaviours when robots were very close to their targets, this makes it less likely for collisions to occur since until robots are sufficiently close to their goals they are still concerned with avoiding their neighbours, in all the experiments that were carried out no collisions of this type were observed.

It is very rare for conflicts to arise between different FAPs since it is never the case that a robot will want to both drop and pickup rubbish at the same time and because the bin and charging posts are positioned far apart (in the experiments carried out in this project) there will never be a conflict between wanting to charge and wanting to drop rubbish. The only conflict that can emerge between FAPs is the desire to pickup rubbish versus the desire to charge, this problem was solved by imposing a hierarchy over the FAPs with the recharge pattern, as deemed the most important, at the top and the pickup and drop rubbish patterns underneath.

**4.2.1 Summary**

The design of a neural endocrine systems requires that several factors be taken into account. The structure of a network is largely determined by the behaviour it is being designed for, but some properties, such as the number of hidden layer nodes are chosen at the designers discretion. The weights of a network can be decided in one of two ways, either by hand or by learning algorithm, if a learning algorithm is used it is necessary to construct training data. The integration of AES and ANN requires a lot of thought, it was decided that all nodes of the same network would be sensitive to the same single hormone and that sensitivity would be defined as either 0 or 1, other factors such as the activation, stimulation and decay rate also need to be taken into account. In this project FAPs will inhibit neural endocrine behaviours and to avoid conflicts between different FAPs, have an imposed hierarchy. In the next sections of this chapter when describing the behaviours of this system, the structure of networks, how the weights were decided upon and the methods by which any training data that was constructed shall all be described. When discussing the hormone glands in particular, how the activation of a gland is calculated and the values of stimulation and decay rate chosen shall all be discussed.

**4.3 System design**

This section starts by presenting a general overview of the system and describing the way in which it works from a high level, a more detailed explanation is then given of the constituent parts of the design, including a
breakdown of the individual classes and their main functions. The final part of this section covers the design of the individual behaviours, each behaviour is introduced in turn and details are included, where appropriate, of each of the design considerations mentioned in the previous section.

### 4.3.1 Overview

A high-level, functional overview of the system is shown in figure 4.4 which demonstrates the how through the combination of different behavioural responses, sensory inputs are mapped to actuator outputs. All of the behaviours in the system, apart from “wander”, receive input from the environment via sensors; although not shown in the diagram, some behaviours also receive input from the internal state of the system. All inputs are pre-processed before they are acted upon to ensure that they provide only the most relevant information, for example, the seek bin behaviour needs only to know the location of the bin and not the locations of the other objects in the environment, consequently it is only presented with information about the bin. Following the processing of inputs some form of calculation is performed to produce appropriate outputs, these calculations are behaviour dependent. In the case of the neural endocrine behaviours (shown in the grey shaded area), inputs are presented to artificial neural networks which in combination with an artificial endocrine system, return the appropriate outputs, more details about this process are given later. The wander behaviour, is not strictly a neural endocrine behaviour as it is not controlled by an artificial neural network, it is however effected by the concentration of a particular hormone. As the wander behaviour takes no inputs its output is solely dependent on the concentration of its associated hormone. The outputs of the other behaviours are much simpler as once activated they always return the same output values.

Once the outputs of all the behaviours have been calculated, they are coordinated to determine how the

Figure 4.4: Overview of the system
actuators are effected. As described earlier the output of the signal bin behaviour always proceeds to the beacon component, regardless of the outputs of the other behaviours. The outputs of the neural endocrine behaviours and the wander behaviour are summed and the result is passed directly to the motor component. The pickup rubbish, drop rubbish and recharge behaviours are fixed action patterns, when any of them are activated, they will suppress the result produced by the neural endocrine and wander behaviours. There is a hierarchy amongst the FAPs such that if the drop rubbish behaviour is activated it will suppress the output of the pickup rubbish behaviour and if the recharge behaviour is activated it will suppress the output of both the pickup and drop rubbish behaviours.

4.3.2 Code overview

All of the main code that makes of the neural endocrine control system was written in C++ and developed inside the Eclipse IDE (Eclipse Foundation, 2009). This section introduces all of the major classes of the system, describing their purpose and their most important functions. The entire code is documented on the attached DVD which may provide a useful accompaniment to this section, as might the diagram in appendix A.1 which shows how all of the classes are related.

There are five main classes that shall be described in this section, they are: Robot, Behaviour, NECAnet, RobotKnowledge and Gland. Robot is the main class from which all others are spawned, it connects to the Player/Stage simulator and from a high level controls the mapping between perceptions and actions. RobotKnowledge keeps track all the sensor values at any one moment in time and is also responsible for preprocessing the behavioural inputs. The Behaviour class is perhaps the most important, there is one Behaviour object for each of the behaviours in the system, they take pre-processed inputs from, and pass outputs back to the RobotKnowledge class, the Robot class then coordinates the outputs of each of the behaviours to form the actions of the robot. The NECAnet class, in combination with Gland class perform much of the low level computation for the Behaviour class, especially that of the neural endocrine behaviours.

NECAnet

The NECAnet class provides the main functioning component of a neural endocrine system, the neural endocrine network. Each neural endocrine network is composed of an artificial neural network and a number of hormone glands, in this project each NECAnet is only associated with one hormone, and hence one gland. Artificial neural networks are constructed from a number of artificial neurons, linked by weighted connections, this neuron-connection pairing is encapsulated in this project by the Neuron and Connection classes. Artificial endocrine glands are provided by the Gland class and secrete hormones that are identified by the enumerated type hormone. The Neuron and Connection classes are described below, Gland, being a much larger class, is described within its own section later on.

Connection The Connection class can be thought of as a simple record, with the three fields: origin, target and weight, which keeps track of the IDs of two nodes and the weight of the connection between them.

Neuron The Neuron class contains a pointer (*connections) to an array of Connection objects that defines the links from this Neuron to others. The function getWeightBetween is able to return the weight between this Neuron and any others it is connected to and the function updateConnectionWeight can be used to adjust these weights. Neuron also contains a field for recording the current input and output values of a node.

The NECAnet class contains the function initNeurons which creates an array of Neuron objects based on the required number of input, output and hidden layer nodes, different layers are defined by the index at which the first node of that layer is located. The class also contains the function initGlands which has a similar purpose but for Gland objects.

The two main functions of the NECAnet class are train and run. The train function uses a back-propagation algorithm, provided by the Fast Artificial Neural Network Library (Nissen, 2009), to train a network based on the contents of a training-data file. The train function also saves the trained network in a format that can be used to recreate the network, using the createFromFile function.

The run function takes an array of input values, propagates them through the network and returns an array of outputs values. The function is based on (4.1) and (2.5), it works by first calculating the hormone concentration of the gland that the particular network is sensitive to, and the, starting with the first node of the hidden layer, uses the hormone concentration to influence the outputs of each neuron, the final values returned by the function are the outputs of the nodes in the output layer.
Gland

Gland is an abstract class of which there are seven subclasses, one for each of the hormones in the system. Each type of gland is activated under different conditions and so has a different function for updating its current activation value (updateActivation). How the updateActivation functions differ between gland types is discussed later when describing the behaviours of the system. The two most important functions of the Gland class are updateStimFB and updateConcentration, both of which are very simple. updateStimFB is based on (4.2), using a negative feedback mechanism it updates the stimulation of a hormone, that is the amount of hormone released at a single timestep, based on the stimulationRate of the hormone, the current hormone concentration and the current activation value. (4.3) shows a segment of code from updateStimFB.

\[
\text{stimulation} = \frac{\text{stimulationRate}}{1 + \text{concentration}} \times \text{activation}; \quad (4.3)
\]

updateConcentration is based on equation (2.8) it calculates the new concentration level of a hormone, based on the current concentration, the gland specific decayConstant and the stimulation as calculated by updateStimFB. (4.4) shows a segment of code from updateConcentration.

\[
\text{concentration} = \text{concentration} \times \text{decayConstant} + \text{stimulation}; \quad (4.4)
\]

Everytime inputs are run through a NECAnet all three of updateActivation, updateStimFB and updateConcentration are called in order to update the hormone concentration.

RobotKnowledge

RobotKnowledge is an abstract class which has only one subclass, RobotKnowledgeSim. Using sensory information provided by the Robot class, RobotKnowledge pre-processes and temporarily stores information about the relative positions of objects and obstacles within the environment, as well maintaining a number of suggested actuator commands. The postions of obstacles and objects are stored in various different arrays and are independently updated by calling different functions of RobotKnowledge. Suggested actuator commands are also split into different categories depending on the actuator they effect and are updated by different Behaviour objects. The ways in which information is stored, along with the update functions that are used are described shortly.

Sensor information

Information provided by the two different sensors, sonar and camera, is stored in two different ways. The sonar device returns the range of the nearest obstacle, or the maximum range of the device if no obstacles are blocking the sensor. Sensors values from the two sonar devices of this system are stored in an array of length two, one cell corresponding to the value returned by the left sonar, and one cell corresponding to that returned by the right. The function addSonar is used to add sonar (or obstacle) information to a RobotKnowledge object, given a range \( r \) and an index \( i \), addSonar normalises \( r \) and adds it to the sonarData array at position \( i \).

It is slightly more complicated to store information from the camera as there is no fixed number of values that the camera device will always return, it depends purely on how many objects are in the field of view. Objects detected by a camera device are stored in arrays of length eight, each cell of the array corresponds to one \( \frac{\pi}{6} \) segment of the cameras field of view, within which the object lies, the eight segments of the cameras field of view are shown in figure 4.5. Segments are referred to by the same numbering scheme as arrays, so in this case from zero to seven. Not by accident and for reasons explained shortly, the second and fifth segments of the cameras field of view line up exactly with the fields of view of the two sonar devices. There is a separate array for each type of object in the environment, the value that a particular cell of an array takes corresponds to the normalised distance between the robot and the nearest object in that segment.
of camera’s field of view. Information is stored for the four main objects found in the environment: rubbish, robots, charging stations and the bin. The general method by which camera data is used to store the positions of objects is as follows. Given the x position of an object in a 2-D projection of the cameras field of view and the distance to the object \( d \), using the x-position and the function `getIndex`, the segment \( (S) \) within which the object lies is calculated. If there is no value currently stored in cell \( S \) or \( d \) is less than the currently stored value, then \( d \) is placed in cell \( S \). Each of the functions for adding objects to arrays is described in more detail shortly.

**addRobot**  This function takes the colour of the robot, the distance to the robot and the x position of the left, right and centre points of the robot in the cameras 2-D projection. The segment within which the robot lies is calculated, if the segment is one of the ones that overlaps a sonar device then the distance to the robot and the left and right x positions are used to determine whether the robot is blocking the sonar sensor, if a robot is blocking the sensor then the appropriate cell of the *sonarData* array is reset to signify that there is no obstacle. It is necessary to make this check because *sonarData* is used by the obstacle avoidance behaviour, and so should not be skewed by the positions of robots, other robots are avoided using the separation behaviour. As well as being stored in a general *robotsData* array, according to their colour the position of robots is recorded in the colour specific arrays: *cyanRobotsData* and *yellowRobotsData*.

**addRubbish**  This function takes the distance to a piece of rubbish and the x position of its centre. The segment in which the rubbish lies is calculated, but rubbish is only added to the *rubbishData* array if it is closer than the currently recorded closest piece. The reason only the closest piece of rubbish is recorded is because with the current set of behaviours there is no advantage for the robot to track more than one piece of rubbish. This approach can cause interference problems with two robots heading for the same piece of rubbish, the ideal behaviour in a situation such as this would be for one robot to realise that rather than risk interference it would be better to choose another piece of rubbish to aim for, even if it is further away. One way this behaviour could be achieved is by assigning each piece of rubbish a *risk* value determined by how close it is to other robots, then scoring each piece of rubbish based on its risk value and how far away it is, the robot would then aim for the piece of rubbish with the best score.

**addBin**  This function takes the distance to a the bin and the x position of its centre. The segment in which the bin lies is calculated and compared to the current nearest recorded bin. The comparison is seemingly unnecessary for the environments of this project where there is only one bin, however it must be remembered that the camera device detects “blobs” and not whole objects, if there was another smaller object in front of the bin, partially covering the line of sight, then to the camera this may be interpreted as three separate blobs. If two or more of the detected blobs were part of the same object, even though they would be very similar in terms of their distance value, is still necessary to make a choice so that the robot can be more decisive, furthermore, it is entirely possible to create environments with more than one bin.

**addCharge**  This function takes the distance to a charging station and the x position of its centre. The segment in which the charging station lies is calculated and compared to the current nearest recorded charging station. As was the case with *addBin* this comparison may seem unnecessary due to the fact that the charging stations are well spread out in this project, but for the same reasons as with *addBin* it is still necessary.

**Actuator information**  Behaviour objects update actuator information to express their preference for particular actions. Five types of actuator information are stored in a *RobotKnowledge* object: *motorData*, *doCharge*, *pickupRubbish*, *dropRubbish* and *signalBin*. *motorData* is an array of length two, one value for the forward speed of a robot and one for the angular. Behaviours that are concerned with the locomotion of the robot add their desired motor commands to the current values of *motorData*. Since the behaviours that are not immediately concerned with the locomotion of the robot have a response which is not effected by the strength of the stimulus, their suggested actuator commands are boolean values. If a behaviour wants the robot to drop rubbish it sets *dropRubbish* to true, these commands are coordinated by the *Robot* class so even if more than one of them is set to true, only one will prevail.

**Robot**  *Robot* is an abstract class, in this implementation it has only one subclass *RobotSim*, named so because it controls a simulated robot. The system was designed so that the *Robot* class was the only one that interfaces with the actual robot, this means that in theory, to use the control system on a different robot (simulated or real), requires only that a new subclass of *Robot* be created. Originally the preprocessing of inputs was done within the *RobotSim* class, however due to some unforeseen “quirks” with the Player/Stage simulator (discussed later) it was deemed appropriate to move the preprocessing to the *RobotKnowledge* class. Moving the preprocessing
to RobotKnowledge meant that to develop a system for another robotic platform would now involve creating
two new classes instead of one, which is still not a major problem, but an annoyance nonetheless. Ways in which
to reformat the system so that only one new class would need to be created were considered, but unfortunately
there was not time to implement them.

As a superclass, Robot provides several functions for its subclasses. Two of the most important functions that
Robot provides are setupBehaviours and go, setupBehaviours simply constructs a Behaviour object
for each of the eleven behaviours and places them into an array pointed to by *behaviours, it is called only
once when the Robot object is initially created. go is called once every timestep, if first updates all the sensor
values and then updates the outputs of each behaviour, both of which are stored in a RobotKnowledge object.
Finally Robot coordinates the outputs and sends the appropriate commands to the actuators.

During construction the RobotSim class connects itself to the Player/Stage simulator using the function
connectToPlayerStage. Subclasses of Robot are required to overwrite a number of functions, including
most importantly fetchSonarData, fetchCameraData and updateMotors. fetchSonarData collects
the current sonar values from the simulator and sends them to a RobotKnowledge object. fetchCameraData
is slightly more complicated, as mentioned, the camera device returns coloured blobs with different colours
corresponding to different objects. According to the colour of a blob fetchCameraData requests that
RobotKnowledge adds them to a different set of data, for example, information about other robots is stored in
robotsData.

A couple of problems arose from the way in which robots detected rubbish, the first problem was that robots
would attempt to steal rubbish from their fellow robots, this was because they couldn’t tell the difference
between a piece of rubbish on the floor and a piece of rubbish in the gripper of another robot. The function
rubbishInGripper was created to prevent robots from stealing from each other, the function compares the
location of a detected piece of rubbish to the locations of all the robots in the field of view, if the rubbish is
sufficiently close to a robot it is assumed to be within that robots possession. A second similar problem was that
robots would attempt to pickup rubbish from within the bin, this was solved by creating the checkForBin
function that determines whether the robot was inside the designated bin area. Before requesting that a rubbish
object be added to the robots knowledge RobotSim first checks that it is not in the bin or within the gripper of
another robot.

The way in which behaviours are coordinated has already been described from a high level, the process
by which coordination happens at the implementation level is now described. Firstly, the two values of the
motorData array, that store forward and angular speed commands, are set to zero, next, the outputs of the six
neural endocrine behaviours and the wander behaviour are in turn added to the contents of motorData. If any
of the FAP behaviours are activated they reset the motorData array and using one of the functions: doCharge,
doPickup and doDrop take full control of the robots actuators until completion, the function executed depends
on which behaviours are activated and the hierarchy over them. After any FAP behaviours have relinquished
control of the actuators, the signal bin behaviour is given the chance to change the state of the beacon actuator,
finally the motors are updated with the current values of motorData.

**Behaviour**

Behaviour is an abstract class of which there are six subclasses, one for each of the three fixed-action
patterns: BehaviourCharge, BehaviourPickup and BehaviourDrop; one for the single reflex behaviour:
BehaviourSignalBin; one for the wander behaviour BehaviourWander; and one for the six neural
endocrine behaviours: BehaviourNECA. The BehaviourNECA class is itself an abstract class of which there are
four subclasses: BehaviourAvoid, BehaviourSeparation, BehaviourCohesion and BehaviourSeek. Figure 4.6 shows the inheritance diagram for the Behaviour class, including where not obvious what behaviour
each class is used for.

All of the behaviour classes read inputs from the RobotKnowledge object and based on these inputs write
outputs back to the RobotKnowledge object, for this purpose subclasses inherit the function updateOutput
from the Behaviour superclass. In this section each of the Behaviour subclasses is introduced in turn, starting
with those designed for the simpler reflex and FAP behaviours before moving on to those designed for the
wander and the neural endocrine behaviours. The inputs and outputs of each classes updateOutput function
are mentioned. When discussing the classes designed for the reflex and FAP behaviours, the follow on effects of
their outputs are also described. When discussing the neural endocrine classes, the structure of the networks
are detailed, along with the methods used to select the weights and how any training data was constructed,
how the glands are activated and the values chosen for the stimulation and decay rates are also discussed.

**BehaviourSignalBin** The inputs for the signal bin behaviour are taken from the cyanRobotsData array, and
the boolean field seeBin which defines whether or not the robot can see the bin. The output is the boolean
field signalBin. If the robot can see the bin or can see another signalling robot in front of itself (a robot is
defined to be “in front” if it is in or between the second and fifth camera segments) then the robot updates its
output to true. Unless a FAP is executing or the robot is already signalling then the go function of the Robot
class will call the function startSignalBin, implemented in RobotSim, which changes the colour of the
robot from yellow to cyan.

It was decided that robots would only signal if they saw another robot in front of them signalling because that
way a trail of signalling robots would always lead towards the bin. When combined with seek bin and cohesion
a “following” behaviour emerges.

**BehaviourPickup** The pickup rubbish behaviour outputs to the boolean field pickupRubbish and takes inputs
from the rubbishData array and the boolean field hasRubbish which defines whether or not the robot is
currently holding rubbish. If the robot is not already holding a piece of rubbish and there is a piece of rubbish
directly in front of it (directly in front, meaning in one of the front two segments, i.e. segments three and
four) less than 0.2 m away then the robot updates its output to true. Provided that none of the other FAPs are
requesting to be activated then doPickup function is called by the Robot class.

The doPickup function has three phases, the approach, the pickup and the retreat. In the approach the
robots slowly move towards the piece of rubbish, using a seek rubbish neural endocrine network to ensure that
they keep heading in the right direction. Once the robot is less than 0.035 m away from the rubbish the pickup
phase starts, the robots gripper is commanded to close, if this successfully results in the piece of rubbish being
picked up then the retreat phase starts, otherwise the gripper is reopened and the FAP ends. The retreat phase
simply involves the robot rotating by $\pi$ radians which, because of the wide angle of its camera is enough to
survey $2\pi$ of its environment. The retreat phase can end before the robot has rotated the full $\pi$ radians if the
robot spots the bin. The retreat phase adds two advantages, firstly it makes the robots more efficient, if a robot
was near the bin but facing away from it, after having picked up a piece of rubbish it would move further away
and it may be a long time before it found then bin again, whereas with the retreat phase it would find the bin
immediately. The second advantage is that if a piece of rubbish is positioned very close to an obstacle or wall,
rotating makes it easier for the robot to avoid the obstacle.

**BehaviourDrop** The drop rubbish behaviour outputs to the boolean field dropRubbish and takes inputs from
the binData array and the boolean field hasRubbish. If the robot is carrying a piece of rubbish and the bin is
directly in front of it less than 0.2 m away, then the robot updates its output to true. As long as the recharge
behaviour is not requesting to be activated then the doDrop function is called by the Robot class.

The doDrop function has two phases, the approach phase and the drop phase. Similarly to doPickup the
approach phase involves the robot slowly moving towards the bin until it is just inside the bin, at which point
the drop phase begins, which simply involves the robot opening its gripper.

**BehaviourCharge** The recharge behaviour outputs to the boolean field doCharge and takes inputs from the
chargeData array and the powerLevel variable which monitors the robots current power level. The output is
the boolean field doCharge. If the robot can see a charging station directly in front of it and its current power
level is less than 60% of full power, then the robot updates its output to true. The recharge behaviour suppresses
all the other FAPs and so when activated it will always execute, it is executed by the calling of the doCharge
function from the Robot class.
when the power is at 60% (the power threshold) the desire is zero, as the power level decreases the desire for
when it finds one, opposingly

phase gradually increments the robots power level until it reaches 100%, at which point the retreat phase starts,

BehaviourWander

The wander behaviour outputs to the motorData array, is always constant at 2 m\(\cdot\)s\(^{-1}\). The angular speed, the second element of the motorData array, changes according to (4.5 & 4.6). Varying the angular speed using (4.5) causes the robot to trace a sinusoidal wave, it is clear to see from (4.5) that increasing the level of hormone will increase the range of the angular speed, this in turn increases the amplitude of the traced wave. It should be noted that increasing the hormone level too much would cause the robot to travel in circles, this situation was prevented through careful design of the hormone gland. Particular values of decay and stimulation rate were chosen such that the maximum level a hormone could ever reach still produced the desirable effects from (4.5).

\[
\text{angularSpeed} = \text{wanderHormone} \cdot 0.2 \cdot \sin(\text{step}(t))
\]

(4.5)

\[
\text{step}(t + 1) = \text{step}(t) + 0.02
\]

(4.6)

BehaviourWander is associated with the gland class GlandWander, the activation of this gland is based on three of the robots desires, the desire to find rubbish, the desire to find the bin and the desire to recharge. When the levels of the desires \(d_r, d_b\) and \(d_c\) are the robots desires to find rubbish, the bin and charge respectively.

\[
a_g = 3.0 \cdot (d_r + d_b + d_c)
\]

(4.7)

The three desires vary between zero and one and are updated every timestep using the updateDesires function. \(d_r\) is incrementally increased as long as the robot does not have a piece of rubbish and is reset to zero when it finds one, opposingly \(d_b\) is increased when the robot has a piece of rubbish and rest to zero when it finds the bin. \(d_c\) is more complicated and takes into account the power level and power threshold of the robot, when the power is at 60% (the power threshold) the desire is zero, as the power level decreases the desire for charge increases to maximum when the robot has ran out of power.

BehaviourNECA

Each of the neural endocrine behaviours is implemented as a BehaviourNECA object which is associated with a Gland object. Table 4.1 shows the relationships between each of the neural endocrine behaviours and the Behaviour and Gland subclasses they are implemented and influenced by. The decay and stimulation rates of each of the glands are also included in table 4.1. The wander behaviour, though not a neural endocrine behaviour, is included for completeness and comparison.

<table>
<thead>
<tr>
<th>Description</th>
<th>Behaviour class</th>
<th>Gland class</th>
<th>Decay rate</th>
<th>Stimulation rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obstacle avoid</td>
<td>BehaviourAvoid</td>
<td>GlandAvoid</td>
<td>0.22</td>
<td>1.5</td>
</tr>
<tr>
<td>Cohesion</td>
<td>BehaviourCohesion</td>
<td>GlandCohesion</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>Separation</td>
<td>BehaviourSeparation</td>
<td>GlandSeparation</td>
<td>0.8</td>
<td>0.85</td>
</tr>
<tr>
<td>Seek rubbish</td>
<td>BehaviourSeek</td>
<td>GlandSeekRubbish</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>Seek bin</td>
<td>BehaviourSeek</td>
<td>GlandSeekBin</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>Seek charge</td>
<td>BehaviourSeek</td>
<td>GlandSeekCharge</td>
<td>0.7</td>
<td>1.5</td>
</tr>
<tr>
<td>Wander</td>
<td>BehaviourWander</td>
<td>GlandWander</td>
<td>0.7</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 4.1:

The doCharge function has three phases the approach phase, the charging phase and the retreat phase. Like the doPickup and doDrop approach phases the doCharge phase involves the robot slowly moving towards the charging station, until it is less than 0.07\(m\) away, at which point the recharge phase starts. The recharge phase gradually increments the robots power level until it reaches 100%, at which point the retreat phase starts, in which the robot rotates on the spot by approximately \(\frac{\pi}{2}\) radians, before ending the pattern.

BehaviourNECA provides the transformOutput function for its subclasses, this allows them to transform the output of the network in some way, this was necessary since the neural networks were using the tanh activation function which only output values between one and minus one. The majority of implementations of transformOutput simply involved multiplying the network outputs by some constant.
BehaviourAvoid The BehaviourAvoid class is used to implement the obstacle avoidance behaviour, inputs are
provided by the two sonar devices and outputs are designed to effect the robots motors. The neural network
of the avoid behaviour contains two input nodes, two hidden nodes and two output nodes (plus a bias node in
both the input and hidden layers, as is the case in all the networks of this project). The weights were determined
using the back-propagation algorithm, the training data was created according to algorithm 1 where \( I_{left} \) and
\( I_{right} \) are the inputs, and \( O_{forward} \) and \( O_{angular} \) are the desired outputs. Algorithm 1 is based on two simple ideas,
firstly that a robot should turn in the opposite direction of the nearest obstacle, and secondly that a robot should
slow down the closer it gets to obstacles. It is clear to see how these two ideas are encapsulated in algorithm 1,
on line 3 the smaller the values of \( I_{left} \) and \( I_{right} \), the more negative the output \( O_{forward} \) becomes, which will
contribute to slowing the robot down. On line 4, inputs are mapped to the desired output \( O_{angular} \) simply by
finding the difference between the two inputs, this causes a positive or negative output depending on which
input was smallest.

Algorithm 1 - used to create the obstacle avoidance training data

```plaintext
1: for \( i = 0 \) to 30000 do
2: \( I_{left}, I_{right} \leftarrow \text{random}(0,1) \)
3: \( O_{forward} \leftarrow \frac{I_{left} + I_{right}}{2} - 1 \)
4: \( O_{angular} \leftarrow I_{left} - I_{right} \)
5: end for
```

The BehaviourAvoid class is associated with the GlandAvoid class. The glands activation is inversely
proportional to the distance to the nearest obstacle \( (4.8) \), this means the the activation of the gland will be much
higher, the closer the robot gets to an obstacle.

\[
a_g = \frac{0.9}{\text{nearestObstacle}}
\]  

(4.8)

BehaviourSeparation The BehaviourSeparation class is used to implement the separation behaviour, inputs
are provided form the camera device, specifically information regarding the positions of robots is provided,
and outputs are created for the robot’s motors. The neural network of the separation behaviour has eight input
neurons, two hidden neurons and two output neurons. The eight input neurons correspond to each of the
eight segments of the camera device’s field of view. The weights are determined using the back-propagation
algorithm, trained on data constructed by algorithm 2.

Algorithm 2 - used for creating the separation training data

```plaintext
1: for \( i = 0 \) to 30000 do
2: \( I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7 \leftarrow \text{random}(0,1) \)
3: for \( I_j \) in \{\( I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7 \)\} do
4: \( O_{forward} \leftarrow O_{forward} + I_j \)
5: if \( j < 4 \) then
6: \( O_{angular} \leftarrow O_{angular} + I_j \)
7: else
8: \( O_{angular} \leftarrow O_{angular} - I_j \)
9: end if
10: \( O_{forward} \leftarrow \frac{O_{forward}}{8} - 1 \)
11: \( O_{angular} \leftarrow \frac{O_{angular}}{4} \)
12: end for
```

Though it looks daunting at first, algorithm 2 is very similar to algorithm 1, except that it involves more
input values. Closer inspection of algorithm 2 shows that the inputs which represent the segments on the
left-hand-side of the robot will contribute positively to \( O_{angular} \) (line 6) whereas those on the right will contribute
negatively (line 8), this is identical to what happens in algorithm 1 on line 4. \( O_{forward} \) is also calculated in the
same way as in algorithm 1, it is simply the sum of all the inputs, scaled to a value between 0 and \(-1\).

The BehaviourSeparation class is associated with the GlandSeparation class. Similarly to the obstacle
avoid behaviour, the glands activation is inversely proportional to the distance to the nearest robot \( (4.9) \),
consequently the activation is higher, the closer the robot gets to an obstacle.

\[ a_s = \frac{1.81}{\text{nearestRobot}} \]  

(4.9)

**BehaviourCohesion**  
The **BehaviourCohesion** class is used to implement the cohesion behaviour, inputs come from the camera device, specially information about the positions of cyan robots is provided, and outputs are created for the motors. The neural network has six input neurons, two hidden neurons and two output neurons. The six input neurons correspond to the six segments of the camera's field of view that are in front of or along side the robot, it was decided that those behind the robot would not be taken into account to allow robots at the front of a group to be more focused on reaching the bin. The weights were determined using the back-propagation algorithm, with data training by algorithm 3.

Algorithm 3 - used for creating the cohesion training data

1: \textbf{for} \( i = 0 \) to 30000 \textbf{do}
2: \( I_0, I_1, I_2, I_3, I_4, I_5 \leftarrow \text{random}(0,1) \)
3: \textbf{for} \( I_j \) in \{\( I_0, I_1, I_2, I_3, I_4, I_5 \}\) \textbf{do}
4: \( X \leftarrow X + \text{calcXCoord}(I_j) \)
5: \( Y \leftarrow Y + \text{calcYCoord}(I_j) \)
6: \textbf{end for}
7: \( X \leftarrow X/6 \)
8: \( Y \leftarrow Y/6 \)
9: \( O_{\text{forward}} \leftarrow \sqrt{X^2 + Y^2} \)
10: \( O_{\text{angular}} \leftarrow \text{normalise}(\arctan(X/Y)) \)
11: \textbf{end for}

Algorithm 3 uses a method similar to that of Reynolds (1999) to determine, based on the position of other robots, the best motor commands for cohesion. Cohesion, as described by Reynolds works by first calculating the average position of all an agents neighbours, and then applying a steering force in the direction of that position. Assuming that a robot found in a segment is located directly in the middle of that segment, and because the arrangement of segments is well defined, trigonometry can be used to calculate the approximate, relative, x and y coordinates of a robot's neighbours. Once the average position has been calculated, the \( O_{\text{angular}} \) output is determined by calculating the angle between the robots heading and the average position of its neighbour, this is done using \( \arctan \) (lin 10). The \( O_{\text{forward}} \) is calculated using pythagoras theorem, so the forward speed is larger the further away a robot is.

The **BehaviourCohesion** class is associated with the **GlandCohesion** class. The glands activation is directly proportional to the average distance of a robots neighbours (4.10), and is also determined by whether or not the robot is carrying rubbish, this is included since there would be no reason for a robot with rubbish to follow others to the bin.

\[ a_s = 2.0 \times \text{averageDistance} \times \text{hasRubbish} \]  

(4.10)

**BehaviourSeek**  
The **BehaviourSeek** class is used to implement the remaining three neural endocrine behaviours: seek bin, seek charge and seek rubbish. The network and weights used by the three seek behaviours are identical, but each is associated with a different **Gland** class. In the separate **Gland** classes the functions for calculating the glands activation and the values of decay and stimulation rate all vary.

The **BehaviourSeek** class receives inputs from the camera device, but the specific information provided depends on the behaviour. The information received by each behaviour intuitive, the seek bin behaviour receives locational information about the bin, seek charge receives information about the charging stations and seek rubbish receives information about the rubbish. The neural network of the **BehaviourSeek** class has six input nodes, two hidden nodes and two output nodes. The six input nodes correspond to the same six segments used by the cohesion behaviour, the reason for only using six segments for this behaviour was that in testing if robot attempted to seek an object that was behind it, since its turning circle was not quick enough, it was often seen to be "chasing its tail" and ending up circle around the object. As described earlier only the nearest bin, charging station or rubbish objects are presented as inputs, this greatly simplifies the calculations of the neural network and meant that when designing the network it was possible to calculate the weights by hand. Weights were chosen so that if there was on object in one of the first three camera segments, the output would suggest motor commands to turn the robot left, and if their was on object in one of the final three segments, the output would suggest commands to turn the robot right. The weights were also chosen so that the further away an object was, the faster the robot would move towards it.
When implementing the seek bin behaviour, the `BehaviourSeek` class is associated with the `GlandSeekBin` class. The glands activation depends on whether it can see the bin and whether it is holding rubbish (4.11).

\[ a_g = 2.0 \times \text{seeBin} \times \text{hasRubbish} \]  

(4.11)

When implementing the seek rubbish behaviour, the `BehaviourSeek` class is associated with the class: `GlandSeekRubbish`. The glands activation depends on whether it can see any rubbish and whether it is already holding some (4.12).

\[ a_g = 2.0 \times \text{seeRubbish} \times \neg \text{hasRubbish} \]  

(4.12)

When implementing the seek charge behaviour, the `BehaviourSeek` class is associated with the class: `GlandSeekCharge`. The glands activation depends on whether it can see a charging station, its current power level and its desire to charge (4.13).

\[ a_g = \text{seeCharge} \times \text{desireCharge} \times \frac{2.0}{\text{powerLevel}} \]  

(4.13)
5 Experimentation

This chapter outlines the experiments that were undertaken in order to determine whether the requirements in section 3.5 were met. One hundred and forty individual runs were carried out across two different environments, with varying numbers of robots between one and seven. This chapter starts by introducing the hardware upon which the experiments were run, before going on to describe the two different environments within which the experiments took place. The later sections of this chapter describes the experimental process that was taken, including details of the individual runs, how these runs were set-up, and how data was collected from them.

5.1 Environments

In order to test the adaptability of the system it was necessary to test the performance of the robots in two different environments, these are described below. Both of the environments were designed with the capabilities of the robots in mind, for example, it was known that because the robots had only two sonar sensors, both of which were located at the front, they would struggle to find their way out of concave obstacles with small internal angles. When faced with concave obstacles robots can be indecisive about which way to turn and in the end may either end up stalling or crashing into the obstacle. Another deficiency caused by the poor sonar coverage is that if an obstacle is too small (smaller than the width of the robot) when a robot approaches it head on, its sonar devices will not recognise it up and the robot will crash, because of these problems, both of the environments were designed to contain no concave obstacles (with small internal angles) and no obstacles smaller than the width of a robot.

World 1

The first environment, referred to as world 1, can be seen in figure 3.1, it contains a single bin, shown by the blue square; three charging stations, represented by the green hexagons; and twenty pieces of rubbish, depicted as small red squares. The world was made deliberately challenging by placing the bin in the centre of the environment and surrounding it with obstacles. The reason for placing the bin in a difficult position, was the expectation that to reach it, robots would fair better if they cooperated with each other, for example by signalling and flocking.

World 2

The second environment, referred to as world 2, is identical to world 1 in terms of its bounding walls and the positioning of objects, such as the bin and the charging posts. The only difference between world 2 and world 1 is that world 2 does not contain the obstacles or chambers, like those found in the centre of world 1.

5.1.1 Hardware

All experiments were carried out on an Apple MacBook1 (Apple Inc, 2008), running Ubuntu 8.04 (Canonical Ltd, 2008), the technical specifications of the hardware are listed in table 5.1.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>2GHz</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>2MB</td>
</tr>
<tr>
<td>System Bus</td>
<td>667MHz</td>
</tr>
<tr>
<td>Memory</td>
<td>512MB of 667MHz DDR2 SDRAM (PC2-5300)</td>
</tr>
<tr>
<td>Hard Drive</td>
<td>60GB 5400-rpm Serial ATA</td>
</tr>
</tbody>
</table>

Table 5.1: Hardware specification of the Apple Macbook1 upon which all of the experiments were carried out.
5.2 Experiments

One hundred and forty experiments were run in total, seventy of which took place in world 1 and seventy of which took place in world 2. The groups of experiments that took place in the same worlds were sub-divided further into sets of seven, in each set, rubbish pieces were placed in exactly the same positions and for each run the number of robots was varied between one and seven.

Each experiment ran for twenty minutes, counted by the clock of the simulator, or until all of the pieces of rubbish were successfully collected. Because of the large number of experiments it took several days to complete them all and so their execution was spread over many sessions. To try and ensure the experiments were fair and free from bias, between each session conditions were replicated as best as possible, each session was started from a cold boot, networking was disabled and only essential processes were running during the experiments, furthermore all experiments from the same set of seven were always carried out within the same session.

5.2.1 Setup

In each experiment, robots started from the same positions, evenly spaced around the bin as shown in figure 3.1. Each experiment contained twenty pieces of rubbish for the robots to collect, these were randomly positioned using a specially created Python (Python Software Foundation, 2009) script. The script randomly chooses a position within the environment and presents it to the user, who verifies that the suggested position is not located on, or too close to, an obstacle, a wall, a charging station or the bin. The process is repeated for all twenty pieces of rubbish, and each of the ten different sets of seven experiments in each world, hence there were twenty different rubbish configurations in total. The script and others used in throughout the experimentation process are included in appendix B.

5.2.2 Data collection

The following data was collected from each of the robots involved in an experiment:

- the number of pieces of rubbish the robot picked up
- at what time, in seconds, each piece of rubbish was picked up
- the number of times the robot recharged itself
- the amount of power used by the robot.

Each robot kept track the relevant information throughout their execution. As well as controlling the robot, each client program contained a section of code that was executed once the simulator was closed to save all of the robots information to a text file.

5.2.3 Hypotheses

- It is expected that the robots will perform well up until a point at which interference will start to play a part and a drop in the performance of the system will be observed.
6 Results and Evaluation

This chapter starts by evaluating the system in general, first looking at the effects that the hormones have over the system before moving on to discuss the individual behaviours. Each of the eleven programmed behaviours and the two emergent behaviours shall be discussed, videos, screenshots and graphs are provided to aid in these discussions. All of the unforeseen problems that arose during the experiments shall be detailed, as well as the unexpected advantages. The results of the experiments outlined in chapter 5 shall be analysed including specific details of how the performance of the system changed as the number of robots was varied. Finally the requirements from section 3.5 shall be revisited in order to help assess the achievements of this project.

6.1 Hormone levels

It is very hard to tell simply from observing, how the hormones of the system are actually effecting the behaviours of the robots. To help describe the influence that the hormones have on the system and to provide evidence that they are producing the desired effects, figures B.3 and B.4 were created. Figure B.3 shows the levels of a single robot’s bin and rubbish hormones, as well as their “desires” to find the bin and rubbish, over a period of 450 seconds. During the period covered by the graph the robot successfully picks up three pieces of rubbish and deposits them in the bin. The first piece of rubbish is picked up at $t = 40$ and is almost immediately dropped into the bin, this can be seen in the graph by the sudden drop in the rubbish hormone as soon as has the rubbish has been picked up and the sudden increase in bin hormone when the bin is detected. The second piece of rubbish is picked up at $t = 73$ and dropped at $t = 192$, between which a gradual increase in the robots desire to find the bin is seen. It is clear that the robots desires to find the bin and rubbish are having the correct effect on the wander hormone, since as the desires increase so too does the concentration of the wander hormone. The final piece of rubbish is picked up at $t = 360$ and dropped at $t = 403$.

Figure B.3 shows only a small amount of the hormones in the system, in figure B.4, for the same robot and time period every hormone and desire of the system can be seen. It is hard to fully analyse a graph with so many components (which is why some parts were extracted into B.3) but one thing that can be seen figure B.4 is that there is a sharp increase in the cohesion hormone right before the first and second pieces of rubbish were dropped in the bin, showing that, on those occasions the robot was drawn towards the bin by following others.

6.2 Behaviours

The system designed here consisted of eleven behaviours in total, five of which were neural endocrine behaviours, five of which were strictly not neural endocrine behaviours and one of which contained an endocrine part but involved no artificial neural network. All of the behaviours were observed to successful complete the actions they were designed to perform, videos were created of each of them in operation, the details of which are included in appendix C. In this section each of the behaviours shall be evaluated in turn, looking at both the situations where they performed well and the situations where they performed poorly.

6.2.1 Basic behaviours

Pickup rubbish  The pickup rubbish behaviour performed well in most situations, when they attempted to, robots were almost always successful at picking up a piece of rubbish. Though rare, there were some situations where the robot would overshoot the rubbish on its approach and hence miss when attempted to grip it, in this situation the robot would simply carry on forward looking for another piece of rubbish. Sometimes the robot would undershoot a piece of rubbish, but the consequences of this were nominal, since the rubbish was still in front of them they would simply shuffle forwards and try again.

Drop rubbish  The drop rubbish behaviour performed well most of the time, when the command was issued the robot would always drop whatever it was holding. Sometimes however, the actual command to drop rubbish was issued in the wrong situation, i.e. when the robot was not inside the bin. This was not so much a problem with the drop rubbish behaviour but rather the method used by the robots to determine whether they were inside the bin. In the simulator the bin was represented as a blue square, though from the topological view it
appears that the whole area the bin encompasses is coloured blue, this is not how the robots see it. Robots see only the boundaries of the bin and so it is quite difficult for them to determine whether they are looking at a boundary from inside or outside of the bin. To help robots decide if they were inside or outside of the bin they maintained a series of states that cycled through: seeBin, onBinBoundary, enteringBin, inBin, leavingBin, onBinBoundary, !inBin. The problems arose if there was a piece of rubbish very near or on the bin boundary, if a robot picked up this piece of rubbish it might not be able to determine when it was on the bin boundary and so once inside the bin the robot might think the boundary that it can see leads into, rather than out of the bin. Though infrequent this problem sometimes led to robots dropping rubbish just outside of the bin, this was not a major problem though as the next time a robot came along it would be able shift the stray piece of rubbish into the bin. It is easy to blame this problem on the simulators representation of objects, however in the real world sensors are not perfect, the fact that this problem arose actually adds a bit more realism to the simulation and the fact that the robots are able to function well despite its presence shows the robustness of the system.

Recharge The recharge behaviour always performed well without fail. There were some problems with congestion at the bin, but this a problem with the seek bin behaviour and so are discussed later.

Signal bin The signal bin behaviour performed well most of the time, however because of the same problem that effected the drop rubbish behaviour, sometimes the robots would believe that they were still inside the bin and hence signal, even if they had left. This meant that robots would distract others throughout the environment, and since the robot believed to be inside the bin it would ignore the task of picking up rubbish. This problem was not persistent though since the next time the robots saw the bin they were able to recognise that they were outside of it.

Wander The wander behaviour performed as expected but since the results of the behaviour are not immediately observable, it is hard to qualitively evaluate its performance, and it would not be sensible to attempt to do so without further experiments which unfortunately there was not time for.

6.2.2 Neural endocrine behaviours

Obstacle avoid The obstacle avoid behaviour worked very well most of the time, however due to the problem anticipated in section 5, the poor sonar coverage of the robots meant that sometimes they would not sense an object until it was too late to avoid it. This problem could be solved by implementing a recovery behaviour, which could take the form of a fixed-action-pattern that is activated when a robot detects that it had crashed. The stimulus for the behaviour could come from a bumper sensor and the response could be the robot reversing for a short period before turning on the spot and continuing forward.

Separation The separation behaviour worked very well and collisions between robots were infrequent. One problem was observed when combining the separation behaviour with the obstacle avoidance behaviour. When two robots were approaching a wall at an angle, from opposite directions, their obstacle avoidance behaviour would attempt to turn them away from the wall, but their separation behaviour would attempt to turn them away from the other robot and hence into the wall, this indecision would sometimes lead to the robots colliding with each other. The indecision problem was reduced by giving the hormone of the obstacle avoidance behaviour a high stimulation rate and a low decay rate, and the separation behaviour fairly high values of both.

Cohesion The cohesion behaviour worked well, sometimes though, as will be mentioned when discussing the emergent behaviours it could have benefited from imposing a bit more of its influence on the global behaviour of the system.

Seek rubbish and seek bin The seek rubbish and seek bin behaviours always worked well and were free from obvious problems.

Seek charge The seek charge behaviour worked the majority of the time, however it would sometimes cause collisions at the charging stations. The charging stations were not restricted by the amount of robots that could charge at them at any moment in time, it was expected the the separation behaviour alone would provide enough of a deterrent to prevent too many robots from charging at the same time. Unfortunately the separation behaviour did not always prevent robots from attempting to dock at a busy charging station, and robots would sometimes try and dock between two other robots where there was not enough room, this usually rendered at least two of the robots immobile. This problem indicates that the strength of the seek charge hormone was too high, and figure B.4 certainly backs up this claim.
6.2.3 Emergent behaviours

Following The “following” behaviour, that is the combination of cohesion, separation, signal bin and seek bin, performed well the majority of the time. Often the behaviour appeared to be very useful in helping robots find the bin, which the results in the next section confirm. One problem with the following behaviour was that once the leader of a group saw the bin its seek bin behaviour would often cause it to speed up, on occasion this caused robots at the back of the group to lose sight of those in front, this problem could be reduced by changing the parameters of the cohesion hormone or increasing the strength with which the behaviour responds.

Dispersion The dispersion behaviour, that is the combination of separation, wander and obstacle avoidance, which causes the robots to explore as much of the environment as possible, worked reasonably well for both of the worlds in which the experiments took place. Figures B.1 and B.1 show the area covered by the robots in both world 1 and world 2 over a period of 160 seconds. To emphasise the effect of the dispersion behaviour alone, when creating figures B.1 and B.1 the robots movements were only effected by their separation, wander and obstacle avoidance behaviours. Figure B.1 shows the area covered in world 1, as can be seen the coverage was generally quite good except for the large chamber just to the right of centre, which only one robot visited over the whole period. Figure B.2 shows how the robots moved in world 2 and shows a fairly good spread of coverage across the entire environment. Both figures demonstrate that the dispersion behaviour has the desired effect and achieves a good coverage throughout the environment.

Peer pressure One final behaviour observed that was neither programmed nor anticipated, is referred to here as peer pressure. Similar to dispersion, but specific to more localised regions, peer pressure is defined to be: the observed behaviour of robots following paths that they would not normally take, as a result purely of the presence of other robots. The behaviour has both positive and negative effects, the negative effects simply amount to the problem of interference, which was expected. The positive effects of the behaviour can be seen when for example in world 1 two robots are travelling towards each other around the outskirts of the world. The desired path of robots around the outskirts, is to be equidistant from any surrounding walls, this is evident in figure B.2, since many pieces of rubbish are located near to the walls peer pressure is helpful in pushing robots off the usual path and closer to the rubbish.

6.3 Known deficiencies

There are two further problems that were not mentioned in the behaviours section that shall be covered here. Firstly, due to a lack of support in the simulator the system implemented does not accurately measure the power consumption of the robots. The simulator included documentation for a power monitoring device but the implementation of the device was incomplete for the version of the simulator used by this project. Unfortunately there was not enough time to attempt to finish the implementation and so instead a very crude power monitoring system was created from scratch. The device created simply decrements a counter at every time step regardless of the current actions of the robot, this is clearly not an accurate representation of a real robot since it doesn’t take into account: whether the robot is moving, at what speed it is moving and which sensors or actuators are being used, all of which will cause the robot to consume power at different rates. Regardless of the simplicity it was still necessary and useful to include the power device so that the effects of the power seek and recharge behaviours could observed. The power consumption of each robot and the amount of times they charged per run were measured but because of the over simplified design no evaluation of their results is carried out in this section.

The second problem relates to that mentioned in the previous section with regard to the simulators representation of the bin. As well as it being difficult for robots to determine if the are inside or outside of the bin, it is also not possible for them to see through the boundaries of the bin, and hence they cannot tell if there is another robot on the other side which, as will be discussed in the next section, can lead to collisions.

6.4 Analysis of results

In this section the results from the experiments outlined in chapter 5 are detailed and analysed. Experiments were carried out across two different environments, the results from each are first analysed separately, before comparisons and further analysis is made between the two. Three different measures are used in this section in order to help evaluate the system. Firstly the success of the system is measured, secondly the speedup is measured and finally, from the speedup the efficiency of the system is calculated. Each of these measures is introduced briefly including details of how they were calculated, before the results themselves are analysed.
It should be noted that although a large amount of experiments were carried out in total, at the level where most of the comparisons were made, that is between the different number of robots within each of the two worlds, the number of experiments was relatively few. Because of this the results can only be considered from a high level.

6.4.1 Success

The success of the system is measured in terms of the amount of rubbish that was collected. Graphs are presented to show how the success of the system changed as the number of robots was varied. The total amount of rubbish collected by the group as a whole, as well as the number of pieces collect per robot are analysed.

6.4.2 Speedup

The speedup measures how much faster multiple robots are at the task over single robots. This measurement is used to determine how effective the collaborative behaviours of the system were. The speedup achieved using \( r \) robots over a single robot (\( S_r \)), is calculated using (6.1), where \( T_i \) is the time taken for \( i \) robots to complete a certain task. Graphs are plotted showing how the speedup varies with the number of robots.

\[
S_r = \frac{T_1}{T_r}
\]  

(6.1)

6.4.3 Efficiency

The efficiency in terms of time is measured, which is closely related to the speedup of the system and is calculated using (6.2). Graphs are plotted showing how the efficiency of the system varies as the number of robots changes.

\[
E_r = \frac{S_r}{r}
\]  

(6.2)

6.4.4 World 1

World 1 was the most challenging environment and included many obstacles and walls. It was hypothesised in the previous chapter that this is where the collaborative behaviours of the robots would prove their worth, as will be seen shortly the results did in fact confirm this.

Success

Figures 6.1a and 6.1b show the success of the robots after periods of 300 and 1200 seconds respectively. Each boxplot shows the results of ten different runs with ten different starting positions for the rubbish. Both the graphs show a strong positive correlation between the number of robots and the number of pieces of rubbish collected up, until the case where five robots were used, at which the performance starts to level out and even drops in figure 6.1b. The levelling out is expected in 6.1b since the maximum number of pieces that can be collected is twenty, but the fact that it is observed in 6.1a and that the performance drops in 6.1b indicates that interference starts to have an effect after five robots. The case with five robots also had the smallest interquartile range showing that five robots not only performed the best, but did so consistently.

The first outlier in 6.1b, where the number of robots was three and the number of pieces picked up was six, was caused by one robot crashing, and the other stop robots crashing into the obstruction that the other robot formed, which emphasises the importance of redundancy in multi-robot systems. The outlier where the number of robots was five and the number of pieces collected was sixteen can be attributed, at least partly, to the simulator and the way the bin is represented. Since robots cannot see the inside of the bin from the outside, there is always the danger that collisions can occur as one robot travels into and one robot travels out of the bin, this is what happen in case of this outlier, two robots crashed whilst entering and leaving the bin which meant that when other robots came to drop rubbish there was a pileup effect. Only one other crash at the bin was observed in the seventy experiments of world 1, again for an experiment involving five robots however in this case it did not involve all of the robots and two were able to continue functioning, resulting in an impressive nineteen pieces being collected.

Figure 6.2 shows the number of pieces of rubbish collected per robot after 300 and 1200 seconds, as to be expected, in both graphs the number of pieces drops as more robots are added. What is interesting about figure 6.2b is that the smallest interquartile range is observed for the case where there were five robots, showing that a
Figure 6.1: Graphs showing the number of pieces of rubbish collected over periods of 300 (a) and 1200 (b) seconds, with varying numbers of robots between one and seven

group of five robots is most consistent on an individual level as well as a group level as indicated by figure 6.1b. The outliers in figure 6.2b relate to the same runs as in figure 6.1b.

Speedup

Figure 6.3a shows the average speed up that multiple robots have over a single robot (blue line) alongside the projected linear speedup (red line). The speedup was calculated using the average time it took the groups of robots to collect six pieces of rubbish, this number was chosen since it was the minimum number of pieces that any group of robots managed to pick up from the seventy experiments in world 1. Figure 6.3b shows that apart from groups of three robots, for the time taken to collect the first six pieces of rubbish the system produced a greater than linear speedup on average. There is a drop in the speedup when the number of robots is three, five and six, this is caused by the outliers that can be seen in 6.1 and 6.2.

Efficiency

Figure 6.3b shows the average efficiency of the system as calculated by 6.2. The graph indicates that on average the system was most efficient at collecting the first six pieces of rubbish when groups of three robots were used. As was noted earlier relatively few samples were used and so these results are subject to scrutiny. To determine whether the apparent improved efficiency between the use of a single and multiple robots was statistically significant a Wilcoxon rank-sum test was carried out. The hypothesis of the test used is given shortly (6.3), where $T_1$ is the sample of results for the time taken for a single robot to collect six pieces of rubbish, and $T_m$ is the sample of results for the time take for $m$ robots to collect six pieces of rubbish.

\[
\begin{align*}
H_0 : T_1 \text{ and } T_m \text{ have different distributions} \\
H_1 : T_1 \text{ and } T_m \text{ have different distributions}
\end{align*}
\] (6.3)

The results of the test are given in table 6.1 and show that at a 95% confidence level, when the number of robots is greater than three it is possible to reject $H_0$ and accept $H_1$.  

58
Figure 6.2: Graphs showing the number of pieces of rubbish collected per robot over periods of 300 (a) and 1200 (b) seconds, with varying numbers of robots between one and seven.

Figure 6.3: Graphs showing the average performance speedup (a) and the average efficiency (b) of robots at collecting the first six pieces of rubbish.
### Table 6.1:

<table>
<thead>
<tr>
<th>No. of robots</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.1212</td>
</tr>
<tr>
<td>3</td>
<td>0.1212</td>
</tr>
<tr>
<td>4</td>
<td>0.0013</td>
</tr>
<tr>
<td>5</td>
<td>0.0017</td>
</tr>
<tr>
<td>6</td>
<td>0.0010</td>
</tr>
<tr>
<td>7</td>
<td>0.00024</td>
</tr>
</tbody>
</table>

#### 6.4.5 World 2

World 2 was the simplest world, it contained to obstacles and no walls other than those bounding the arena. It was expected that the robots would show a marked improvement over the performance of robots in world 1, this was in fact the case as will be seen in this subsubsection.

**Success**

The success of the system was much greater in world two with all runs collecting 14 pieces of rubbish or more, and 51 out of the 70 collecting all twenty within the time limit of 1200 seconds. The success of the system after 1200 seconds does not show any quantity of robots to perform noticeably better than any other and so only the success of the system after 300 seconds is considered here. Figures 6.4a and 6.2a show the number of pieces of rubbish picked by a group of robots, and the number of pieces of rubbish picked up per robot after 300 seconds. From figure 6.4a it can be seen that the success of the system improves as more robots are added, up until six robots where the median is at the maximum of 20 and the mean is at its highest for all of the runs in world 2. As can be seen in figure 6.4a, the results of experiments that involved six robots had the smallest interquartile range, indicating that for world 2, six robots achieved the greatest and most consistent amount of success.

Figure 6.4b shows the number of pieces of rubbish picked up per robot after a period of 300 seconds. The experiments with six robots are again seen to be the most consistent, which could indicate that beyond six robots interference starts to effect the system.

The three outliers that appear in figures 6.4a and 6.4b belong to the same runs. The outlier that is seen in the experiment with four robots which after 300 seconds had collected eleven pieces of rubbish, was caused by two robots crashing in the bin and the other two later joining them. The outlier in the experiment with five robots was caused again by two robots crashing in the bin, two more robots later joined the crash in the bin, one robot survived but appeared to struggle to find the bin without the others to aid it. The outlier in experiment six was caused by a single robot crashing into a wall whilst carrying a piece of rubbish, the other robots went on to collect nineteen pieces of rubbish but the experiment still appears as an outlier, even after 1200 seconds, because all the other experiment involving six robots performed so well, collecting all twenty pieces of rubbish.

**Speedup**

Figure 6.5a and 6.5b show the average speedup that multiple robots provided over a single robot in world 2. Since the robots were more successful in world 2 it was possible to test the speedup over a larger task, in figure 6.5b the speedup over the time taken to collect fourteen pieces of rubbish is shown. To allow comparisons to be made with the experiments of world 1 the speedup achieved over the time taken to collect six pieces of rubbish was also plotted, and can be seen in figure 6.5b. Both graphs show a linear speedup, although in figure 6.5a the speedup is consistently greater than linear, the amount is not significant.

**Efficiency**

Since the speedup is effectively linear, the efficiency of the system in world 2 remains constant as the number of robots is increased and so consequently does not warrant a graph.

#### 6.4.6 Comparisons

In terms of the success of the system, the results of the experiments from both world 1 and world 2 showed there to be an optimal value for the number of robots for specific amounts of time. World 1 showed that after 1200 seconds five robots were the most consistent both in terms of the success of the system and the distribution of work amongst the robots. World 2 showed similar findings after 300 seconds for six robots. These results
Figure 6.4: Graphs showing the number of pieces of rubbish collected (a) per robot (b) over a period of 300 seconds, with varying numbers of robots between one and seven.

Figure 6.5: Graphs showing the number of pieces of rubbish collected (a) per robot (b) over a period of 300 seconds, with varying numbers of robots between one and seven.
indicate that beyond five robots in world 1 and six robots in world 2 the interference between robots starts to have a detrimental effect on the system. The fact that the optimal value was observed with five robots in world 1 and six in world 2 adds weight to the argument since the area of world 2 is larger and free from obstacles it would be natural for interference to become a factor with more robots. To fully confirm this proposal more experiments would need to be carried out with more robots.

Both systems were able to show at least a linear speedup when using multiple robots over a single robot, the speedup was much more significant in world 1 with four robots showing the biggest increase. The fact world 1 showed such a performance increase is attributed to the fact that it was a more challenging environment, for the robots to perform well in world 1 they needed the help of other, whereas in world 2 where no obstacles were found it was always possible for robots to succeed even without the presence of others.

Outliers in the results had a noticeable effect on some of the measurements, in particular the speedup measured in world 1, at one point shows a big drop which a single outlier is responsible for, this adds further support to the suggestion that more experiments would be beneficial.

6.5 Satisfication of Requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>How satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>All nine of the individual behaviours were successfully and effectively exhibited by the robots, as described in sections 6.2.1 and 6.2.2.</td>
</tr>
<tr>
<td>F2</td>
<td>Individual robots were able to successful complete the rubbish collection task, cases in which they did so can be seen in the graphs throughout section 6.4</td>
</tr>
<tr>
<td>F3</td>
<td>Both of the collaborative behaviours were successfully exhibited by the robots, as described in section 6.2.2.</td>
</tr>
<tr>
<td>F4</td>
<td>Both of the emergent behaviours were successfully exhibited by the robots, as described in section 6.2.3.</td>
</tr>
<tr>
<td>F5</td>
<td>Groups of up to seven robots were able to successfully complete the rubbish collection task, cases in which they did so can be seen in graphs throughout section 6.4</td>
</tr>
<tr>
<td>F6</td>
<td>When one or more robots failed, on many occasions the other robots were able to carry on and successfully complete the task of rubbish collection without them, on some occasions though, as was discussed in section 6.4 the failure of one robot could have catastrophic effects on the rest of the system.</td>
</tr>
<tr>
<td>N1</td>
<td>Experiments in both world 1 and world 2 showed that multiple robots produced a speedup over a single robot at the task of rubbish collection</td>
</tr>
<tr>
<td>N2</td>
<td>Experiments in both world 1 and world 2 showed that multiple robots produced a linear speedup over a single robot at the task of rubbish collection</td>
</tr>
<tr>
<td>N3</td>
<td>Experiments in both world 1 showed that multiple robots produced a greater than speedup over a single robot at the task of rubbish collection</td>
</tr>
</tbody>
</table>

6.6 Summary

All of the eleven programmed behaviours were shown to be effective, as were the two collaborative behaviours and the two emergent behaviours, an extra behaviour referred to as peer pressure was observed which had a positive effect on the system. The key result from the experiments was the faster than linear speedup found in world 1 when using multiple robots, in world 2 the speedup was seen to be linear which is impressive in itself. All of the requirements were met to some degree.
7 Conclusions

In this chapter the main aims of this project are revisited and the ways in which they were met are detailed and summarised. Some suggestions are made for future extensions to this work or new work in a similar vein. As introduced in section 1 the main aims of this project were:

1. To investigate whether the neural endocrine control architecture is capable of controlling a multi-robot system
2. To investigate how effective the architecture is at controlling a multi-robot system
3. To investigate the capabilities of the architecture at a new and complex task

7.1 Project aims

In terms of meeting the project aims, the neural endocrine architecture implemented in this project was not only seen to be capable of controlling the agents of a multi-robot system, but it was shown to do so in an extremely effective manner. For one situation the system showed nearly a 100% increase in efficiency using multiple robots, when compared to a single robot. In world 1, the most complicated environment, as the number of robots was increased a greater than linear speedup was observed for the time taken to collect the first six pieces of rubbish. In world 2, the simpler environment, a linear speedup was consistently seen in the performance, this is in itself an achievement and could be an indication that the robots were able to nullify the effects of interference, however it could also mean that the population density was not great enough for interference to take place, further experiments are required to investigate interference, either in situations with more robots or a smaller world.

The architecture handled the complex design, that included eleven unique behaviours, with little trouble. Many of the problems that were come across across during the implementation and testing of the system related to the simulator rather than a deficiency in the neural endocrine architecture. Combining behaviours was tricky in some instances, combining the separation behaviour with the obstacle avoidance behaviour in particular was challenging but once it was achieved the system performed very well. The task of rubbish collection including the requirement to dock and recharge, was more challenging than other tasks that the neural endocrine architecture has been set to before, but the architecture proved itself capable.

7.2 Requirements

All six of the functional requirements were met by the system and all of the non-functional requirements were met by the experiments that took place in world 1, in world 2 only the first two non-functional requirements were met since the system did not show a faster than linear speedup.

7.3 Future work

There are many suggested directions which could be taken in the future, the inclusion of an artificial immune system has been one of the main aims of the neural endocrine control architecture from the start and if achieved could produce an even more adaptable system, as might the on-line-learning techniques currently being investigated by Timmis et al. (2009).

All of the work in this project was done in simulation, it would be very interesting to see how well a complex multi-robot system such as that presented here performed in the real world on real hardware, whilst attempting to achieve a similar collaborative task.

The promise shown in this project provides motivation for further work into the use of the architecture as a method of control for multiple robots. The effectiveness of the collaborative behaviours and whether they are solely responsible for the performance increase shown in this work requires further investigation, the performance of the system in new more challenging environments also warrents investigation.
Bibliography


64


A Diagrams

Figure A.1: Overview of the system
B Experiments

Figure B.1: The distance covered by seven robots in world 1 after 160 seconds

Figure B.2: The distance covered by seven robots in world 2 after 160 seconds
Figure B.3: Graph showing the levels of a single robot’s bin, rubbish and wander hormones; along with their desires to find rubbish and the bin. Over 450 seconds the robot picks up and drops off three pieces of rubbish.
Figure B.4: Graph showing all the hormone and desire levels of a robot over a period of 450 seconds.
C Submitted electronically

The files listed in the table below were submitted electronically:

<table>
<thead>
<tr>
<th>Filename</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>avoid.mpg</td>
<td>Video of the obstacle avoidance behaviour</td>
</tr>
<tr>
<td>seek_bin.mpg</td>
<td>Video of the seek bin behaviour.</td>
</tr>
<tr>
<td>seek_rubbish.mpg</td>
<td>Video of the seek rubbish behaviour</td>
</tr>
<tr>
<td>seek_charge.mpg</td>
<td>Video of the seek charge behaviour</td>
</tr>
<tr>
<td>wander.mpg</td>
<td>Video of the wander behaviour</td>
</tr>
<tr>
<td>cohesion.mpg</td>
<td>Video of the cohesion behaviour</td>
</tr>
<tr>
<td>separation.mpg</td>
<td>Video of the separation behaviour</td>
</tr>
<tr>
<td>pickup_rubbish.mpg</td>
<td>Video of the pickup rubbish behaviour</td>
</tr>
<tr>
<td>drop_rubbish.mpg</td>
<td>Video of the drop rubbish behaviour</td>
</tr>
<tr>
<td>recharge.mpg</td>
<td>Video of the recharge behaviour</td>
</tr>
<tr>
<td>signa_bin.mpg</td>
<td>Video of the signal bin behaviour</td>
</tr>
</tbody>
</table>