The Suitability of the Dendritic Cell Algorithm for Robotic Security Applications

Robert Foster Oates MEng.

A Thesis presented for the degree of
Doctor of Philosophy

Automated Scheduling Optimisation and Planning

Research Group
School of Computer Science
University of Nottingham
England

January 2010

Dedicated to

my parents. They inspired me to start this journey and supported me as I made it. For that, and so much more, I am eternally grateful.

The Suitability of the Dendritic Cell Algorithm for Robotic Security Applications

Robert Foster Oates

Submitted for the degree of Doctor of Philosophy
January 2010

Abstract

The implementation and running of physical security systems is costly and potentially hazardous for those employed to patrol areas of interest. From a technial perspective, the physical security problem can be seen as minimising the probability that intruders and other anomalous events will occur unobserved. A robotic solution is proposed using an artificial immune system, traditionally applied to software security, to identify threats and hazards: the dendritic cell algorithm. It is demonstrated that the migration from the software world to the hardware world is achievable for this algorithm and key properties of the resulting system are explored empirically and theoretically. It is found that the algorithm has a hitherto unknown frequency-dependent component, making it ideal for filtering out sensor noise. Weaknesses of the algorithm are also discovered, by mathematically phrasing the signal processing phase as a collection of linear classifiers. It is concluded that traditional machine learning approaches are likely to outperform the implemented system in its current form. However, it is also observed that the algorithm's inherent filtering characteristics make modification, rather than rejection, the most beneficial course of action. Hybridising the dendritic cell algorithm with more traditional machine learning techniques, through the intro-

duction of a training phase and using a non-linear classification phase is suggested as a possible future direction.

Acknowledgements

I count myself very fortunate to have received contributions from a great number of very talented people.

The staff of the ASAP and IMA research groups have been friends, advisors and inspirations. People who deserve a special mention include Julie Greensmith, who interested me in artificial immune systems in the beginning; William Wilson and Peer-Olaf Siebers who always made themselves available for discussion and advice; my office mates, Phil Birkin, Jan Feyereisl and Feng Gu for sharing their technical experience and excellent company; and both Sam Allen and Sven Groenemeyer who were generous with their time, hospitality and insights. Thanks must also go to my supervisors Graham Kendall and Jonathan M. Garibaldi for the time and energy they spent throughout this project.

I have also been incredibly lucky to have been welcomed into both the robotics and artificial immune systems communities. From robotics, the staff from the School of Information Technology and Engineering at The University of Queensland were generous hosts and invaluable colleagues, in particular, Gordon Wyeth and Michael Milford made Chapter 6's work possible. From artificial immune systems Thomas Stibor from The University of Munich deserves a special mention for his fascinating conversations and challenging arguments. Without his enthusiasm and hard work, Chapter 7 would not have happened.

It would be amiss of me to not thank M. Imran from The University of

Durham for publishing the LaTeXtemplate that provided the basis for this thesis and Markus Hammonds who provided the scale, vector graphics rendering of the robot pen from Chapter 3. Finally, many thanks to MobileRobots Inc. for financially supporting this project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"...as if a shuttle should weave of itself, and a plectrum should do its own harp playing." - Aristotle, The Politics [circa 230 B.C.]

1.1 Motivation

1.1.1 Physical Security Systems

In financial terms, building security is a large business. In 2006 the UK Security Industry Authority (SIA) estimated the private security industry to be worth approximately 4 billion in the United Kingdom alone [128]. The roles of security staff are diverse. However primary goals include the prevention of unauthorised intrusion, the inspection of rooms for missing items and the detection of dangerous events, such as fires or other factors that could compromise a building's integrity. All of these goals can be summarised under the broader heading of "the detection of anomalous events".

The current solution to this problem set is "manned security". This describes a solution where a number of security staff patrol a building looking for anomalous events. This is not without its hazards. Coming between a malicious intruder and their goal obviously carries a great deal of risk. For this reason amongst others, hybridisation of manned security systems with static sensor networks is a common practice. Most basically, this occurs in the form of using closed-circuit television (CCTV) to augment a staffed solution. These sensors provide additional decision-making data for the staff and also record events in a manner that can be admissible as evidence for court proceedings and/or investigations. In addition, CCTV allows a single guard to monitor multiple points of interest regardless of their physical location. While CCTV is the most common static sensor type, others are widely used, such as X-ray machines, chemical sensors, water sensors and smoke alarms.

However, existing building security solutions are not without their problems. Systems making use of CCTV suffer from "information overload" and the detection rate of "situations of note" reduces dramatically

as the number of screens observed by a single operator rises. Tickner et al. [137] estimated that the detection rate for a human operator fell from 83% for a four camera system, to 64% for a 16 camera system. In addition, the quality of a static sensor system is obviously highly dependent on the initial placement of the sensors. This is highlighted by the breadth of work which has been done in the field of optimum sensor placement algorithms and is typified by computer science's classic "Art Gallery Problem" where the objective is to identify the minimum number of static "guards" required to provide full coverage of an irregular shaped room [112]. This is made more complex when non-visual sensors are also used. Devices such as mass-spectrometers, X-ray scanners and environmental detectors typically have a very limited effective range. As a result, in order for such sensor types to be useful, they are required to be placed in bottle necks (e.g. security checkpoints at airports) or at specific points of concern, (e.g. water sensors on the ground floor of buildings prone to flooding and smoke alarms in hallways and stairways.) Once the location of static sensors has been committed, people with the intention of maliciously bypassing them can attempt to find alternative routes in order to avoid detection. Where it is not possible to circumnavigate a static sensor, they are vulnerable to problems of occlusion and frame of reference. In situations where the sensor's line of sight is occluded or the frame of reference is insufficient to provide data about a specific target, the majority of static sensors are limited to simple pan, tilt, zoom operations which are not sufficient for all scenarios.

Wholly manned security solutions have additional drawbacks. Manned security is obviously reliant on rest-periods for the people involved and the results and quality of detection are highly variable as a result. For the purposes of investigations, a wholly manned security solution has no stored log of events other than the highly-subjective human memory.

During regular operation, security staff are vulnerable to social prejudices that could make them less effective at stopping certain groups of people. In addition, manned security has considerably more running costs than other solutions as it requires labour and training of staff.

1.1.2 Robotic Security

A solution to the building security problem that has been explored in the literature is a robotic system. While the focus on the specific aspects of the problem change within the literature, there is some agreement on the basic architecture such a solution would have. This basic architecture is similar to a static sensor system, with a human guard at a control station, observing the outputs of autonomous vehicles [20,32,115]. Implemented naively this solution could potentially make the system worse, as attempting to observe situations of interest is a more complex task with a moving image than a static one. However, an autonomous solution would mean that a certain amount of local processing could be done on the sensor inputs and situations of interest could be highlighted automatically. This level of intelligence could be implemented for a static sensor system, however there are further advantages that a robotic solution can offer, which a static solution cannot. Primarily, the sensors can be moved around the situation of interest, providing a much richer data set. This can be used to overcome occlusion and to move sensors with a limited effective range closer to the situation as it unfolds. Problems with bypassing sensors are also reduced, as an autonomous solution means that the "blind spots" of the system are constantly moving. Work has been done on increasing the unpredictability of the patrol routes for such systems, to take full advantage of this property [80]. Obviously an autonomous solution is not reliant on rest periods, has an electronic record of events and is not vulnerable to the social prejudices of a wholly

manned solution. A robotic security guard would also have sociological benefits. The nature of the security application is such that a robot would be minimally invasive into the regular workings of a building, but would allow people to become more accustomed to the presence of robots in the workplace. This would help to remove many of the social barriers when introducing robots into other positions within society.

The implementation of a working robotic security solution poses many challenges to the scientist. A final solution will be required to implement goal-based path planning, sensor fusion and perform classifications based on the environmental conditions to decide if the situation that the device is in, is within the standard usage of a building or not. All this must be done using noisy, often conflicting sensor data and in a cluttered human-orientated environment. These requirements place several strict constraints over any candidate solution. Firstly, the solution must be computationally inexpensive enough to allow the robot to robustly perform the basic tasks of path planning and obstacle avoidance. A computationally expensive solution would starve these systems of CPU time and would lead to the robot's reaction times falling to levels unsuitable for navigating a real-world environment. A second, related, issue is that the system should be scalable. For a system to be effective for all building sizes, it must be able to support multiple robots operating in parallel. While there is no strict requirement that these devices need to work together, (though it is obviously advantageous) it is necessary that they perform the vast majority of their processing locally to make the system robust against network failure and to prevent limitations of server-side computation placing upper bounds on the number of robots that the system can incorporate. Thirdly, the final system must have some form of adaptive intelligence to enable it to adapt to the usage characteristics of a specific building. Without this, the system will re-

quire a significant level of manual adjustment before it is possible to apply it to a previously unencountered building.

1.1.3 Artificial Immune Systems

A family of algorithms known for being scalable and adaptive are those associated with the field of artificial immune systems, (AISs). In addition, some algorithms within this field have been shown to be computationally inexpensive, compared to their counterparts from other areas of computer science [42, 105]. This field attempts to develop models and subsequently algorithms based on the operation of the (typically mammalian) immune system. There are many appealing properties of the immune system that make it suitable as the basis for computational algorithms. The system is scalable, asynchronous and exhibits properties of memory and signature-based processing [21]. One algorithm of particular interest within a security context is the dendritic cell algorithm (DCA). This algorithm, based on the operation of biological dendritic cells, has been used successfully within computer security [47]. The attraction of this algorithm stems from its low computational cost, sensor fusion [46] and its capability to make decisions based on multiple, often conflicting, data sources. While its batch processing nature means that it is not directly applicable to a robotic security problem, it is of interest to see if an algorithm that has been successfully applied to computer security can make the transition into a physical security problem.

1.2 Thesis Structure

This thesis is primarily an investigation of the dendritic cell algorithm, using the robotic security problem as a case study application to provide complex, noisy data. As a result, the structure of the thesis is as follows:

Chapter 2 is a critical analysis and review of the related literature from the fields of artificial immune systems and robotics; Chapter 3 presents the development and experimental validation of a robotic dendritic cell algorithm; Chapter 4 explores the findings of the prototype through a theoretical analysis of the algorithm in the frequency domain; Chapter 5 discusses the short-comings of a frequency driven technique for parameter tuning and population-wide modelling; Chapter 6 uses the findings from the previous chapters to develop an adaptive security system based on the dendritic cell algorithm; Chapter 7 presents a theoretical analysis based on linear classifier systems to analyse how the dendritic cell algorithm stands up against more common machine learning techniques and Chapter 8 contains a discussion and suggested future work.

1.3 Research Questions

In this section the research questions that this thesis intends to address will be presented. Each question will subsequently be discussed in turn and the motivation for exploring these issues will be explained.

1.3.1 Migrating from Software to the Physical World

> Can an immune-inspired, anomaly detection algorithm be adapted to solve threat detection problems in the physical world, through the medium of a robot?

Robots provide computer science with the opportunity to give previously abstract software agents the means to interact with the real world. This is particularly pertinent to bio-inspired algorithms, where the underlying mechanism is abstracted from a physical system to begin with. In these cases, the return to an embodied system acts to close the loop between the physical world and the computational abstractions taken from it, and

goes some way to validating that the abstraction retains some properties that make it able to function in the real world. Artificial intelligence and image processing are both heavily connected to robotics, with some researchers claiming that artificial intelligence is an unachievable goal without a robotic solution bridging the gap between the software and the real world [117]. The possibility that an algorithm designed to safeguard the virtual resources of a computer could also safeguard physical resources is an appealing one. However, the decoupled approach of developing software and then porting it to robotics has been criticised in the past for generating unwieldy computational systems or unrealistic abstractions [16]. Despite this, at the application level, it is still possible that there are strategies that can still be imported. By demonstrating that it is possible to move some, or all of a computer security decision making algorithm to a robotic platform, a step is taken towards characterising those algorithms that are able to make this transition. In addition the successful implementation of such a system has the obvious practical benefits of demonstrating that existing technology is capable of delivering a physical security solution.

1.3.2 Emergent Properties of the Dendritic Cell Algorithm

> Does the dendritic cell algorithm have properties that were not explicitly added as part of its design, which could be advantageous to a robotic application?

Little work has been done to explore the dendritic cell algorithm (DCA) from a theoretical viewpoint. It is the aim of this investigation not to simply provide another black-box, empirical exploration of the DCA, but to also examine how a collection of seemingly simple agents are able

to make collective decisions in the face of conflicting, noisy data. This exploration will be taken specifically from the perspective of a robotic application, focussing on issues such as noise and the relationship between this algorithm and more traditional machine-learning decision making techniques. For this investigation properties which can be described as 'emergent' are defined as "properties that were not explicitly added as part of the design". Properties that are considered to be advantageous to a robotic system are tolerance to noise and the ability to make decisions in the presence of conflicting information. These are considered useful to robotics as they are properties which allow intelligent agents to function despite the noise and complexity of real-world data.

1.3.3 Applying The Dendritic Cell Algorithm to a Robot

> Is it possible to adapt the dendritic cell algorithm from being a batch system to a system that can operate on a robotic platform?

Current work on the DCA uses the algorithm as a batch processing system. In addition, as a relatively new algorithm, the DCA has not been reviewed to assess its computational cost. For this application and this investigation to be successful, it must be possible to transform the existing algorithm into something that can provide answers fast enough to be of use within a dynamic environment. However, it is important that such a transformation should not come at the cost of removing the core properties of the abstractions that lie at the heart of the algorithm. An important step in identifying if the algorithm is fit for purpose is to recognise the hard and soft real-time constraints placed upon the system.

1.3.4 The Benefits of the Dendritic Cell Algorithm

> Are there other algorithms with functional equivalence to the dendritic cell algorithm, which can outperform it in terms of reduced computational complexity or superior performance, for the threat detection problem?

It is not sufficient to simply demonstrate that the DCA is able to approach the threat detection problem. It must also be demonstrated that doing so has advantages over other techniques. However, to make such a claim based on empirical evidence alone carries with it significant risks. Any direct comparisons with respect to speed or quality of performance would rely on a functional like-for-like implementation of a comparable algorithm. As shall be discussed in greater depth in Chapter 2, this is not a trivial task as the DCA is yet to be fully characterised. Also, many decision making algorithms in the literature require a training phase, so an empirical comparison would either entail using a trained algorithm or the construction of another system based on expert knowledge. Not only would the comparison between a trained system and the DCA be non-equivalent, but the selection of training data could unfairly bias the results of such a comparison. The performance of an expert system obviously relies heavily on the knowledge from the expert and the way in which it is captured and used, another potential source of bias. A more sound approach is to theoretically characterise the operations that the algorithm performs and then explore its relative benefits with other approaches function by function from a theoretical perspective, thus reducing the risks of bias from input data selection and non-equivalence.

1.4 Thesis Contributions

This thesis provides several contributions to the fields of robotics and artificial immune systems.

- It is demonstrated that the dendritic cell algorithm is capable of running on a real robot without impairing the robot's ability to perform its normal tasks such as obstacle avoidance and route planning. Related publication:
  - Oates, R., Greensmith, J., Aickelin, U., Garibaldi, J., and Kendall, G. The application of a dendritic cell algorithm to a robotic classifier. In ICARIS '07:Proceedings of 6th international conference on Artificial Immune Systems, 204–215
- The properties of a single, virtual, dendritic cell in the frequency domain are derived and expressed as a function of the input parameters to the algorithm. Related publication:
  - Oates, R., Kendall, G., and Garibaldi, J. Frequency analysis for dendritic cell population tuning. Evolutionary Intelligence 1, 2 (June 2008), 145-157
- It is clearly demonstrated that frequency analysis alone is insufficient to create a full operational understanding of the algorithm. Related publications:
  - Oates, R., Kendall, G., and Garibaldi, J. M. The limitations of frequency analysis for dendritic cell population modelling. In ICARIS 08: Proceedings of the 7th international conference on Artificial Immune Systems, 328–339
  - Oates, R., Kendall, G., and Garibaldi, J. Frequency analysis for dendritic cell population tuning. Evolutionary Intelligence 1, 2 (June 2008), 145-157

- A novel, adaptive robotic security application is developed through the interfacing of the dendritic cell algorithm to a neural model. Related publication:
  - Oates, R., Milford, M., Wyeth, G., Kendall, G., Garibaldi, J. The Implementation of a Novel, Bio-Inspired, Robotic Security System. In ICRA '09: Proceedings of the 2009 IEEE International Conference on Robotics and Automation, 1875–1880
- The limitations of the dendritic cell algorithm are highlighted and future improvements suggested by analysing the algorithm using linear classifier modelling. Related publication:
  - Stibor, T., Oates, R., Kendall, G., Garibaldi, J.M.: Geometrical insights into the dendritic cell algorithm. GECCO 2009: Proceedings of the Genetic and Evolutionary Computation Conference 2009 1275–1282

Chapter 2

Related Work

"Books serve to show a man that those original thoughts of his aren't very new after all." - Abraham Lincoln (1809-1865)

2.1 Mobile Robotics

2.1.1 Introduction

The field of robotics is a diverse and challenging research area. A wealth of active research is being carried out internationally, with the general aims of improving the reliability and applicability of robotics as a tool. It is necessary to present a brief explanation of the history and language of robotics before a full understanding of the current research landscape can be conveyed. The subject of this document forms part of the field of mobile robotics, (i.e. the study of intelligent vehicles) as opposed to manufacturing or industrial robotics. Mobile robotics has become a complex, multi-disciplinary topic requiring mechanical, electrical and electronic engineering as well as software engineering to make a fully functioning system. The field's progress has been supported by the evolution of micro-electronics and has, unsurprisingly, had a strong connection to the progress of artificial intelligence. The first two sections of this chapter provide a review and discussion of the literature pertinent to some of the key issues in the field of mobile robotics in general and robotic security in particular. The latter two sections examine a group of biologically inspired algorithms known as 'artificial immune systems' and applications of those algorithms on robotic platforms.

2.1.2 Architectures for Mobile Robotics

Many early robotics projects, such as The Stanford Cart [102], used complex models and symbol systems to represent and reason about the world. This 'sense-model-plan-act' (SMPA) approach has parallels with the symbol systems approach to artificial intelligence, known colloquially as GOFAI, (Good Old-Fashioned Artificial Intelligence). However, this approach was computationally intensive and required sensor information

to be streamed to a separate computer for analysis before the final model was generated. This has obvious drawbacks as the models could easily become out of date and unusable in the time it took to transmit and compute them.

The separation of robotics, artificial intelligence and image processing left many researchers looking to more powerful computers as a solution to the problem of achieving real-time control. However, just as the advent of trainable, multi-layer neural networks saw sub-symbolic artificial intelligence gain popularity, robotics also went through a transition which led to the emphasis of the 'intelligence' moving from model building to 'reactive' and 'physically grounded' architectures. In 1990 Rodney Brooks' seminal paper "Elephants Don't Play Chess" [16] launched a scathing critique on the use of symbolic artificial intelligence as a means of robot control. In addition, Brooks goes on to draw, explicitly, the parallels between GOFAI and the 'sense model plan act' school of robotics and provides several examples of robots that have achieved relatively complex behaviours without using a model at all. This paper encapsulated a growing rejection of SMPA architectures in favour of techniques that abandoned the concept of using a model of the real-world altogether. The journey away from modelling had already been started by several researchers who had produced encouraging results from architectures that simply reacted to the current sensory input to the system, known as reactive architectures.

In 1984 Braitenberg produced vehicles that performed relatively complex tasks, not only without the use of a model, but without the use of a digital processor [14]. The "vehicles" experiments relied entirely on weighted connections between sensors and actuators, demonstrating the power of systems modelled on the "reflex-arcs" found in biological organisms. Brooks' earlier research was also part of this rejection of SMPA

and had produced a robot control architecture known now as 'the subsumption architecture' [15]. This architecture shuns the standard, serial process of sensing, modelling, planning and then finally acting. Instead it uses a more parallel technique which relies on several, much-simpler systems reacting to the sensory data directly and each requesting an action from the actuators. These behaviours are connected in a hierarchical fashion which ensures that, in the event of a conflict, the lowest-level behaviours take priority over the higher-level behaviours. The aim of the subsumption architecture was that, at the bottom, behaviours to prevent damage to the robot or its environment, such as emergency stopping criteria, would take precedence over more complex and ultimately more 'luxurious' behaviours, such as reasoning about the world. Such was the impact of this technique that today many manufacturers of mobile robots provide APIs (Application Programming Interfaces) that are deliberately structured in this way, (for example MobileRobots Incorporated's ARIA API). Reactive architectures have now been applied to a wide-variety of robotic control techniques. Under-water navigation, elaborate animal-inspired behaviours [18, 25, 126] and a variety of obstacle-avoidance and tracking tasks [15,16,126], have been successfully implemented reactively.

Reactive architectures were far from a panacea. From the outset many researchers argued about both the technical and philosophical ramifications of the reactive paradigm. Ziemke [155] summarizes what he feels are the five key, contradictory definitions of the phrase "embodiment", from the most relaxed "structural coupling", (requiring only that the agent be capable of changing the state of the environment and vice-versa), to the most extreme "organismic embodiment", (requiring the agent to actually be constructed from organic materials). However, the philosophical debates were overshadowed by the more practical issues of the algorithmic limitations of a system without a model. Many re-

searchers found that not having a model meant that architectures could be vulnerable to unexpected interactions and fail to infer things from information that was available to them. The capacity to guard against uncertainty and infer information from experiences were properties of the traditional SMPA architectures. In [150] the history of several museum tour robots is presented, and illustrates many of the problems with the reactive architecture. These issues include the robot failing to find its docking bay because the security guards turned off the lights prematurely, causing the robot to lose its way. It is argued that with a model, and a dead-reckoning algorithm, the robot would have been more resilient to this type of sensor failure. Local minima problems were also encountered, where the reactive controllers failed to spot cyclic behaviours. The roots of these criticisms stem from the difficulties associated with representation within a reactive architecture. With no model, the 'knowledge' within the system is represented in a much more abstract way than for an SMPA architecture. This can make the design and the predictability of reactive systems extremely complex. In [7] Arkin summarises these issues and presents a case for hybrid architectures using reactive behaviours for lower-level systems and behaviours with formal models for higher-level systems.

2.1.3 Localisation

A fundamental problem with mobile robotics is that of localisation. Many applications are only achievable if the robot has some estimate of its position in the environment. For example, reconnaissance tasks are useless without some knowledge of where the robot's data is coming from and transportation tasks are of little use without the robot being able to guarantee that the items being transported will reach their goal. Many algorithms exist to localise a robot, such as Kalman filters [66] and Bayesian

Methods such as particle filters [41]. However, in order to localise a robot in space, a map of that environment is required to support localisation hypotheses. This is an acceptable constraint for known environments, but there are several applications where this is not possible. Situations where the robot must localise itself and build a map of the environment at the same time fall under the heading of "simultaneous localisation and mapping" (SLAM), an active research area in robotics. One of the most popular techniques for solving SLAM involves iteratively building a probabilistic map, using a specially modified version of the Kalman Filter, first proposed in [129]. However, different hardware configurations and environments place different constraints onto the problem and a large repertoire of SLAM algorithms are now available to the roboticist to match the diverse challenges posed by these constraints. A relatively new addition to this repertoire is the biologically inspired RatSLAM algorithm, for vision-based SLAM in dynamic environments.

The RatSLAM Algorithm

In 2003 Milford et al. put forward a prototype algorithm for performing SLAM using only vision sensors [96]. The initial work was inspired by biological research that had suggested that rats have collections of cells which represent their position and orientation in space within their hippocampus [65,87,111,136]. These cells, termed 'head direction cells' and 'place cells' represent the rat's estimate of its orientation and position in space. The initial prototype was able to emulate orientation cells using a continuous attractor network, (a type of neural network similar to a Hopfield network). By stimulating the head direction cells, the network dynamics eventually settle into a state where one cell is significantly more stimulated than the others. Each cell is associated with a specific angular range. The 'winning' cell is considered to be the most likely estimate of

the robot's orientation. Initially the input to the network was based on a simple vision system which estimated the colour, distance and direction of a series of coloured cylinders in an artificial environment. Hebbian learning is used to associate specific 'local views' (estimates of cylinder direction and distance) to a direction cell. This process ensures that if a specific scene is viewed again, it stimulates the associated local view cell and updates the robot's direction estimate. In [93] this work was extended to be able to support multiple estimates of the robot's pose through network dynamics that allowed several collections of cells within the attractor network to remain stimulated, even in the face of a clear winner. This extension was demonstrated to operate within an indoor environment, and was able to estimate both orientation and location for a mobile robot. In [120] the 'local view' cells were significantly improved. This work demonstrated that the algorithm was able to identify visual landmarks of note, without the need for an artificial environment. The new local view cells used feature extraction to associate automatically generated templates with specific poses. A similarity metric assessed each scene presented to the robot and checked it against the existing local view cells. If it matched a given template, that template was allowed to stimulate the pose associated with it. However, if the difference exceeded a given threshold, a new template was built and added to the environment. This improved version of the algorithm was not only able to map an unknown test environment indoors and outdoors, but was also shown to be robust against 'kidnapping', (moving the robot so that its estimate of its location was compromised) [94, 121].

In 2005 the algorithm was improved further through the addition of 'experience maps' [91]. Up until this point the maps produced had been topologically accurate but spatial information was held in a highly abstract way which made generating a human-readable map difficult.

Experience maps were an additional collection of objects that explicitly linked local view cell information to x, y, θ values. By recording these and the odometry information that linked them, human-readable maps could be produced. Again, the visual information presented to the robot was used to determine which 'experience' most closely resembled the current experience. If no close matches were found, a new experience was produced. Experience mapping also reduced the number of pose cells required to represent a given space as the experience map allowed pose cells to be reused more efficiently [92, 95].

The RatSLAM algorithm has been shown to be able to map large outdoor environments and small indoor environments with no changes to its parametrisation [90] and has even mapped an entire city-suburb using only a webcam [97, 98].

2.1.4 Swarms of Agents

Robotics provides computer science with the means to bestow previously abstract algorithms the capacity to directly interact with the physical world. In section 2.1.2 architectures were introduced which placed the 'intelligence' of a system into the interactions between the agent and its environment. This concept of 'intelligence through interaction' has been taken further by the various fields which utilise swarms of agents. In these areas agents must not only interact with their environment, but also other agents attempting to perform the same overall goal. This section will explore two fields that utilise swarms of agents, swarm robotics and particle swarm based optimisation techniques. The aim of this review is to provide the reader with an insight into the power behind collections of interacting agents, (such as those found within the dendritic cell algorithm) and to justify the assertion that any candidate solution for the threat detection problem should be inherently scalable, so that it could

reap the rewards of a multi-robot implementation in the future.

As with many relatively new disciplines, there is much debate in the literature about the exact definition of a swarm [125]. For the purposes of this work, the definition outlined by Hinchey et al. will act as a guideline [59]. The most fundamental requirement is that a swarm is a collection of agents, this implies that each member of the swarm has some individual intelligence. In addition, Hinchey et al. go on to assert that swarm members must be able to interact with both the environment and a subset of the swarm group [59], thus allowing complex behaviours to arise from the interaction of simpler parts.

Arkin describes the properties of swarms that make them such an appealing concept to robotics developers [7]. Firstly, tasks that are "naturally decomposable" can obviously be performed quickly and efficiently in parallel by a swarm of robots. Secondly, robotic swarms facilitate "task enablement", i.e. there may be tasks that an individual robot cannot physically or computationally achieve, but a group of similar robots can. Thirdly, a swarm of robots provides distributed sensing and acting, allowing a robot to make decisions using more complete environmental knowledge and effect actions on several, distant locations. Finally, swarms are highly fault tolerant, as they typically have a significant amount of functional redundancy. There are of course negative aspects to creating a robotic swarm. Arkin describes four key issues with the implementation of a swarm robotic system, as opposed to a single robot system: Interference; Communications Cost; Uncertainty and System Cost. The issue of 'interference' stems largely from the physical instantiation of robots. Robots can occlude line of sight and obstruct paths, which can make goal achievement difficult or impossible. The cost of communications is a significant problem, as power is a scarce resource for a mobile robot and any unnecessary transmission is detrimental. Uncertainty can be

introduced by swarm robotic systems, as with no centralised control, a poorly designed system can result in agents competing for resources and reducing the efficiency of the group as a whole. Finally the issue of system cost arises from the concept that multiple simple robots may cost more than a smaller number of more complex robots.

The problems associated with interference, communications cost and uncertainty are addressable at the system architecture level. By carefully selecting the protocol and rules for interaction it is possible to limit the effects of these issues. Amongst the more popular swarm architectures is the "Alliance" architecture, presented in [114]. This architecture is an extension of Brooks' subsumption architecture [15], discussed in section 2.1.2. Like the subsumption architecture, Alliance allows the overall behaviour of a robot be determined by the interactions between a series of simple, low-level controllers. However, these controllers are not always active. Instead, groups of low-level behaviours are identified that achieve different, useful actions. In turn the active group or "behaviour set" is selected by a series of "motivational behaviours" which determine the best action to take using both local information and the active actions of near-by robots. By only communicating the active action to nearby neighbours the communications cost is quite small and there can be no uncertainty about the intentions of the robots in a given neighbourhood. However, problems caused by physical occlusion are not fully addressed. Parker et al. extended the original Alliance architecture to allow the relationships between actions and motivational behaviours to be dynamically adjusted to improve group-wide performance [113]

The applications for swarms of robots are varied. In [10] the authors attempt to discern a taxonomy of the published application areas for swarm robotics. In the paper, the authors are able to subdivide swarm robotics into eight smaller groups, though four of these primarily revolve

around structured movement for problem solving based on flocking. So far we have considered agents that have been physically instantiated as robots. However, a common use of the power of flocking is the particle swarm optimisation technique, an algorithm utilising purely software agents.

Particle swarm optimisation techniques were first developed in [67]. Since that point they have been successfully applied to a wide range of applications such as electrical and electronic design problems, control parametrisation and signal processing problems [118]. In such techniques, agents are distributed throughout the search space and explore it as a swarm. The velocity of each member of the swarm is calculated using a velocity update equation which is typically a linear combination of the position of the best point found by an agent's social group, the position of the best point found by the individual agent and random noise variables used to invoke exploration of the space. The social groupings of the agents are typically predetermined before the search begins, and the position of the agent is updated by integrating the velocity vector defined by the velocity update equation. There are various forms of the velocity update equation, as new variants have been introduced to compensate for problems with unconstrained movement and unstable trajectories through the search space [118]. In [119] an overview is provided of the more popular velocity update equations and social network topologies is provided. In [119], Poli et al. also present a series of dynamic strategies for redefining the network topology as the search is made, in order to exploit discovered information about the search space. Of interest for this investigation, Poli et al. also describe attempts that have been made to theoretically analyse particle swarm techniques, concluding that the stochastic, multi-agent nature of the algorithm makes analysis using standard techniques difficult.

Much of the research presented in this section has focussed on specific algorithms for localisation or cooperative swarms. However, general research into robotics and agents is rarely conducted from a purely theoretical perspective, instead applications are utilised to provide useful, real-world problems with which to explore the algorithms and hardware of robotic systems. The rest of this investigation of related work will take this application-orientated approach to the literature.

2.2 Robotic Security

Chapter 1 introduced the benefits of introducing robotics into a security setting. A major problem with the concept of robotic security is that very few researchers in the field agree on a definition of the problem to be solved. In [60] an attempt is made to make clear distinctions between several, related problem areas which the author describes as 'exploratory missions'. Seven problem types are identified including military reconnaissance, urban reconnaissance and security patrolling. The two reconnaissance tasks are separated by the fact that the environment in urban reconnaissance poses additional challenges in that the field of view is limited and wireless communication range is restricted. Security patrolling is described vaguely as the act of keeping "an area safe", though no definitions are provided of the potential hazards. Further work from the same research group places an emphasis on sensor coverage through the deployment of multiple 'scout' robots from a larger 'ranger' robot [123]. In this paper, the threat detection algorithm is a simple motion detector based on frame differencing. Frame differencing is a severely limited form of motion detection as it cannot compensate for the movement of the robot itself, instead requiring that the robot remain stationary during its detection phase. This negates many of the advantages of using a

robot for a security application.

A much more useful definition for robotic security comes from the work of Massios and Voorbraak [83, 145]. The focus of Massios' work is for the surveillance and detection of building fires [82]. Probability distributions based on the likelihood of fires breaking out are used to plan an efficient patrol route around an area which will minimise the cost of any potential fire and minimise the probability that any fire will go un-noticed. Contributions include a detailed comparison on the performance of several heuristics designed to reduce the planning time for near-optimal solutions [83]. Massios explicitly states several assumptions of the work, including that the robot is always perfectly localised, it can always sense a threat when in the presence of one and there is only one 'virtual sensor' to be monitored, considered to be the result of sensor fusion. These assumptions implicitly define the robotic security problem as one that requires localisation, threat detection and sensor fusion. In earlier work, Voorbraak and Massios explicitly state that it is very important, that in the future, researchers focus on the automatic detection of threats, to complement their contribution of an efficient routing algorithm [145]. Figure 2.1 is a proposed architecture for a robotic security system inferred from the work of Massios [83]. The proposed architecture divides the robotic security problem into two parts, 'routing' and 'problem identification'. The routing aspect of the problem is further subdivided into three parts: the basic avoidance of obstacles, the minimisation of the probability that an important event will be missed and the localisation of the robot within its environment. The problem identification aspect of the problem is subdivided into two parts: the fusion of sensor data from multiple sources, allowing the on-board 'model' of the environment to have high fidelity to the real world and the classification of that data to identify situations of note.

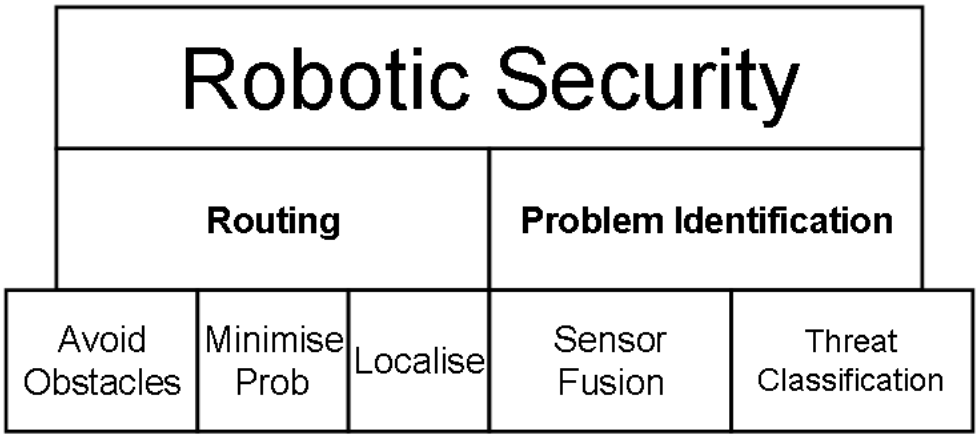


Figure 2.1: The Structure of the Robotic Security Problem

Automatic threat detection is not a new topic in the world of static sensors. In [9] a formal logic system is developed using Prolog that is intended to work with multiple sensor types to identify the presence of threats using static sensors. By assigning costs to the use and processing-times associated with each sensor, game theory is applied to determine which sensors should be used at which times. As with any formal logic system, this approach cannot detect any events that the developer has not reasoned about beforehand, nor can it adapt to persistent changes in the environment, such as long-term engineering works in the field of view of the system. The work presented is only theoretical with no results from actual sensor systems. Threat-detection in static sensors has additional complexity as without the mobility of a robotic solution, once recognised, a target of interest must be tracked across multiple cameras as it moves through the observed space. Work presented in [89] suggests a modified version of the Kalman filter to facilitate multi-sensor tracking. However the Kalman filter is sensitive to the parametrisation of the probability distributions used to represent the noise characteristics of the sensors and the target's movement behaviour. Mikic et al. make no attempt to modify these distributions on-line, suggesting an assumption

that all targets will have the same underlying characteristics [89]. The results presented are from artificial data, so it is difficult to judge the practicality of such a system. Strategies from static sensor networks may not translate to a robotic setting. Static sensors can be connected with higher fidelity to powerful computer systems, whereas a mobile security robot may become transiently out of communications range from a given base-station. In addition, static sensors do not need to compensate for optical changes that arise from movement, such as optic flow, changes in lighting conditions and transient occlusion. Despite these limitations, there are still things to learn from the field of static sensor threat detection. Work based on colour analysis and clustering techniques has been shown to translate into a robotic setting, with the caveat that a database of images exists for the robot's environment which define the 'normal' state of the building [20]. This assumption does place a large memory requirement on the system and is limited to the detection of missing objects and the presence of non-standard objects. As with the majority of robotic threat detection literature, the system would be useless during the hours of operation of a building, as the presence of any dynamic object, (such as a person) would trigger an alarm.

2.2.1 Military Case Studies

Combat robots have received considerable coverage in the media, but the majority of them are simply teleoperated devices with no on-board intelligence. Within the literature, two military projects to automate physical security tasks using robotic platforms are documented. Both projects come from the Mobile Detection Assessment and Response System (MDARS) project, a joint research collaboration between the American army and navy. The aim of the project is to develop a collection of robots and associated control software, capable of automating many of

the standard patrol-based security tasks involved with securing a military base [19]. A military base provides an excellent prototyping area for a security robot as the environment is known and well-constrained, under 'friendly control' and can support limited modifications to make navigation and sensing easier [33]. As it was recognised that the constraints and requirements for patrolling the inside of a warehouse are significantly different to those posed by patrolling an outdoor area, the project was split into MDARS-I (internal) and MDARS-E (external). Ultimately, the aim is to reintegrate the projects at the user level, so that a single operator will be able to monitor both sets of robots [73]. The project's overall structure fits the template given in Figure 2.1, separating the threat detection and routing problems. The MDARS project defines the physical security problem as both the identification of missing inventory items and also the detection of fires, flooding and intruders [33].

The external phase of the project uses a modified four-wheel drive vehicle to travel autonomously around the military base. The external vehicles check doorways and barriers to ensure that the magnetic locks are in place and are equipped with RFID scanners to check that inventoried items are in the locations that they should be [29]. In addition, they are fitted with thermal imaging cameras to perform intruder detection [115]. A paintball gun is mounted on some of the robots to investigate the potential use of teleoperated non-lethal weaponry. However, several limitations to the external project bring into question its usefulness. Barrier detection is performed by polling sensors already built into the doors. As the location of doors is somewhat constrained, it is curious that the sensors are not simply polled remotely. The RFID tags on inventoried objects could easily be detected at bottle-necks by stationary readers, to prevent them from being taken off-site. Were a malicious intruder to remove the RFID tag and leave it behind, the robot would consistently

report that the object is safely stored, potentially introducing a delay in the detection of missing military equipment. Finally, the intruder detection algorithm is extremely simplistic and cannot perform while the robot is moving. Instead, the infra-red cameras detect heat and cause the robot to stop, allowing it to perform motion detection.

The internal project has undergone the most change, with several robots being prototyped and re-engineered since 1982. The internal robots perform the same simple intruder detection as the external robots, using infra-red sensors and low resolution cameras. They also perform the same RFID-based inventory checking. In order to improve the reliability of the robot's navigation through a warehouse environment, a hybrid system uses reflective markers to localise itself in certain areas and a reflexive navigation behaviour to move between the markers [32]. In Chapter 1 the possibility was raised that using robots to perform security patrols could reduce the effectiveness of the security system by making the patrol routes predictable and therefore, avoidable. This is compensated for by the MDARS-I project by identifying 'idle' robots that are neither in service nor charging, and sending them on random patrols. Despite the constrained nature of a military warehouse, there are several documented problems that have arisen throughout the history of the project. In [34] a detailed description is given of the process of moving the robot from a laboratory space to the real-world and the associated issues. These included the discovery of software failures that had previously gone undetected, (overflows from sensor readings due to the larger space being monitored). Several of the issues were caused by the brittleness of the system. Most worryingly, the system's reliance on a modified environment was undermined, when one of the 'immovable' markers was moved by a forklift truck which crashed into one of the pillars.

The latest robot to be used as part of the MDARS-I project is Robart III. This is a highly advanced system which is capable of deploying additional slave robots to help it to achieve various tasks [24]. The slave robots are capable of acting as wireless relays to prevent the main system losing communications with the robot. In addition, the slave robots can be deployed at bottlenecks to prevent intruders from leaving a room unnoticed while the master robot is patrolling it. Despite the extremely simplistic intruder detection algorithm, Robart III can be put into a fully autonomous mode, allowing it to fire its non-lethal weaponry at targets [39]. This weaponry includes a tranquilliser-dart firing Gatling gun, an optional BB gun and a powerful light and sound system designed to disorientate humans in close proximity to the robot. Placing the control of even non-lethal weaponry with such a simplistic intruder detection system raises significant moral questions. It is of note that the work presented in this thesis is intended to augment a human-orientated security solution where any intervention should be performed by humans directly, limiting the robotic system to highlighting threats to human operators.

Much can be learned from the case studies presented here. The problems that the MDARS project has encountered highlight the importance of autonomy for mobile security robots. Even in a military base it was not possible to guarantee that the robot would have permanent communications with the base station, leaving potential blind spots in the system. In addition the lack of adaptation by the software meant that the resulting system was brittle and unable to compensate for changes in the environment that resulted from the day-to-day operation of the building. It is also of note that the MDARS-I system fails to perform any real sensor-fusion, which limits the range of possible threats that it can detect.

So far this exploration of related literature has focussed on robotics.

However, robots are a tool and it is difficult, or even impossible, to assess them without either their environment nor their goals to provide the context for their actions. As a result, the specific application area of security robotics has been explored, but was found to be lacking scalable solutions capable of sensor fusion and adaptation. Not only is this a gap in the knowledge of robotics which could be both useful and interesting to cover, but the robotic security setting also provides a rich and complex setting with which computer science can investigate existing algorithms and explore their limits. An area of research that has been shown to provide algorithms and solutions which are scalable, capable of adaptation and data fusion is the field of artificial immune systems. This relatively new field of research focusses on copying the properties of the biological immune system, a decentralised system which processes multiple sources of information and exhibits the ability to learn and adapt [57]. In the next section the field of artificial immune systems shall be explored, and it will be demonstrated that there are algorithms within the field that not only exhibit the properties that are required to solve the robotic security problem, but also are yet to be fully characterised, so would benefit from being applied to such a challenging area.

2.3 Artificial Immune Systems

2.3.1 Introduction

Artificial immune systems (AISs) are a diverse collection of algorithms and models which attempt to capture or mimic the behaviour of some part of the immune system. While some work has been carried out on modelling immune functionality purely for the purposes of furthering the understanding of the system in biological terms, the majority of the work in AIS is centred around the creation and application of computational

algorithms.

The remainder of this review of related work has four aims. Firstly, to explain the basic metaphors that underpin the most common artificial immune system algorithms. Secondly, to explore, in depth, the existing research on the dendritic cell algorithm. Thirdly, to introduce and explore work relating to the theoretical analysis of artificial immune systems. Finally, to introduce and explore work relating to the application of artificial immune systems on robotic platforms.

As immunological concepts are not common knowledge, the pertinent metaphors for the standard immune-inspired algorithms will be introduced. Also, in order to fully explain the attraction of immunology as an inspiration for computational research, some of the differing viewpoints from the biological community will also be discussed. However, this work is from a computer science perspective and, just as it would be incongruous to discuss, for example, the functionality of neuronal sodium-potassium pumps in a neural networks work, every attempt will be made to prevent biological content from detracting from the algorithmic perspective of this review. A more biologically orientated introduction to immunology from a technical perspective is given in [116].

2.3.2 Immunological Concepts

Biological systems are often thought of as physically bounded constructs of cells. The cardiovascular system, for example, is clearly delineated as the heart, lungs and blood vessels, despite its pervasive interactions with the rest of the body. However, in modern biology there is little agreement on the boundaries of such a separation for the immune system. The fundamental structure of the body is designed to protect itself from external pathogens, (such as parasites and bacteria). From the structure of the epidermis [151], to the chemical contents of cells [127], the microscopic

and macroscopic design of the body interacts with and participates in the process of immunity. Like Lovelock's 'Daisyworld' [148], a form of homeostatic control arises, not through dedicated sensing and acting systems, but as an emergent property of the physical interactions between unknowing agents. As a result there are several paradigms for thinking about the operation of the immune system and the operation of the cells that participate in it.

The concepts presented here can be found in any introductory immunology text such as Chapter 14 of [35], [116] or [151]. An understanding of the basic operation of the immune system revolves around the structure and properties of antigen. Antigen are molecules or fragments of molecules that are associated with cells from the body's tissue or from external pathogens (e.g. viruses and bacteria). The differing structures and make-up of antigen allow them to be used as representations of their cell of origin in an analogous way to fingerprints. The first time a pathogen infects a body, the first immune cell that it is likely to interact with is the immature dendritic cell. The immature dendritic cell is a 'phagocyte', roaming throughout the body absorbing antigen and cellular debris. Through interacting with the tissue the immature dendritic cell undergoes a physical change into either a semi-mature dendritic cell or a mature dendritic cell. The process which determines this change is a matter of debate, but under the 'Danger Theory' [84, 85], maturation into a mature dendritic cell is caused by exposing immature dendritic cells to evidence of cellular damage. Both semi-mature and mature dendritic cells move into a lymph node.

Lymph nodes contain, amongst other things, a second type of immune cell, known as T cells. T cells are equipped with receptors which allow them to identify pathogens via their corresponding antigen. The T cell then attacks the identified pathogen. The match between a T cell

receptor and the antigen does not have to be perfect. Instead they are said to have a level of 'affinity' or similarity. The receptor shape of a T cell is generated randomly, thus allowing the immune system to react to a wide variety of threats. However, this random generation means that it is possible for a T cell to be created that will attack cells from the body's own tissue. To prevent this, a process known as 'negative selection' destroys T cells with a high affinity to the host. T cells will not attack unless they are presented with antigen fragments by a mature dendritic cell. Semi-mature dendritic cells also present antigen to T cells, but have an inhibitory effect. When stimulated by a mature dendritic cell, T cells move into the tissue and perform one of two roles. Primarily T cells attack pathogens that match their receptor. However 'helper T cells' do not attack pathogen. Instead they locate a third type of immune cell: the B cell.

Helper T cells co-stimulate B cells with similar receptors to themselves. If a given B cell is also exposed to a pathogen for which its receptors have a high affinity, it becomes active. Once active, B cells enter a 'proliferation' phase where they repeatedly clone themselves. The cloning process introduces random variation in the receptor shapes via a process known as hypermutation. Those new cells which either have a weak affinity for the pathogen, or fail to be stimulated by a helper T cell, die off. The need for both a pathogen *and* a T cell to be present for a B cell to be stimulated means that the successful removal of T cells that react to the host also prevents B cells from attacking the host. This process of proliferation of high affinity matches and cell death for low affinity matches is referred to as 'clonal selection' [17]. Those B cells which survive follow one of two paths. One route, is that a B cell can differentiate into a plasma cell. Plasma cells mass produce antibodies which react to the same antigen as the original B cell, destroying the

associated pathogen. As with much immunology, the second route is a matter of debate. For the purposes of this discussion the point of view of Jerne [64] will be presented as it forms the basis of one of the more popular algorithms. Jerne suggests that B cells can also stimulate each other, causing continued proliferation after the helper T cell and the pathogen have gone. By maintaining a population of B cells with similar receptor shapes to the originally stimulated B cell, reinfection by the original pathogen will be immediately destroyed by the remaining B cells. This gives the immune system the property of memory.

2.3.3 Common Immune-Inspired Algorithms

As the primary focus of this thesis is a theoretical analysis of the DCA, it is important to look to other attempts to theoretically analyse artificial immune algorithms. However, such a discussion would be nonsensical without first describing the form and function of those algorithms. Here we will explore the three types of algorithm that have dominated the field of artificial immune systems for many years. These are: the negative selection algorithm, based on the phenomena that removes T cells that react to the host from the population; the clonal selection algorithm, based on the hypermutation and affinity selection that activated B cells go through; and idiotypic networks, based on the Jerne's co-stimulating B cell hypothesis [64]. Here the 'vanilla' implementations of these algorithms are outlined. The interested reader will find [57, 138, 142] provide wide ranging literature reviews on all of these algorithms.

Negative Selection

The negative selection algorithm was one of the earliest attempts at producing an artificial immune system [37]. The algorithm was developed by Forrest et al. as a computer security algorithm [36]. The negative se-

lection algorithm is a supervised learning algorithm, requiring a period of training. During this period 'detectors' are generated at random. In the work by Forrest et al. these detectors were simply bit strings. Training data aimed at encapsulating the 'normal' operation of a computer was created by encoding function calls by active processes. This data was presented to the detector sets and detectors with a high affinity for the normal activity of the machine were deleted. Affinity was measured using a metric called 'r-contiguous bits'. This process compares two bit strings and states that they match *iff.* r or more contiguous bits in one string match r bits in the other. Once a large population of detectors has been produced that does not match the normal operation of the computer, the algorithm enters its second phase. In this phase the function calls of the current computer use are encoded into bit strings in a similar manner and matched with the generated detector set. By process of elimination, if a match is found, it is assumed to be 'abnormal' behaviour and the user is informed.

The work of Newborough and Stepney attempts to find a common framework for expressing population based algorithms, including the negative selection algorithm [104]. This framework is outlined in Figure 2.2. In order to clarify the algorithmic properties of the negative selection algorithm, the description of the negative selection algorithm from [104] is provided in Figure 2.4. However, in order to provide the reader with some context, the Newborough and Stepney's description of the more commonly used genetic algorithm is also provided, in Figure 2.3 [104].

Clonal Selection

Clonal selection algorithms are a family of optimisation techniques based on the proliferation and selection of B cells. Several implementations exist including, (in chronological order) opt-AINET [2], Clonalg [22], the

Figure 2.2: A Generic Population Algorithm [104]

Individuals: An individual is a set of 'characteristics' which define a solution

Characteristics: Characteristics define the space that the individuals inhabit

Create: This phase introduces novel members into the population

Evaluate: Evaluates each individual to find its fitness or affinity to the solution

Test: A test to identify if a termination condition has been met

Select: Selection criteria to identify individuals from the current generation to be used in the creation of the next

Spawn: The creation of new individuals for use in the next generation

Mutate: The changing of selected individuals

Figure 2.3: The Genetic Algorithm, as expressed in [104]

Individuals: `Chromosomes`

Characteristics: `Genes`

Create: `Individuals assigned random characteristics, giving broad coverage of the search space`

Evaluate: `A 'fitness function' evaluates the chromosomes based on their genes`

Test: `On termination the the solution is the individual with the highest fitness`

Select: `Different variants use different techniques. Typically the fittest members of the population become parents to the next generation`

Spawn: `Characteristics of pairs of selected parents combined using a crossover mask to generate two new individuals`

Mutate: `Genes subject to random perturbation to reintroduce lost characteristic values`

Figure 2.4: The Negative Selection Algorithm, as expressed in [104]

Individuals: `Antibody`

Characteristics: `Shape receptor`

Create: `Individuals assigned random characteristics`

Evaluate: `The shape receptors are compared to the target region using a 'lock and key' metaphor, (typically some distance metric measuring the difference between the target and the receptor)`

Test: `On termination the the solution is a set of individuals with affinities above a user-specified threshold`

Select: `A threshold is applied to the members of the current population, those above that threshold are continued to the next generation`

Spawn: `Individuals identified by` *`Select`* `are passed to the next generation. Population level maintained by using the` *`Create`* `policy to introduce new members`

Mutate: `No mutation`

B cell algorithm [140] and opt-IA [26]. The core functionality of clonal selection algorithms is similar to the functionality of a genetic algorithm (GA). Each cell represents a possible solution for the optimisation problem, in a similar way to a member of the population in a GA. An affinity function, (analogous to a GA fitness function) evaluates the quality of each solution. Successful solutions are able to proliferate, using a hypermutation operator to allow the algorithm to traverse the search space. Unsuccessful solutions are removed from the population. The key difference between a GA and a clonal selection algorithm is that at no point do the cells in a clonal selection algorithm perform a cross-over operation. These algorithms have been applied to several optimisation problems, though of note is the work of Ciccazzo et al. who not only apply a clonal selection algorithm to a real-world problem, but also perform a comparison with a standard genetic algorithm [23]. The problem domain is automated analogue circuit design and in this case the clonal selection algorithm used (eIP), outperforms the GA approach. It is hypothesised that this may be due to the absence of cross-over, resulting in a less rapid convergence of the population, which would be a benefit in certain search spaces.

As with the negative selection algorithm, the algorithmic description of the clonal selection algorithm presented in [104] is provided in Figure 2.5.

2.3.4 The Dendritic Cell Algorithm

With its first formal proposal in 2005, [45], the dendritic cell algorithm (DCA) is a relatively new addition to AIS. Several versions of the DCA have been developed since that point as Greensmith et al. continued to explore their algorithm. The majority of the development of the DCA up to 2007 is described in Greensmith's PhD thesis [42]. In this work,

Figure 2.5: The Negative Selection Algorithm, as expressed in [104]

Individuals: Two populations, one consisting of antibodies and one consisting of memory cells

Characteristics: Shape receptor

Create: Antibodies are randomly created, memory cells are not created, they are introduced at the *spawn* phase

Evaluate: The shape receptors are compared to the target region using a 'lock and key' metaphor, (typically some distance metric measuring the difference between the target and the receptor)

Test: On termination the the solution is the set of memory cells

Select: The antibody population is selected by identifying the n best members and applying a threshold to enforce a minimum standard. All members of the memory cell population are selected for use next iteration

Spawn: Antibodies are cloned, where the number of clones is proportional to the fitness/affinity of the parent. Memory cells are cloned.

Mutate: All clones, (antibodies and memory cells) are mutated by an amount inversely proportional to their affinity

Appendix B provides pseudocode for every version discussed here. This algorithm forms the majority of the focus for this investigation, as a result, this algorithm's discussion will be more detailed than those previously. What follows within this section is a discussion of the mechanisms behind the most commonly used version of the DCA, as described in [47] (pseudocode in Appendix B.4). The aim is to provide the reader with an engineering-centred model of the algorithm and to remove the biological terms from its presentation where possible.

The DCA accepts four streams of data as input, three time-varying signals and an application-specific list of symbols. The time-varying signals are sourced from application-specific heuristics and are termed "PAMP", "Danger" and "Safe".

- The PAMP heuristic provides a signal which increases proportionally to the presence of data with a strong correlation to a positive or 'anomalous' situation. The biological PAMP is a 'pathogen associated molecular pattern', in other words a recognised signature of a dangerous invader.

- The Danger heuristic provides a signal which increases proportionally to the presence of data with a weaker correlation to a positive or 'anomalous' situation.

- The Safe heuristic provides a signal which increases proportionally to the presence of data with a strong correlation to a negative or 'normal' situation.

The list of application-specific enumerations, termed "antigen", acts as a cyclic buffer, storing the symbols which describe the current environment for the algorithm.

An artificial dendritic cell (DC), uses these input signals to produce three internal signals, termed "CSM", "$IL10$" and "$IL12$" after their

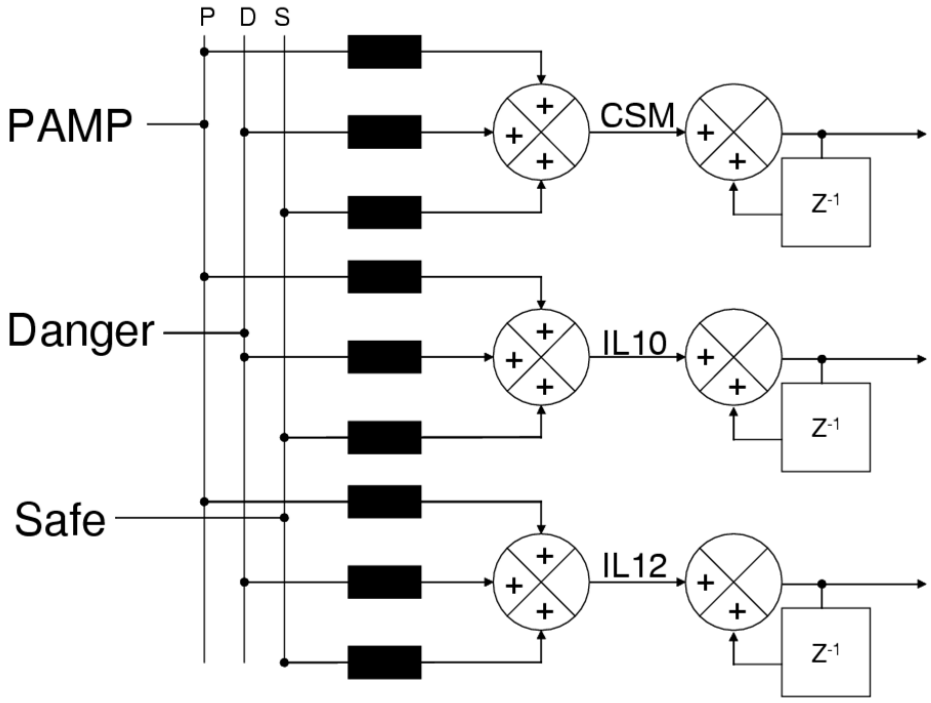


Figure 2.6: The Block Diagram of a Dendritic Cell

The variable z^{-n} is used to denote a delay of n sampling iterations. The output from the summation of the CSM signal is compared against the migration threshold to determine the time of migration. At this time the relative size of the other two summations determines the output signal of the cell.

biological counterparts. 'CSM' stands for "co-stimulatory molecule", an important part of the immune response process. It is thought to be a crucial part of the antigen presentation process, where DCs present their sampled antigen to T Cells [88]. Each of these signals is the integral of a weighted sum of the input signals (see Figure 2.6).

CSM accumulates throughout the cell's lifetime and rises proportionally to the cell's exposure to any input signals. PAMP and Safe signals typically have a greater impact on the accumulation of CSM than Danger. $IL10$ is a cumulative value that rises proportionally to the cell's exposure to the Safe signal. $IL12$ is a cumulative value that rises proportionally

to the cell's exposure to PAMP and Danger, but can be decreased by exposing the cell to Safe signals. Each time a cell polls the environment to update its internal variables it also removes 'antigen' from the input buffer and stores these symbols locally.

At the end of every algorithm cycle the cumulated CSM value of each cell is compared to its migration threshold. This threshold is allocated to every cell at random, using an application-specific probability distribution. If the accumulation of CSM is greater than or equal to the threshold, the cell is said to 'migrate'. The migration process triggers a decision to be made based on the relative concentrations of $IL10$ and $IL12$. If the cell has accumulated more $IL12$ than $IL10$, its decision is positive, otherwise its decision is negative. When the decision is reported, the cell also presents all of its locally-stored antigen so that the symbols can be correlated with the decision. When a cell has migrated, it is removed from the population and a new cell is put in its place. The migration threshold of the new cell is randomly assigned using the probability distribution.

A History of the DCA

The initial proposal of the DCA in [45] outlines the algorithm as it is explained in Appendix B.2. A fixed set of weightings for calculating the intermediate signals from the input signals was outlined that had been inspired by biological experiments performed as part of the Danger Project [3]. As part of the proposal a prototype algorithm had been written and applied to the UCI Wisconsin Breast Cancer machine learning data set [13]. The PAMP and Safe signals were generated as a dual of one another, based on the status of one of the key factors in the data set. This has since been found to be a poor design decision as it neither represents a 'signature' of danger, as a PAMP is supposed to be,

nor does it allow situations where the PAMP signal is suppressed by the Safe signal to be created [42]. The classification accuracy is reported as 99% although the authors point out that the DCA is not meant to be a classification algorithm. This is indicative of the early work in the DCA where its population-based approach and its correlative stage meant that fitting it into one of the traditional machine learning or algorithmic fields was difficult. Even so, classification was performed and was achieved by aggregating all of the cell states from the entire run of the algorithm. If a specific antigen was found to have more mature cells associated with it than semi-mature cells, it was considered anomalous.

In [47] the algorithm was applied to a computer security problem, namely port scan detection. This version of the algorithm had a slightly different aggregation metric (pseudocode in Appendix B.4). Instead of simply casting votes for a given antigen the algorithm reported the number of mature cells that had migrated for a given antigen, as a fraction of the total number of cells that had sampled that antigen. This metric is still the most commonly used aggregation technique and is referred to as the MCAV, (mean context antigen value). This metric has proved popular, as it effectively provides a probability that a given antigen is anomalous. This work was the first to raise one of the key properties of the DCA, the emergence of a 'guilty by association' heuristic. In the data set two 'innocent' processes and two 'malicious' processes were running and the aim of the algorithm was to identify which were which. However, one of the innocent processes was required to run one of the malicious processes, so there was a regular, temporal correlation between the two. This meant that dendritic cells with large migration thresholds tended to see both processes and detect the Danger and PAMP signals from the latter. As a result one of the innocent processes was assigned an 18% probability of being malicious. This is in many ways a pleasing result

as it demonstrates that the algorithm was able to detect an unsuspected correlation in the data.

The results from both [47] and [45] are summarised within [49]. This work goes on to use the previous two data sets to further explore the properties of the DCA. However, this version also introduces a new inflammation metric not used in the previous works, (pseudocode in Appendix B.3). Unfortunately, at this stage, there was still no comparison between the DCA and an established machine learning algorithm. The major contribution of [49] is the introduction of an inflammation signal into the algorithm. Inflammation was selected as another input heuristic that acted as a multiplier to the other input heuristics. Its inclusion seemed to have a derogatory effect on the experiment, though a thorough investigation into a range of parametrisation was not performed.

In 2006 Kim et al. performed experiments using the DCA, (as described in [47] and Appendix B.4) to detect the 'interest cache poisoning attack' in sensor networks [69]. The paper contributes little to the knowledge base of the DCA as, not only is the 'interest cache poisoning attack' an attack specially designed for the paper, but the paper is only conceptual, outlining how the algorithm could theoretically be used to detect such an attack.

Further evidence for the confusion surrounding the functionality of the algorithm is found in [46], where the authors describe the DCA as a 'sorting algorithm', though with little further explanation it is difficult to understand the justification for this description. In this paper the version of the algorithm from [47] is compared with several incorrectly parametrised versions. The aim of this is to demonstrate that the two functionally similar signals, Danger and PAMP can be interchanged under these conditions, but the correct parametrisation of the Safe signal is more crucial. In addition, [46] performs sensitivity analysis on the

signal weightings. The range of values used by the sensitivity analysis is dubious as, for all of the experiments, the ratios between the weights is kept constant with the exception of the weighting between the Safe signal and the $IL10$ signal, which is held at one for the duration of the experiments. As a result, rather than exploring the sensitivity of the weights, as reported, the experiment really explores the effect of differing the weight ratios between CSM, $IL12$ and $IL10$. Even this is not quite true, as the $IL10$ weights are fixed throughout the entire experiment.

In 2008 Al-Hammadi et al. carried out a further empirical study, applying the DCA to a bot net detection problem [4]. In this work a new technique is put forward for aggregating the results of the algorithm, called the MAC. This new version of the algorithm is described using the pseudocode in Appendix B.6. This technique is still a batch process, but scales the original MCAV calculation for a given antigen by the number of antigen used to calculate the MCAV in order to add some measure of confidence to the measurement. This result is then normalised by the total amount of antigen sampled to return it to a 0–1 range. At first glance this seems like a sensible metric. However, if one rearranges these calculations as shown below, an obvious flaw becomes apparent.

$$\text{MAC}_x = \frac{\text{MCAV}_x \times \text{Antigen}_x}{T} \tag{2.1}$$

$$\text{MCAV}_x = \frac{\text{Mature}_x}{\text{Antigen}_x} \tag{2.2}$$

Substituting 2.2 into 2.1

$$\begin{aligned} \text{MAC}_x &= \frac{\text{Mature}_x}{\text{Antigen}_x} \times \frac{\text{Antigen}_x}{T} \\ &= \frac{\text{Mature}_x}{T} \end{aligned} \tag{2.3}$$

Where:

MAC_x is the MAC value for antigen x

MCAV_x is the MCAV value for antigen x

Antigen_x is the sum of antigen samples collected for antigen x

Mature_x is the total amount of antigen samples that were collected by mature DCs for antigen x

T is the total amount of antigen sampled by all DCs for the application.

As a result, the MAC for an antigen is simply the total number of antigen sampled in a mature context of the same type, as a fraction of the total number of antigen sampled overall. For large data sets or applications where the distribution of antigen is not roughly even, this could result in infrequently occurring antigen types being overwhelmed. Though it is of note that in smaller data sets, this technique could yield a lower false positive rate, a known issue with the DCA [42].

In [48] a comparison was made between the DCA and an established machine learning algorithm, the self-organising map (SOM) [71]. The comparison was performed using the port scan data set from [47]. As discussed, it is a challenging task to empirically compare the DCA to an existing machine learning algorithm as its intended functionality does not easily fit into any of the established categories. In this version of the algorithm, the concept of inflammation is reintroduced and combined with the other innovations introduced by other versions of the algorithm, the combined algorithm is outlined in Appendix B.5. The inflammation model uses a Boolean value that either multiplies the other signals by two or not. The results of the comparison demonstrated that, for that data set, the DCA out-performed the SOM for classifying anomalous processes but there was no difference for classifying normal processes.

While this is a useful result, tests with a broader range of data sets would have to be performed to characterise the areas where the DCA performs well. An additional problem with this comparison is that the SOM is a supervised learning algorithm requiring a training phase. This means that its performance is heavily dependant on the selection of the training data. For this comparison only 'normal' data was used to train the SOM for reasons that are not made entirely clear. A fuller set of tests, exploring the effects of different samples of training data would also have benefited this investigation.

Lay and Bate approach a different anomaly detection problem in their two DCA works, [77,78]. Both works focus on an attempt to identify possible over-runs which cause tasks on a real-time system to fail to meet their output deadlines. The major contribution of the first paper from Lay and Bate, [77], is an attempt to dynamically tune the weights for the signal processing phase of the dendritic cell algorithm. In this approach, the weights for the signal processing phase of the algorithm are randomly assigned. Each cell is rewarded or punished based on its performance classifying a simple data set with a known ideal response. If a cell's performance is too low, its signal weights are mutated. A moving average of the punishment/reward system for the entire cell population is used to determine the threshold for "poor performance". The results from this approach are shown to be very different to the results of the standard DCA, though no statistics are used to demonstrated if they are significant differences. Sadly, the graphs used to display the data are difficult to interpret in the black and white format of the conference proceedings, so it is difficult to independently assess the quality of the changes made. In the journal paper this adaptive technique is abandoned in favour of the traditional DCA, (the version presented in [47]). This paper focuses on comparing the DCA with a more common-place statistical

technique for assessing process over-runs. The comparison demonstrates that the DCA performs comparably with the statistical technique, but is heavily dependent on the parametrisation of the DCA. Of particular note for this work, Lay and Bate also provide a discussion about the differences between hard and soft real-time constraints. As many of the future chapters of this thesis will be exploring a robotic system, a summary of that discussion will be presented here. Lay and Bate make a distinction between 'hard' and 'soft' real-time constraints [78]. A "real-time system" is defined as a system where the quality of a solution is not simply a function of solution accuracy, but also a function of when that solution is presented. The constraint placed on the system, in terms of its timeliness, is said to be a "hard constraint" if failing to meet that constraint will cause a serious system failure, resulting in the system being no longer fit for purpose. If failing to meet a constraint is undesirable, but not system critical, that constraint is said to be a "soft constraint".

The history of the DCA contains evidence that it has promise as an anomaly detection algorithm. Its performance on selected data sets has been shown to be comparable or better than established machine learning algorithms, albeit in a limited manner [43, 48, 49]. However, there is much criticism about its application and possible uses. Much of this criticism stems from the absence of good analysis of the algorithm and a working understanding of what problems the algorithm is best suited to solve. Empirical experiments on new data sets will support the case for the dendritic cell algorithm by providing more insight into its application potential. However, only theoretical analysis will be able to prove its worth against other techniques. Theoretical analysis will provide definitive 'rules' about the types of problem that the DCA can work with and will shed light on, what is for many, a black-box algorithm. Ultimately it will be theoretical analysis that will definitively add or

remove the DCA from the AIS field by demonstrating its merits and its flaws. Other algorithms within AIS have undergone this process and have been suffered/benefited accordingly. What follows is a review of the theoretical analysis that has been performed on AIS algorithms in the past, which both provides context for the importance of such analysis and is a potential source of theoretical methods which could be used on the DCA.

2.3.5 Theoretical Analysis of Artificial Immune Systems

In 2008 Timmis et al. produced a review paper outlining the state of the art in theoretical analysis of artificial immune systems algorithms [142]. Here we will revisit some of that work, augment it with some more recent publications and place the known work on DCA analysis in context.

By far the most analysed family of artificial immune systems are negative selection algorithms. As discussed in 2.3.3 the original negative selection algorithm used an affinity measure called r-contiguous bits [36]. At that early stage, time complexity analysis revealed that the sheer number of detectors required meant that detector generation was a computationally expensive task, especially for large values of r. An alternative affinity function called r chunking was suggested in [8]. Three crucial flaws were pointed out in successive years from 2001 to 2003. Firstly [68] highlighted the additional time complexity issues as the dimensionality and size of shape space increases to the sizes required for real world problems. Secondly, [30] highlighted that positive selection is simply more practical for most real-world applications. For the majority of real-world tasks, the space of anomalous behaviours can be huge or even infinite. With a finite number of detectors, this makes covering the complement of that set, i.e. the normal behaviours, much more feasible. Thirdly, Freitas and Timmis

highlighted the risks of arbitrarily picking affinity measures without considering the possible biases that this could introduce into the behaviour of the algorithm. In addition to these problems the work of Stibor et al. highlighted many more shortcomings of the algorithm revolving around shape space coverage, computational complexity and the inefficiency of random detector generation [130, 132, 133]. Despite the extent of the work demonstrating the problems with negative selection Elberfeld and Textor were able to demonstrate that a solution does exist for producing detectors in polynomial time for string based negative selection [31].

Theoretical analysis has also been applied to clonal selection algorithms. In [142] a description is given of Markov chain analysis of a clonal selection algorithm to prove its convergence properties. While it is possible to show the algorithm does converge, the upper bound of that convergence is not shown. In 2009, Jansen and Zarges were able to demonstrate an upper bound for somatic contiguous hypermutations, a specific type of mutation operator [63].

Idiotypic networks have had very little theoretical analysis performed on them. This is likely to be due, in part, to the diverse nature of the algorithms. In [142] techniques based on differential equation analysis are suggested as possible approaches for exploring this type of algorithm. Such techniques have been extremely successful when applied to complex, biological networks based on neuronal interaction [152].

It is thought by the author that the work of McEwan et al. is particular importance to the field of AIS [86]. This work questions one of the fundamental principles of artificial immune systems, the computational representation of affinity. This concept permeates all three of the most commonly used algorithms in the field, so improvements and optimisations have the potential to yield a wide variety of positive results. As discussed earlier in this section, affinity is generally performed within

some form of shape space, which can suffer from the 'curse of dimensionality'. McEwan argues that, while kernel methods can help when the data starts as a low dimensional vector but needs to perform operations in high dimensional space before returning to a lower dimension, they cannot help when the input data starts as being high dimensional. Instead it is suggested that the mechanics of receptors and other constructs within the immune system may be more analogous to feature extraction, selecting lower dimensional, yet representative samples of the available data.

Some theoretical analysis has been attempted on the DCA. In [44] Greensmith presents a modified version of the algorithm that removes all stochastic elements from the algorithm and reduces the dimensionality of the input data by ignoring the PAMP signal. This simplification is aimed at making algorithm analysis easier in the future. The paper presents a revisiting of the port scan detection data set, to verify that the core functionality of the algorithm is still present. The work of Gu et al. has also explored properties of the DCA [51,52]. These papers have explored techniques for using the DCA as a real time algorithm, partially based on the work presented in Chapter 3 of this thesis.

2.4 Robotic Applications of AIS

Biologically inspired algorithms are at their best when the link between the software and the underlying biological principles is not a purely semantic one. There are a great number of benefits to be gained from stripping away biological complexity from an algorithm, but that process assumes a deeper level of connection to the biological inspiration than simply a naming convention. A great deal of robotics literature from AIS suffers from this purely semantic link. What is presented here

is a summary of applications with a more fundamental connection to models of the immune system.

Some of the earliest robotics work from AIS was mobile robot controllers. Watanabe et al. present an idiotypic network where the cells represent behaviours [147]. In a similar way to Brooks' subsumption architecture [15] the complexity of the robot's behaviour comes from the interaction of simple robotic behaviours. Instead of using a hierarchical approach, Watanabe et al. allow the behaviours to be excitatory or inhibitory of each other, with additional input from the sensors. Designing the mechanics of such a network would be a challenging task, so a genetic algorithm is used to design the network dynamics using a simulator. The results demonstrate that the technique is able to find the optimal solution for a simple collection problem.

Lau and Ko present a framework for search and rescue robots based on the interaction of T cells with the inflammatory response called the 'general suppression control framework' or GSCF [70, 76]. Each behavioural component of the robot is modelled as a T cell with a receptor that gives it antigen specificity. The receptor models the behaviour's appropriateness in a given situation. The robot's sensors feed into an affinity evaluator which selects the appropriate behaviour based on that receptor. A series of signatures such as the location of a target or the immediate presence of an obstacle cause an anti-inflammatory reaction, suppressing the cell behaviours and stopping the robot. Unfortunately, no statistical results are presented on the effectiveness of the algorithm. Hart et al. go further than simply arbitrating behaviours and instead propose a system for growing the behaviours themselves [56]. The system also uses an affinity metric to determine the appropriateness of a given behaviour, but the behaviours themselves can also stimulate or inhibit other behaviours in the style of an idiotypic network. An initial set of behaviours are given

to the robot as a starting point. New behaviours are presented to the network and pruned if they are unable to stay stimulated. Stimulation depends on the sensory input to the robot and the interaction with the other cells. A simulated implementation of the system demonstrated that the robot was able to create new experiences and was able to correlate simple actions to their effects. Later work by Whitbrook et al. generates the simple behaviours using a genetic algorithm and uses an on-line reinforcement learning technique to form those behaviours into an idiotypic network [149]. This technique performs the behaviour generation in a simulator and then applies it to a search and rescue task on a real robot. Several experiments were performed leaving out various stages of the architecture and replacing them with more traditional controllers, however the idiotypic network-based controllers outperformed the others and never failed to locate the target.

As a meta-heuristic for optimisation, clonal selection can readily be applied to a wide variety of problems. In [79] clonal selection is used to find the optimal parametrisation for the PID, (proportional, integral and differential) controller on an underwater autonomous vehicle (for an overview of PID control see [146]). The results of the clonal selection technique are compared to the results obtained from the standard Ziegler-Nichols technique for identifying PID control parameters [154]. In the experiment, clonal selection identifies a set of parameters that appear to give a much better performance than the traditional technique. In particular, the yaw controller produced by the clonal selection algorithm is much better. However, there are two, key problems with these results. Firstly the results are obtained via simulation. While simulation techniques can be an excellent way to try ideas, their correspondence to the real world performance of a robot can be tenuous. Secondly the affinity function used by the clonal selection algorithm fails to consider

any other stability criteria than the overall error. The Ziegler-Nichols technique does not just promise a fast solution; it also guarantees a stable solution, deliberately avoiding some regions of parameter space to prevent the controller from being too close to instability. A fairer comparison could be achieved if the affinity function were altered to factor these conditions into the search.

The work of Neal and Timmis attempts to incorporate concepts of immunity into robotic systems to make them more robust to failure [103]. As purely mechanical devices, robots are vulnerable to wear over prolonged periods of use. However, the failure of a single system need not mean that a robot becomes inoperable, as long as the failure of that system is identified. A controller architecture is described that combines a neural network with ideas from artificial immunology and artificial endocrinology.

In this architecture the role of the immune system is to remove components from the controller that have been identified as detrimental and to identify environmental or fault based changes that require controller adjustment. The neural network element of the system provides the direct control, but its behaviour is adjusted and manipulated by the other two components of the system. The initial stages of this work have been applied to real robotic systems by Timmis et al. in 2009 though it currently focuses on the interplay between the endocrine system and the neural controller [143]. Lau et al. have also presented a framework for using AIS for fault detection, with the emphasis on foraging tasks for swarms of robots [75]. Faults are characterised as anomalies and can include environmental hazards that prevent agents from achieving goals. This work is carried out from an immuno-engineering perspective as outlined in [141]. As a result this initial research focuses on characterising the problem rather than a direct application of AIS.

Since work presented in this thesis was published there has also been an application of the dendritic cell algorithm to a mobile robotic platform. In [101] a modified version of the DCA is presented for detecting faulty sensors. This version of the DCA has a fixed migration threshold and a modified, integer representation of CSM. The cell population for this implementation is considerably smaller, with each cell monitoring a single sensor. Each cell stores the history of its past eight migration states. If all are mature, the sensor is considered to be faulty and the robot no longer responds to it. A type of adaptation is introduced by having partial redundancy between the sensors. This should prevent neighbouring sensors from reporting drastically different values and therefore allow anomalous readings to be detected. The results from a simulator and a real robot experiment were presented. Two faults were artificially introduced, a stuck sensor, reporting the last 'good' value continuously and a sensor reporting random noise. The results demonstrated that the DCA was able to detect these faults. While the author does point out that the small number of cells allows the algorithm to be implemented on a micro-controller with very limited resources, its departure from the population-based nature of the algorithm does effectively reduce it to a slight variation of a hand-tuned linear classifier, (see Chapter 7). In addition, its reliance on neighbouring sensors sharing partial information could potentially cause problems in highly cluttered environments with legitimate structures that cause neighbouring sensors to report drastically different values. Such situations should only occur transiently while the robot is moving, so the counter should protect the robot against them, but there will be situations where the robot may be stationary while performing a complex task. A better solution may be to make the duration of the counter dependent on the encoder data from the wheels and hence the movement of the robot.

2.5 Conclusions

The information collected within this review can now be used to frame the research questions presented in Chapter 1 within the context of the current state of the art within the fields of artificial immune systems and robotics.

2.5.1 Migrating from Software to the Physical World

> Can an immune-inspired, anomaly detection algorithm be adapted to solve threat detection problems in the physical world, through the medium of a robot?

The current state-of-the-art in security robotics has several key limitations. Most importantly, all of the systems encountered were unable to tolerise against dynamic objects, constraining them to being used in empty warehouses or outside the hours of operation for a building. Secondly, the inability of the robots to perform threat detection while moving severely limits their applicability in the real world. In AIS computer security applications there has already been a recognition that 'normal' requires a more complex description. It would be totally impractical to have a computer security system that treated all network activity as a threat. As a result, there are already algorithms, such as negative selection and the DCA that attempt to differentiate between normal and anomalous behaviour in a more intelligent way.

2.5.2 Emergent Properties of the Dendritic Cell Algorithm

> Does the dendritic cell algorithm have properties that were not explicitly added as part of its design, which could be advantageous to a robotic application?

While the dendritic cell algorithm has potential as a threat detection algorithm, it is yet to be fully characterised in the literature. As a population-based and highly stochastic algorithm, it is likely that there are emergent properties that are yet to be identified. Of particular interest is the algorithm's ability to deal with imprecise and noisy data, a feature of robotic systems. As highlighted by the problems with other AIS techniques, it is important to ensure that abstractions of immune concepts are not simply 're-inventing the wheel' so any investigation with the DCA at its core, must assess it from a theoretical stand point, not simply an empirical one.

2.5.3 Applying The Dendritic Cell Algorithm to a Robot

> Is it possible to adapt the dendritic cell algorithm from being a batch system to a system that can operate on a robotic platform?

As discussed, there is little work on theoretically analysing the DCA. However, there is precedent for alternatives to the batch system of generating the MCAV. This suggests that the core functionality of the algorithm is independent of the cell analysis phase. When this investigation was begun, the implementations of the DCA were all performed within the 'libtissue' framework and very slow. It is an important step to guarantee that the algorithm can be made light-weight enough to run on a robot on-line for the system to be a feasible solution. The work of Lay and Bate, while not directly applicable, raises important issues relating to identifying soft and hard real-time constraints within the system which must be addressed to answer this question [78].

2.5.4 The Benefits of the Dendritic Cell Algorithm

> Are there other algorithms with functional equivalence to the dendritic cell algorithm, which can outperform it in terms of reduced computational complexity or superior performance, for the threat detection problem?

While weaknesses in the current solutions to the robotic security problem have been identified, a solution based on the DCA should be critically appraised. With so few results published in the literature, it is difficult to see how a direct like-for-like comparison could be produced. However, its performance should be at least as good as a random classifier, to demonstrate that information about the problem is being used to make appropriate decisions. In addition, it should be clarified if the DCA has functional equivalence to other techniques, or a collection of other techniques, and if such a system could outperform the algorithm.

Chapter 3

The Robotic Dendritic Cell Algorithm

"You see, but you do not observe. The distinction is clear."
- Sir Arthur Conan Doyle (Sherlock Holmes), A Scandal in Bohemia (1892)

3.1 Introduction

The work presented in this chapter is based on [107].

As discussed in Chapter 1, technologies and protocols designed to enforce security are now pervasive in society. Many houses now have burglar alarms and CCTV is common-place in towns and cities. Both Chapters 1 and 2 highlight the need for each robot within a robotic security system to be able to run as a self-contained agent. As a result any algorithm employed by a robotic security system must prove itself to be capable of running in its entirety on a mobile robot, without having a detrimental effect on core functions, such as obstacle avoidance.

At the time that this research was carried out the dendritic cell algorithm (DCA) had not been used as a real-time algorithm. In order for it to be useful as a processing algorithm for a mobile robot it is necessary to address architectural limitations that have previously constrained it to be a batch processing algorithm. Once engineered for use with a real-time processing system, the DCA must be interfaced to a robotic operating system. In addition to these modifications, the implementation of any version of the DCA requires the input heuristics to be specified and designed. Finally, it is necessary to parametrise the population distribution for the specific application. This is vitally important as it has been shown in [42] that the performance of the DCA is significantly effected by the correct identification of this distribution.

At the prototype stage it is important to verify that the performance of the DCA is of a sufficient quality to pursue. Later chapters will explore the DCA's performance relative to other classification techniques using theoretical analysis. In this chapter the target shall be to outperform the calculated classification rate based on the expected results for a classifier operating at random. To outperform such a classifier is a minimum requirement of any potential algorithm.

In short the hypotheses that this chapter shall explore are as follows:

- It is possible to modify the DCA to work on a real-time system while retaining the core properties of the DCA. This hypothesis is rejected if the algorithm is unable to return classifications within the soft real-time constraint of being within 5 seconds of witnessing an event or if the algorithm requires more computation time than the hard real-time constraint of 250ms per allocated processing slot. More information about these constraints is provided in Section 3.2.4.

- Untrained and using unfiltered sensor data, the DCA outperforms a random classification technique. This hypothesis can be accepted if a single parametrisation can be found which causes the average classification error for the DCA to be lower than a random classifier, theoretically calculated from the layout of the environment.

In Section 3.2 the modifications required to implement the DCA on a real-time system are discussed. In Section 3.3 an experiment is designed to test the classification accuracy of the system. In Section 3.4 the preliminary results are presented and analysed and problems with the prototype are highlighted. In Section 3.5 a modification to the original system is presented that corrects one of the major short-comings of the initial system. In Section 3.6 the results of the improved system are presented and analysed. Finally, Section 3.7 discusses the contributions made by these experiments and work that has subsequently been carried out by other researchers.

3.2 Adapting the DCA for use on a Robotic Platform

3.2.1 Robotic Platform

Throughout this work, the platform used for all physical robotics experiments is a MobileRobots Pioneer 3DX. This robotic system has a broad variety of sensors, including a laser range finder (LRF), an array of sonar sensors and a pan-tilt-zoom camera. On-board processing is performed using an 850MHz Pentium III processor running Debian Linux, (kernel version 2.6.10). A full technical specification can be found in Appendix C. The manufacturer's "Aria" library is used to control the device. The Aria control system is an object-orientated (C++) library which is structured to support a standard robot control architecture known as "the subsumption architecture". All software compilation is performed using g++ version 4.0.2.

The hardware architecture of the system is controlled via a single interface board which enables communications between the computer and all of the transducers with the exception of the vision system. The interface board receives commands via the on-board serial port on the motherboard. The camera's pan-tilt-zoom mechanisms and image transfer are controlled via a direct serial connection to the camera itself.

It is of note that the Pioneer 3DX is extremely computationally powerful. This means that it makes an excellent prototyping platform for algorithms as it can withstand a large processing load. However, it is acknowledged that enabling the DCA to work on such a powerful platform does not necessarily mean that all robotic platforms will have the necessary computational power.

3.2.2 DCA Optimisation

The starting point for this work is the DCA version outlined in Appendix B.4, as described in [47]. This version of the DCA was a batch processing algorithm. This meant that little effort had gone into improving the running speed of the system. In addition, this version was developed as part of a larger project and relied on several libraries which implemented simulations of tissue and other biologically inspired constructs. For the purposes of optimisation the DCA has been re-implemented in C++ using only the commonly used standard template library (STL) to support faster manipulation of collections of objects.

The next step of speeding up the DCA was to identify how to minimise the processing performed by each cell. This was achieved by rearranging the block diagram in Figure 2.6. Primarily any computation that does not require any persistent state can be moved outside each cell and into the 'tissue' (signal pre-processing). This reduces the overall processing requirements for the DCA as any computation that can be performed instantaneously from the input signals can be done once per iteration for the entire population, rather than once per cell, per iteration.

Equation 3.1 demonstrates a simple, generic rearrangement that allows a further optimisation to be carried out.

$$X > Y \equiv (X - Y) > 0 \tag{3.1}$$

This rearrangement allows two important changes to be made. Firstly, substituting $IL12$ for X and $IL10$ for Y, results in a comparison between their difference and 0. A comparison between a number and 0 is slightly computationally faster than a comparison between two non-zero values. Secondly, rather than storing $IL10$ and $IL12$ separately, one can represent them both as a single, abstract quantity K, thus reducing the memory requirements of each cell. Equation 3.2 defines K as the instan-

taneous difference between the two original intermediate variables.

$$K = IL12 - IL10 \tag{3.2}$$

If the summation of K is greater than 0 the outcome is positive. If the summation of K is less than or equal to 0, the outcome is negative. As both $IL10$ and $IL12$ are expressible as weighted sums of the three input signals, K can be expressed as a single weighted sum, where the weights are given by Equation 3.3.

$$W_{sk} = W_{s12} - W_{s10} \tag{3.3}$$

Where: W_{XY} is the weighting between input signal X and internal signal Y.

This allows K to be computed directly from the input signals, removing the need to calculate $IL10$ and $IL12$ altogether.

Figure 3.1 is the new block diagram for a DC. In this new model, CSM and K can be considered as inputs from the signal preprocessing to the cell population. This reduces the number of instructions per cell, per iteration from twenty one to two. Dr. Julie Greensmith provided test data from the experiments published in [47] to allow verification that the new system is functionally equivalent to the original DCA. The results are available from http://www.cs.nott.ac.uk/ rxo/thesis/ and the raw data is available from Dr. Julie Greensmith on request. A paired Wilcoxon test could not reject the hypothesis that the two were from the same sample set using a significance of 0.05.

3.2.3 DCA Modifications

This section discusses the final stage of the process which lead from the version of the DCA outlined in Appendix B.4 to the version outlined in Appendix B.7. Transforming the DCA into a near-real-time system

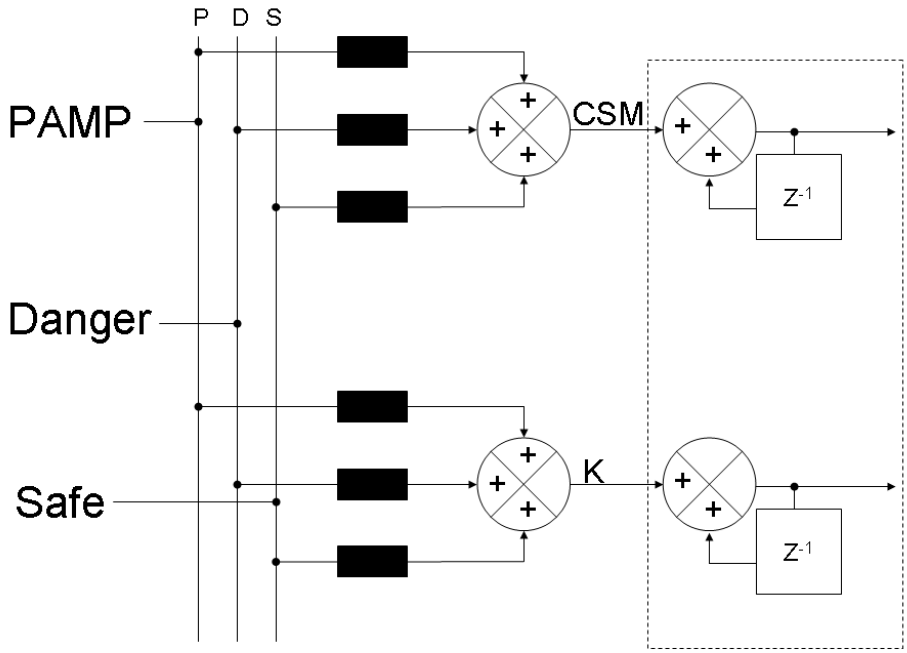


Figure 3.1: The Optimised Block Diagram for a Dendritic Cell

Only processing contained within the dotted line needs to be carried out on a per cell basis. All other processing can be performed within the tissue.

required a modification of the way the DCA reports its outputs. In the original DCA it was possible to produce the MCAV for every antigen over the entire data set. To enable a close to real-time response the DCA was modified to calculate and report the MCAV at the end of each one second time window. In the event that no cells mature within a given window this statistic is reported to the log and the system continues. Since this investigation was published, additional work has been carried out by Gu et al. which compares this technique with an antigen-orientated segmentation technique [52].

As the antigen is generated based on a specific robot location, it is possible that an ineffective amount of antigen will be generated. One solution for this is to add multiple copies of each antigen to the DCA environment as suggested in [144]. A novel extension to the DCA for this application is an antigen multiplication function. This function adds varying amounts of each antigen depending on the speed of the robot. Areas passed through slowly are made to contribute more antigen than areas passed through quickly. This is done because areas passed through quickly contribute less signal to the DCA environment, as less time is physically spent within that area. The weighting function is given in equation 3.4. The aim of the function is to add between 2 and 102 copies of each antigen to the input buffer depending on the speed of the robot. The lower limit of 2 is to ensure that at least two cells sample an antigen and the upper limit was selected to guarantee that in the upper case, all cells sampled the given antigen.

$$W(v,\dot{\theta}) = \left\lfloor 75\left(1 - \left|\frac{v}{v_{max}}\right|\right) + 1 + 25\left(1 - \left|\frac{\dot{\theta}}{\dot{\theta}_{max}}\right|\right) + 1 \right\rfloor \tag{3.4}$$

In equation 3.4 v is the velocity of the robot, $\dot{\theta}$ is the rotational velocity of the robot and W is the amount of antigen added to the environment.

The smallest amount of antigen that can be added is 2, when the

robot is at maximum velocity and maximum rotational velocity. The maximum amount of antigen that can be added is 102, when the robot is totally stationary.

The full pseudocode for this new version of the DCA can be found in Appendix B.7.

3.2.4 Interfacing to a Robotic Operating System

Developing a robotic system is a demanding task as robust, real-time control is difficult to achieve. Brooks' "subsumption architecture" (first proposed in [15]), has been shown to be an effective way of designing robotic control systems [16]. Such architectures rely on the development of a family of simple "behavioural modules" that interact to produce more complex behaviour. For example, the complex behaviour of wandering through a dynamic environment, without hitting obstacles can be achieved through the interaction of two trivial behavioural modules. Figure 3.2 illustrates a simple subsumption architecture. The lower priority behaviour simply moves the robot forwards at a constant velocity. In the event of a higher priority behaviour detecting an obstacle, it can subsume the output of the low priority behaviour and steer the robot away or, in emergencies, stop. The interaction between the two modules ensures that the robot is always moving when possible, without hitting obstacles. This architectural style has now become so common-place that software libraries for the creation of robot controllers are often structured specifically to support it. This is true of the Aria library provided to control the Pioneer robot.

The Aria Library

The Aria library provided to control the Pioneer robot is geared towards creating subsumption-based programmes. Aria is an object-orientated

library which allows the programmer to create individual behaviours, (termed 'actions' in the Aria language) and collections of behaviours, (termed 'action groups' in the Aria language.) Action groups are wrapped in a container class called a 'mode' which activates the action group and stores additional information such as help tips and text descriptions of the intended behaviour implemented by the action group. Action groups are hierarchical, as with the subsumption architecture, and each behaviour within a task is allocated an integer value determining its importance. The integers are within the range of 0 to 100, with 100 meaning a high-priority action which is required to be run as often as possible, and 0 meaning a low-priority action, which can be run infrequently. Tasks with higher priorities have the ability to over-ride the movement commands of lower priority tasks, as with the subsumption architecture.

The robotic DCA is implemented as a stand-alone action for compatibility with a subsumption architecture. Figure 3.3 illustrates the architecture which implements a simultaneous wandering and DCA classifying behaviour. Note that, as is standard for diagrams of subsumption architectures, the highest priority behaviours are illustrated at the bottom and the lowest priority behaviours are illustrated at the top. This stems from the original paper, where the distance from the bottom is used to symbolise the 'distance' that the action is from directly interacting with the sensors and transducers [15]. This action group extends the Aria library's standard 'wander' architecture with an additional module for image processing and an additional module for executing the DCA. In this action group, the DCA is given priority 10, the camera processing is given priority 25 and the other actions occupy the range 50 - 100. This ensures the fundamental behaviour of moving around safely within the environment is prioritised above all other actions. In addition to the wandering and classifying action group, there is also a tele-operation, (re-

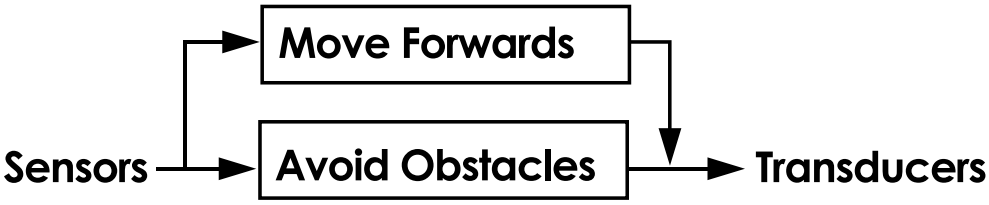


Figure 3.2: A Simple Subsumption Architecture

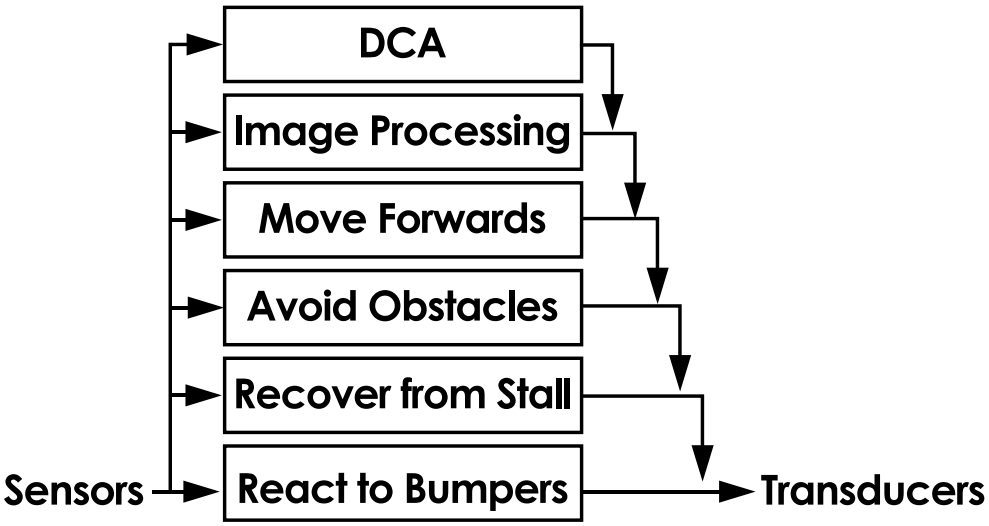


Figure 3.3: The DCA as Part of a Subsumption Architecture

motely controlling the robot from a networked machine) and classifying action group. The image processing behaviour has been given a higher priority than the DCA to ensure that the robot's internal representation of the environment is updated more regularly than the DCA samples it.

Real-Time Constraints

Hypothesis 1 discusses the soft and hard real-time constraints that the system must comply with for it to be said that the DCA has been implemented on a real-time system. Inspired by the definitions discussed in Lay and Bate's work the constraints put on the system have been separated into hard and soft real-time constraints [77, 78]. As discussed in Chapter 2, the definition of a real-time system is a system where the

quality of a solution is not simply a function of solution accuracy, but also a function of when that solution is presented. Hard real-time constraints cause serious system failure if they are not met where as soft real-time constraints only cause undesirable effects. From a functional perspective, the soft-real time constraint is that the classification of a given situation as being normal or anomalous is reported quickly enough for a user to respond to that potential threat. This is set at 5 seconds, this time limit means that the robot will not have travelled very far before realising that there is a potential threat, allowing the operator to hypothetically engage the tele-operated action group and investigate further manually. The hard real-time constraint is that a single iteration of the DCA should not require more than 250ms. This limit is set by the robot operating system in order to prevent the bumper and obstacle avoidance actions from being starved processor time. Obviously, were this to happen, the robot would crash into obstacles and potentially damage itself or an individual or item in its environment.

3.3 Experimental Conditions

3.3.1 Experiment Design

To accept or reject both hypotheses discussed in Section 3.1 it is necessary to devise an experiment that will test the DCA's performance as a classifier on a real robot. For this prototype an artificial environment was created with a tightly controlled layout. By controlling the environment so tightly it is possible to explore the effects of changing the parametrisation of the DCA with confidence that the effects of changes in the environment are not unduly influencing the results. The layout of the environment is shown in Figure 3.4. Obstacle A is a pink cylinder, with a height less than 330mm. This means that the object is visible to

the sonar array's conic field of view but not visible to the LRF's planar field of view. Obstacle B is also a pink cylinder, with a height greater than 330mm. This interacts with both the LRF and the sonar array. By choosing appropriate heuristics for the DCA's inputs it is hoped to produce a system that will react to the cylinders in the environment. The short cylinder should trigger an 'anomalous' response where as the tall cylinder should have its response inhibited by the Safe signal. An additional advantage of controlling the environment is that it is possible to identify which areas should be associated with an anomalous response and which areas should be associated with a normal response prior to the experiment. This is done by implementing a simply ray-tracing algorithm in Java. Figure 3.5 is a screen shot of the theoretical performance generated by the ray-tracer. It is acknowledged that errors in object placement and possibly even the ray-tracing program could introduce an experimental bias. However, this technique is preferable to the alternative of using a human operator to determine the "truth" for the reasons outlined in Chapter 1.

3.3.2 Signal Heuristic Selection

For the version of the DCA outlined in [49] (Appendix B.3) three signals are used as inputs to the DCA. These signals are the Safe signal, the Danger signal and the PAMP signal. The former acts to suppress the full maturation of the dendritic cells, whilst the other two stimulate the maturation. All signals contribute to the migration of the cells.

It was necessary to select appropriate sensors to provide the inputs to the DCA running on the prototype. The decision was made to interface the PAMP signal to the robot's vision system, the Safe signal to the robot's LRF and the Danger signal to the sonar array. Chapter 2 introduced the concept that PAMP signals are signatures of pathogens

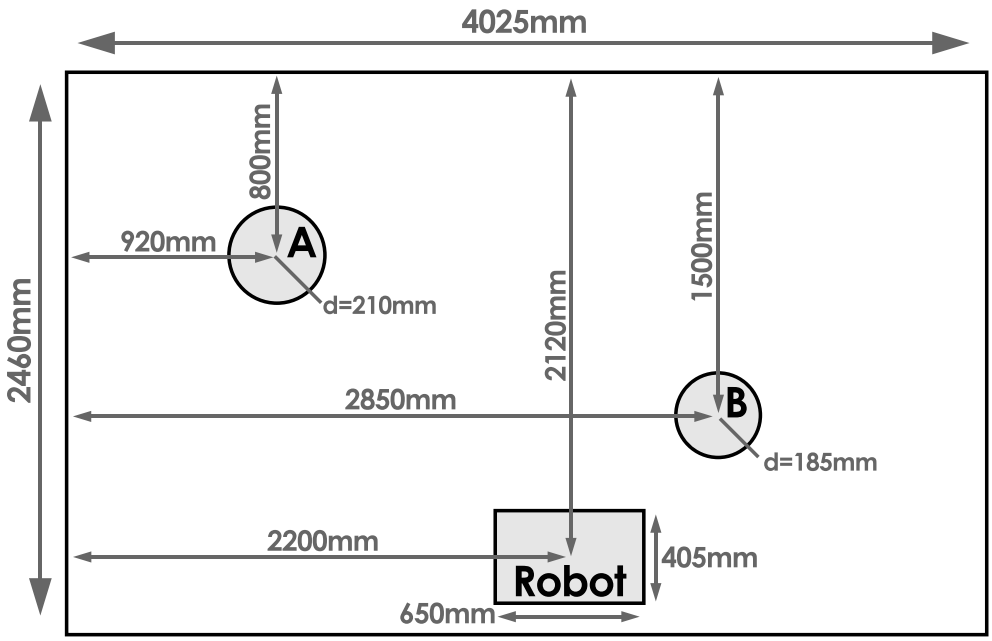


Figure 3.4: The Robot Pen

The starting conditions for each experiment. Cylinder A is the 'dangerous' obstacle, cylinder B is the 'safe' obstacle.

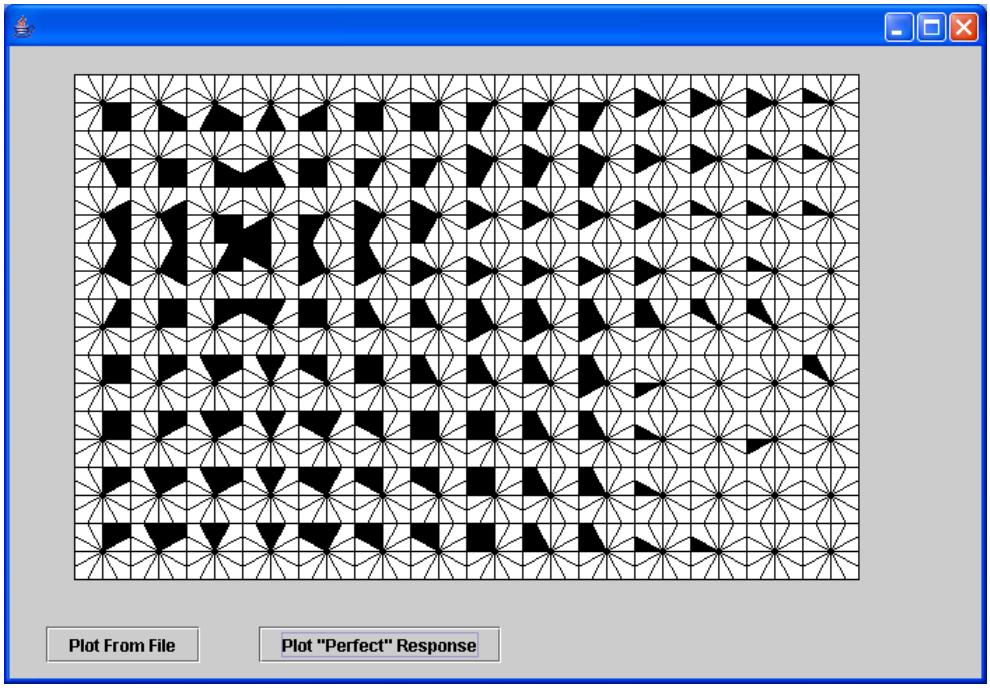


Figure 3.5: The Output from the Ray-Tracing Program

A screen shot of the programme used to determine the "true" state of each segment in the pen. Segments marked in white are normal and segments marked in black are anomalous. Note that occlusion of the shorter cylinder by the taller cylinder is taken into account.

that the immune system of a species evolves to identify. It is possible to connect the sonar or the vision system to the PAMP signal, however, the vision system provides much richer environmental data than the ranged sensors, thus it is the most applicable for implementing a signature based signal. As discussed, the relative heights of the LRF and the sonar array, in conjunction with their respective field of view (FOV) shapes, mean that it is possible for a floor-mounted object to trigger the sonar array without triggering the LRF. This holds with the biological analogy of the Safe signal being used to suppress existing environmental factors.

For the PAMP signal the input from the camera is transformed into a real-valued time-varying signal using the histogram back-projection algorithm [135]. The back-projection algorithm uses a single training image to identify the colour properties of an object of interest. All pixel groups within the image that share the same statistical properties are identified and contours are drawn around those clusters. To increase the robustness of the algorithm to changes in the ambient light, the HSV colour space was used, (as opposed to the RGB colour space where different light levels cause non-linear movement throughout the space for the same hue). The final output from the image processing library is the area, in pixels, of the largest region which matches the properties of the test image. To normalise this output within the range of 0–100, (as is expected by the DCA) this area was scaled. The scaling factor used was calculated from test data generated by a seven minute random walk around the pen. 0 corresponded to an area of 0 pixels and 100 corresponded to a value 50% greater than the median value of the trial. Anything above 100 was considered to be saturated at 100. The value of %150 of the median value was selected to prevent outliers from making the scaling range insensitive. All aspects of the image processing was performed using Intel's "OpenCV" library.

Table 3.1: Object Distance and Signal Strength for Ranged Sensors

Distance (mm)	Safe Signal Strength
0	100
300	90
600	50
900	20
1200	0

The LRF is used as the source for the Safe signal so objects taller than 330mm will produce high values of the inhibitory signal. The field of view (FOV) of the LRF extends from -90° to +90° (where 0° is directly in front of the robot). A 44° FOV is used, ranging from -22° to +22°. A narrow FOV reduces the risk of erroneous interference from walls. The distance to the closest object within the Safe FOV is returned to the signal processor. The signal processor calculates the magnitude of the Safe signal. This is performed using a look up table which relates distance to signal strength. For values that lie between those specified in the look up table, linear interpolation is used to calculate the signal strength. The values used are given in Table 3.1. For clarity, the response is plotted in Figure 3.6.

The Danger signal is sourced from the sonar array which has a 360° FOV. The Danger signal FOV coincides with the Safe signal FOV. The same look up table (see Table 3.1) is used to normalise the sonar to provide the Danger signal. The values from the look-up table were selected to roughly approximate a decay based on the Gaussian distribution, where the width of the distribution was selected to hit zero at approximately 1200mm. This places a greater emphasis on objects close to the robot

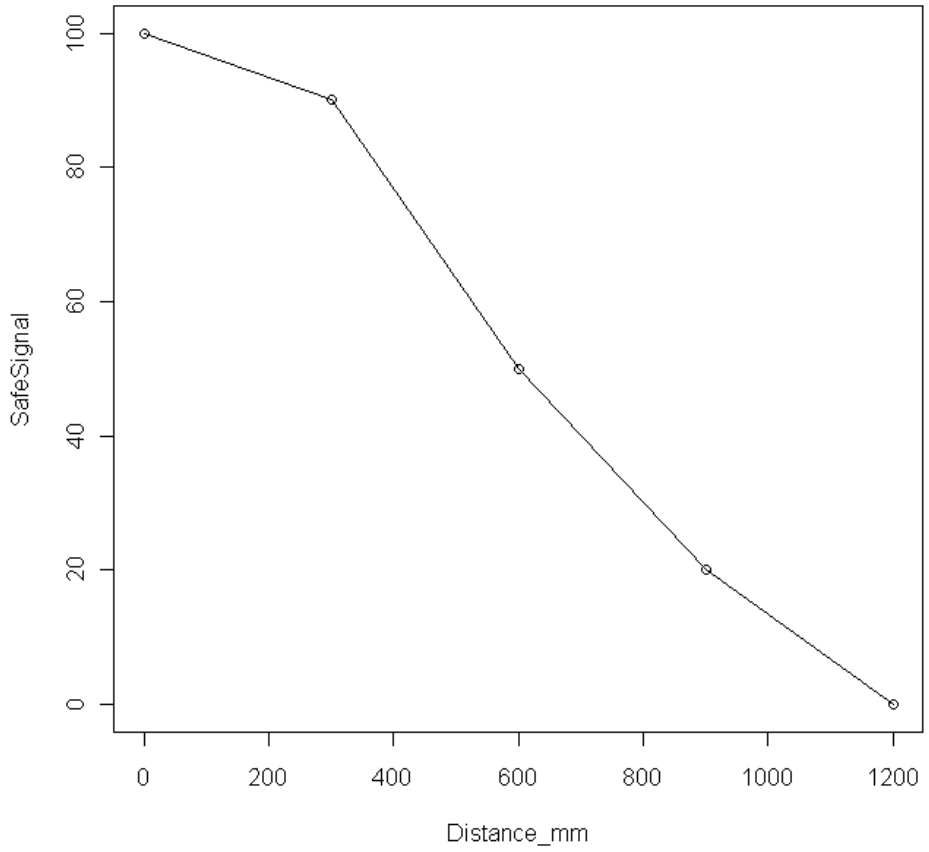


Figure 3.6: Safe Signal vs Distance

A plot of Table 3.1, showing the Safe signal as a function of distance.

and an exponentially smaller emphasis on objects as they become further away. The upper limit of 1200mm was chosen based on the parameters of the robot pen.

3.3.3 Antigen Heuristic Selection

In a practical robotic security solution, the antigen could be a vector based on the estimated location of the anomalous situation. Object-based approaches for antigen generation within a robot system have been put forward by Krautmacher et al. in [72]. For this simple implementation the antigen is an integer number which uniquely identifies a segment of the test pen. This encapsulates a small range of positions and orientations of the robot. The actual position and orientation of the robot is estimated using a 'dead reckoning' algorithm. Dead reckoning estimates the position and orientation of the robot from encoders mounted on the wheels, the fixed starting position of the robot and the diameter of the tyres. The antigen generated enumerates a 300mm grid square within the pen and a 30° segment within that square. Generating antigen based on the current location of the robot is more practical than object-based antigen, which requires a deeper knowledge of the environment to compute.

3.3.4 Experiment Parameters

Each run of the experiment allowed the robot to wander around the test pen for ten minutes. As discussed, the version of the DCA used for these experiments is outlined in Appendix B.7 The parametrisation of the DCA relies heavily on the values assigned to the cells which dictate when they perform a classification, termed the "migration threshold". The values are typically assigned to the cells using a random, uniformly distributed process, centring on a median and ranging from ±50% of

the median. The classification experiment was repeated three times for each value of migration threshold median. In each ten minute experiment approximately 800 segment classifications are performed. Table 3.2 shows the migration medians and tolerances for each experiment. In addition the experiments are assigned names to allow them to be referenced more easily.

Table 3.2: Migration Parameters for Each Experiment

Migration Median	Migration Bounds	Experiment Name
15	± 7	E1
30	± 15	E2
60	± 30	E3
120	± 60	E4
240	± 120	E5

A threshold of 0.6 is applied to the MCAV values from the DCA. Values less than or equal to 0.6 are counted as a negative or 'normal' classification, values above are counted as a positive or 'anomalous' classification.

3.4 Initial Results

The DCA was able to perform its classifications in a timely manner, (within 5 seconds of an event) and did not trigger any timing errors from the robot operating system, (meaning that its computational load did not require more than 250ms). This means that the first hypothesis is not rejected. Figure 3.7 shows the false positive and false negative rates for the DCA. To provide a benchmark for the success of the DCA the theoretical response of a random classifier is marked on the results. The

random classifier's false positive and false negative rates are calculated using equations 3.6 and 3.5.

$$P(\text{FP}) = P(\text{Negative} \cap \text{Positive Reported}) = 0.794 \times 0.5 = 0.397 \quad (3.5)$$

$$P(\text{FN}) = P(\text{Positive} \cap \text{Negative Reported}) = 0.206 \times 0.5 = 0.103 \quad (3.6)$$

Where $P(\text{FP})$ is the probability of a false positive and $P(\text{FN})$ is the probability of a false negative. The values for the probability of a negative or a positive occurring are estimated from the proportion of negatives and positives within the ray-traced simulation i.e. the ratio of black segments to white segments in Figure 3.5. It is acknowledged that the route the robot takes also influences the true probability of those events. Though with no explicit bias designed into the route, the true probabilities should be approximately equal to those estimated. The high proportion of negative segments to positive segments means that the random classifier's false positive rate is very high. Every parametrisation used outperformed this estimate. However, the rarity of positive events means that the random classifier's false negative rate is very low. In fact, none of the attempts managed to beat the performance of a random classifier. This means that the second hypothesis cannot be accepted.

The reason for this poor performance becomes apparent if one explores the performance of the DCA over time. The raw data was split into segments, each representing one minute of the robot's performance. This is illustrated in Figure 3.8. Each point on the chart shows the misclassified antigen rate from the beginning of the experiment up to the time indicated on the x-axis. Each series is the average classification error from three runs with the specified migration rate.

The classification error rates rise throughout the experiments. By taking the samples of false positives after one minute and the samples

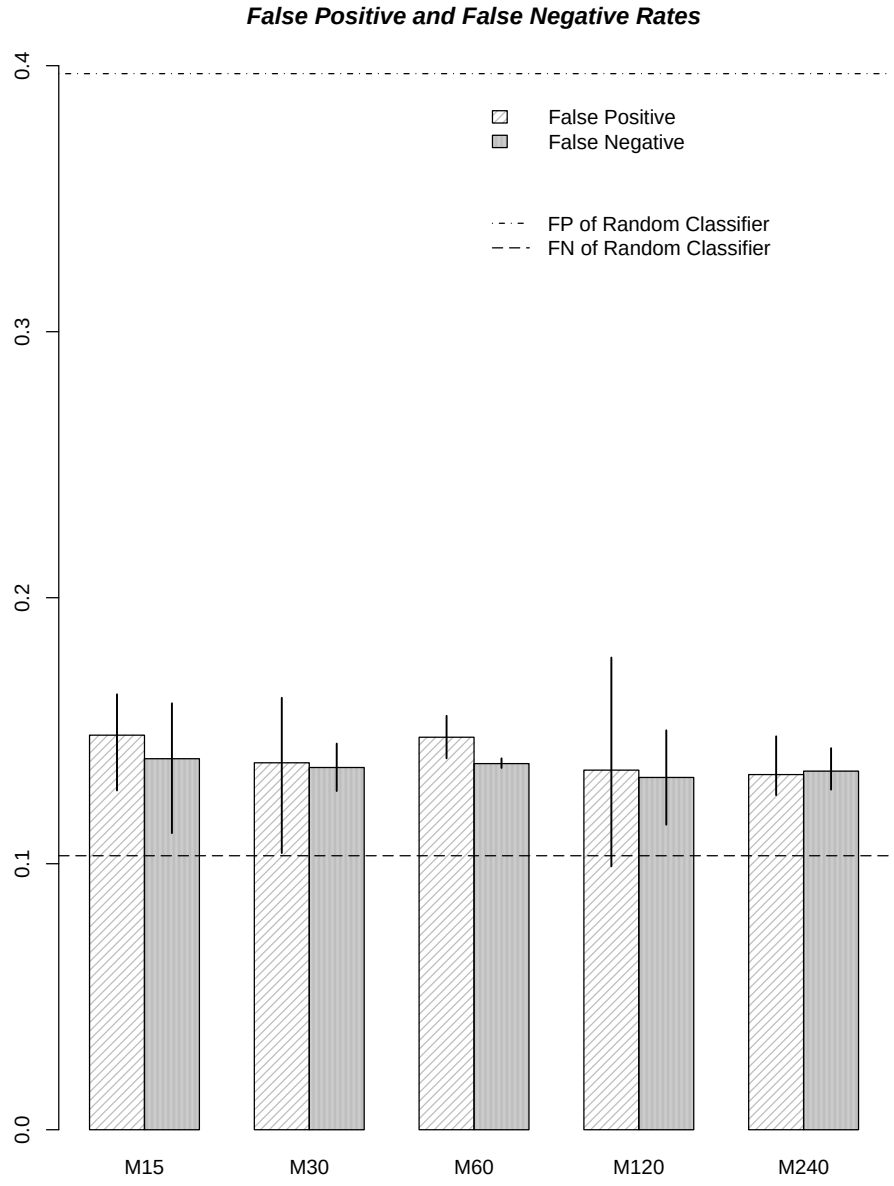
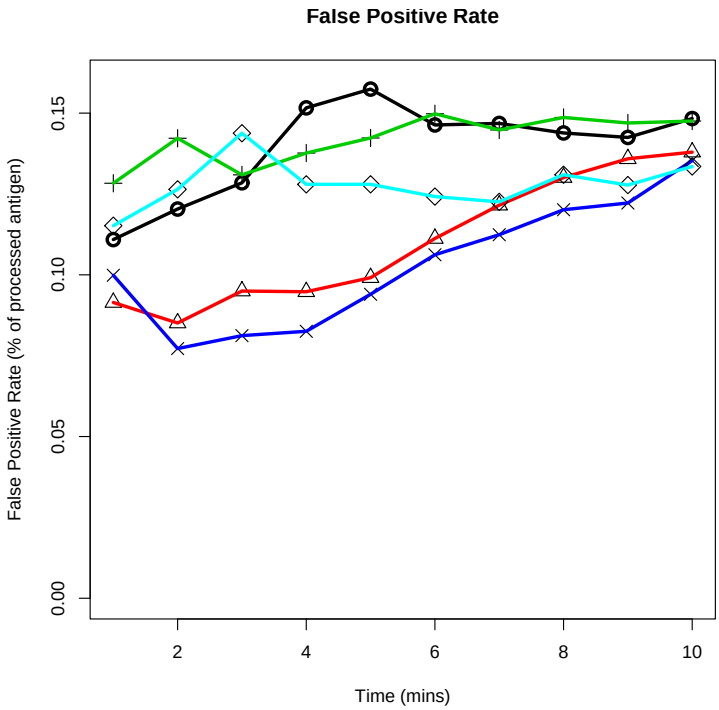
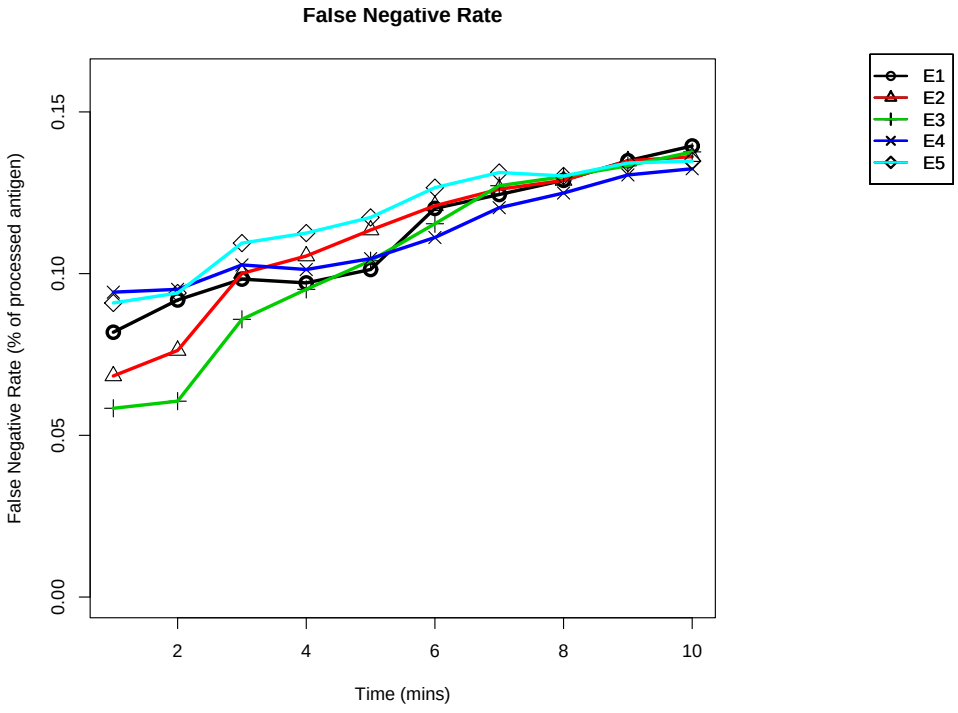


Figure 3.7: Error Rates from the Initial DCA Experiments

The dotted lines indicate the performance of a random classifier. The vertical axis is measured in terms of a fraction of the total number of classifications made. The error bars indicate the minimum and maximum error values.



(a)



(b)

Figure 3.8: The Classification Error Rates Changing Over Time

Part (a) is the false positive rate and part (b) is the false negative rate.

of false positives after ten minutes, a Wilcoxon test was able to reject the hypothesis that the two sets of data have the same medians (with a confidence of 95%), demonstrating that some statistically detectable difference has occured. Likewise, a similar Wilcoxon test on the false negatives also demonstrated a statistically detectable change, rejecting the hypothesis that the samples had the same medians. This is important as the error rate should be approximately the same throughout the experiment. It is theorised that this can be attributed to integration errors from the dead reckoning algorithm causing the robot's concept of where it is in the world to drift. This is supported by the fact that this is the only time-dependant element of the system. To minimise the effects of this external error influencing the performance of the DCA it was decided to make a change to the system used to localise the robot in the environment.

3.5 Localisation Experiment

There are many techniques available to improve the localisation of a robot, however particle filters (first suggested in [40]) are amongst the most common. Particle filters rely on generating several estimates of the pose of a robot, (in terms of the robot's translation in x and y and the angle of the robot, θ). By loading in a map of those environmental factors which are expected to remain static, (i.e. the walls of the pen) the particle which corresponds to the most likely position of the robot, (given the current ranged-sensor data and how that relates to the map) is considered to be the location of the robot. The Aria libraries that are provided with the robot have a particle filter function as standard. This was added to the system and run in parallel with the rest of the software.

In addition to the particle filter, the run time of each experiment was

reduced from 10 minutes to 2 minutes. It was identified that the robot's path was a complete circuit, so the additional passes contributed little to the end results. In fact, as this experiment is designed to explore the DCA's performance, it was advantageous to limit the effects of integrated localisation errors.

3.6 Results

The effect of the particle filter is visible from the two contrasting plots of telemetry data from the robot taken from two runs within the same pen in Figure 3.9. The black line is a plot of the telemetry without a particle filter. Integration errors from the dead reckoning algorithm are introduced by the wheels slipping instead of producing useful torque. The accumulation of these errors produces a stereotypical "spirograph" effect, with the same shape being incrementally redrawn by an increasingly erroneous rotational offset.

A Wilcoxon test on the false positives was unable to reject the hypothesis that the data after 1 minute had a different median to the data after 2 minutes (confidence level of 95%). This doesn't prove that the data has the same median, but suggests an improvement in the consistency. A similar Wilcoxon on the false negative data did reject the hypothesis that they had the same medians, suggesting that some changes in performance do occur. Performing the same statistical test for difference on the results from the experiments after 2 mins before and after the localisation changes, also rejects the hypothesis that they are the same (confidence at 95%). With the effects of localisation error reduced, it is possible to better judge the performance of the DCA. Figure 3.10 illustrates the results from the experiments with the particle filter localisation implemented. For additional clarity Table 3.3 also contains these results.

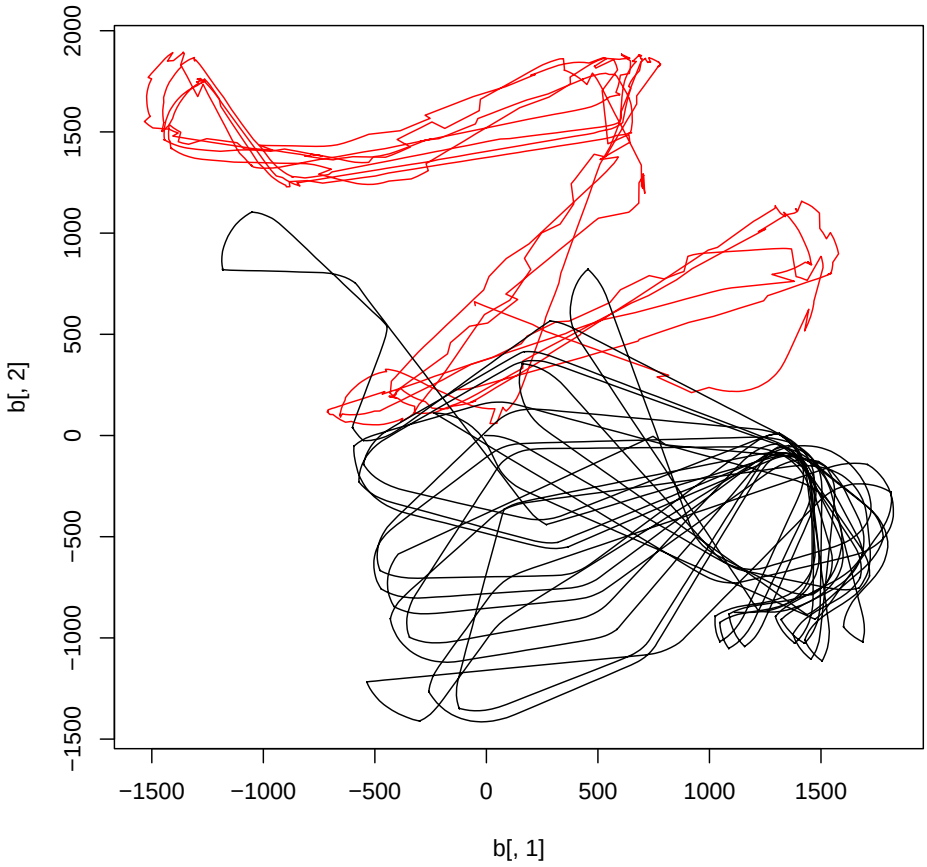


Figure 3.9: Telemetry Data With and Without the Particle Filter

The black line is the telemetry without the particle filter. The "spirograph" effect of a similar shape being repeated, slightly offset is a classic sign of integration errors in the dead-reckoning system. The red lines overlap significantly more as the estimate is repeatedly corrected.

Table 3.3: Results from the Localised Classification Experiment

	E1	E2	E3	E4	E5
False Positive Rates	0.1370	0.1288	0.0987	0.1347	0.0978
False Negative Rates	0.0596	0.0916	0.0884	0.1032	0.1735

All of the false positive rates are better than the random classifier performance. Three out of five false negative rates are better than a random classifier performance. This allows the second hypothesis of the experiment to be accepted as three parametrisations have been found which outperform the random classifier. It is an intuitive result that there would be a trade-off between a system which errs on the side of caution, (resulting in a high false negative rate) and a system which is more susceptible to triggering (resulting in a higher false positive rate). An additional phenomena is also present in the DCA. As all cells receive the same input signals, cells with smaller migration thresholds will tend to migrate after observing a smaller amount of data than cells with larger migration thresholds. This means that small migration thresholds can lead to snap judgements based on noise or sudden changes in circumstance and that large thresholds can have their classification coloured by old, sometimes out-of-date data. These effects mean that the migration threshold distribution must be selected very carefully.

These results support the concept that the DCA is a potentially useful algorithm for a robotic security system. Despite the data being unfiltered, the DCA demonstrated that, if parametrised correctly, it can perform at least better than a random classifier.

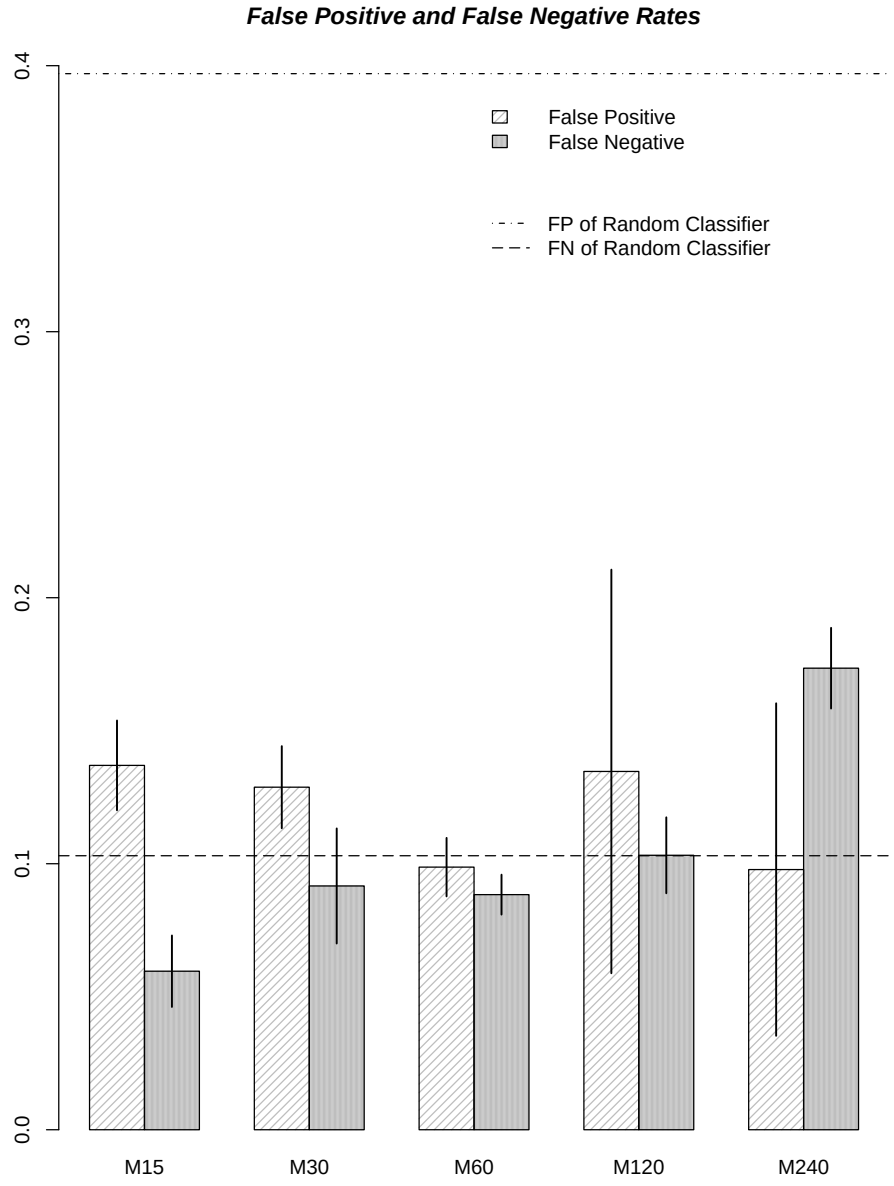


Figure 3.10: Error Rates from the Modified DCA Experiments

The dotted lines indicate the performance of a random classifier. The vertical axis is measured in terms of a fraction of the total number of classifications made. The error bars show the minimum and maximum errors.

3.7 Conclusions

This experiment has demonstrated several, key points. Firstly that the DCA, a previously batch-processing algorithm, can be modified to work with a real-time system. Both the hard and soft real-time constraints were met, meaning that the system provided a timely response for as well as providing sufficient computer time for vital functions. Secondly, the importance of using a good localisation algorithm has been clearly shown. The performance of the DCA without the particle filter enabled was worse than a random classifier in terms of its false negative rate. The cause of this was that the mapping produced by the antigen generation function was losing its correspondence to the robot's location. Localisation algorithms are typically computationally expensive and the requirement to pre-load a map limits the adaptability of the system. However, a statistically detectable improvement was produced by adding the localised data.

This suggests several areas of further work. The first area concerns the DCA's tolerance to noise. The reaction of the DCA to the unfiltered inputs seems to be tolerant to noise. The multiple time scales that the cells sample over, as a result of their varied migration thresholds, would seem to be influencing the way the DCA handles input data. As noise is a typically high-frequency signal, it is a valid hypothesis that the frequency of a signal within the input data has some effect on the weighting it receives in the decision-making process. Chapter 4 is an exploration of this hypothesis, utilising frequency-domain analysis of a single dendritic cell. Now that the performance of the DCA has been shown to be at least promising in this limited experiment, further work developing a localised, adaptive system that can use more complex heuristics should be carried out to generate a feasible security system. This is explored in Chapter 6. It is also crucial to compare the performance of this DCA with an

established, equivalent system in order to gauge exactly how well it is performing. This is done through a theoretical analysis of the algorithm in Chapter 7.

3.7.1 Summary

In this chapter a new, optimised version of the DCA was designed and implemented on a mobile robot. The new version was able to produce results as the robot patrolled around an arena demonstrating that the system was both fast enough to prevent interference with the primary functions of the robot and produce results on-line instead of in a batch fashion. The parametrisation of the DCA was explored for a mock security scenario which featured a signature based metric for the PAMP signal, a Danger signal that had a high false positive rate and an inhibitory Safe signal that prevented the system reacting to anomalies that had been identified as acceptable. It was found that with the correct parametrisation, the DCA can at least outperform a purely random classifier at this task.

3.7.2 Contributions

Novel contributions provided by this chapter are:

- The use of an antigen multiplication function, as opposed to a static antigen multiplier.
- The conversion of a batch-processing version of the DCA into one suitable for use on a real-time system. The interested reader is directed to [52] which provides a full exploration of real-time methods for the DCA, including the technique used for this experiment.
- The implementation of the abstract K variable, which makes possible future analysis of the DCA.

Chapter 4

Frequency Analysis of the Dendritic Cell Algorithm

"I don't think necessity is the mother of invention. Invention, in my opinion, arises directly from idleness, possibly also from laziness - to save oneself trouble." - Agatha Christie, An Autobiography (1977)

4.1 Introduction

The work in this chapter is based on [108].

Despite a large volume of research being carried out on the DCA, its behaviour in neither the time nor the frequency domain is well understood. Results from Chapter 3 suggested that the algorithm treats signals of differing frequency differently, but the relationship between the algorithm's input parameters and its behaviour in the frequency domain has never been characterised.

In this chapter a single dendritic cell will be modelled in the frequency domain as a digital filter. Section 4.2 demonstrates how the mathematics of the signal processing equations can be rephrased as a filter. Section 4.3 outlines an experiment to compare the results of the resulting model with the original algorithm and Section 4.4 presents the results of that experiment. Section 4.5 discusses the findings of the chapter and the limitations of the single-cell model.

4.2 Modelling the DCA as a Filter

To analyse the flow of information through the DCA, it was decided to model a single cell as a filter. The transfer function of the cell should provide insight into the information that is used to make a decision and the information that is ignored. As presented in Chapter 3, the algorithm's signal processing phase relies on two internal signals, CSM and K. The CSM signal controls the rate of cell output and the K signal is the 'useful' information being processed. For the purposes of this model $K(t)$, (K varying in time) will be considered the input being filtered by the cell. Despite recording additional information, the workings of the DCA here are identical to those presented in Appendix B.7. To model a cell as a filter, it was necessary to perform signal reconstruction from

the output of the DCA. The standard technique for assessing the output from the DC, (Dendritic Cell) population is to calculate the MCAV [47]. The MCAV is a symbol-specific (i.e. antigen-specific) calculation that identifies all of the cells that have voted for a given symbol within a fixed time-frame. An average is then calculated for that symbol. A value of one is attributed to cells with a positive vote and zero to cells with a negative vote. Thresholding the MCAV provides a final decision for the presented symbol. This technique has the advantages of being both computationally inexpensive and able to provide a measure of confidence. The further the result is from 0.5, the more confident the result.

However, this technique does not allow us to make detailed inferences about the input signal K. For this analysis, a different technique will be used to allow a reconstruction of the K signal and thus allow inferences to be made about the information passed by the algorithm. Alternatives to the MCAV are not without precedent. In [4] an alternative that takes into account the volume of antigen as well as the output signals is used to make more informed decisions about which antigen represents malicious code. The DCA version used in [4] is outlined in Appendix B.4. In this case we shall pass the time the cell spent accumulating signals (measured in sample steps) with the accumulated K signal. By dividing the latter by the former we can estimate the mean value of K that the cell was exposed to. This is given in Equation 4.1. Were the window size for a cell to approach 0, the reconstructed signal would be identical to the input signal.

$$\hat{K} = \frac{\sum_{n=0}^{W_L-1} K[n]}{W_L} \tag{4.1}$$

Where

W_L is the length of time the cell is accumulating signal.

$\hat{K}$ is the estimate of K

This technique not only allows an estimate of the K signal to be constructed but the magnitude of $\hat{K}$ also provides a measure of confidence in the decision.

For the purposes of this investigation we will simplify the model by assuming a constant CSM value of C, which allows the window length to be calculated using Equation 4.2.

$$W_L = \left\lceil \frac{M_i}{C} \right\rceil \tag{4.2}$$

Where

M_i is the migration threshold of cell i

C is the constant value of CSM

The fraction is rounded up as the cell can only migrate after an integer number of steps and will only do so if the accumulated CSM is greater than M_i.

Inspection of Equations 4.1 and 4.2 allows us to infer that the output of a single cell will be an average value of K, taken over W_L steps and reported every W_L steps.

4.2.1 Equivalence to Other Filters

The simplest technique for deriving the frequency response of a DC is to identify equivalence with established filters with known frequency responses. The description of the cell's output, given in Section 4.2, is similar to that of a moving-average filter with a length of W_L. Equation 4.3 describes the behaviour of a moving average filter in the time domain, (taken from [153]).

$$y[n] = \frac{\sum_{i=(n-(L-1))}^{n} x[i]}{L} \tag{4.3}$$

Where

n is the current time step

y is the output of the filter

x is the input to the filter

L is the length of the filter

Figure 4.1 compares the output of a moving average filter with the output from a DC. A moving average filter continuously reports the average of the previous L values for each value of n. In contrast, the output of a DC can be viewed as a series of pulses at the sample rate of the algorithm. The majority of those pulses are 0's, however every W_L pulses the magnitude is the output of the moving average filter at that point in time. This type of system can be realised using the transfer function expressed using the block diagram in Figure 4.2.

The transfer function of the moving average filter in the frequency domain is known to be given by Equation 4.4.

$$Y(\omega) = \frac{\sum_{g=0}^{L-1} e^{-jg\omega}}{L} \tag{4.4}$$

For this investigation the imaginary constant will be indicated by j as is standard for engineering applications. ω is the frequency of the signal and L is the length of the filter.

In [134] the transfer function of a downsampled, then upsampled filter in the frequency domain is given by Equation 4.5.

$$V(\omega) = \frac{\sum_{g=0}^{M-1} X(\omega + (2g\pi))}{M} \tag{4.5}$$

Where $X(\omega)$ is the original transfer function to be downsampled, M is the downsampling factor and $V(\omega)$ is the effective transfer function.

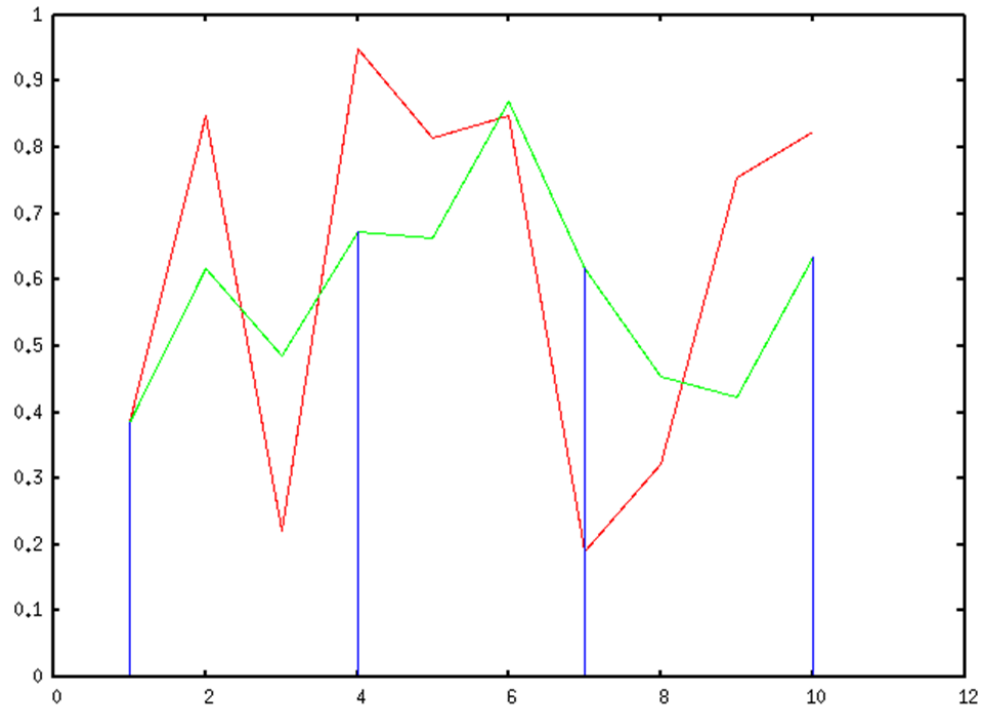


Figure 4.1: The Output from a DC for Constant C and M_i

The x-axis is in sample steps and the y-axis is arbitrary. The red line is a randomly generated input signal. The green line is a moving average of that signal with a window size of 3. The specific values of M_i and C are irrelevant, it is the ratio of the two variables which determines the window size. The blue impulses are the value of the average every 3 steps. It is of note that this output is independent of the version of the DCA used.

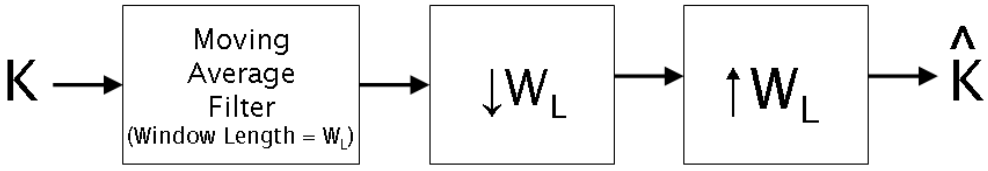


Figure 4.2: A Dendritic Cell in the Frequency Domain

The first box is a moving average filter of window length W_L, the second box is a downsampler and the third box is an upsampler.

To apply Equation 4.4 to the DCA we must substitute the window length L for the DC window length W_L. To apply Equation 4.5 to the DCA we must first substitute the downsampling factor M for the DC window length W_L. It is now possible to substitute the transfer function in Equation 4.4 for X in Equation 4.5 to find the frequency response of the DC. This result is given in Equation 4.6.

$$H(\omega) = \frac{\sum_{g=0}^{W_L-1} \sum_{b=0}^{W_L-1} e^{-jb((\omega+(2g\pi)))}}{W_L^2} \tag{4.6}$$

This is an important proof as it is common in engineering applications to precede a downsampling block with a filter [61]. Such a filter-downsampling pair is called a "decimator". This structure is used when a device with a high data output rate is connected to a device with a lower input rate. The downsampling reduces the rate to match the input rate of the lower-frequency device, but the filter ensures that those samples which do get through are indicative of the set of samples for that time frame. A filter precedes the downsampler because if the original input signal contains any frequencies that satisfy the inequality in Equation 4.7, the downsampling process will introduce harmful aliasing artefacts into the output.

$$f > \frac{f_s}{2M} \tag{4.7}$$

Where

f_s is the sampling frequency

M is the downsampling factor

In engineering terms, the 'cut-off frequency' or 'corner frequency', f_c of a filter is an important property. The gain for all frequencies greater than f_c is considered to be negligible enough for these frequencies to be ignored. Any frequency below f_c is said to be 'passed'. This frequency

is generally considered to be the point where the gain has dropped to $1/\sqrt{2}$. The cut-off frequency for a moving average filter is defined by Equation 4.8 [153].

$$f_c = \frac{0.443 f_s}{L} \tag{4.8}$$

In the case of a DC, the downsampling factor and the filter window length are both equal to W_L. With the inequality in equation 4.7 and the expression for f_c in equation 4.8 in terms of the ratio of f_s to W_L, it is trivial to demonstrate that the highest frequency that the DC filter will pass is $0.443 f_s/W_L$, well beneath the 'dangerous frequency' of $0.5 f_s/W_L$. This result means that the DC will never make decisions based on data erroneously introduced by aliasing effects.

4.3 Verification of the Model

In order to verify this model we can compare the predicted frequency response with the actual frequency response of a single DC with a constant CSM input and migration threshold. This was generated by making K equal to various sine waves at different frequencies and recording the magnitude of the resultant sine waves from both the model and the dendritic cell. To have confidence in the model this must be performed for a variety of CSM values and migration thresholds.

All of the experiments are performed using frequencies between 0 and the Nyquist frequency of the system. The Nyquist frequency, f_n, of a system is half the system's sampling rate. This is a valid test as the frequency response of a system from 0 to f_n is exactly the same as the frequency response for the system from any $X f_n$ to $(X+1) f_n$ where X is an even number. For frequency responses in regions $Y f_n$ to $(Y+1) f_n$ where Y is an odd number, the response is the mirror of the response

Table 4.1: The Parameter Values Used for Experiments 1-9

For experiments 1-9 f_s is held at 1Hz	
Migration Thresholds (M_i)	CSM Signal Values (C)
30	10
60	20
120	30

Table 4.2: The Parameter Values Used for Experiments 10-21

For experiments 10-21 C is held at 10	
Migration Thresholds (M_i)	Sampling Frequencies(f_s)
30	0.5
60	1
120	2
	4
	10

from 0 to f_n. Thus, establishing that the model is accurate from 0 to the Nyquist frequency establishes that the model is accurate for any frequency.

Two sets of experiments were run. Firstly, nine runs of the experiment were performed keeping the sampling rate at 1Hz, testing every migration threshold against every CSM signal value listed in Table 4.1.

Secondly a further fifteen experiments were performed, keeping the CSM signal at 10 and using every value of the migration threshold with every sampling frequency in Table 4.2.

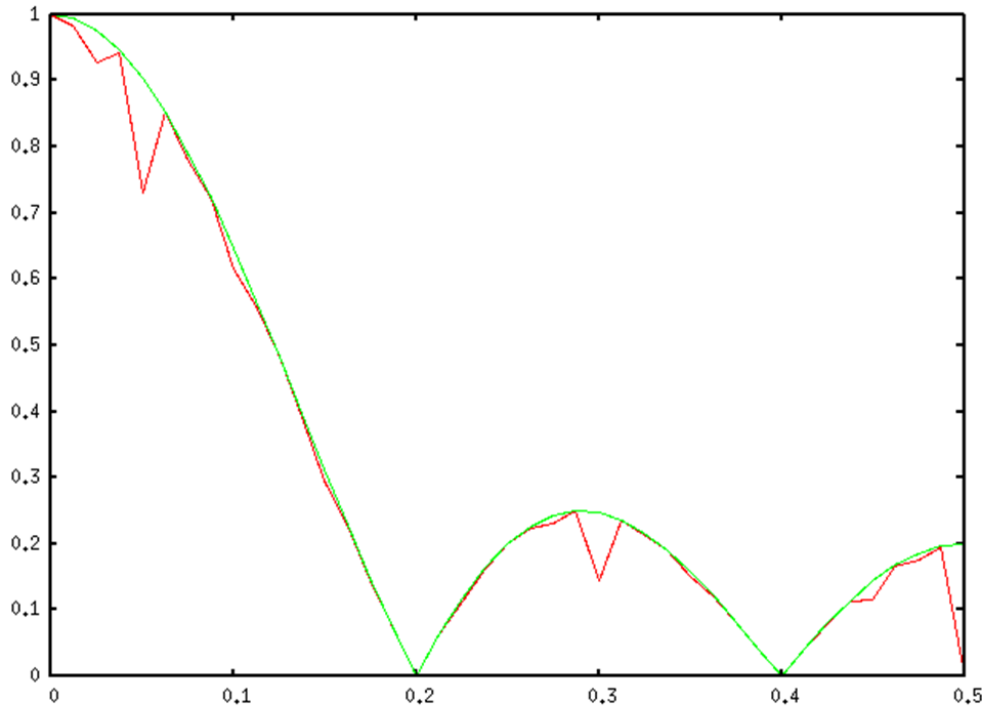


Figure 4.3: The Output from the Worst Performing Model Prediction

The green line represents the model's prediction and the red line represents the actual frequency response of the algorithm for M_i = 120 and CSM = 30

4.4 Results

In all cases the response followed a similar overall shape. The gain has an initial drop from 1 to 0 and in the higher frequencies, oscillations in the gain value can be observed. These oscillations are known as "ripple". The total, absolute error over the entire frequency range for the model can be defined as the Euclidean distance between its predictions and the actual response from 0 to the Nyquist frequency. In the worst case, (M_i = 120, C = 30) the total, absolute error was approximately 0.69 and is shown in Figure 4.3.

It is clear from the plot that the error arises from circumstances where the actual DC gain transiently drops. This is not a serious concern for

this investigation as we are interested in the frequency ranges that are passed / rejected by the DCA. It is the general shape of the response that is important. These transient drops are the result of the sample rate causing the peaks of the input sine waves to be consistently missed. As a result the algorithm 'sees' a different input wave to the intended one.

Figure 4.4 shows the effects of changing the CSM value for a constant value of M_i (60). As the CSM value is increased the initial drop-off slope becomes slower. The ripple in the cut-band becomes larger with a lower CSM. Both of these aspects are generally considered to be negative features in a filter. A gradual cut-off slope means that more of the unwanted frequencies are close enough to the cut-off frequency to have an effect on the output signal. A high magnitude of ripple means that some higher frequencies have a disproportionately high effect on the output signal. This reinforces the idea that low values of migration threshold relative the CSM signal can lead to too much information being passed into the output. High values of the migration threshold relative to CSM cause a fast drop-off in the cut band, but this means that if the useful information is in the cut band, its information will be lost. In all of these cases the model predicts the performance of the algorithm accurately.

Predictably, increasing the migration threshold had the inverse effect to increasing the CSM value. This supports previously seen experimental behaviour of the algorithm. Again the model's predictions were accurate across the entire range. These results can be seen in Figure 4.5.

For the next set of experiments the sampling frequency of the algorithm was altered. For all experiments the effect was to simply scale the same shaped frequency response between 0 and the new Nyquist frequency. This can be seen in Figure 4.6. In all cases the error rate was identical for all migration threshold / CSM pairs for all sampling

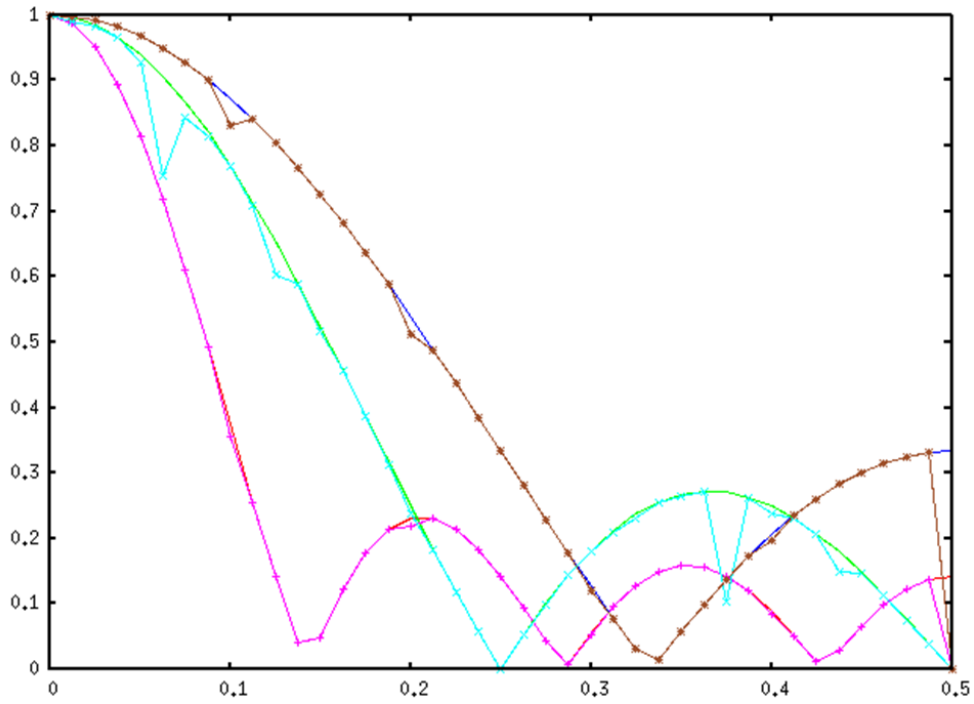


Figure 4.4: Gain vs Frequency(Hz) for Varying CSM Values

For these experiments the value of the migration threshold is set to 60 and the CSM value is 10, (red) 20, (green) and 30, (blue). In each case the purely solid line is the model prediction and the solid line with markers is the actual response.

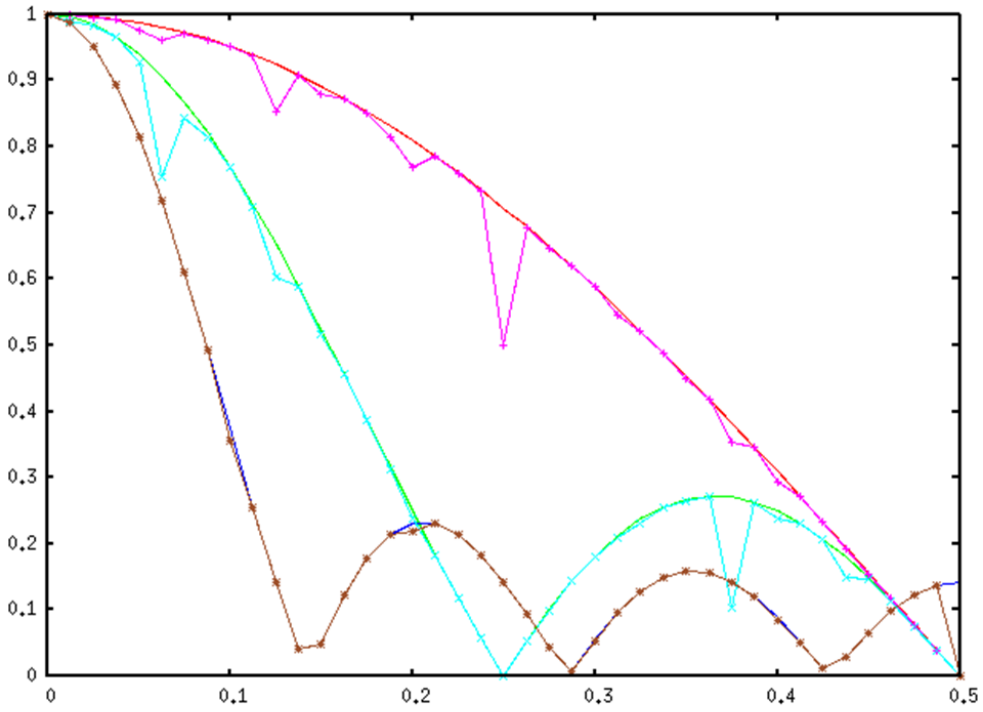


Figure 4.5: Gain vs Frequency(Hz) for Varying Migration Thresholds

For these experiments the value of the CSM is held at 20 and the migration threshold is 30 (red), 60 (green) and 120 (blue). In each case the purely solid line is the model prediction and the solid line with markers is the actual response.

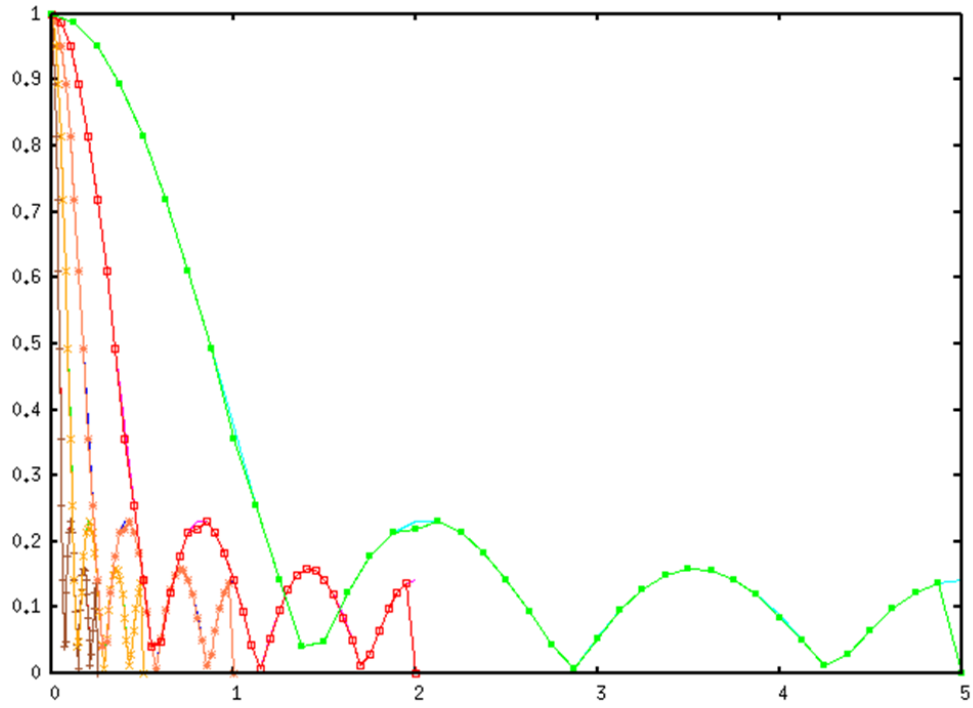


Figure 4.6: Gain vs Frequency for Varying Sampling Frequencies

The CSM was held at 10, the migration threshold was held at 60 and the sampling frequency was changed from 0.5 (red), 1 (green), 2 (blue), 4 (pink) and 5 (turquoise).

frequencies. It is an intuitive result that given a stationary CSM, an increasing sampling rate will proportionally increase the rate at which the cells migrate

4.5 Analysis and Discussion

For all parameters the predictions from the model were very accurate. The significant effect of changing the ratio between the migration threshold and the CSM signal highlights the importance of correctly parametrising the algorithm. This further emphasises the need for a more informed way of selecting the migration threshold. Figures 4.4, 4.5 and 4.6 clearly

demonstrate that correct selection of the migration threshold will significantly alter which information is passed and which information is cut. As the DCA always acts as a low pass filter any application which relies on high frequencies to make a decision will require bass cut filters to be incorporated within the signal generation heuristics.

The scaling effects of varying the sampling frequency had no impact on the accuracy of the model. The scaling effect is an important result as it demonstrates that selecting the correct sampling frequency for a given application is crucial. When applying the DCA to real-time systems this means that the algorithm is potentially vulnerable to drift in the sampling rate. However, it is likely that this is rectified by the population-based nature of the algorithm, so long as the spectrum of migration thresholds is picked appropriately.

The construction of an accurate frequency domain model of a DC has ramifications for the usage of the algorithm in the future. With representative samples of the CSM and K signals for a given application it may now be possible to tune the probability density function for the migration thresholds of the population to provide an optimum response for the application, a concept which is explored in Chapter 5. By selecting a migration threshold that rejects data that could be potentially misleading to the decision making process it is theorised that the error rate of the algorithm could be significantly reduced. This of course assumes that the irrelevant information has a higher frequency than the useful information. This assumption is true for many complex problems as sources of noise are generally transient perturbations, thus occupying higher frequencies [61]. The experiment in Chapter 3 uses unfiltered input data so much of the misleading information comes from high-frequency sensor noise.

4.5.1 Limitations of the Frequency Model

The assumptions made to derive the model limit how useful the results are for predicting the response of the DCA. Here we discuss the key assumptions and the effects that these assumptions have on the model.

Constant CSM

The model assumes that the CSM signal is kept constant over the lifetime of the cell. This is unlikely, as the K signal and the CSM signal are both weighted sums of the same three input signals, so whilst it is possible for one to move independently of the other, it is highly unlikely. However, it is doubtful that this is a factor in the model's accuracy. The CSM signal is accumulated by a cell over its lifetime. This means that any constant model of CSM is equivalent to any selection of the CSM signal with the same cumulated total over the lifetime of the cell. The implication of this is that the model allows the user to inspect the cell's behaviour for a small range of CSM values. Thus predictions based on this model only hold for transient periods of the algorithm's use.

Antigen Correlation

The model makes no attempt to take into account the antigen correlation of the algorithm, so it can make no predictions about how this element of the algorithm is effected by the input parameters. For applications where the correlation between antigen presentation and signal presentation is trivial, this is unimportant. For example, if there is no delay between the antigen being presented and its effects being felt in the input signals, the results of the model would be adequate for representing the application's needs. However, when there is a delay between antigen presentation and the resulting signal presentation, or where the relationship between antigen presentation and signal presentation is combinatorial, (i.e. no one

antigen is responsible for a positive decision, but certain combinations of antigen can cause this to happen) the model will be less useful. It is of note that there are no applications of the DCA in the literature where combinatorial effects have been investigated. The model could be used in the future to investigate the effects of M_i on cases where there is a time delay between antigen presentation and signal presentation, as the phase of the $\hat{K}$ signal will provide information about the lag introduced by the algorithm and thus, the largest possible time between sampling an antigen and a cell migrating.

Single-Cell Modelling

The model only considers a single cell operating in isolation from the rest of the population. This considered to be the most significant drawback to the practical use of this model for migration threshold tuning. The DCA relies on the use of a population of cells to ensure that samples are processed regularly and to gather a wide range of data from multiple frequencies. By ignoring the interaction between a population of cells it is likely that the model is an oversimplification. For this reason it was decided to extend the model to incorporate multiple cells.

4.5.2 Conclusions

The DCA has been optimised in this chapter to improve the speed of execution for future implementations of the algorithm. This optimised model has been evaluated mathematically to derive the transfer function for a single DC under the condition that the CSM value is held constant. The accuracy of this model was verified experimentally and found to be an excellent indicator for the performance of a DC with a constant CSM input. Insight has been gained into the relationship between the CSM signal, the migration threshold and the sampling frequency of the DCA.

A low migration threshold has been shown to indicate a more gradual reduction in gain as frequency is increased. This increase in the volume of information being used in the decision making process is intuitive. In contrast, a high migration threshold means that higher frequencies are cut from the decision making process.

The derivation process demonstrated that a useful measure of the output of a DC is $\hat{K}$. This estimate of the overall signal that the cell was exposed to during its sampling phase provides an indication of how certain the cell is of the final decision. Cells with low magnitudes of $\hat{K}$ have been exposed to low signal values or approximately equal amounts of positive and negative signal. This could potentially be incorporated into future versions of the DCA to provide an alternative to the MCAV that is more resilient when cells are exposed to multiple, conflicting events during their sampling phase. A negative feature of this technique is that it requires some measure of 'age' in order to calculate the estimate, a measurement which is not found in the biological model.

4.5.3 Summary

In this chapter a model was produced of a single dendritic cell's frequency response. The model's validity was demonstrated empirically. By examining the DCA from the frequency domain it was possible to use well-established techniques from the field of signal processing to explore properties of the algorithm. These include its inherent ability to remove potentially misleading aliasing frequencies and its equivalence to a series of known frequency domain structures. It was observed that these structures are commonly used in the field of signal processing to lower the bandwidth from one unit to another, while maintaining crucial information.

4.5.4 Contributions

Novel contributions provided by this chapter are:

- A mathematical model of a dendritic cell's behaviour in the frequency domain which has provided an insight into how the algorithm chooses which information to use and which information to ignore as part of the decision making process.
- Verification of that model's correspondence to the real algorithm's performance.

Chapter 5

Population Tuning and Multi Cell Modelling

"Viewed as a geometric figure, the ant's path is irregular, complex, hard to describe. But its complexity is really a complexity in the surface of the beach, not a complexity in the ant." - Herbet A. Simon, The Sciences of the Artificial (1996)

5.1 Introduction

The work presented in this chapter is based on experiments published in [108] and [109].

Sensitivity analysis carried out on the DCA has shown that appropriate selection of the migration thresholds is crucial to the performance of the algorithm [42]. This result is supported by the experiments in Chapter 3 and the theoretical analysis in Chapter 4. These findings are likely to be caused by the windowing phenomena that occurs as a cell samples signal from the environment. This phenomena means that if a migration threshold is too low, a cell will migrate too quickly and will not be able to gather a representative sample of the input signals. If a migration threshold is too high, the cell will migrate too slowly and will mis-classify the gathered antigen. A balance between these two extremes is found in part by the population-based nature of the algorithm. However, this is still dependant on a suitable selection of migration thresholds within the population.

Currently no techniques exist for tuning the DCA's migration thresholds. Generally applications are tuned experimentally using trial and error. This can be time-consuming and does not guarantee to find an optimal set. In Chapter 4 a model was presented that provided a clearer insight into the strategies used by the algorithm to select which information is used as part of the decision-making process and which information is ignored. It is proposed that such a model should be able to provide a tuning methodology for the DCA's input parameters, based on removing those frequencies that are deemed to contain misleading, noisy data. This work is based on the DCA version outlined in Appendix B.7.

This chapter is organized as follows: in Section 5.2 a tuning methodology is designed, based on the mathematical model presented in Chapter 4; in Section 5.3 an experiment is performed to test the performance of the

tuning methodology; the results of the experiments are discussed in Section 5.4; Section 5.5 discusses the results of the experiments and outlines the limitations of frequency analysis for generating a tuning methodology.

5.2 Tuning the DCA using the Frequency Domain Model

Chapter 4 provides an equation which relates frequency of input signal to gain (4.6), considering a dendritic cell as a filter. This relationship relies on the CSM and K signals that are presented as inputs to the cell and the cell's migration threshold property. CSM and K will be dependent on the application-specific heuristics, but the migration threshold is a user-defined parameter. The premise that this can be used as the basis for an effective tuning strategy relies on the underlying assumption that the data to be removed has a higher frequency than the data to be observed. For the majority of real-world applications, this assumption holds.

It is necessary to overcome two, crucial problems before a tuning methodology can be created. Firstly, as the model only accepts a constant level of CSM as an input, a value must be identified which captures the CSM value for the problem being solved. Secondly, once that value has been identified, the 'corner frequency' or 'cut-off frequency' of the filter must be calculated. In standard filter design, this is the frequency at which all data above that frequency is considered to be 'cut' or removed and all data below that frequency is considered to be 'passed' or kept. This is commonly held to be the frequency at which the gain of the system drops below $1/\sqrt{2}$ [153].

Identifying these values for a given application will obviously involve analysing a sample of test data from the application-environment. An expert should be able to identify 'events of interest' within this sample.

Here, an 'event of interest' is defined to be a situation where the CSM and K signals have increases in magnitude which highlight a situation that is of use for the application area. The peak of interest with the smallest magnitude in CSM will provide information about the minimum sensitivity required by the system. Using the division of the peak's height by the peak's width as the approximate value of CSM will ensure that in systems where the smallest magnitude peak is a narrow pulse, that the CSM used in the model will be relatively large compared to that of a peak with a similar magnitude but a wide pulse. A large value of CSM, for a given cut-off frequency, implies that the resulting migration threshold will be set to a smaller, more sensitive value, resulting in a cell better equipped to deal with higher-frequencies. The cut-off frequency will be classed as the median frequency of the events of interest for the application being tuned for. Using the median value will remove the effects of outliers.

Below the final tuning methodology is defined.

Step One Take an indicative sample of the typical values of K and CSM that the system will encounter for the application.

This sample acts as a training example for the tuning methodology. As with any training sample, the underlying assumption that it is indicative will cause the algorithm performance to suffer if the data is not indicative of the application.

Step Two Identify the median period between events of interest. The application-specific heuristics should highlight these events with peaks in the CSM and K values.

Identifying the period at which the events of interest are occurring in the data is motivated by the need to identify a corner frequency for the filter. As filters have a smooth response, using this period to calculate

the corner frequency will not in practice remove all frequencies directly above it, allowing some of the higher frequency events of interest to pass through, albeit slightly attenuated.

Step Three Identify the length of the event of interest with the lowest CSM value associated with that event. Events of interest will generate a spike in the CSM value of the system. The smallest spike is indicative of the most difficult event to extract from the input data. Record the median value of the spike and its duration.

The model used to calculate the parameters requires a CSM value to operate. By using the lowest CSM value associated with an event of interest, the aim is to ensure that the resultant system can detect even the smallest event required.

Step Four Calculate the target corner frequency. The target corner frequency is the inverse of the period calculated in step two. As discussed in step 2 the filtering process has a very gradual cut off, the fact that this is an approximation should not prevent any useful information from passing through to the decision making process.

Step Five Calculate the constant CSM value for the model. The median value of the spike identified in step three, divided by the duration of that spike provides a good approximation of the value of C. C is the constant CSM value to use in the model. By dividing the median value by the length of the spike we ensure that the quantity of CSM is sufficient to generate a single window for a DC to monitor. This is the smallest artefact within the signal that is examined.

Step Six The transfer function is used to search the space of available frequency responses to find one with a corner frequency which is

approximately equal to the target corner frequency.

This search is a simple exhaustive search going in integer steps from 1 to 500.

5.3 Revisiting the Robotic Classification Problem

In order to test this methodology the experiment outlined in Chapter 3 was repeated using the migration threshold values specified by the methodology.

5.3.1 Tuning The Algorithm

A sample of the CSM and K values for a typical run of the experiment were taken. The sample rate used was 4Hz. Figure 5.1 shows the first 400 samples covering a period of 10 seconds, from the robot.

The estimated peaks from the sample in Figure 5.1 are given in Table 5.1. The median distance of 30.5 samples gives a target corner frequency of approximately 0.131Hz.

The event of interest with the lowest CSM signal was identified as the small peak in CSM between samples 297 and 302 where the robot moved so the anomalous cylinder briefly came into view of the camera. This is evident in the data as the K signal associated with this period is positive. Here, the term 'interesting' refers to any event that the robot should report back to the user. The median CSM value between these points is approximately 22.9. This equates to a constant CSM value of approximately 4.5.

After searching the parametrised space of frequency responses a migration threshold of 54 was identified as an optimal result. This result was compared to the results from Chapter 3. In addition, the experi-

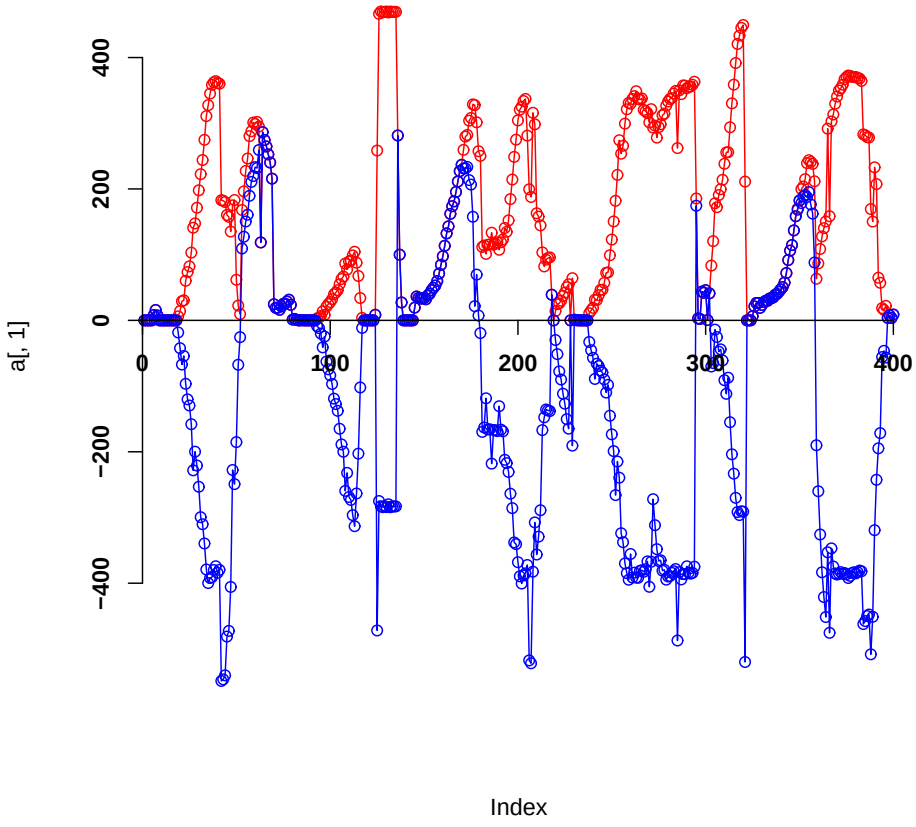


Figure 5.1: A Plot of the Magnitude of K and CSM Against Time

The time is measured in sample steps. A 10 second sample from the robot application. The pink line is the K signal and the blue line is the CSM value.

Table 5.1: The Sample Numbers of the Peaks Within Figure 5.1

Sample Number	Distance From Last Peak
42	N/A
62	20
114	52
133	19
178	45
205	27
230	25
264	34
299	35
321	22
356	35
380	24

Table 5.2: Results From Using a Distribution of ±50%.
"FN" is used to denote "False Negative" and "FP" is used to denote "False Positive".

M_i	Average FP %	FP Deviation	Average FN %	FN Deviation
15	13.697	0.132	05.959	0.070
30	12.877	0.098	09.164	0.103
54	04.769	0.089	10.089	0.117
60	09.872	0.159	08.838	0.093
120	13.469	0.157	10.317	0.150
240	09.781	0.170	17.351	0.175

ments were repeated, but using a distribution width of ±10% (as opposed to the standard ±50%). By reducing the distribution width, the overlap between the populations is reduced and the effects of the tuning methodology can be more clearly seen.

5.4 Results of the Tuning Experiments

Table 5.2 shows the results from the experiments on the robot using a distribution of ±50%. The error rates presented are the average percentage of incorrect readings over 10 blocks of twelve seconds. For illustrative purposes, the data is plotted in Figure 5.2.

Examining the results presented in Table 5.2 and Figure 5.2, the tuning algorithm appears to have performed comparably with the other parameters in terms of the false positive rate. 4.769% is less than half the false positive rate of the best result found using trial and error, a migration threshold of 60. It also has one of the smallest standard devia-

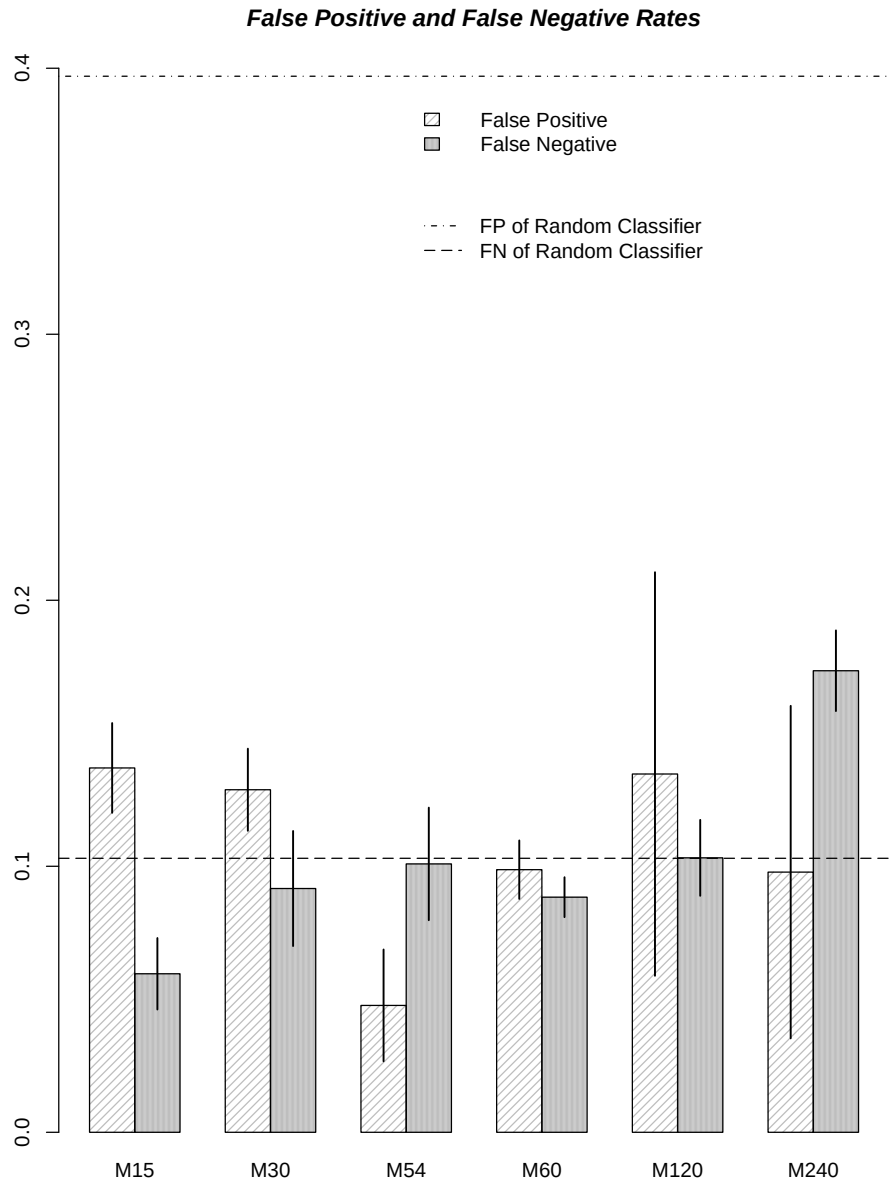


Figure 5.2: The Results from the Tuning Experiment

Using a migration distribution of $\pm 50\%$. The dashed lines indicate the performance of a random classifier.

Table 5.3: Results From Using a Distribution of ±10%
"FN" is used to denote "False Negative" and "FP" is used to denote "False Positive".

M_i	Average FP %	FP Deviation	Average FN %	FN Deviation
15	07.054	0.129	07.838	0.091
30	09.113	0.101	07.406	0.093
54	15.007	0.193	07.706	0.091
60	05.000	0.127	06.830	0.105
120	11.697	0.103	12.305	0.140
240	02.500	0.079	09.687	0.116

tions of the set, suggesting that the entire range of migration thresholds around the tuned value appear to work well for the problem. However, this is not the case when exploring the more taxing problem of the false negative rate. The tuned result barely outperforms a random classifier and is the fourth worst performing parametrisation of the algorithm. The most likely cause of this is that that the techniques for calculating the constant value of CSM and the target corner frequency are flawed and lead to an incorrect value of the migration threshold to being generated. Another possible cause is the broad spectrum of migration thresholds. By using a distribution with a width of $\pm(0.5M_i)$ the effect of tuning may be lost. To explore this idea the experiments were repeated using a narrower band of migration thresholds.

The results presented in Figure 5.3 and Table 5.3 are from the second set of experiments, using the narrower migration threshold band. Applying the Wilcoxon test to the two sets of data with a migration median of 54, (p-values of 0.172 for the false positive rates and 0.843

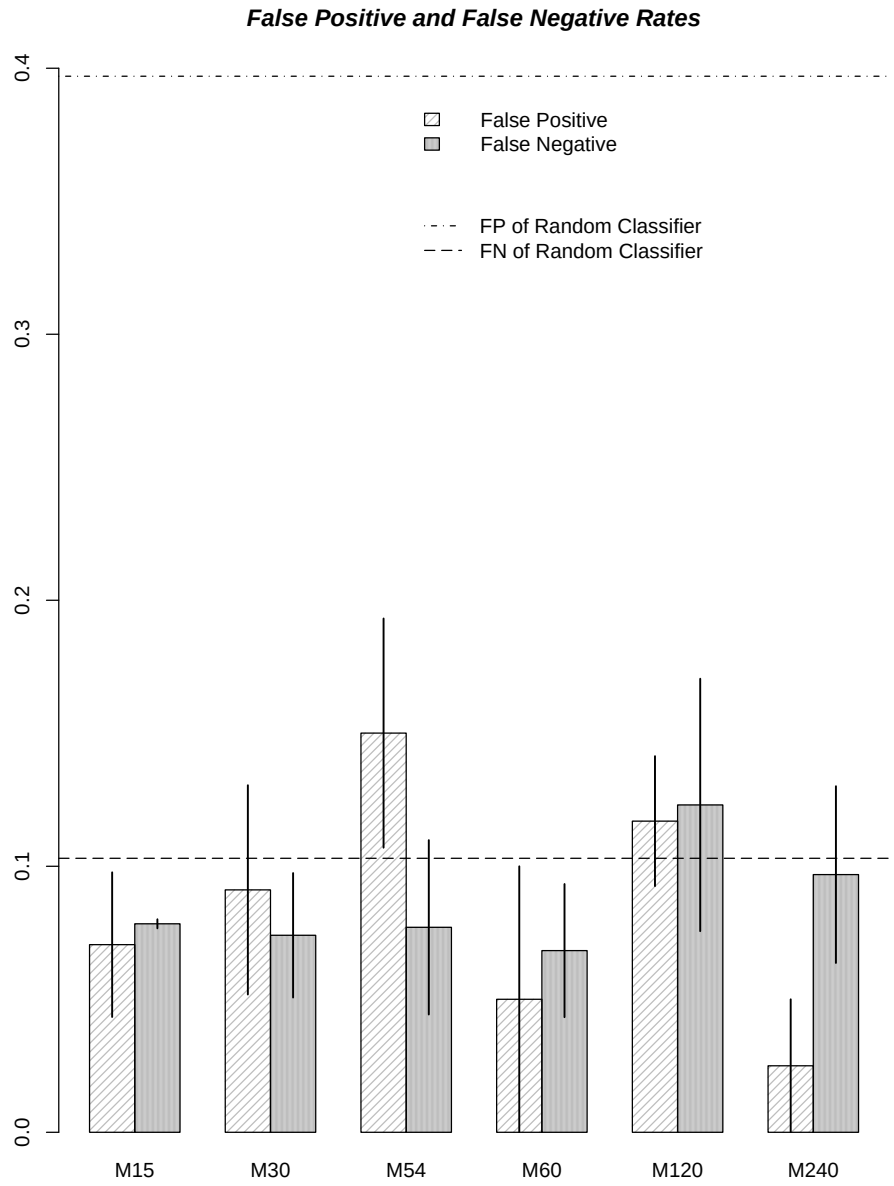


Figure 5.3: The Results from the Tuning Experiment

Using a migration distribution of $\pm 10\%$. The dashed lines indicate the performance of a random classifier.

for the false negative rates) failed to confirm that the two sets of data had significantly different medians (using a 95% confidence threshold). This is unsurprising as the two data sets both have the same central point for their migration threshold distribution. Comparing the mean false positive rates demonstrates that the narrower migration threshold distribution worsens the mean error. However, the standard deviations between the two data sets indicate that narrowing the distribution width actually increases the range of the false positive rates. These factors indicate that narrowing the migration threshold distribution causes the false positive rate to jump to much worse values for short periods of time, but for the majority of cases, remains approximately the same. This suggests that the tuning methodology has failed to capture the CSM value appropriately for all cases and that a wider distribution has simply masked this short-coming by including cells that are appropriate for the cases where the tuned threshold fails. This is supported by the fact that the false positive rates for a migration median of 60 are much better in the narrow band case, indicating that slightly higher migration thresholds would 'wash-out' the poor performance of the other cells. Comparing the false negative rates between the wider and narrower migration bands for the tuned results, indicates that the band has had a limited effect.

5.5 Discussion

The transient poor performance of the system when using the tuned value indicates that the identification of a stationary equivalent to the varying CSM signal is not a trivial task. In addition, if variances in the CSM signal are causing one value of the migration threshold to be good in some cases, but bad in another, it is much more important for a tuning methodology to not simply identify a median point about which

the cells' migration thresholds are to be distributed, but to identify a good distribution shape for best dealing with the sampled CSM data. As a result, a much better understanding of how an entire population of cells reacts in the frequency domain is required.

5.5.1 Multi-Cell Modelling

In order to model multiple cells in the frequency domain, it is necessary to specify how they will interact in the time domain. To produce a population-wide $\hat{K}$ we must find a reliable way of combining the data from a population of cells. For the purposes of this investigation it was decided to simply periodically sample the cell population and check for migrated cells. The $\hat{K}$ output from each migrated cell would be averaged together to produce an population-wide estimate of K for that window. By averaging together the output from multiple cells, the process of generating a multi-cell model is made much easier. In the frequency domain, the averaged output from multiple filters can be modelled as simply the sum of the gains. The averaging process has no effect on the shape of the response, but scales it to be in the range 0-1. To explore the effects of this multi-cell model a 2 cell system was created using one cell with a migration threshold of 90 and one cell with a migration threshold of 110. The CSM signal was held at 20 and the sampling rate was held at 1Hz. The outputs of the cells were checked every algorithm cycle. All of the experiments were carried out using the Octave environment.

Figure 5.4 shows the frequency response of the actual system and the predicted output from equation 4.6 in Chapter 4. The two lines clearly diverge significantly more than the other models. The source of the difference is a combination of the asynchronous nature of the dendritic cell algorithm and the way in which the actual system gain is calculated. To calculate the gain of the actual system, the peak value of the output is

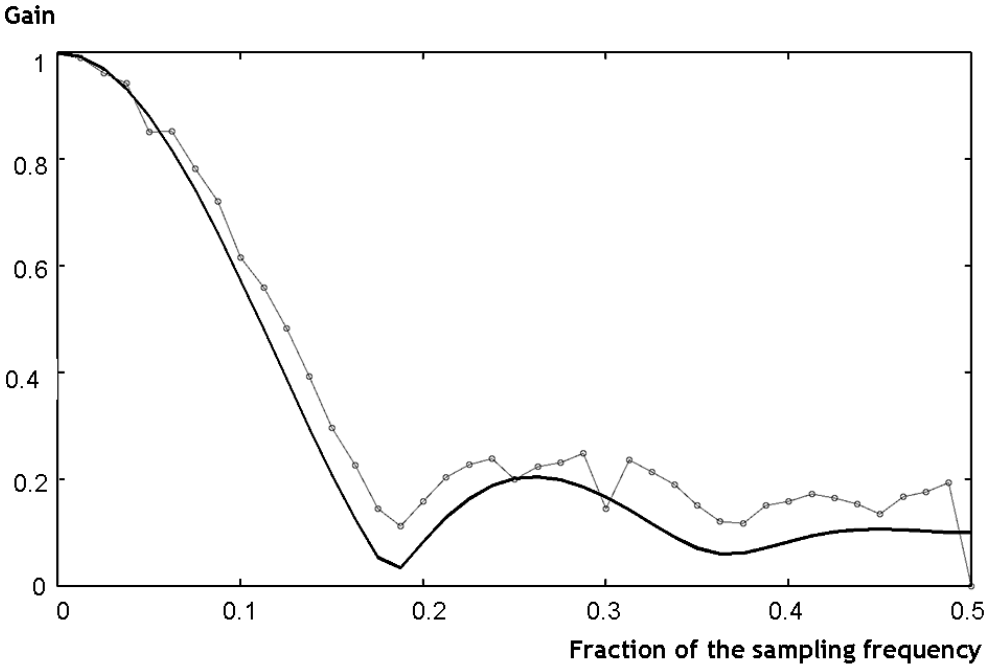


Figure 5.4: The Frequency Response of the Two Cell System

The thick, dark line is the predicted response and the light, dashed line is the actual response.

recorded by the simulator. As the cells have different migration thresholds there will be occasions when one cell reports and the other does not. On other, rarer occasions, both cells will synchronise and report at the same time. As the maximum peak is recorded as a measure of gain, the cell with the larger gain for that frequency will dominate the results from the simulator. This can be verified by comparing the measured response from the algorithm with the maximum of the two single-cell model predictions. In Figure 5.5 the output from the actual system clearly follows the maximum path of the two model predictions. Figure 5.6 shows an example of the asynchronous system outputting three different sized gains for a single input frequency.

The construction of a model capable of predicting the response of a population of DCs is a non-trivial task. The asynchronous nature of the population means that the differing phases of the cells will have a

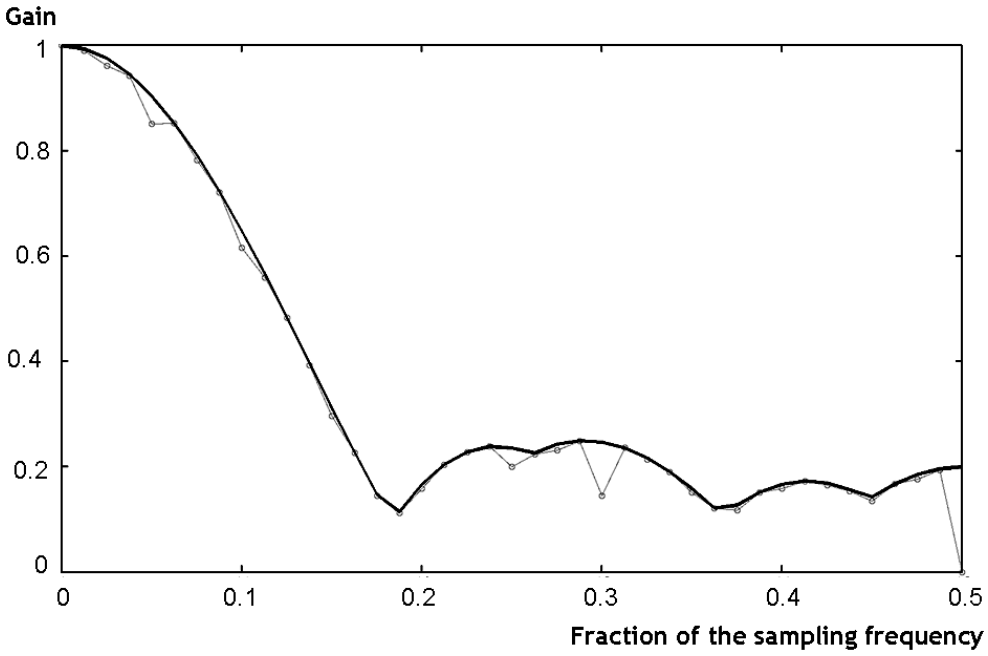


Figure 5.5: The Revised Frequency Response of the Two Cell System

The thick, dark line was generated by using the largest gain out of the two, single cell predictions for each frequency. The light, dashed line is the actual response.

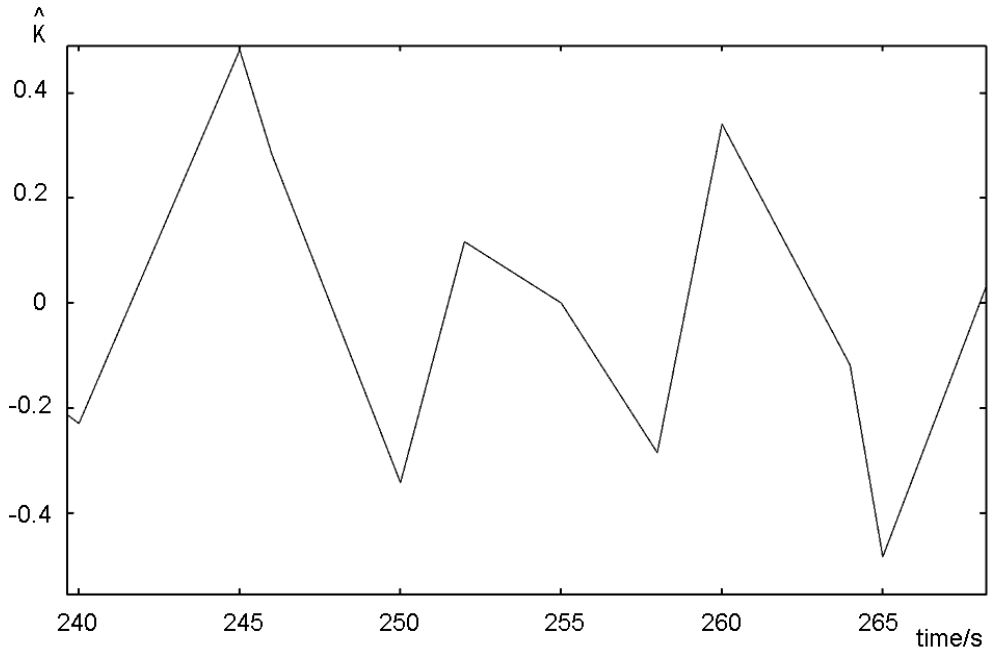


Figure 5.6: An Example of the Output for a Two-Cell DCA

The sample frequency is $1Hz$ **and the input frequency is a sine wave at** $0.125Hz$ **with a magnitude of** 1**. The first peak is the gain of the cell with a migration threshold of** 90 **(approximately** 0.48**), the second peak is the gain of the cell with a migration threshold of** 110 **(approximately** 0.31**) and the third peak is the average gain of each cell (approximately** 0.40**)**

significant effect on the output of the system. Effectively the relationship between gain and input frequency has ceased to be expressible using conventional means, as the gain for a given frequency is a range of values, depending on the relative phases of the cell population. For a two cell system there are four possible gains for each frequency, the gain of cell 1, the gain of cell 2, the average gain of cell 1 and cell 2 and a gain of zero, when neither cell migrates. It is possible to derive that the number of possible gains for a single input frequency, for a population of cells is given by:

$$N_g = 2^P \tag{5.1}$$

Where N_g is the number of possible gains and P is the number of cells in the population. This a worst-case that assumes that it is possible for all cells to simultaneously drift in and out of phase with one another. For a standard 100 cell implementation of the DCA this evaluates to approximately 1.27×10^{30}. Whilst it is possible to calculate the average response, it is questionable if this will be sufficient to provide enough information to effectively tune the system. It is possible that the cells drifting in and out of phase with one another adds another level of filtering to the system. A transient spike will be picked up by some, but not all of the cells migrating at a given interval, thus the average output over the population will potentially remove some of the noise from the inputs.

These results cast doubt on the usefulness of traditional frequency-based techniques for modelling the DCA. An effective, multi-cell model, potentially needs to be able to take into account the differing phases of the cells, but even for standard implementations the space of possible gains is huge. The average response could be calculated with knowledge of how often combinations of cells drift in and out of phase with one another.

This is calculable for a constant CSM system by using the different values of W_L. Such a model would only be a guideline for the general case of the algorithm and the computational complexity of evaluating such a model could potentially outweigh the benefits of automated parameter tuning vs. the trial and error approach.

Since this work was published, Gu et al. have explored an alternative technique for automatically parametrising the algorithm based on principal component analysis. This still requires the migration threshold distribution to be manually tuned, but attempts to automatically generate the source of the input signals (PAMP, Danger and Safe) from available domain data [54].

5.5.2 Summary

In this chapter a tuning methodology was designed to automatically identify the input parameters for the DCA, given a sample of training data. The methodology was explored empirically, but was found to be inferior to a trial and error approach to identifying the parameters. This poor performance was attributed to limitations of the model. An attempt was made to improve the model by extending it to take into account the effect of multiple cells operating simultaneously. It was identified that the asynchronous nature of cell migration, enforced by the broad spectrum of migration thresholds, creates a one to many relationship between input frequency and gain. This is not a relationship that standard frequency analysis techniques are able to model and suggests that this line of enquiry has been exhausted. However, Chapter 3 raised other questions about the DCA that are still in need of investigation. Namely, the practicality of using the DCA for a real-world anomaly detection algorithm and the DCA's performance in that field compared to traditional machine learning algorithms. These issues shall be explored in Chapters 6

and 7.

5.5.3 Contributions

Novel contributions provided by this chapter are:

- An initial tuning methodology for the DCA has been designed, but was demonstrated to be of limited practical use.
- It was demonstrated that while standard frequency analysis techniques are good at modelling the behaviour of a single dendritic cell, they are not a useful tool for future analysis of an entire population of dendritic cells.
- The reason for the tuning methodology's failings and the frequency analysis' inability to model a collection of cells was identified. The asynchronous nature of cell migration increases the complexity of modelling a population of cells to the point that standard frequency analysis tools are unable to further explore the algorithm.

Chapter 6

A Robotic Security Application

"Quis custodiet ipsos custodes? (Who will guard the guards themselves?)" - Juvenal, Satire VI [circa 100 A.D.]

6.1 Introduction

The work presented in this chapter is based on the experiments published in [110].

Chapter 3 identified the importance of good localisation for implementations of the DCA using location as antigen. Such a system applied to building security also requires the use of heuristics that will assist the algorithm in identifying anomalous or dangerous situations. In this chapter we present a technique for implementing a robotic security system that combines a neural-inspired algorithm for localisation and mapping and an immune-inspired algorithm for adaptive anomaly detection. By combining these two algorithms it is hoped that a scalable security solution can be constructed which takes advantage of localisation data to detect anomalous situations. This investigation is based on the DCA version outlined in Appendix B.7.

This chapter is structured as follows: in Section 6.2 a biologically inspired security system is outlined, which uses both the RatSLAM algorithm and the DCA; in Section 6.3 an experiment is designed to test this prototype system; in Section 6.4 the results of this experiment are presented and in Section 6.5 the meaning of these results is discussed.

This chapter concerns the merging of the DCA to the RatSLAM algorithm. As much information as possible is provided though, due to intellectual property issues concerning the commercial exploitation of the RatSLAM algorithm, information about the code and parametrisation is limited.

6.2 A Bio-Inspired Physical Security System

6.2.1 Combining the DCA with RatSLAM

As discussed in Chapter 2, RatSLAM is a biologically inspired algorithm for mapping and localising within a dynamic environment.

A crucial stage in the implementation of the DCA is the selection of appropriate input heuristics. For this application normality can be viewed as the standard operation of the building. For a physical security application the most analogous source of a PAMP signal would be a specific recognition algorithm for a universally dangerous situation, such as a building fire or the absence of an item of specific interest. It is the interaction between the neural model and the immune model that is of interest so this signal will be left out. Using the DCA without the PAMP signal for simplicity has precedent in other areas [44], (though for that publication, the authors also made significant changes to the algorithm itself, as outlined in Appendix B.8). For the Danger signal it was decided to monitor RatSLAM's localisation score, S. In RatSLAM a low score implies that the robot is lost and a high score indicates that the input data corresponds to the algorithm's current estimate of pose. From analysis of the algorithm's performance it is possible to discern that any value of S greater than 80% represents a good estimate of pose [1], while any value below that represents an increasingly poor estimate, with 0% representing a total localisation failure. Using this knowledge

[1] From personal communication with Dr. Michael Milford

the Danger heuristic was generated using equation 6.1.

$$Danger(S) = \begin{cases} 100 - S, & \text{if } S \geq 20 \\ 0, & \text{otherwise} \end{cases} \tag{6.1}$$

The purpose of the Safe signal is to ensure that the immune response is tolerised against circumstances where a certain level of abnormality is acceptable. An example of this mechanism in action within biology is found in the stomach, which is filled with foreign proteins as part of its normal operation so an immune response would be unwanted. As discussed in Chapter 2, the RatSLAM algorithm stores "experiences" to assist in the mapping of an environment. The number of experiences required to map dynamic environments is higher than the number of experiences required to map largely static or uniform environments. As a result it is possible to quantify the typical level of 'abnormality' for a given environment by calculating the density of the experience points around that position. Figure 6.1 demonstrates the variation between experience point density for four regions of space. In order to convert experience density for a given point in space into a usable Safe heuristic it is necessary to define two parameters: the radius r of the circle to be drawn around that point; and the upper density N to be used that will correlate to 100% signal output. These parameters will be explored as part of this investigation. It is of note that the RatSLAM algorithm regularly prunes experiences that have been unhelpful. In this way the algorithm should adapt to night-time conditions and re-tolerise against busy environments in the morning after a period of adjustment. The Safe signal is expressed in equation 6.2

$$Safe(x, y) = \frac{100 \times \sum_{i=0}^{E} f(x, y, x_E, y_E)}{N} \tag{6.2}$$

Where N is the number of points required to achieve the maximum

Safe signal of 100, E is the total number of experience points created and $f(x, y, x_E, y_E)$ is given in equation 6.3.

$$f(x, y, x_E, y_E) = \begin{cases} 1, & \text{if } \sqrt{(x - x_E)^2 + (y - y_E)^2} < r \\ 0, & \text{otherwise} \end{cases} \tag{6.3}$$

In previous robotic applications of the DCA a grid-based system was used to relate pose to an enumeration of state. This was ultimately flawed as using an enumerated type to describe a physical position in space limited the area that the algorithm could be used in. RatSLAM uses a toroidally mapped, three-dimensional grid to represent pose. It has been shown that this wrapping of the representation of space is effective over significant distances [99]. As there is clearly a finite number of pose cells within the RatSLAM algorithm, it is easy to map the most active pose cell as the antigen being presented to the dendritic cell population.

6.3 Monitoring a Cluttered Office Environment

The DCA has shown itself to be capable of operating on a robotic system, (see Chapter 3) and robust to noise, (see Chapter 4). The focus of this experiment is to explore the algorithm operating within a real-world environment. To fully test the practicality of the system, the environment should be unmodified and contain both static and dynamic obstacles. The hypothesis of this experiment is that the DCA will outperform a random classifier at the same task.

To explore the properties of the security system, it was presented with captured data from a cluttered, unmodified office environment. This data was gathered using a Pioneer 3DX robot using a camera attached to a

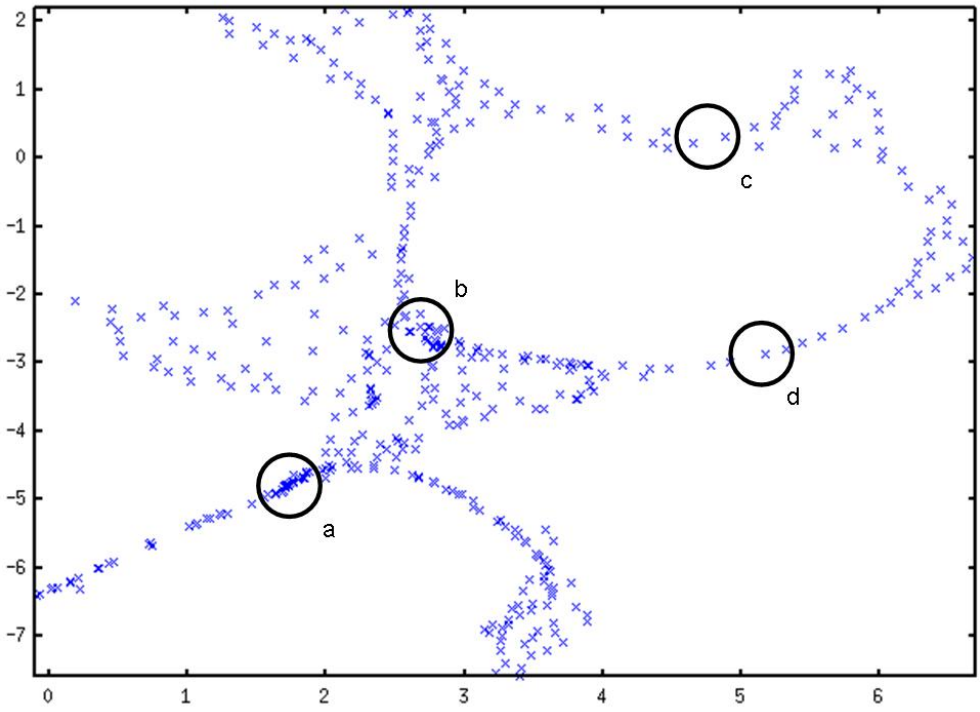


Figure 6.1: Generating the Safe Signal

The experience density for a given point in space is calculated by identifying the number of experiences (crosses) that are contained within a circle of radius r centred on that point. The parameter N scales the number of experiences to be between 0 and 100 where 100 correlates to N or more experiences. Using this metric, the points at the centre of circles a and b would generate high levels of Safe signal and the points at the centre of circles c and d would generate low levels of Safe signal.

parabolic mirror for localisation and a laser range finder for obstacle avoidance. Approximately 30 minutes of data was analysed. Logged data was used to aid repeatability for exploring the parametrisation of the system.

For the RatSLAM part of the system, the standard parametrisation and physical configuration was used. The visual input was from the lower half of a feed from a parabolic camera attached to the top of the robot. Laser and sonar sensors were used for obstacle avoidance.

It is standard practice to use a uniform distribution of migration thresholds centred around 15, 30, 60, 120 and 240 when identifying the appropriate range to use [42]. In each case the width of the distribution is set to $\pm 10\%$ of the centre. As the Safe signal heuristic is also parametrisable, a range of values of r will also be explored. The results for $r = 0.05, 0.1.0.2, 0.3$ and 0.5 will be presented here, (units in metres). N will be set to the maximum experience density found in the first few minutes of test data.

In each experiment, the performance of the algorithm will be compared to the results of a human operator identifying people walking around within the environment. Rather than simply identifying peaks where the algorithm successfully identifies a threat, the instantaneous difference between the human operator and the algorithm will be calculated, thus penalising the algorithm for a late identification or for prematurely returning to the "safe" state. For the purposes of analysis the output from the first 300 frames, (30 seconds) will be ignored. In this time the robot is considered to be lost, as it attempts to localise in the environment so the output from the algorithm is invalid.

In each case the width of the migration distribution will be set to $\pm 10\%$ of the centre point. Table 6.1 shows the number of experience points required for each radius size to generate 100% Safe signal.

Table 6.1: Parameter Values Used for N

In each case N is the number of experiences within the circle that caused 100% output from the Safe signal.

r (m)	N
0.05	10
0.1	19
0.2	28
0.3	35
0.5	48

6.4 Results

A full video of the experiment can be found at
http://www.cs.nott.ac.uk/˜rxo/thesis/
Figure 6.2 displays the results from 25 experiments using different values of r and the migration threshold distributions.

Increasing the radius of effect for the Safe signal reduced the false positive rate considerably. However, there was a sudden fall from 0.1 to 0.2 which was associated with the largest rise in the false negative rate. None of the chosen parametrisation outperformed a random classifier in terms of their false negative rate.

6.5 Discussion

The input parameters had a noticeable effect on the performance of the algorithm which was to be expected for a system which relies so heavily on expert knowledge for tuning. It is arguable that as a security system

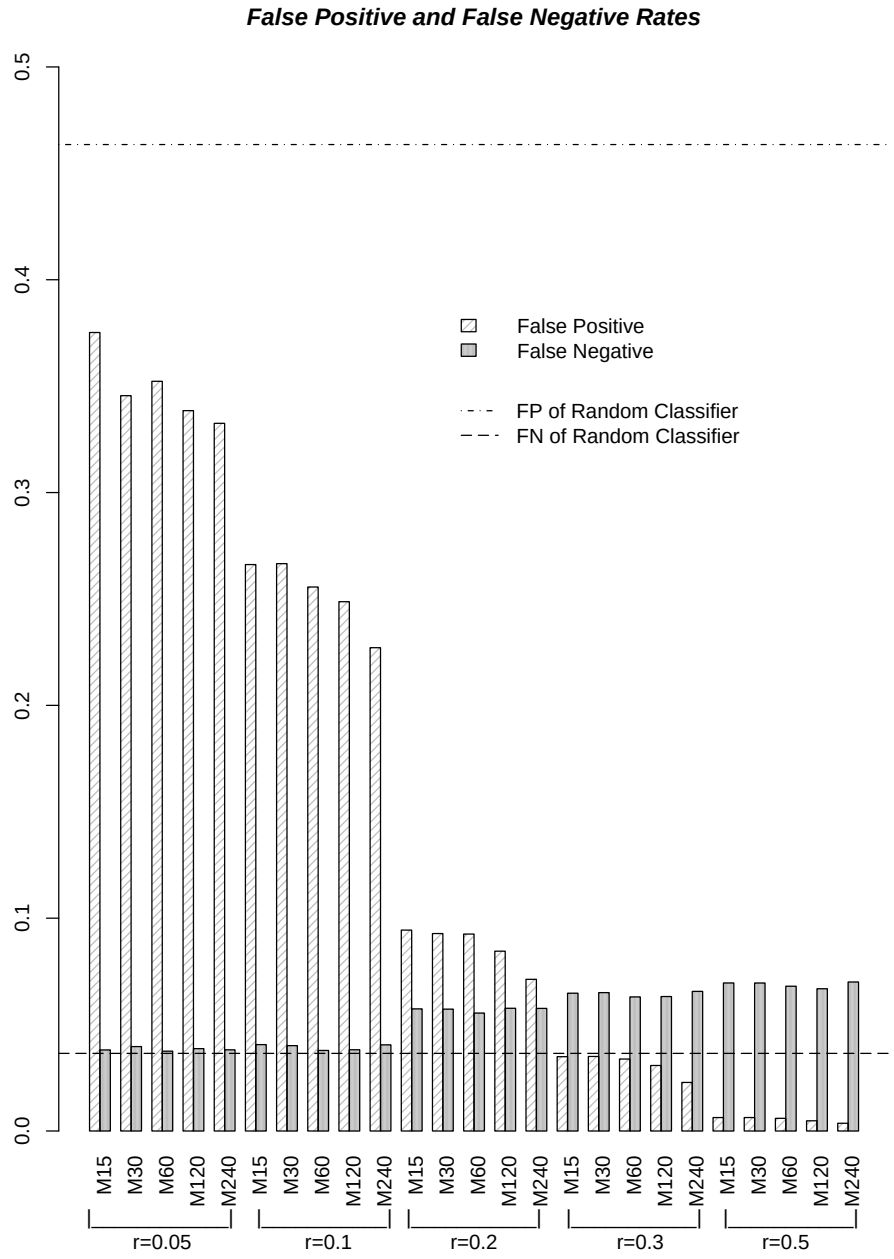


Figure 6.2: The Results from the 25 Experiments

The small x-axis labels indicate the centre point of the migration threshold and the larger x-axis labels indicate the groups of experiments with the same r value. The dashed lines indicate the theoretical results of a random classifier based on the ratios of positive and negative events from the human classification.

the performance of the algorithm with a radius of 0.3 and a migration threshold distribution centred on 240 had the best performance. This combination of inputs gave a false positive rate of 0.02 and a false negative rate of 0.07. As an augmentation of a manned security system these are promising results as the low false positive rate is likely to prevent alerts from being ignored by the operator. However, as the true positive rate, as determined by a human operator, was such an unlikely event, the theoretical performance of a random classifier was hard to beat. In a genuine security situation it is likely that the majority of what a security guard observes is "normal" behaviour and requires no intervention, a key advantage of this system is that it performs consistently regardless of how long it is in service. Figure 6.3 shows the first 3000s of the output of the best performing system over time.

With such a large migration threshold centre, it is likely that some of the false negatives have been caused by a delay between the input data being presented to the algorithm and the algorithm generating an immune response. In addition, the output from the algorithm appears to be pulsing (due to the larger differences between the cells), which is generating more false negatives. Assuming that any real system would raise an alert for several seconds at a time from the first immune response, it is likely that the performance of the system will be better than it originally appears. Certain key pulses are missed. This is likely to be for two reasons. Firstly, the algorithm is only observing the bottom half of the video feed, which could potentially mean that some inputs were missed. For example the pulse at iteration 9800 corresponds to the image in figure 6.4 which has a limited effect on the lower half of the image.

Secondly the human operator was only looking for individuals walking within the environment. The algorithm would ignore those pulses found in areas that were usually busy. The missed pulse at 2960 corresponds to

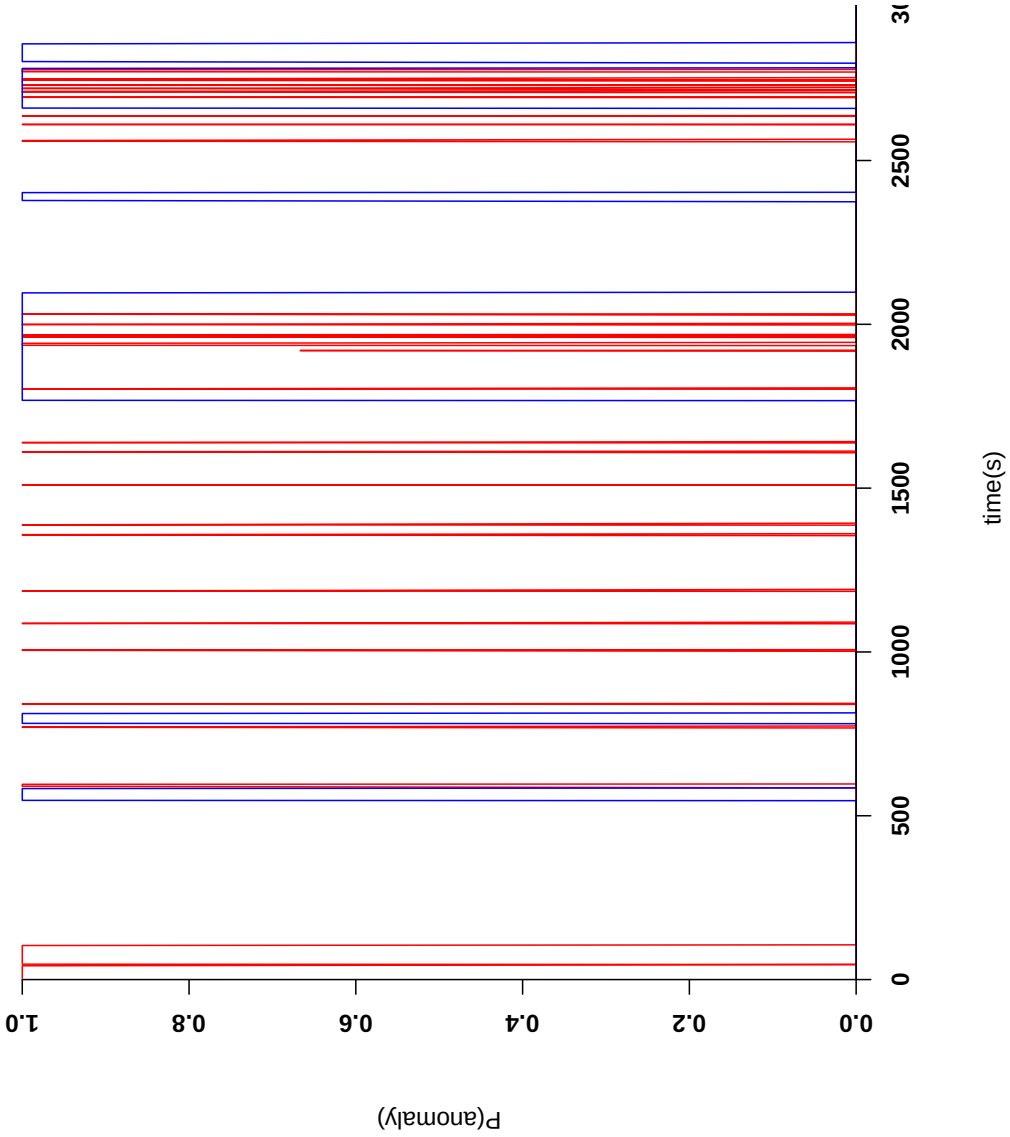


Figure 6.3: The Output from the Best Performing System

The first 3000s of the best performing system. The red line is the output from the system and the blue line is the output from the human operator

Figure 6.4: Problems With Occlusion

The figure (circled) has the bottom half of their body occluded by an object. As only the bottom half of the image is processed by the algorithm, this generates a false negative.

an office environment, where the objects, such as chairs etc are regularly moved. In this case the Safe signal rises consistently to approximately 20 which inhibits the response to someone walking in front of the camera. For a security system, ignoring areas which are typically cluttered and crowded is an advantage with obvious potential drawbacks.

In the future this system would benefit from the addition of a PAMP signal to recognise specific situations. As the PAMP signal is supposed to be a signature of known threats, it would be analogous to provide the robot with a smoke detector for identifying fires or an RFID detector to identify if specific pieces of equipment were present or not. This could be achieved using a standard classifier trained on data from artificial re-creations of those circumstances, or in the case of fires, a heat or smoke sensor could be physically connected to the robot. Additional sources of Danger and Safe to encapsulate information such as time of day could potentially vastly increase the usefulness of this system. For example it would be possible to automatically increase the Danger signal at night to increase the reactivity of the system or suppress the system when the robot's pose estimate moved into a foyer or other such area.

6.5.1 Summary

This chapter presented the design and implementation of a bio-inspired robotic security system. The system uniquely combined a neural model of a hippocampus with an immune system model. The neural model provided both sensor fusion and robot localisation. The immune system was able to utilise metrics used by the neural model to provide anomaly detection. The system demonstrated the ability to learn the "normal" state of a building and was able to adapt to long term changes. Empirical exploration demonstrated that the system was able to outperform a random model. This supports the supposition that the algorithm is making use of the data presented to it, but as yet there is no information about how well the algorithm will perform compared to traditional machine-learning techniques. However, minimising the bias in a direct empirical comparison would be extremely challenging due to the fact that there is no direct functionally equivalent algorithm available. In addition, comparing the DCA against a system that made use of training data would potentially bias the results, depending on the quality of the training data. As a result, the next logical step is to return to a theoretical perspective, this time examining the algorithm as a one class classifier.

6.5.2 Contributions

Novel contributions provided by this chapter are:

- A novel physical security system which combines the anomaly detection properties of an immune model with the adaptive properties of a neural model.
- The production of a synergistic bio-inspired model which takes advantage of key metrics to facilitate both anomaly detection and localisation and mapping.

Chapter 7

Geometric Analysis of the DCA

"Geometry is not true, it is advantageous." - Henri Poincare (1854-1912)

7.1 Introduction

The work presented here is based on [131].

Chapter 6 describes the implementation of a novel, bio-inspired physical security system. While it was possible to ascertain the algorithm's performance against a theoretical, random base-line, how well it performs against other techniques is not known. Comparing the DCA to known machine learning algorithms has been attempted before [48], however it is problematic. As discussed in Chapter 4, the DCA performs a variety of roles, filtering information, performing temporal correlation with antigen and eventually performing a classification for each sampled antigen based on the associated environmental signals. To perform a naïve comparison between the DCA and another algorithm which fulfils only one of the roles that it performs is difficult. If the base-line outperforms the algorithm for one application, it is difficult to be sure that it will outperform the DCA over all. If it outperforms the DCA at its specialty, but does not provide any of the other functionality, what weight should be placed on the DCA's additional functions? If one were to construct a hybrid algorithm, using traditional machine learning and signal processing techniques (say, for example, a population-based filtering algorithm and an ensemble classifier), how would we attribute success/failure to individual components? Could the system be let down by a poor choice of training data? Could the frequency responses of the filters be introducing a bias? All of these issues limit the usefulness of a direct, empirical comparison. Instead it was decided to pick apart the elements of the DCA and compare them to the behaviours of stand-alone algorithms from a theoretical stand-point. This makes comparisons fairer and the results more useful from a practical perspective. To some extent, Chapter 4 has already begun this process by treating the algorithm as a filter. In this chapter the classification phase of the algorithm is analysed by demonstrating its

equivalence to an established machine learning algorithm.

For this analysis the Deterministic DCA [44] shall be used. Amongst other changes, this version of the algorithm ignores the PAMP signal and concentrates on the interplay between the Danger and Safe signals. It is outlined in full in Appendix B.8. Ignoring the PAMP signal reduces the dimensionality of the input data and makes analysis simpler to visualise. It is of note that the techniques outlined here will work for all versions of the DCA, but the two dimensional version is simpler to visualise on paper.

7.2 Linear Classification

Linear classifiers are a well-established form of machine learning. Within machine learning, the classification problem can be viewed as determining whether an example input, presented to the classifier in the form of an N-dimensional vector, has membership to a given set or not. Classifiers are typically supervised learning algorithms, with two distinct phases: a training phase and a classification phase. In the training phase the linear classifier is presented with a series of N-dimensional vectors, each with a corresponding label indicating if the example is a member of the set to be identified or not. During this phase a learning algorithm is applied which attempts to construct an $(N-1)$-dimensional hyperplane, (a single point for one dimensional space, a line for two dimensional space and so on) which will act to separate set members from set non-members. As explained, in the classification phase, observations of the training set are used to identify if the presented example is part of the set of interest or not. There are a huge number of different implementations and learning algorithms within this field. However, once trained, linear classifiers can always be expressed as a weight vector $\mathbf{w}$ and a bias term w_0 which define

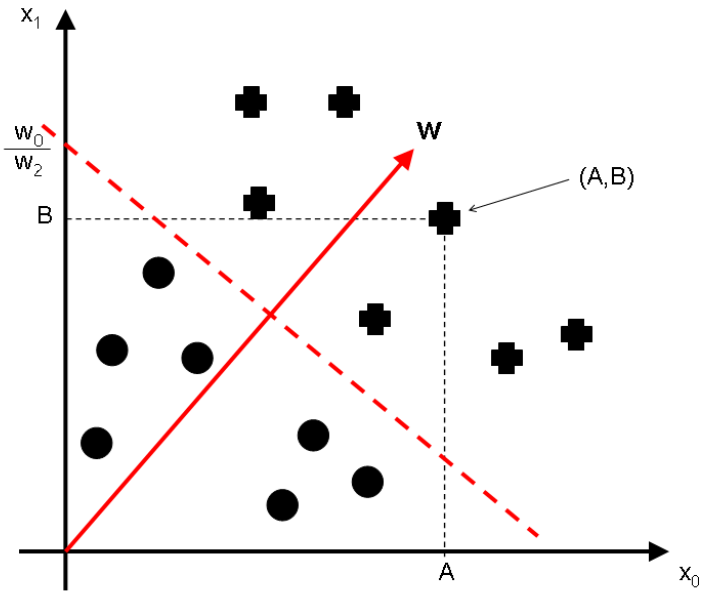


Figure 7.1: A Linear Classification Example

In this representation the members of the set of interest, (represented by crosses) can be separated by the dashed red line. Each example is represented by a point in the input signal space. The dashed red line can be represented by a weight vector w and a bias w_0.

the position of the hyperplane in the input signal space. The orientation of the dividing hyperplane is always perpendicular to the vector described by **w** and its offset from the origin is proportional to the w_0 bias. In the two dimensional case, the y-intercept of the decision boundary is given by the ratio between w_0 and w_2, the second term of the **w** vector. These concepts, in their two dimensional case are illustrated in Figure 7.1.

The weight vector representation of a linear classifier is implemented using the dot product. This mathematical concept is an operation which can be performed on two vectors of equal length. In this case the first vector is the weight vector that represents the linear classifier and the

second vector is the N-dimensional vector representing the point in signal space to be classified. The dot product is simply the sum of the products between corresponding elements of each vector and is represented by equation 7.1.

$$\mathbf{w}.\mathbf{x} = \sum_{i=0}^{N} w_i \times x_i \tag{7.1}$$

In [100] the significant limitations of linear classifiers were outlined. These were seen as a severe limitation to neural networks as a whole. However, non-linear classification methods, such as multi-layer perceptrons, have found techniques to compensate for these limitations [58]. These limitations revolve around the linear nature of the classification boundary, which not only severely limits the datasets for which they can be applied to, but prevents any benefit from applying them in a cascade as the linear combination of several linear classifiers can ultimately be expressed as a single linear classifier.

7.3 Representing The DCA as an Ensemble Classifier

The deterministic DCA uses equations 7.2 and 7.3 to calculate the intermediate control signals for each cell.

$$\mathrm{CSM}_n = S_n + D_n \tag{7.2}$$

$$\mathrm{K}_n = D_n - 2S_n \tag{7.3}$$

Where D_n is the current sample of the danger signal, S_n is the current sample of the safe signal, CSM_n is the current CSM value and K_n is the current value of the K signal.

The means of collating the outputs from a population of cells varies between versions. A comparison between two popular collation techniques for the DCA is performed in [4]. The most common technique is to use the "mean context antigen value" or MCAV [47]. This technique has been discussed in previous chapters and can be seen in pseduocode form in Appendix B.4.

As discussed in Section 7.2, for typical classifier analysis it can be instructive to view the positions of the classification hyperplanes with respect to the input signal space. However, the DCA does not operate on the input signals directly, rather the sum of the signals experienced over a given cell's lifetime. This presents a challenge as the population is asynchronous and a cell can migrate at any time relative to the rest of the population depending on its input history. Conventional techniques for visualising linear classifiers that use their input history (as opposed to their instantaneous inputs) usually treat each sample as a new input dimension. However, the number of samples a dendritic cell analyses varies depending on signal magnitude via the CSM gating mechanism, so the resulting space would have varying dimensionality between cells and input magnitudes. To overcome this challenge, rather than using the instantaneous input signals as axes, we can use the cumulated input signals. In this space the algorithm becomes easier to explore.

Each cell generates two planes in this space. The first is the decision boundary. The cell will not make a decision, (i.e. migrate) until its current decision boundary is reached. At the first instance where the cell's decision boundary is reached, the point is compared to the second plane, the classification boundary, to decide if the cell is voting for normality or 'anomalousness'.

The inequality defining when a cell migrates is given in equation 7.4

$$M_i \leq \sum_{n=0}^{T} \mathrm{CSM}_n \tag{7.4}$$

Where M_i is the migration threshold of cell i and T is the index of the current sample. Substituting the definition for CSM_n from equation 7.2 into equation 7.4 and rearranging for $\sum$safe gives equation 7.5.

$$\sum_{n=0}^{T} S_n = -1 \times \sum_{n=0}^{T} D_n + M_i \tag{7.5}$$

Where $\sum S_n$ is used as the y axis and $\sum D_n$ is used as the x axis. However, this only holds in the individual cell's frame of reference. To express equation 7.5 in the global co-ordinate frame, the line must be offset by the starting position of the cell in signal space, (x_P, y_P). Equation 7.5 shows that all decision boundaries are parallel going from $(x_P + M_i, 0)$, to $(0, y_P + M_i)$ with a constant gradient of -1. The classification boundary can be defined similarly, using the inequality in equation 7.6, which defines 'normality'.

$$\sum_{n=0}^{T} K_n > 0 \tag{7.6}$$

Substituting equation 7.3 and rearranging as before provides equation 7.7, the classification boundary.

$$\sum_{n=0}^{T} S_n = 0.5 \times \sum_{n=0}^{T} D_n \tag{7.7}$$

This also gives a constant gradient, of 0.5 in this case, implying that all of the classification boundaries are also parallel. This demonstrates that not only can an individual cell be modelled as a linear classifier, but a population of cells is limited to only produce those classification boundaries that can be expressed as combinations of parallel planes.

Rearranging equations 7.5 and 7.7 into the dot product representation for a linear classifier yields equations 7.8 and 7.9 respectively.

$$y_D(\mathbf{x}) = \langle \mathbf{W_D}, \mathbf{x} \rangle - M_i \tag{7.8}$$

Where y_D is the decision boundary, W_D is a vector representing the weightings of the Safe and Danger signals towards the CSM signal and M_i is the migration threshold of the cell being considered. For the current version of the DCA $\mathbf{W_D}$ is $[1, 1, 1, 1, ...]$ and $\mathbf{x}$ is $[D_n, S_n, D_{n-1}, S_{n-1}...]$ where the length of the vectors is defined by the number of samples required to satisfy equation 7.4.

$$y_C(\mathbf{x}) = \langle \mathbf{W_C}, \mathbf{x} \rangle \tag{7.9}$$

Where y_C is the classification boundary and W_C is a vector representing the weightings of the safe and danger signals towards the K signal. For the current version of the DCA $\mathbf{W_C}$ is $[1, -2, 1, -2, ...]$ and $\mathbf{x}$ is the same varying length vector described in equation 7.8.

7.3.1 Visualising a Single Dendritic Cell

Figure 7.2 illustrates the visualisation of the decision boundary and classification boundary for a single dendritic cell.

The x axis is the cumulated danger signal that the cell has been exposed to and the y axis is the cumulated safe signal that the cell has been exposed to. Each cross represents a change in the input signal experienced by the cell and the connecting solid black line represents the path through the signal space that the algorithm's previous inputs have described. It is of note that this path is constrained to move neither to the left nor in a downwards direction. This is because the input signals are positive values only and the input space is cumulative. The 'cell start point' is the point at which the cell last migrated and had its cumulated CSM and K signal reset to 0. The thin, solid line indicates the decision boundary, the first input signal that breaches that boundary will lead to a classification. If the input signal causes the path to enter the white region, (marked 'A') the cell will classify all collected

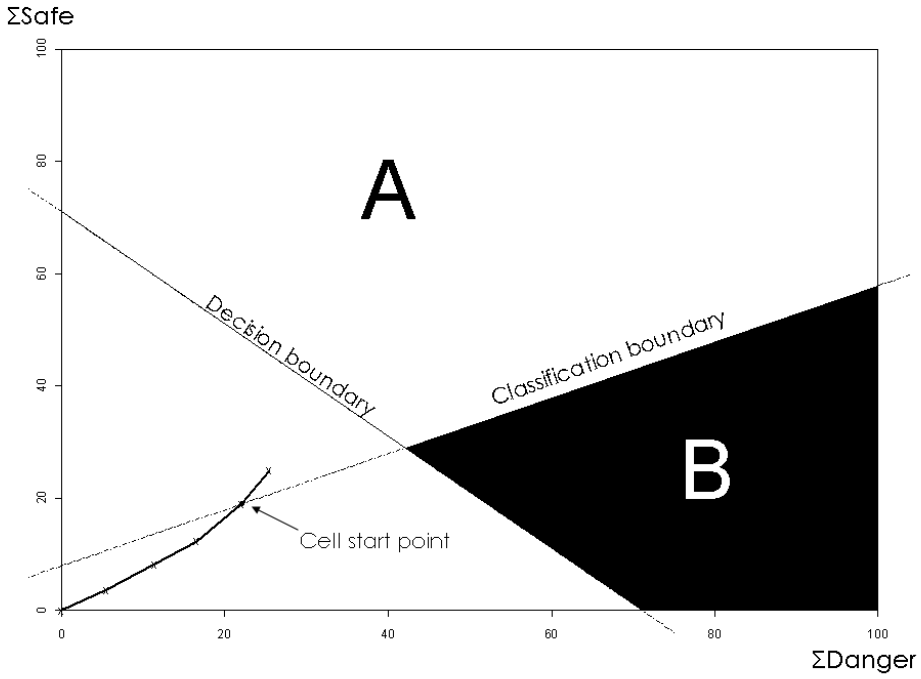


Figure 7.2: A Single Dendritic Cell's Cumulative Input Signal Space

antigen as being "safe". Conversely, if the input signal enters the black region, (marked 'B') the cell will classify all collected antigen as being "anomalous". These two regions are divided by the thin, dashed line, which illustrates the classification boundary for the cell. If the next signal change fails to breach the decision boundary, no classification shall be returned by the cell and it shall simply wait until sufficient data has been sampled.

To illustrate how this would work in practice, consider two data sources presenting input data to the cell, one stream indicative of a normal situation and another indicative of an anomalous situation. Each time a new item of data was presented to the cell, the line representing the input history of the cell, (solid, marked with 'x's) would navigate through the space. The cell would not respond to that input until the line passed through the dashed line, marked the 'Decision Boundary'.

The first data item that caused the input history to breach the decision boundary would be classified as normal or anomalous. This decision would be based on the position of that point with respect to the 'classification boundary'. If the point was above it, (the region marked with 'A') the cell would report a normal situation and if the point was below it, (the region marked with 'B') the cell would report an anomalous situation.

7.3.2 Visualising a Population of Dendritic Cells

In order to visualise a population of cells, it is necessary to define how the output of the cells will be combined. For this chapter it shall be assumed that the MCAV is calculated after every iteration. To simplify analysis, it will also be assumed that there is only one type of antigen being classified. The first assumption is reasonable for a real-time implementation of the algorithm. The second assumption is not realistic in an application sense. However, it is instructive for demonstrating the different classification region shapes that the cell population can construct. For the multi-cell visualisation, the same key can be used to identify the positions of the classification and decision boundaries, but instead of a black and white region, a grey-scale map will be produced that illustrates the possible outcomes for the next signal input. As before, black illustrates an anomalous region, (i.e. the probability that the data is normal approaches 0) and white illustrates a normal region, (i.e. the probability that the data is normal approaches 1). To illustrate a population of cells, random data shall be presented to the algorithm, with a Gaussian distribution. By using Gaussian distributions for both signal sources, it is hoped that the generated data will exemplify 'normality' for the most part, but will also produce situations where 'anomalous' signal characteristics will be generated at the extremes of the Gaussian curves. Two

sets of data shall be used, both with a standard deviation of 1, but the first shall use a mean of 5 and the second shall use a smaller mean of 2. In each case a population of 100 cells will be used, with a uniformly distributed set of migration thresholds, ranging from 15 to 45, (a standard range for the DCA.) The two different values for the mean of the Gaussian distributions will alter the ratio of signal strength to migration threshold for the cells, a property which is known to alter the response from the algorithm, (see Chapter 4). Larger steps should cause more cells to migrate in unison and larger steps should maintain greater diversity in the population throughout the lifetime of the algorithm.

7.4 Results

Figure 7.3 illustrates four steps through the algorithm, using randomly generated input data with a mean of 5. Step five was chosen as a starting position as it allows the algorithm to settle into its usual operation. At step 0 all cells have been exposed to exactly the same amount of signal (i.e. 0), so the classification boundaries are all exactly the same.

In Figure 7.3(a) there are only three distinct classification boundaries, (the dashed lines). This is because the cells have formed into three groups of cells. This separation is also observable by inspecting the decision boundaries, which have clearly split into three clusters. The gaps between the decision boundary groups are formed by the magnitude of the input data to the algorithm. Large steps cause many cells to migrate at once. The black circles on the data path mark the steps at which the cells that are part of the current population were last reset. Only three circles exist, showing that this population has no cells older than three iterations. The shape of the classification regions are all constructed from an averaging of linear classification boundaries. The

shading demonstrates that in certain regions the classification is highly sensitive to change. As a result, in the boundary between normal and anomalous, small changes in the input signal can result in drastically different classification outputs. In this case it is even possible for a counter-intuitive situation to arise, where increasing the danger signal slightly can cause the classification to become normal. This is because cells that have been exposed to more normal signal in their overall lifetime can be forced to migrate by either signal, and can outnumber the cells voting for 'anomalousness'. This is a potential source of error for the classification, as it cannot be pre-trained or controlled, it is simply a function of the input data. Figure 7.4 is an expanded view of an example boundary region between $\sum \text{Safe} = 20$ and $\sum \text{Safe} = 40$. Here it is possible to discern that the shading is not gradiated, but pseudo-randomly distributed, according to the density of the overlapping decision boundaries and the previous inputs. This expanded view shows that at the boundaries between the white (normal) and black (anomalous) regions the varitation in colour is not always through equally distributed shades of grey. The solid diagonal lines travelling from the top left corner of the plot towards the bottom right corner, represent decision boundaries of cells within the population. The shades of grey are proportional to how anomalous the population wide classification will be, (darker indicating more anomalous). In this space, the presentation of safe signal causes the input data point to travel upwards and the presentation of danger causes the input data point to travel to the right. The patterns of shading show that in certain rare cases, there are points within this landscape where if the input data line were travel directly upwards, it would enter a darker region than if it were to travel to the right. This illustrates the above point, that sometimes safe signal can sometimes actually cause a more anomalous classification than danger signal!

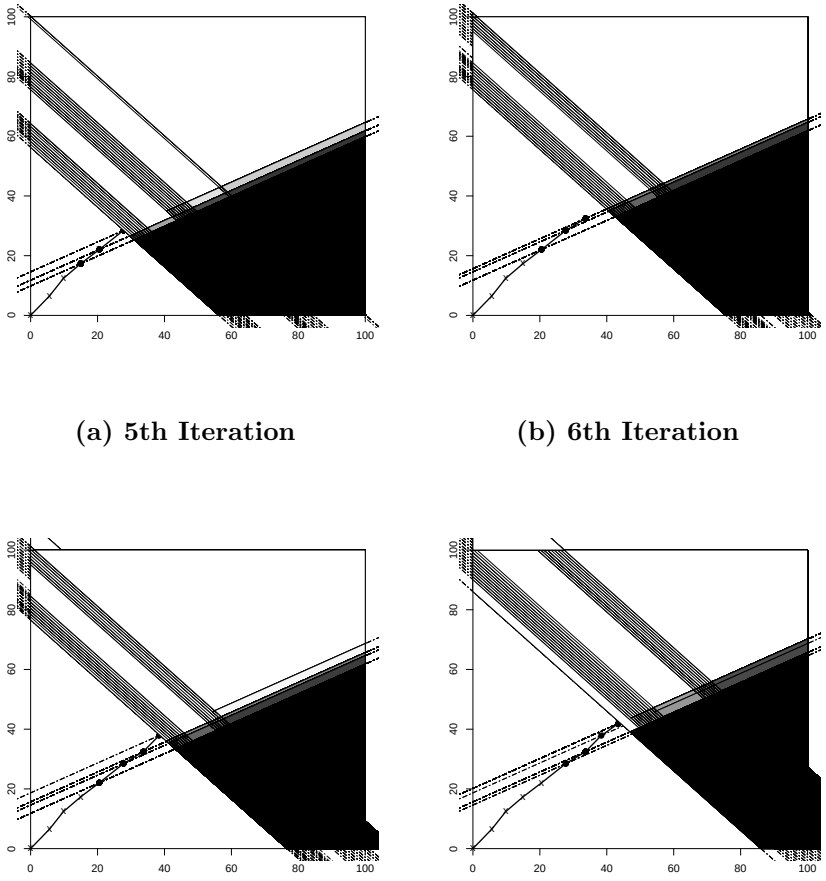
In Figure 7.3(b) many of the cells are forced to migrate as the input signal path passes through their decision boundaries. All of the cells in this case classify their collected data as normal, though this is unsurprising given that the random input signals have equal means and that the safe signal is weighted more heavily than the Danger signal in the decision making process. Note how the first and second set of cells are pushed closer together by the migration process. This highlights a further potential problem with the algorithm, as the number of cells that migrate, for a given input, is going to be exceptionally hard to predict, as the density of decision boundaries is going to be highly dependant on the input data.

In Figure 7.3(c) the input signal fails to breach any decision boundaries. This results in the classification boundaries staying as they are. In practical terms, this would represent the algorithm not generating any output, and waiting to receive more data.

Finally, in Figure 7.3(d), several cells migrate and widen the 'grey area' between normal and anomalous. The wider region is indicative of the size of the signal that caused the migration to occur. In other words, the width of the classification boundaries is representative of the diversity in accumulated CSM for the cells in the population.

By using a Gaussian distribution with a smaller mean the effects of smaller input signals, relative to the selected migration thresholds can be observed. Figure 7.5 shows steps 5 through 8 for the algorithm, using randomly generated data with a mean of 2.

In Figure 7.5(a) there are only two distinct classification boundaries. The black spot on the origin is a sign that many of the cells are yet to migrate. The light colour of the region at the anomalous/normal border indicates that in fact, very few cells have migrated. Again, the large width of that region is indicative of the large distribution of accumulated



(a) 5th Iteration

(b) 6th Iteration

(c) 7th Iteration

(d) 8th Iteration

Figure 7.3: DCA Boundaries Moving (Gaussian input, mean=5)

These figures illustrate the 5th, 6th, 7th and 8th iterations of the algorithm responding to randomly generated input data. The population size is 100, and the migration thresholds are between 15 and 45. The input data is generated using a Gaussian probability distribution, with a mean of 5 and a standard deviation of 1.

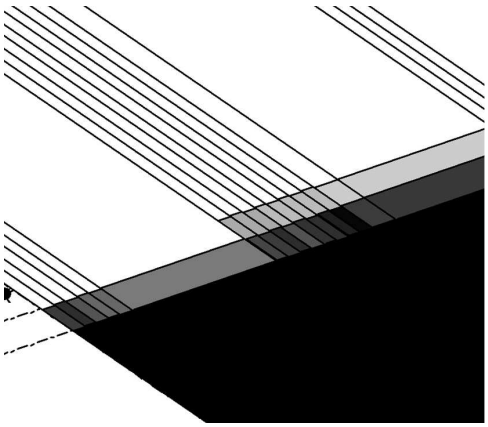


Figure 7.4: An Expanded View of an Example Boundary Region

Here we can see that the classification distribution is not gradiated, but varies significantly for minor changes in input signal.

CSM within the population.

Figure 7.5(b) shows the input signal path moving with a gradient approximately equal to the gradient of the classification boundaries. This is the equivalent of receiving sufficient volumes of conflicting data in the short term to cause the certainty of the classification to drop. However, in this case, the majority of the cells are yet to migrate, so the longer-term view of the input signals causes the population to be heavily biased towards a normal classification. Contrasting the positions of the decision boundaries between Figure 7.5(b) and Figure 7.3(b) demonstrates some of the effects of smaller magnitudes of input data. The decision boundaries are more clustered together, indicating that if the signal strength remains at this level, the population will continue to output classifications at a quite steady rate, rather than the more erratic output from a larger input signal magnitude. This is further supported by Figures 7.5(c) and 7.5(c) where each subsequent presentation of input data causes an additional set of cell migrations.

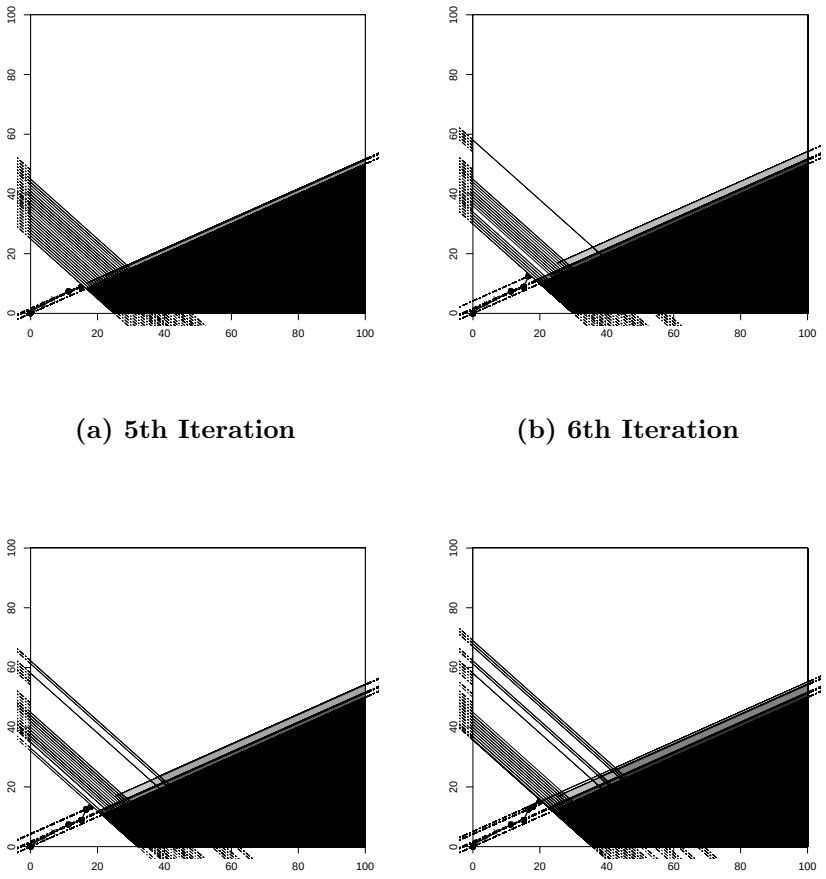
The large number of black spots in Figure 7.5(d) indicates that the population has a diverse number of classification boundaries, though they

are extremely close together.

7.5 Conclusions

The equations defining the decision and classification boundaries for the DCA raise many issues with the algorithm. Demonstrating equivalence to a collection of linear classifiers, implies that all of the known weaknesses of linear classifiers can be levelled at the DCA. These include the severe limitations on the data sets that the algorithm will be able to assess. This is made worse by the fact that the gradients of the boundaries are constant, applying still further restrictions onto the regions in signal space that the algorithm can discriminate between. It is likely that for the performance of the DCA on complex data sets to be competitive with other classification-based techniques that it will become necessary to introduce non-linearity into the classification boundaries. In contrast to learning machines where the model is inferred from the data set, the DCA requires the user to construct a model a priori to fit the hard coded weights. However, as with all expert system solutions, the optimal separation for the problem is unlikely to be found by the user. An alternative to this is discussed in [55]. In this work, the author suggests that a kernel method could be introduced to the DCA to expand the possible regions in signal space that could be discriminated between. This would in itself be challenging, as parametrising the appropriate kernel would be a non-trivial task. This overcomes the limitation of the achievable classification shapes, but not the issues associated with requiring the user to generate the model. In fact, this task is made more difficult through the introduction of non-linearity.

On a positive note for the DCA it is apparent that the way in which these linear classifiers are combined is a function of the input data and



(a) 5th Iteration

(b) 6th Iteration

(c) 7th Iteration

(d) 8th Iteration

Figure 7.5: DCA Boundaries Moving (Gaussian input, mean=2)

These figures illustrate the 5th, 6th, 7th and 8th iterations of the algorithm responding to randomly generated input data. The population size is 100, and the migration thresholds are between 15 and 45. The input data is generated using a Gaussian probability distribution, with a mean of 2 and a standard deviation of 1.

therefore, usually non-linear. This means that it is not possible to represent the entire population as a single linear classifier. However, this benefit is of limited use as, for non-trivial applications, the input data is not known prior to the execution of the algorithm. Another positive point for the DCA is that neural networks do not by default have decision boundaries, only classification boundaries. The phenomena whereby the DCA delays making a decision while it aggregates information is not found within a standard neural network implementation. As this is the mechanism which gives the DCA its robustness to noise (see Chapter 4), this may be a property that should be carried through to any revised versions of the DCA.

In this chapter a novel visualisation technique for exploring the DCA was suggested. The technique has made it possible to make predictions about how the algorithm will react to input data. It also makes it possible for inferences about the flow of data and the shape of the classification boundaries to be made. For the first time it is possible to predict signal combinations that will result in the algorithm's behaviour becoming highly sensitive to subtle changes in signal magnitude, and in the future may facilitate better tuning of the algorithm based on test data. The visualisation technique also makes it possible to comment on the diversity of the population and how that impacts on the classification for a given sample of input data. In short, the technique allows a deeper insight into the families of problems for which the DCA is an appropriate tool.

This model has a weakness in that it does not include the antigen sampling phase of the algorithm. However, as the antigen sampling phase has no direct link to the classification phase the results of the analysis still hold.

7.5.1 Summary

This chapter demonstrates how a dendritic cell can be modelled as a linear classifier. This has allowed a novel visualisation techqniue to be presented that makes it possible to explore how the algorithm as a whole reacts to its input signals. The equivalence of a single DC to a linear classifier indicates that known weaknesses of linear classifiers could be present in the DCA. These include the inability to solve linearly inseparable problems. In addition, with no training phase, the DCA is reliant on the system designer to transform the domain knowledge from the problem into a linearly separable problem.

7.5.2 Contributions

Novel contributions provided by this chapter are:

- A geometric analysis of the DCA demonstrating areas of equivalence to linear classifiers.
- A new technique for visualising the DCA's operation within its input signal space.

Chapter 8

Conclusions and Discussion

"The open mind never acts: when we have done our utmost to arrive at a reasonable conclusion, we still. . . must close our minds for the moment with a snap, and act dogmatically on our conclusions." - George Bernard Shaw, Androcles and the Lion (1912)

In order to frame the conclusions from this work in an appropriate context they will be discussed with reference to the original research questions discussed in Chapter 1.

8.1 Migrating from Software to the Physical World

> Can an immune-inspired, anomaly detection algorithm be adapted to solve threat detection problems in the physical world, through the medium of a robot?

The initial experiment presented in Chapter 3 demonstrated that it was at least possible to migrate a computer security algorithm into the physical world through the medium of a robot. It raised several important practical issues, such as the need for good quality, enumerable localisation. 'Good quality' as the system's performance requires the robot to not only classify the 'anomalousness' of its surroundings but also communicate that information in a useful fashion to an end user. 'Enumerable' as computer security algorithms are typically applied on running processes with unique identifiers, so making the transition into the physical world is made easier by applying a similar classification to the robot's surroundings. Initially problems with localisation were combated using a standard continuous measure of space, augmented with a particle filter which was then transformed into a grid pattern and presented to the dendritic cell algorithm as an antigen stream. This was computationally costly and required the system to be given a pre-constructed map of the prospective environment. However, in Chapter 6 a more elegant solution was presented, making use of the RatSLAM algorithm's pose cells to represent space, facilitate better localisation and build its own, scalable map of the environment.

It has been demonstrated that the algorithm's performance is statistically different when it has an effective localisation technique, and that it performs better than a theoretical classifier acting randomly on the input data.

In the future this could be further improved by returning to a more 'computer security' approach and introducing signatures of known threats, through the PAMP signal. These could include detectors for fires, chemical agents or any application-specific threat which could be accurately characterised. The system is already able to accept multiple sources of Danger and Safe signal, so other social factors could be incorporated into the system with ease, such as increasing the Danger signal at night when the system should be more wary of intruders.

8.2 Emergent Properties of the Dendritic Cell Algorithm

> Does the dendritic cell algorithm have properties that were not explicitly added as part of its design, which could be advantageous to a robotic application?

The work in Chapter 3 not only showed that the algorithm could make the transition from computer security to physical security, but it also demonstrated that the algorithm seemed capable of handling raw, unfiltered signals from physical sensors, with no modification. Chapter 4 went on to prove this hypothesis using frequency analysis to construct an accurate model of a single dendritic cell and demonstrating clearly that the cell had a frequency-dependent component. This frequency-based removal of noise is a significant find for the dendritic cell algorithm and clearly improves its usefulness for robotic applications. A crucial finding was that reducing the migration threshold for a cell, increases the fre-

quency range that it is able to pass, making the algorithm more sensitive to changes in input signal.

However Chapter 5 demonstrated several limitations on the practical use of this emergent property. While the failure of the author to construct a working tuning methodology based on frequency rejection is in no way evidence that such a thing cannot be managed, it pointed at a deeper problem surrounding the complexity of such a task. This was confirmed by the analysis of a population of dendritic cells which indicated that the one-to-one mapping between gain and frequency assumed by most traditional frequency analysis, simply does not hold for the dendritic cell algorithm as the population can asynchronously drift in and out of phase with one another. It is the belief of this author that simplifying the algorithm by removing the asynchronous migratory behaviour of the cells would be to deny the algorithm of one of its most interesting and, potentially, one of its more useful features. Not only does this population-based approach allows decisions to be made over several different time-scales, it also holds with the original, biological influence of the algorithm. This property is clearly visible in the visualisation used in Chapter 7, where small increases in the Danger signal could occasionally cause counter-intuitive classifications of antigen as being 'normal'.

8.3 Applying The Dendritic Cell Algorithm to a Robot

> Is it possible to adapt the dendritic cell algorithm from being a batch system to a system that can operate on a robotic platform?

This question is answered in part by addressing the movement of the DCA from the software world to the physical world discussed in Section

8.1. Again Chapter 3 lays the foundations for making the claim that it is possible, with the introduction of time-based segmentation, to increase the turn-over of antigen processing. However there is additional work in Chapter 4 which illustrates several key optimisations to the algorithm, which vastly reduce the computational complexity of the algorithm by moving several calculations outside of the dendritic cell and into the signal preprocessing layer, (sometimes termed 'tissue'). This optimisation not only made the implementation of a practical robotic system in Chapter 7 possible, but also provided the basis for much of the theoretical analysis that occurred in Chapters 4, 5 and 7, as it removed the biologically named signals and transformed them into an easier to analyse, single variable system.

8.4 The Benefits of the Dendritic Cell Algorithm

> Are there other algorithms with functional equivalence to the dendritic cell algorithm, which can outperform it in terms of reduced computational complexity or superior performance, for the threat detection problem?

This work's purpose is ultimately to answer this question. Again, it is important to acknowledge Chapter 3's demonstration that it is possible to move the DCA from the software world to the physical world. Chapter 4 demonstrated that the algorithm did indeed present some benefits as it filters noisy input data as part of its standard implementation. While Chapter 5 does conclude this line of enquiry, by suggesting that this is as much information as frequency analysis will be able to provide us on this algorithm, it also suggests that the population-based nature of the algorithm introduces a previously unexplored level of complexity. Moving

on from frequency analysis, Chapter 6's empirical study demonstrates that not only does the algorithm have emergent filtering properties, but it can also both perform information fusion and work synergistically with a SLAM system to solve the classification stage of the security problem. Despite these promising results, Chapter 7 indicates that the performance of the algorithm is limited by the known weaknesses of linear classifiers. In addition, constant gradients of the decision making boundaries impose additional restrictions on the algorithm's ability to make useful decisions. While other machine learning techniques do not provide the noise removal that the DCA does, they can be trained and adapted to suit a much more diverse range of application areas and in all likelihood, outperform the DCA at the the areas that it has been used on.

8.5 Limitations

Like any investigation, this work has limitations that should be explored and minimised in the future. Any robotic investigation must make the choice of whether to explore the system in simulation or in reality. The key advantage of using a simulator is that multiple configurations can be run extensively and compared. However, a simulation will never provide the rich variation of problems that a real robotic experiment will. A real robot was used for all of the empirical studies performed in this thesis. The pen used in Chapters 3 and 5 was kept the same for all experiments, to explore how the parametrisation effected the results for that problem. Unfortunately this also meant that no statements could be made about how well the algorithm performed in general, only about how it performed for that specific scenario. Again, by keeping the pen as an artificial environment, the beneficial effects of running a real robot are reduced. The alternative would be to run the robot in a totally unconstrained

environment, but this would limit the repeatability of the experiment. Chapter 6 used a different approach. Data was recorded on a real robot and replayed back to the algorithm. This was an acceptable solution as the feasibility of running the algorithm on the Pioneer had already been verified, (though admittedly without the additional computational load of running RatSLAM). This approach introduced a new problem, as in an unconstrained environment, there is no way to impartially calculate the baseline for "truth" in the experiment, (if that were possible, the problem would be solved). Instead a human classified the data, introducing a new source of potential bias.

As discussed in the Chapters 4 and 5, there are several weaknesses with the model of the DCA in the frequency domain. The most dramatic of these is the inability to model the interaction between multiple cells. After highlighting the importance of parametrisation of the system, it was still not possible to develop a successful tuning mechanism. In addition, the tuning mechanism suggested failed to take into account the fact that some applications may respond better to different probability density functions for selecting the migration thresholds of the cells.

Across many chapters it was clearly demonstrated that the DCA could be modified to operate on a robot to provide a solution to the physical security problem. However, no empirical comparison was performed against a competing system. This is partially offset by the demonstration of equivalence to linear classifiers, (Chapter 7) and decimators, (Chapter 4), so the well documented discussions about their weaknesses can be applied to the DCA. However, the magnitude of the effect of these weaknesses cannot be assessed without an empirical study.

8.6 Future Work

8.6.1 The DCA

A criticism from [42] is that the DCA has a propensity to have a high false positive rate. Looking back to the biological model, negative selection of T cells prevents false positives from biological dendritic cells from influencing the body. While the dendritic cell algorithm has obvious abstractions that place a large degree of separation between the algorithm and the biological dendritic cell, a possible improvement is the introduction of a similar barrier in order to improve the algorithm's false positive rate. This could simply be in the form of identifying signatures of "normality" at the antigen level and simply preventing those antigen from being allowed to report as anomalous. This is the functional equivalent of a DC being unable to locate a T cell with a high affinity to antigen that it has classed as dangerous, due to the negative selection process.

This work has highlighted several useful features of the DCA, including its inherent filtering, its capacity to make decisions over several time-scales and the potential it has for working with other biologically inspired algorithms. It has also pointed out several limitations, largely to do with the difficulties tuning and adapting the algorithm to new and complex data sets. It is the belief of this author that the DCA does have potential as a useful tool for roboticists and computer scientists alike, with some changes to its structure. To reliably adapt to different applications it is unavoidable that some form of tuning and learning should be introduced. Population based machine learning algorithms are not without precedent and include ensemble classifiers such as the "mixture of experts" system outlined in [58]. Where the DCA has the potential to stand out from such systems is its inherently asynchronous and multi-timescale nature. Traditional ensemble classifiers rely on different

classifiers being experts at specific regions of the input signal space, or families of problems. Instead, the internal structure of the classifiers throughout the population could be different. By ensuring that cells can accepting different numbers of input samples, as is the way in the existing algorithm, the multi-timescale nature of the algorithm can be retained. This could actually be achieved by simply applying a decimator to the incoming data, to condense and filter the information before sending it on to a classifier. At the time of writing the authors had not found any precedent for such a system in the literature. Such an architecture has the potential to graft the best elements of the dendritic cell algorithm onto well-established machine learning techniques.

8.6.2 Robotic Security

The field of robotic security is still in its infancy. There is little work in the literature and much of that has been identified as severely lacking in terms of practical use. Most notably, the heuristics for automatically detecting problems were found to be extremely simplistic and often required the robot to be stationary to be effective. This thesis attempts to not only define the robotic security problem, but also structure the problem into its respective sub-problems, which should allow techniques from other areas to be imported. For example, the routing problem could be approached by combining the work of Massios and Voorbraak [81–83,145] with combinatorial optimisation techniques. More advanced image-based algorithms from image processing and artificial intelligence could be imported to solve the automatic detection problem. This work also leaves the open question "Are there techniques which could take advantage of a multi-robot system, not just in terms of increasing coverage, but in terms of making more informed decisions using distally separate data sources?". It is the view of this author that the field of artificial im-

mune systems could lend a great deal to this question, by creating a self-organising network of robots, with the aim of achieving a ubiquitous maintenance solution, rather than a simple threat-detection. Such a system could control the ambient temperature, lighting and other environmental controls of a building in addition to providing security. In such a model, the human security guards would be an extension of the system, acting in an analogous way to the T Cells and B Cells of the immune system, by responding directly to threats. Inflammation models could be used to deliberately alter the sensitivity of the system in possible problem areas, and even dynamically alter the environment. For example shutting security doors to create bottle necks or changing environmental controls, (rapidly altering lighting levels, increase in ambient temperature) to disorientate or slow down intruders. However, any system which could manifest changes in the physical world would need to be thoroughly ethically explored before implementation. Even a non-lethal automated response must not be allowed to act without a human choosing whether or not such 'force' is warranted, as the programmer of such a system could never pre-empt every eventuality and any sweeping generalisations would inevitably create situations where an individual's rights could be compromised.

8.6.3 Artificial Immune Systems

This work has examined the field of artificial immune systems and specifically explored the dendritic cell algorithm. The field of AIS has traditionally focussed on models and algorithms based on the behaviour of an individual cell-type. This approach could be viewed as an extension of the cellular biology on which the field relies, where cells are often examined in isolation to ensure the validity of experiments. However, this approach is not necessarily appropriate for a chiefly computational field. Rather

than exploring the immune system from a cellular perspective, why not maintain a bottom up approach, but explore the system from a functional perspective? Questions such as "What characteristics are required for decentralised control?" and "Are there advantages to morphological diversity in systems with the functional property of homoeostasis?" could easily be explored computationally, and shed light on a broader range of topics. Such an approach seems to have a greater chance of giving information back to biologists, as it asks generic questions about how functional phenomena can be implemented, allowing hypotheses to be created about information flow and individual behaviour that are irrespective of the minutiae of their bio-chemical implementation. Thus, hypotheses can be constructed for specific biological systems, based on these established generic properties.

The artificial immune systems community has no immediate need for a 'killer-app' to make it a more main-stream topic, rather, it is uniquely situated to create models and techniques with applications in biology and engineering, rooted in the properties of complex groups of heterogeneous agents.

Bibliography

[1] *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2006, 16-21 July 2006, Vancouver* (2006), IEEE.

[2] Abbass, H. A., Sarker, R. A., and Newton, C. S., Eds. *Data Mining: A Heuristic Approach*. IGI Global, 2001.

[3] Aickelin, U., Bentley, P., Cayzer, S., Kim, J., and Mcleod, J. Danger theory: The link between AIS and IDS? In Timmis et al. [139], pp. 147–155.

[4] Al-Hammadi, Y., Aickelin, U., and Greensmith, J. DCA for bot detection. In *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI) Hong Kong* (2008), pp. 1807–1816.

[5] Al-Sheshtawi, K. A., Abdul-Kader, H. M., and Ismail, N. A. Artificial immune clonal selection algorithms: A comparative study of clonalg, opt-ia, and bca with numerical optimization problems. *International Journal of Computer Science and Network Security 10*, 4 (2010), 24–30.

[6] Andrews, P. S., Timmis, J., Owens, N. D. L., Aickelin, U., Hart, E., Hone, A., and Tyrrell, A. M., Eds. *Artificial Immune Systems, 8th International Conference, ICARIS 2009, York,*

UK, August 9-12, 2009. Proceedings (2009), vol. 5666 of *Lecture Notes in Computer Science*, Springer.

[7] Arkin, R. C. *Behaviour-Based Robotics*. MIT Press, 1998.

[8] Balthrop, J., Esponda, F., Forrest, S., and Glickman, M. Coverage and generalization in an artificial immune system. In Langdon et al. [74], pp. 3–10.

[9] Barrouil, C., Castel, C., Fabiani, P., Mampey, R., Secchi, P., and Tessier, C. A perception strategy for a surveillance system. In *the proceedings of the European Conference on Artificial Intelligence* (1998).

[10] Bayindir, L., and Sahin, E. A review of studies of swarm robotics. *Turkish Journal of Electrical Engineering 15* (2007), 115–147.

[11] Bentley, P. J., Lee, D., and Jung, S., Eds. *Artificial Immune Systems, 7th International Conference, ICARIS 2008, Phuket, Thailand, August 10-13, 2008. Proceedings* (2008), vol. 5132 of *Lecture Notes in Computer Science*, Springer.

[12] Bersini, H., and Carneiro, J., Eds. *Artificial Immune Systems, 5th International Conference, ICARIS 2006, Oeiras, Portugal, September 4-6, 2006, Proceedings* (2006), vol. 4163 of *Lecture Notes in Computer Science*, Springer.

[13] Blake, C. L., and Merz, C. J. UCI repository of machine learning databases. `http://www.ics.uci.edu/`, 1998.

[14] Braitenberg, V. *Vehicles: Experiments in Synthetic Psychology*. MIT Press, 1984.

[15] Brooks, R. A. A robust layered control system for a mobile robot. *IEEE J. Robotics and Automation* (1986), 14–23.

[16] Brooks, R. A. Elephants don't play chess. *Robotics and Autonomous Systems 6* (1990), 3–15.

[17] Burnet, F. M. *The Clonal Selection Theory of Acquired Immunity*. Cambridge University Press, 1959.

[18] Carreras, M., Ridao, P., Garcia, R., and Ursulovici, Z. Learning reactive robot behaviors with a neural-q learning approach. Internal Report, Universitat de Girona, 2002.

[19] Carroll, D., Everett, H. R., Gilbreath, G., and Mullens, K. Extending mobile security robots to force protection missions. In *AUVSI Unmanned Systems 2002, Lake Buena Vista, FL* (2002), pp. 9–11.

[20] Castelnovi, M., Miozzo, M., Scalzo, A., Piaggio, M., Sgorbissa, A., and Zaccaria, R. Surveillance robotics: analysing scenes by colours analysis and clustering. In *CIRA* (2003), pp. 229 – 234.

[21] Castro, L. N. D., and Timmis, J. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer Verlag, 2002.

[22] Castro, L. N. D., and Zuben, F. J. V. The clonal selection algorithm with engineering applications. In Langdon et al. [74], pp. 36–37.

[23] Ciccazzo, A., Conca, P., Nicosia, G., and Stracquadanio, G. An advanced clonal selection algorithm with ad-hoc network-based hypermutation operators for synthesis of topology

and sizing of analog electrical circuits. In Bentley et al. [11], pp. 60–70.

[24] Ciccimaro, D. A., Everett, H., Ciccimaro, D. A., Phillips, C. B., Everett, H. R., and Bruch, M. H. A supervised autonomous security response robot. In *American Nuclear Society, 8th Topical International Meeting On Robotics and Remote Systems, Pittsburgh, PA, 25-29 April* (1999).

[25] Cliff, D. *The Handbook of Brain Theory and Neural Networks*. MIT Press, 1995, ch. Neuroethology, Computational?, pp. 626–630.

[26] Cutello, V., Nicosia, G., and Pavone, M. Exploring the capability of immune algorithms: A characterization of hypermutation operators. In Nicosia et al. [106], pp. 263–276.

[27] Cutello, V., Nicosia, G., and Pavone, M. An immune algorithm with stochastic aging and kullback entropy for the chromatic number problem. *J. Comb. Optim. 14*, 1 (2007), 9–33.

[28] de Castro, L. N., Zuben, F. J. V., and Knidel, H., Eds. *Artificial Immune Systems, 6th International Conference, ICARIS 2007, Santos, Brazil, August 26-29, 2007, Proceedings* (2007), vol. 4628 of *Lecture Notes in Computer Science*, Springer.

[29] Doriann, D. C., Carroll, D., (csc, D. J., and Mullens, K. Automating barrier assessment with mobile security robots. In *AUVSI Unmanned Systems in International Security 2003 (USIS 03* (2003), pp. 9–12.

[30] Ebner, M., Breunig, H.-G., and Albert, J. On the use of negative selection in an artificial immune system. In Langdon et al. [74], pp. 957–964.

[31] Elberfeld, M., and Textor, J. Efficient algorithms for string-based negative selection. In Andrews et al. [6], pp. 109–121.

[32] Everett, H., Gilbreath, G., Heath-Pastore, T., and Laird, R. Controlling multiple security robots in a warehouse environment. In *AIAA/NASA Conference on Intelligent Robots* (1994).

[33] Everett, H. R., and Gage, D. W. From laboratory to warehouse: Security robots meet the real world. *Internat. J. Robotics Research 18* (1999), 760–768.

[34] Everett, H. R., Gage, D. W., Gilbreath, G., Laird, R. T., and Smurlo, R. P. Real-world issues in warehouse navigation. In *In Proceedings of the SPIE Conference on Mobile Robots IX* (1994), pp. 235–2.

[35] Flower, D. R., and Timmis, J., Eds. *in silico Immunology*. Springer, 2008.

[36] Forrest, S., Hofmeyr, S. A., Somayaji, A., and Longstaff, T. A. A sense of self for unix processes. In *In Proceedings of the 1996 IEEE Symposium on Security and Privacy* (1996), IEEE Computer Society Press, pp. 120–128.

[37] Forrest, S., Perelson, A. S., Allen, L., and Cherukuri, R. Self-nonself discrimination in a computer. In *In Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy* (1994), IEEE Computer Society Press, pp. 202–212.

[38] Freitas, A. A., and Timmis, J. Revisiting the foundations of artificial immune systems: A problem-oriented perspective. In Timmis et al. [139], pp. 229–241.

[39] Gilbreath, G., Ciccimaro, D., and Everett, H. An advanced telereflexive tactical response robot, 2000.

[40] Gordon, N. J., Salmond, D. J., and Smith, A. F. M. Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation. *Radar and Signal Processing, IEEE Proceedings F 140*, 2 (1993), 107–113.

[41] Gordon, N. J., Salmond, D. J., and Smith, A. F. M. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F 140*, 2 (August 2002), 107–113.

[42] Greensmith, J. *The Dendritic Cell Algorithm*. PhD thesis, The University of Nottingham, Computer Science, Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, 2007.

[43] Greensmith, J., and Aickelin, U. Dendritic cells for syn scan detection. In *GECCO* (2007), H. Lipson, Ed., ACM, pp. 49–56.

[44] Greensmith, J., and Aickelin, U. The deterministic dendritic cell algorithm. In Bentley et al. [11], pp. 291–303.

[45] Greensmith, J., Aickelin, U., and Cayzer, S. 'introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection'. In Jacob et al. [62], pp. 153–167.

[46] Greensmith, J., Aickelin, U., and Tedesco, G. Information fusion for anomaly detection with the dendritic cell algorithm. *Information Fusion* (To be published in 2010), 21–34.

[47] Greensmith, J., Aickelin, U., and Twycross, J. Articulation and clarification of the dendric cell algorithm. In Bersini and Carneiro [12], pp. 404–417.

[48] Greensmith, J., Feyereisl, J., and Aickelin, U. The DCA: SOMe comparison. *Evolutionary Intelligence 1*, 2 (June 2008), 85–112.

[49] Greensmith, J., Twycross, J., and Aickelin, U. Dendritic cells for anomaly detection. In *IEEE Congress on Evolutionary Computation* [1], pp. 664–671.

[50] Gu, F., Greensmith, J., and Aickelin, U. Further exploration of the dendritic cell algorithm: Antigen multiplier and time windows. In Bentley et al. [11], pp. 142–153.

[51] Gu, F., Greensmith, J., and Aickelin, U. Exploration of the dendritic cell algorithm using the duration calculus. In Andrews et al. [6], pp. 54–66.

[52] Gu, F., Greensmith, J., and Aickelin, U. Integrating real-time analysis with the dendritic cell algorithm through segmentation. In Rothlauf [122], pp. 1203–1210.

[53] Gu, F., Greensmith, J., and Aickelin, U. Integrating real-time analysis with the dendritic cell algorithm through segmentation. In Rothlauf [122], pp. 1203–1210.

[54] Gu, F., Greensmith, J., Oates, R., and Aickelin, U. PCA 4 DCA: The application of principal component analysis to the dendritic cell algorithm. In *Proceedings of the 9th Annual Workshop on Computational Intelligence (UKCI 2009)* (2009).

[55] Guzella, T. S., Mota-Santos, T. A., and Caminhas, W. M. Artificial immune systems and kernel methods. In Bentley et al. [11], pp. 303–315.

[56] Hart, E., Ross, P., Webb, A., and Lawson, A. A role for immunology in "next generation" robot controllers. In Timmis et al. [139], pp. 46–56.

[57] Hart, E., and Timmis, J. Application areas of ais: The past, the present and the future. *Applied Soft Computing 8*, 1 (2008), 191–201.

[58] Haykin, S. *Neural Networks A Comprehensive Foundation*. Prentice-Hall, 1999.

[59] Hinchey, M. G., Sterritt, R., and Rouff, C. A. Swarms and swarm intelligence. *IEEE Computer 40*, 4 (2007), 111–113.

[60] Hougen, D. F., Erickson, M. D., Rybski, P. E., Stoeter, S. A., Gini, M., and Papanikolopoulos, N. Autonomous mobile robots and distributed exploratory missions. In *in Distributed Autonomous Robotic Systems 4, Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems* (2000), Springer, pp. 221–230.

[61] Ifeachor, E., and Jervis, P. B. *Digital Signal Processing: A Practical Approach*, 2 ed. Prentice Hall, 2001.

[62] Jacob, C., Pilat, M. L., Bentley, P. J., and Timmis, J., Eds. *Artificial Immune Systems: 4th International Conference, ICARIS 2005, Banff, Alberta, Canada, August 14-17, 2005, Proceedings* (2005), vol. 3627 of *Lecture Notes in Computer Science*, Springer.

[63] Jansen, T., and Zarges, C. A theoretical analysis of immune inspired somatic contiguous hypermutations for function optimization. In Andrews et al. [6], pp. 80–94.

[64] Jerne, N. K. Towards a network theory of the immune system. *Annales d'immunologie 125C*, 1-2 (January 1974), 373–389.

[65] Jr. Ranck, J. Head direction cells in the deep cell layer of dorsal presubiculum in freely moving rats. *Society Neuroscience Abs. 10* (1984), 599.

[66] Kalman, R. E. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering 82*, Series D (1960), 35–45.

[67] Kennedy, J., and Eberhart, R. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on* (August 2002), vol. 4, pp. 1942–1948 vol.4.

[68] Kim, J., and Bentley, P. J. An evaluation of negative selection in an artificial immune system for network intrusion detection. In *GECCO* (2001), L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, Eds., Morgan Kaufmann, pp. 1330–1337.

[69] Kim, J., Bentley, P. J., Wallenta, C., Ahmed, M., and Hailes, S. Danger is ubiquitous: Detecting misbehaving nodes in sensor networks using the dendritic cell algorithm. In Bersini and Carneiro [12], pp. 390–403.

[70] Ko, A., Lau, H. Y. K., and Lee, N. M. Y. Ais based distributed wireless sensor network for mobile search and rescue robot tracking. In Bentley et al. [11], pp. 399–411.

[71] Kohonen, T. *Self-Organizing Maps*. Springer, 2000.

[72] Krautmacher, M., and Dilger, W. AIS based robot navigation in a rescue scenario. In Nicosia et al. [106], pp. 106–118.

[73] Laird, R. T., Everett, H. R., Gilbreath, G., Heath-pastore, T. A., Inderieden, R. S., and Grant, K. Mdars multiple robot host architecture. In *Association of Unmanned Vehicle Systems, 22nd Annual Symposium and Exhibition (AUVS '95* (1995), pp. 10–12.

[74] Langdon, W. B., Cantú-Paz, E., Mathias, K. E., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M. A., Schultz, A. C., Miller, J. F., Burke, E. K., and Jonoska, N., Eds. *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA, 9-13 July 2002* (2002), Morgan Kaufmann.

[75] Lau, H., Bate, I., and Timmis, J. An immuno-engineering approach for anomaly detection in swarm robotics. In Andrews et al. [6], pp. 136–150.

[76] Lau, H. Y. K., and Ko, A. An immuno robotic system for humanitarian search and rescue (application stream). In de Castro et al. [28], pp. 191–203.

[77] Lay, N., and Bate, I. Applying artificial immune systems to real-time embedded systems. In *IEEE Congress on Evolutionary Computation* (2007), IEEE, pp. 3743–3750.

[78] Lay, N., and Bate, I. Improving the reliability of real-time embedded systems using innate immune techniques. *Evolutionary Intelligence 1*, 2 (2008), 113–132.

[79] Lee, J., Roh, M., Lee, J., and Lee, D. Clonal selection algorithms for 6-dof pid control of autonomous underwater vehicles. In de Castro et al. [28], pp. 182–190.

[80] Martins-Filho, L. S., and Macau, E. E. N. Trajectory planning for surveillance missions of mobile robots. *Studies in Computational Intelligence 76* (August 2007), 109–117.

[81] Massios, N., and Voorbraak, F. Hierarchical decision-theoretic robotic surveillance. In *IJCAI'99 Workshop on Reasoning with Uncertainty in Robot Navigation* (1999), pp. 219–226.

[82] Massios, N. A. *Decision-Theoretic Robotic Surveillance*. PhD thesis, Institute for Logic, Language and Computation,Universiteit van Amsterdam, 2002.

[83] Massios, N. A., and Voorbraak, F. Planning strategies for decision-theoretic robotic surveillance. In *In NAIC'98* (1998), pp. 117–126.

[84] Matzinger, P. Tolerance, danger, and the extended family. *Annual review of immunology 12*, 1 (1994), 991–1045.

[85] Matzinger, P. Friendly and dangerous signals: is the tissue in control? *Nature Immunology 8*, 1 (2008), 11–13.

[86] McEwan, C., Hart, E., and Paechter, B. Boosting the immune system. In Bentley et al. [11], pp. 316–327.

[87] McNaughton, B. L., Barnes, C. A., and O'Keefe, J. The contributions of position, direction, and velocity to single unit activity in the hippocampus of freely-moving rats. *Experimental Brain Research 52*, 1 (1982), 41–49.

[88] Medzhitov, R., and Janeway, C. Decoding the patterns of self and nonself by the innate immune system. *Science 296*, 5566 (2002), 298–300.

[89] Mikic, I., Santini, S., and Jain, R. Video processing and integration from multiple cameras. In *In Proceedings of the 1998 Image Understanding Workshop, Morgan-Kaufman* (1998), Cambridge University Press, pp. 183–187.

[90] Milford, M. Featureless vehicle-based slam with a consumer camera. In *the proceedings of the Australasian Conference on Robotics and Automation* (2007).

[91] Milford, M., Prasser, D., and Wyeth, G. Experience mapping: Producing spatially continuous environment representations using ratslam. In *the proceedings of the Australasian Conference on Robotics and Automation* (2005).

[92] Milford, M., and Wyeth, G. Spatial mapping and map exploitation: A bio-inspired engineering perspective. In *the proceedings of the Conference on Spatial Information Theory* (2007), pp. 203–221.

[93] Milford, M., Wyeth, G., and Prasser, D. Ratslam: a hippocampal model for simultaneous localization and mapping. In *the proceedings of the International Conference on Robotics and Automation* (2004), pp. 403–408.

[94] Milford, M., Wyeth, G., and Prasser, D. Simultaneous localization and mapping from natural landmarks using ratslam. In *the proceedings of the Australasian Conference on Robotics and Automation* (2004).

[95] Milford, M., Wyeth, G., and Prasser, D. Ratslam on the edge: Revealing a coherent representation from an overloaded rat brain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2006), pp. 4060–4065.

[96] Milford, M. J., and Wyeth, G. Hippocampal models for simultaneous localisation and mapping on an autonomous robot. In *the proceedings of the Australasian Conference on Robotics and Automation* (2003).

[97] Milford, M. J., and Wyeth, G. Mapping a suburb with a single camera using a biologically inspired SLAM system. *IEEE Transactions on Robotics* (2008), 1038–1053.

[98] Milford, M. J., and Wyeth, G. Single camera vision-only slam on a suburban road network. In *the proceedings of the International Conference on Robotics and Automation* (2008), pp. 3684 – 3689.

[99] Milford, M. J., and Wyeth, G. Single camera vision-only SLAM on a suburban road network. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation* (2008), pp. 3684 – 3689.

[100] Minsky, M., and Papert, S. *Perceptrons*. MIT Press, 1969.

[101] Mokhtar, M., Bi, R., Timmis, J., and Tyrrell, A. M. A modified dendritic cell algorithm for on-line error detection in robotic systems. In *IEEE Congress on Evolutionary Computation* (2009), IEEE, pp. 2055–2062.

[102] Moravec, H. *Obstacle avoidance and navigation in the real world by a seeing robot rover*. PhD thesis, Stanford University, 1980.

[103] Neal, M., and Timmis, J. *Recent Developments in Biologically Inspired Computing (Leandro Nunes de Castro and Fernando J. Von Zuben eds)*. Idea Group, 2005, ch. Once more unto the breach: towards artificial homeostasis, pp. 340 – 366.

[104] Newborough, J., and Stepney, S. A generic framework for population-based algorithms, implemented on multiple fpgas. In Jacob et al. [62], pp. 43–55.

[105] Nicosia, G. *Immune Algorithms for Optimization and Protein Structure Prediction.* PhD thesis, University of Catania, Department of Mathematics and Computer Science, University of Catania, Italy, 2004.

[106] Nicosia, G., Cutello, V., Bentley, P. J., and Timmis, J., Eds. *Artificial Immune Systems, Third International Conference, ICARIS 2004, Catania, Sicily, Italy, September 13-16, 2004* (2004), vol. 3239 of *Lecture Notes in Computer Science*, Springer.

[107] Oates, R., Greensmith, J., Aickelin, U., Kendall, G., and Garibaldi, J. M. The application of a dendritic cell algorithm to a robotic classifier. In de Castro et al. [28], pp. 204–215.

[108] Oates, R., Kendall, G., and Garibaldi, J. M. Frequency analysis for dendritic cell population tuning. *Evolutionary Intelligence 1*, 2 (June 2008), 145–157.

[109] Oates, R., Kendall, G., and Garibaldi, J. M. The limitations of frequency analysis for dendritic cell population modelling. In Bentley et al. [11], pp. 328–339.

[110] Oates, R., Milford, M., Wyeth, G., Kendall, G., and Garibaldi, J. M. The implementation of a novel, bio-inspired, robotic security system. In *The Proceedings of the 2009 IEEE Conference on Robotics and Automation* (2009).

[111] O'Keefe, J., and Dostrovsky, J. The hippocampus as a spatial map: preliminary evidence from unit activity in the freely moving rat. *Brain Research 34* (1971), 171–175.

[112] O'Rourke, J. *Art gallery theorems and algorithms*. Oxford University Press, Inc., New York, NY, USA, 1987.

[113] Parker, L. E. L-alliance: Task-oriented multi-robot learning in behavior-based systems. *Advanced Robotics 11*, 4 (1996), 305–322.

[114] Parker, L. E. Alliance: An architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation 14* (1998), 220–240.

[115] Pastore, T., Everett, H., and Bonner, K. Mobile robots for outdoor security applications. In *American Nuclear Society 8th International Topical Meeting on Robotics and Remote Systems (ANS'99)* (1999).

[116] Perelson, A. S., and Weisbuch, G. R. Immunology for physicists. *Reviews of Modern Physics 69* (1997), 1219–1267.

[117] Pfeifer, R., and Scheier, C. *Understanding intelligence*. MIT Press, Cambridge, MA, USA, 1999.

[118] Poli, R. Analysis of the publications on the applications of particle swarm optimisation. *J. Artif. Evol. App. 2008* (2008), 1–10.

[119] Poli, R., Kennedy, J., and Blackwell, T. Particle swarm optimization. *Swarm Intelligence 1*, 1 (June 2007), 33–57.

[120] Prasser, D., Wyeth, G., and Milford, M. Biologically inspired visual landmark processing for simultaneous localization and mapping. In *the proceedings of the International Conference on Intelligent Robots and Systems* (2004), pp. 730–735.

[121] Prasser, D. P., Wyeth, G. F., and Milford, M. J. Experiments in outdoor operation of ratslam. In *the proceedings of the Australasian Conference on Robotics and Automation* (2004).

[122] Rothlauf, F., Ed. *Genetic and Evolutionary Computation Conference, GECCO 2009, Proceedings, Montreal, Québec, Canada, July 8-12, 2009* (2009), ACM.

[123] Rybski, P. E., Stoeter, S. A., Erickson, M. D., Gini, M., Hougen, D. F., and Papanikolopoulos, N. A team of robotic agents for surveillance. In *In Proc. of the Int'l Conf. on Autonomous Agents* (2000), pp. 9–16.

[124] Secker, A., Davies, M. N., Freitas, A. A., Timmis, J., Clark, E., and Flower, D. R. An artificial immune system for evolving amino acid clusters tailored to protein function prediction. In Bentley et al. [11], pp. 242–253.

[125] Sharkey, A. J. C. Swarm robotics and minimalism. *Connect. Sci 19*, 3 (2007), 245–260.

[126] Sharkey, N. E., and Heemskerk, J. N. The neural mind and the robot. In *Neural Network Perspectives on Cognition and Adaptive Robotics* (1996), IOP Press, pp. 169–194.

[127] Shi, Y., Evans, J., and Rock, K. Molecular identification of a danger signal that alerts the immune system to dying cells. *Nature 425*, 6957 (2003 Oct 2), 516–21.

[128] SIA. The security industry authority annual report and accounts. Available at http://www.the-sia.org.uk/, 2005-2006.

[129] Smith, R., Self, M., and Cheeseman, P. Estimating uncertain spatial relationships in robotics. *Autonomous robot vehicles* (1990), 167–193.

[130] Stibor, T., Mohr, P. H., Timmis, J., and Eckert, C. Is negative selection appropriate for anomaly detection ? In *GECCO* (2005), H.-G. Beyer and U.-M. O'Reilly, Eds., ACM, pp. 321–328.

[131] Stibor, T., Oates, R., Kendall, G., and Garibaldi, J. M. Geometrical insights into the dendritic cell algorithm. In Rothlauf [122], pp. 1275–1282.

[132] Stibor, T., Timmis, J., and Eckert, C. Artificial immune systems for it-security. *it-Information Technology (Systems Biology and Information Technology) 48*, 3 (2006), 168–173.

[133] Stibor, T., Timmis, J., and Eckert, C. On permutation masks in hamming negative selection. In Bersini and Carneiro [12], pp. 122–135.

[134] Strang, G., and Nguyen, T. *Wavelets and Filter Banks*. Wellesley College, 2009.

[135] Swain, M., and Ballard, D. Color indexing. *International Journal of Computer Vision 7*, 1 (1991).

[136] Taube, J., Muller, R., and Jr, J. R. Head-direction cells recorded from the postsubiculum in freely moving rats. i. description and quantitative analysis. *Journal of Neuroscience 10* (1990), 420–435.

[137] Tickner, A. H., and Poulton, E. Monitoring up to 16 synthetic television pictures showing a great deal of movement,. *Ergonomics 14*, 4 (1973).

[138] Timmis, J., Andrews, P., Owens, N., and Clark, E. An immune algorithm with stochastic aging and kullback entropy for the

chromatic number problem. *Journal of Evolutionary Intelligence 1*, 1 (2008), 5–26.

[139] Timmis, J., Bentley, P. J., and Hart, E., Eds. *Artificial Immune Systems, Second International Conference, ICARIS 2003, Edinburgh, UK, September 1-3, 2003, Proceedings* (2003), vol. 2787 of *Lecture Notes in Computer Science*, Springer.

[140] Timmis, J., Edmonds, C., and Kelsey, J. Assessing the performance of two immune inspired algorithms and a hybrid genetic algorithm for function optimisation. In *In Proceedings of the Congress on Evolutionary Computation* (2004), IEEE, pp. 1044–1051.

[141] Timmis, J., Hart, E., Hone, A., Neal, M., Robins, A., Stepney, S., and Tyrrell, A. Immuno-engineering. In *2nd IFIP International Conference on Biologically Inspired Collaborative Computing* (2008), IEEE Press, pp. 3–17.

[142] Timmis, J., Hone, A., Stibor, T., and Clark, E. Theoretical advances in artificial immune systems. *Theor. Comput. Sci. 403*, 1 (2008), 11–32.

[143] Timmis, J., Neal, M., and Thorniley, J. An adaptive neuro-endocrine system for robotic systems. In *IEEE Workshops on Computational Intelligence* (2009), pp. 129–136.

[144] Twycross, J., and Aickelin, U. Libtissue - implementing innate immunity. In *IEEE Congress on Evolutionary Computation* [1], pp. 499–506.

[145] Voorbraak, F., and Massios, N. A. Decision-theoretic planning for autonomous robotic surveillance. In *University of Brighton* (1998), pp. 23–32.

[146] Warwick, K. *An Introduction to Control Systems*. World Scientific, 1996.

[147] Watanabe, Y., Ishiguro, A., Shirai, Y., and Uchikawa, Y. Emergent construction behavior arbitration mechanism based on the immune system. In *ICEC '98: In Proceedings of the IEEE International Conference on Evolutionary Computation* (1998), pp. 481–486.

[148] Watson, A. J., and Lovelock, J. E. Biological homeostasis of the global environment: the parable of daisyworld. *Tellus Series B Chemical and Physical Meteorology B 35* (1983).

[149] Whitbrook, A. M., Aickelin, U., and Garibaldi, J. M. An idiotypic immune network as a short term learning architecture for mobile robots. *CoRR abs/0910.3115* (2009).

[150] Willeke, T., Kunz, C., and Nourbakhsh, I. The history of the mobot museum robot series: An evolutionary study. In *in Proceedings of FLAIRS 2001* (2001).

[151] Wilson, D. L. *Introduction to Biology*. WileyBlackwell, 1999.

[152] Wilson, H. R. *Spikes, Decisions, and Actions: The Dynamical Foundations of Neuroscience*. Oxford University Press, 1999.

[153] Young, S. S. *Computerized Data Acquisition and Analysis for the Life Sciences: A Hands-on Guide*. Cambridge University Press, 2001.

[154] Ziegler, J., and Nichols, N. Optimal settings for automatic controllers. In *Trans. ASME 64* (1942), pp. 759–768.

[155] Ziemke, T. Are robots embodied? In *Lund University Cognitive Studies* (2001), pp. 75–83.

Appendix A

Pseudocode for Common AIS Algorithms

A.1 Negative Selection

Adapted from [38]

Input:

```
S, the set of 'normal' examples
a(x,y), an affinity (similarity) function between x and y
T, a user defined threshold for similarity
```

Output:

```
P, a set of 'mature' T-Cells that do not match any example in
S
```

```
WHILE (stopping criterion not met)
   t := Random 'immature' T-Cell;
   FOR (every example s, in S)
      Evaluate a(t,s);
   END;
   IF (a(t,s) > T for any s)
      THEN discard t;
      ELSE add t to P;
```

A.2 opt-AINET

Taken from [124]

Input:

```
g(x), the objective function
T, Threshold of what constitutes an improvement  Output:
P, a population of candidate solutions

P := Randomly generated cells;
WHILE (stopping criterion not met)
   FOR (each cell c in P)
      Evaluate g(c);
      Add Clone(c) to the population;
      Mutate(c) based on the fitness of its parent;
      Determine the fitness of all new clones;
   END;
   FOR (each parent cell, p in P)
      Select fittest clone for survival;
   END;
   G_n := Average fitness of the population;
   IF (G_n - G_{n-1} > T)
      THEN Restart loop;
      ELSE
      Remove least fit cells from population;
      Replace cells with randomly generated cells
```

A.3 Clonalg

Adapted from [5]

Input:

Ab, the initial population of antibodies

Ngen, the maximum number of generations,

n, number of antibodies to be cloned

d, the number of antibodies to be removed and replaced

L, Length of the clone receptor

β, mutation rate

decode(x), a function which calculates the solution and evaluates its quality

Output:

Ab, A population of antibodies representing candidate solutions

f, A series of values in the range 0-1 representing the quality of the solutions

```
FOR (t = 1 to Ngen)
   f := decode(Ab);
   Ab_n := select(Ab, f, n);
   C := clone(Ab_n, β, f);
   C* := hypermut (C, f);
   f* := decode(C*);
   Abn := select(C*, f*, n);
   Ab := insert (Ab, Ab_n);
   Abd := generate(d, L); (Randomly generate d antibodies of length
L)
   Ab := replace(Ab, Abd, f);
END;
```

```
f := decode(Ab);
```

A.4 The B Cell Algorithm

Adapted from [5]

Input:

```
g(x), the objective function
s(x), the contiguous somatic hypermutation function
```

Output:

```
P, A population of candidate solutions
```

```
WHILE(stopping criterion not met)
   P := Random population of individuals
   FOR (each cell, v, in P)
      evaluate g(v);
      v' := clone(v);
      add v' to clonal pool C;
   END;
   c := Random clone from C;
   Randomly change each element of c;
   FOR (each clone c, in C)
      c := s(c);
      evaluate g(c);
      IF (c has higher affinity than its parent, v)
         THEN v := c;
   END;
```

A.5 opt-IA

Adapted from [27]

Input:

l, the length of the B-Cell representation

d, the size of the B-Cell population

dup, the number of clones to make per static clone operation

τ_B, the maximum age of a cell

c, constant controlling the behaviour of the hypermutation operator

g(x), objective function

Output:

P, a population of B-Cells representing candidate solutions

NOTE:

InverseHypermutation(P,c,l) is a function which mutates a number of elements in each member of population P, inversely proportionally to the fitness of that member

```
P := Randomly Generated Population;
Evaluate g(P);
WHILE(stopping criterion not met)
   P' := StaticClone(P);
   P' := InverseHypermutation(P',c,l);
   Evaluate g(P');
   P' = Age(P',τ_B);
   P = Fittest, Unique Survivors of (P');
```

Appendix B

Pseudocode for DCA Versions Used

B.1 Data Structures

While many data structures have been used to implement the DCA the fastest technique has proved to be using a sequence of arrays to represent the cell population. In each case an array of doubles can be used to store the intermediate variables, (the current state of CSM, IL10, 1L12 and/or K) and an array of integers can be used to keep track of which antigen have been detected by the algorithm.

B.2 Version 0.0

Introduced in: [45]

Used in: [45]

Signal Weights:

PAMP-CSM = 2

PAMP-IL10 = 0

PAMP-IL12 = 2

Danger-CSM = 1

Danger-IL10 = 0

Danger-IL12 = 1

Safe-CSM = 2

Safe-IL10 = 3

Safe-IL12 = -3

Input:

P_t a heuristic providing a time-varying PAMP signal from 0-1

D_t a heuristic providing a time-varying Danger signal from 0-1

S_t a heuristic providing a time-varying Safe signal from 0-1

$\mathbf{A}_t$ a heuristic providing a time-varying vector of antigen identifiers

Output:

A_{id}, a list of antigen identifiers with no duplications

A_C, a list of binary classifications for the values in A_{id}

Algorithm:

```
P := 100 New Cells;
FOR (every cell c in the population)
   Migration Threshold[c] = 10;
END;
WHILE (Update(P_t, D_t, S_t, A_t) still has data)
   FOR (every antigen a in A_t)
      IF (a never encountered before)
         THEN Add a to A_id;
      P_s := A subset of 10 random cells in P;
      FOR (every cell c in P_s)
         Store a;
         Calculate CSM;
         Calculate IL10;
         Calculate IL12;
         IF (CSM > 10)
            THEN
               Remove c from P;
               IF (IL10 > IL12)
                  THEN Vote 0 for all stored antigen in c;
                  ELSE Vote 1 for all stored antigen in c;
               Replace c with a new cell with Migration=10;
      END;
   END;
END;
FOR (each antigen a in A_id)
   IF (more 0 votes than 1 votes for a)
      THEN A_C entry for a := 0;
```

```
ELSE A_C entry for a := 1;
```

B.3 Version 1.0

Introduced in: [49]

Used in: [49]

Signal Weights:

PAMP-CSM = 2

PAMP-IL10 = 0

PAMP-IL12 = 2

Danger-CSM = 1

Danger-IL10 = 0

Danger-IL12 = 1

Safe-CSM = 2

Safe-IL10 = 3

Safe-IL12 = -3

Input:

P_t a heuristic providing a time-varying PAMP signal from 0-1

D_t a heuristic providing a time-varying Danger signal from 0-1

S_t a heuristic providing a time-varying Safe signal from 0-1

I_t a heuristic providing a time-varying Inflammation signal 0-2

$\mathbf{A}_t$ a heuristic providing a time-varying vector of antigen identifiers

R a range specifying the upper and lower bounds for the migration thresholds

P_n the number of cells in the population

Output:

A_{id}, a list of antigen identifiers with no duplications

A_C, a list of binary classifications for the values in A_{id}

Algorithm:

```
P := P_n New Cells;
FOR (every cell c in the population)
   Migration Threshold[c] = RandomValue(R);
END;
WHILE (Update(P_t, D_t, S_t, I_t, A_t) still has data)
   P_t := I_t*P_t;
   D_t := I_t*D_t;
   S_t := I_t*S_t;
   FOR (every antigen a in A_t)
      IF (a never encountered before)
         THEN Add a to A_id;
      Store a in a random cell from P;
   END;
   FOR (every cell c in the population P)
      Calculate CSM;
      Calculate IL10;
      Calculate IL12;
      IF (CSM > Migration Threshold[c])
         THEN
            Remove c from P;
            IF (IL10 > IL12)
               THEN Vote 0 for all antigen stored in c;
               ELSE Vote 1 for all antigen stored in c;
            Replace c with a new cell with Mig = RandomValue(R);
      END;
   END;
END;
```

```
FOR (each antigen a in A_id)
   IF (more 0 votes than 1 votes for a)
      THEN A_C entry for a := 0;
      ELSE A_C entry for a := 1;
```

B.4 Version 2.0

Introduced in: [47]

Used in: [46, 47, 50]

Signal Weights:

PAMP-CSM = 2

PAMP-IL10 = 0

PAMP-IL12 = 2

Danger-CSM = 1

Danger-IL10 = 0

Danger-IL12 = 1

Safe-CSM = 2

Safe-IL10 = 3

Safe-IL12 = -3

Input:

P_t a heuristic providing a time-varying PAMP signal from 0-1

D_t a heuristic providing a time-varying Danger signal from 0-1

S_t a heuristic providing a time-varying Safe signal from 0-1

$\mathbf{A}_t$ a heuristic providing a time-varying vector of antigen identifiers

R a range specifying the upper and lower bounds for the migration thresholds

P_n the number of cells in the population

Output:

A_{id}, a list of antigen identifiers with no duplications

A_{MCAV}, a list of gradiated classifications for the values in A_{id} in the range 0-1

Algorithm:

```
P := P_n New Cells;
FOR (every cell c in the population)
   Migration Threshold[c] = RandomValue(R);
END;
WHILE (Update(P_t, D_t, S_t, A_t) still has data)
   FOR (every antigen a in A_t)
      IF (a never encountered before)
         THEN Add a to A_id;
      Store a in a random cell from P;
   END;
   FOR (every cell c in the population P)
      Calculate CSM;
      Calculate IL10;
      Calculate IL12;
      IF (CSM > Migration Threshold[c])
         THEN
            Remove c from P;
            IF (IL10 > IL12)
               THEN Vote 0 for all antigen stored in c;
               ELSE Vote 1 for all antigen stored in c;
            Replace c with new cell with Mig = RandomValue(R);
      END;
   END;
END;
FOR (each antigen a in A_id)
   Entry for a in A_MCAV := (Number of 1 votes for a/ Total number
of votes for a);
```

B.5 Version 2.1

Introduced in: [43]

Used in: [43, 48]

Signal Weights:

PAMP-CSM = 2

PAMP-IL10 = 0

PAMP-IL12 = 2

Danger-CSM = 1

Danger-IL10 = 0

Danger-IL12 = 1

Safe-CSM = 2

Safe-IL10 = 3

Safe-IL12 = -3

Input:

P_t a heuristic providing a time-varying PAMP signal from 0-1

D_t a heuristic providing a time-varying Danger signal from 0-1

S_t a heuristic providing a time-varying Safe signal from 0-1

I_t a heuristic providing a binary time-varying Inflammation signal

$\mathbf{A}_t$ a heuristic providing a time-varying vector of antigen identifiers

R a range specifying the upper and lower bounds for the migration thresholds

P_n the number of cells in the population

Output:

A_{id}, a list of antigen identifiers with no duplications

A_{MCAV}, a list of gradiated classifications for the values in

A_{id} in the range 0-1

Algorithm:

```
P := P_n New Cells;
FOR (every cell c in the population)
   Migration Threshold[c] = RandomValue(R);
END;
WHILE (Update(P_t, D_t, S_t, I_t, A_t) still has data)
   IF (I_t == true)
      THEN
         P_t = 2*P_t;
         D_t = 2*D_t;
         S_t = 2*S_t;
   FOR (every antigen a in A_t)
      IF (a never encountered before)
         THEN Add a to A_id;
      Store a in a random cell from P;
   END;
   FOR (every cell c in the population P)
      Calculate CSM;
      Calculate IL10;
      Calculate IL12;
      IF (CSM > Migration Threshold[c])
         THEN
            Remove c from P;
            IF (IL10 > IL12)
               THEN Vote 0 for all antigen stored in c;
               ELSE Vote 1 for all antigen stored in c;
            Replace c with new cell with Mig = RandomValue(R);
```

```
      END;
   END;
END;
FOR (each antigen a in A_id)
   Entry for a in A_MCAV := (Number of 1 votes for a/ Total number
of votes for a);
```

B.6 Version 2.2

Introduced in: [4]

Used in: [4]

Signal Weights:

PAMP-CSM = 4

PAMP-IL10 = 0

PAMP-IL12 = 8

Danger-CSM = 2

Danger-IL10 = 0

Danger-IL12 = 4

Safe-CSM = 3

Safe-IL10 = 1

Safe-IL12 = -6

Input:

P_t a heuristic providing a time-varying PAMP signal from 0-1

D_t a heuristic providing a time-varying Danger signal from 0-1

S_t a heuristic providing a time-varying Safe signal from 0-1

$\mathbf{A}_t$ a heuristic providing a time-varying vector of antigen identifiers

R a range specifying the upper and lower bounds for the migration thresholds

P_n the number of cells in the population

Output:

A_{id}, a list of antigen identifiers with no duplications

A_{MAC}, a list of gradiated classifications for the values in A_{id} in the range 0-1

Algorithm:

```
P := P_n New Cells;
FOR (every cell c in the population)
   Migration Threshold[c] = RandomValue(R);
END;
WHILE (Update(P_t, D_t, S_t, A_t) still has data)
   FOR (every antigen a in A_t)
      IF (a never encountered before)
         THEN Add a to A_id;
      Store a in a random cell from P;
   END;
   FOR (every cell c in the population P)
      Calculate CSM;
      Calculate IL10;
      Calculate IL12;
      IF (CSM > Migration Threshold[c])
         THEN
            Remove c from P;
            IF (IL10 > IL12)
               THEN Vote 0 for all antigen stored in c;
               ELSE Vote 1 for all antigen stored in c;
            Replace c with new cell with Mig = RandomValue(R);
      END;
   END;
END;
FOR (each antigen a in A_id)
   Entry for a in A_MCAV := (Number of 1 votes for a/ Total number
of votes for a);
```

Entry for a in A_{MAC} := ((Entry for a in A_{MCAV})*(Total Votes for a))/(Total Antigen Processed)

B.7 Version 3.0

Introduced in: [107]

Used in: [107, 108, 110]

NOTE: In [108, 110] the value of the input heuristic $\mathbf{X}_t$ is set permanently to 1.

Signal Weights:

PAMP-CSM = 2

PAMP-K = 2

Danger-CSM = 1

Danger-K = 1

Safe-CSM = 2

Safe-K = -6

Input:

$\mathtt{P}_t$ a heuristic providing a time-varying PAMP signal from 0-1

$\mathtt{D}_t$ a heuristic providing a time-varying Danger signal from 0-1

$\mathtt{S}_t$ a heuristic providing a time-varying Safe signal from 0-1

$\mathbf{A}_t$ a heuristic providing a time-varying vector of antigen identifiers

$\mathtt{X}_t$ a heuristic providing a time-varying antigen multiplier from 1-102

R a range specifying the upper and lower bounds for the migration thresholds

$\mathtt{P}_n$ the number of cells in the population

SegLim the number of samples to take before calculating MCAV values

Output:

$\mathtt{A}_{id,t}$, a time-varying list of antigen identifiers with no duplications

$A_{MCAV,t}$, a time-varying list of gradiated classifications for the values in A_{id} in the range 0-1

Algorithm:

```
P := P_n New Cells;
FOR (every cell c in the population)
   Migration Threshold[c] = RandomValue(R);
END;
count := 0;
WHILE (true)
   count = count + 1;
   Update(P_t, D_t, S_t, X_t, A_t)
   FOR (i = 1 to X_t)
      FOR (every antigen a in A_t)
         IF (a never encountered before)
            THEN Add a to A_id;
         Store a in a random cell from P;
      END;
   END;
      Calculate CSM;
      Calculate K;
   FOR (every cell c in the population P)
      IF (CSM > Migration Threshold[c])
         THEN
            Remove c from P;
            IF (IL10 > IL12)
               THEN Vote 0 for all antigen stored in c;
               ELSE Vote 1 for all antigen stored in c;
            Replace c with new cell with Mig = RandomValue(R);
```

```
   END;
   IF (count == SegLim)
      THEN
         FOR (each antigen a in A_{id,t})
            Entry for a in A_{MCAV,t} := (Number of 1 votes for a/
Total number of votes for a);
         END;
         Output(A_{id,t},A_{MCAV,t});
         Reset(A_{id,t},A_{MCAV,t}, Votes);

   END;
```

B.8 Version 4.0

Introduced in: [44]

Used in: [44, 131]

Signal Weights:

Danger-CSM = 1

Danger-K = 1

Safe-CSM = 1

Safe-K = -2

Input:

D_t a heuristic providing a time-varying Danger signal from 0-1

S_t a heuristic providing a time-varying Safe signal from 0-1

$\mathbf{A}_t$ a heuristic providing a time-varying vector of antigen identifiers

R a range specifying the upper and lower bounds for the migration thresholds

P_n the number of cells in the population

Output:

A_{id}, a list of antigen identifiers with no duplications

$A_{K\alpha}$, a list of gradiated classifications for the values in A_{id}. Higher magnitudes indicate high certainty, negative values indicate 'normal' and positive 'safe'

Algorithm:

```
P := P_n New Cells;
FOR (every cell c in the population)
   Migration Threshold[c] = A uniform distribution of R;
```

```
END;
P_counter = first cell;
WHILE (Update(D_t, S_t, A_t) still has data)
   FOR (every antigen a in A_t)
      IF (a never encountered before)
         THEN Add a to A_id;
      Store a in the cell specified by P_counter;
      P_counter = next cell;    END;
   Calculate system wide CSM;
   Calculate system wide K;
   FOR (every cell c in the population P)
      IF (CSM > Migration Threshold[c])
         THEN
            Remove c from P;
            Store K against all antigen stored in c;
            Replace c with new cell with the same migration as
original
      END;
   END;
END;
FOR (each antigen a in A_id)
   Entry for a in A_Kα := Average K value reported for each antigen
type
```

B.9 Version 4.1

Introduced in: [53]

Used in: [53]

Signal Weights:

PAMP-CSM = 2

PAMP-K = 2

Danger-CSM = 1

Danger-K = 1

Safe-CSM = 1

Safe-K = -2

Input:

P_t a heuristic providing a time-varying PAMP signal from 0-1

D_t a heuristic providing a time-varying Danger signal from 0-1

S_t a heuristic providing a time-varying Safe signal from 0-1

$\mathbf{A}_t$ a heuristic providing a time-varying vector of antigen identifiers

R a range specifying the upper and lower bounds for the migration thresholds

P_n the number of cells in the population

Output:

A_{id}, a list of antigen identifiers with no duplications

$A_{K\alpha,t}$, a list of gradiated classifications for the values in A_{id}. Higher magnitudes indicate high certainty, negative values indicate 'normal' and positive 'safe'

Algorithm:

```
P := P_n New Cells;
FOR (every cell c in the population)
   Migration Threshold[c] = A uniform distribution of R;
END;
P_counter = first cell;
count := 0;
WHILE (true)
   Update(P_t, D_t, S_t, A_t)
   FOR (every antigen a in A_t)
      IF (a never encountered before)
         THEN Add a to A_id;
      Store a in the cell specified by P_counter;
      P_counter = next cell;    END;
   Calculate system wide CSM;
   Calculate system wide K;
   FOR (every cell c in the population P)
      IF (CSM > Migration Threshold[c])
         THEN
            Remove c from P;
            Store K against all antigen stored in c;
            Replace c with new cell with the same migration as
original;
      END;
   END;
   IF (count == SegLim)
      THEN
         FOR (each antigen a in A_id,t)
            Entry for a in A_Kα,t := Average K value reported for
each antigen type
```

```
        END;
        Output(A_{id,t},A_{K\alpha,t});
        Reset(A_{id,t},A_{K\alpha,t}, Stored K values);
END;
```

B.10 Version 4.2

Introduced in: [54]

Used in: [54]

Signal Weights:

PAMP-CSM = 2

PAMP-K = 2

Danger-CSM = 1

Danger-K = 1

Safe-CSM = 1

Safe-K = -2

Input:

P_t a heuristic providing a time-varying PAMP signal from 0-1

D_t a heuristic providing a time-varying Danger signal from 0-1

S_t a heuristic providing a time-varying Safe signal from 0-1

X_t a heuristic providing a time-varying antigen multiplication signal between 15 and 100

$\mathbf{A}_t$ a heuristic providing a time-varying vector of antigen identifiers

R a range specifying the upper and lower bounds for the migration thresholds

P_n the number of cells in the population

Output:

A_{id}, a list of antigen identifiers with no duplications

$A_{K\alpha,t}$, a list of gradiated classifications for the values in A_{id}. Higher magnitudes indicate high certainty, negative values indicate 'normal' and positive 'safe'

Algorithm:

```
P := P_n New Cells;
FOR (every cell c in the population)
   Migration Threshold[c] = A uniform distribution of R;
END;
P_counter = first cell;
count := 0;
WHILE (true)
   Update(P_t, D_t, S_t, X_t, A_t)
   FOR (every antigen a in A_t)
      IF (a never encountered before)
         THEN Add a to A_id;
      FOR (i = 1 to X_t)
         Store a in the cell specified by P_counter;
         P_counter = next cell;       END;
   END;
   Calculate system wide CSM;
   Calculate system wide K;
   FOR (every cell c in the population P)
      IF (CSM > Migration Threshold[c])
         THEN
            Remove c from P;
            Store K against all antigen stored in c;
            Replace c with new cell with the same migration as
original;
      END;
   END;
   IF (count == SegLim)
```

```
      THEN
         FOR (each antigen a in A_{id,t})
            Entry for a in A_{Kα,t} := Average K value reported for
each antigen type
         END;
         Output(A_{id,t},A_{Kα,t});
         Reset(A_{id,t},A_{Kα,t}, Stored K values);
END;
```

Appendix C

Specification of the Pioneer P3-DX Robot

C.1 Robot Base

Manufacturer: MobileRobots Inc. Length: 45cm

Width: 40cm

Height without accessories: 24.5

Ground Clearance: 6.5cm

Weight (with min. battery capacity): 9kg

Rated Payload of base platform (with included battery): 17kg (flat surface), 10kg (@ 13% grade)

Absolute Maximum Payload of base platform (with included battery):23kg (flat surface), 14kg (@ 13% grade)

Body: Powder-coated 1.6mm aluminium

IP Rating: 20

Operating Temperature Range: 0^{o}C to 35^{o}C

Battery Voltage: 12V

Battery Capacity: 7Ah per battery

Battery Chemistry: Sealed lead-acid

Battery Access: Hinged, latched access door (hot swappable)

Continuous run time, (base platform, new batteries): 8hrs (no laser or computer)

Full Recharge time: 2.4 (high cap charger), 12 (standard charger)

Docking station available: Yes

Drive: 2-wheel differential drive, with rear balancing caster

Wheel composition: Foam filled nylon, hard casters

Drive Wheel Diameter: 20cm

Drive Wheel Width: 5cm

Pushing force: 6kg

Swing radius: 27cm, (32cm without bumpers)

Max Translation Speed: 1.2 ms^{-1}

Max Traversable Step: 2.5cm

Max Traversable Gap: 5cm

Max Traversable Slope: 25%

Traversable Terrains: Wheelchair accessible

C.2 Sensors

Note, the Pioneer 3DX has several possible sensor configurations, the configuration for the robot used in this thesis is listed here.

C.2.1 Sonar Sensors

Position: 8 front-facing, 8 rear-facing

Range: 15cm-500cm

C.2.2 Laser Sensor

Position: 1 front-facing

Light source: Infrared (905nm)

Laser class: 1 (EN/IEC 60825-1), Eye-safe

Field of view: 180^o

Scanning frequency: 75 Hz

Operating range: 0m - 80m

Max. range with 10% reflectivity: 10m

Angular resolution: 0.25^o, 0.5^o, 1^o

Fog correction: no

Resolution: 1mm

Operating voltage: 24V dc 15%

Power consumption: 20W

Enclosure rating: IP 65

Protection class: 2, insulated

Weight: 4.5kg

Dimensions: 156mm x 155mm x 210mm

Housing: Aluminium die-cast

Ambient operating temperature:0^oC - 50^oC

Storage temperature: -30^oC - 70^oC, with heating plate -12^oC - +50^oC

Permissible relative humidity: 90%, non-condensing

C.3 On-Board Computer

vendor id : GenuineIntel

cpu family : 6

model : 8

model name : Pentium III (Coppermine)

stepping : 10

cpu MHz : 848.335

RAM: 250MB

Linux Version: 2.6.10, no.4 Wed Oct 19 16:53:04 BST 2005 i686 GNU/Linux

Linux Flavour: Debian

Appendix D

Glossary of Terms

Antigen

In biology antigen is any molecule that is detected by the immune-system. In the DCA antigen is a unique identifier that communicates the presence of a specific element within the environment.

Affinity

A measure of similarity, usually between a cell receptor and an antigen.

Aria

An object-orientated control library for creating robotic behaviours. Made by MobileRobots Inc. See also, Pioneer.

B-Cell

In biology B-Cells are cells within the immune system which generate antibodies to attack intruders. They also play a role in immune memory. They are the inspiration for the clonal selection algorithm in artificial immune systems.

CCTV

Abbreviation for "Closed Circuit Television". A common augmented security technique relying on cameras to monitor distally separate locations.

Classification Boundary

Part of DCA terminology. The abstract boundary in signal space which separates regions that represent anomalies and regions that represent normal inputs. See also, Decision Boundary

Clonal Selection

A process which causes cells with high-affinity matches to antigen present in the immediate environment to proliferate and cells with low-affinity matches to die out. The basis of a common artificial immune algorithm. See also B-Cell.

CSM

Abbreviation for "Costimulatory Molecule". In biology this is a chemical which facilitates the communication process. In the DCA this is used as a measure of sampled information.

Cut off Frequency

In frequency analysis, the frequency at which it is assumed a frequency dependent component ceases to pass information.

Danger

In biology danger signals are chemical markers indicating necrosis, (unexpected cell death). This suggests the presence of a harmful pathogen. In the DCA this is a time varying input which rises to indicate the pres-

ence of a suspected anomaly.

DC

Abbreviation for "Dendritic Cell". In biology, a dendritic cell is part of the immune system, which is responsible for absorbing antigen from the body. DCs are also referred to as "antigen presenting cells" as they go on to locate T-Cells within the lymph node and present samples of the antigen that they have absorbed.

DCA

Abbreviation for "Dendritic Cell Algorithm" an artificial immune systems algorithm based on the operation of DCs.

Decision Boundary

Part of DCA terminology. The abstract boundary in signal space which separates regions that represent situations when a given cell has not sampled enough input signal to attempt a classification and regions where a given cell has sampled enough input signal to attempt a classification. See also, Classification Boundary

Dendritic Cell

See DC

FOV

Abbreviation for "Field of View". The shape in physical space that describes the region where a sensor can detect targets.

IL10

Abbreviation for "Interleukin 10". In biology IL10 is a chemical signal

used to communicate to cells within the immune system. In the DCA this is the name a variable which keeps track of how much evidence the cell has aggregated for a 'normal' classification. See also IL12

IL12

Abbreviation for "Interleukin 12". In biology IL12 is a chemical signal used to communicate to cells within the immune system. In the DCA this is the name a variable which keeps track of how much evidence the cell has aggregated for an 'anomalous' classification. See also IL10

Linear Classifier

A machine learning construct which attempts to identify set membership based on training data. Linearity refers to the shape that the classification boundary describes in signal space.

LRF

Abbreviation for "Laser Range Finder". A time-of-flight device for detecting physical obstacles based on laser reflections. LRFs have a planar shaped FOV. See also Sonar.

MCAV

Abbreviation for "Mature Context Antigen Value". A technique for interpreting the output from the DCA. See Appendix B.4.

Migration Threshold

DCA terminology. The threshold of CSM that a cell will absorb before performing a classification. Defines the shape and position of the decision boundary.

Negative Selection

In biology negative selection is the process where by immune cells with the potential to harm the body are removed before being allowed to interact with other cells. The basis of an artificial immune system algorithm of the same name.

Neural Network

A machine learning algorithm, commonly used to perform classification tasks.

PAMP

Abbreviation for "Pathogen Associated Molecular Pattern". In biology a PAMP is a signature that identifies a specific threat with a known response. In the DCA PAMP is used as a signal which rises proportionally to the presence of evidence of an anomaly.

Pioneer

A family of robot made by MobileRobots Inc, see Aria.

Ripple

In frequency analysis ripple is an undesired rise (and generally a subsequent fall) of gain with respect to increased input frequency, after the cut off frequency.

Safe

in biology Safe signal is a chemical marker used to indicate apoptosis, (desired cell death). In the DCA safe is used as an inhibitory signal which rises in proportion to evidence of 'normal' operation.

SOM

Abbreviation for "Self-Organising Map" an unsupervised machine learning algorithm used to produce a low dimensional representation of high dimensional data where distance in the new representation provides some measure of similarity.

Sonar

A time-of-flight device for detecting physical obstacles based on sound wave reflections. Sonar devices have cone-shaped FOVs. See also LRF

STL

Abbreviation for "Standard Template Library". An open source collection of object-orientated data structures.

T-Cell

A T-Cell is an immune system cell used primarily to attack pathogens. They interact with DCs and are the inspiration for the negative selection algorithm in artificial immune systems.