Immune-Inspired Fault Diagnosis for Robot Swarms

James O'Keeffe

Doctor of Philosophy

University of York

Electronic Engineering

May 2019

UNIVERSITY OF YORK

Abstract

Electronic Engineering

Immune-Inspired Fault Diagnosis for Robot Swarms

by James O'Keeffe

Previous work has shown that robot swarms are not always tolerant to the failure of individual robots, particularly those that have only partially failed and continue to contribute to collective behaviours. A case has been made for an active approach to fault tolerance in swarm robotic systems, whereby the swarm can identify and resolve faults that occur during operation. Existing approaches to active fault tolerance in swarms have so far omitted fault diagnosis, however this thesis proposes that fault diagnosis is a feature of active fault tolerance that is necessary if robot swarms are to achieve long-term autonomy. This thesis presents a novel method for fault diagnosis in robot swarms that attempts to imitate some of the observed functions of natural immune system. The experimental results presented in this thesis, which were obtained in software simulation and with actual robot hardware, show that the proposed fault diagnosis system is flexible, scalable, and improves swarm tolerance to various electro-mechanical faults in the cases examined.

Contents

Abstract **iii**

Acknowledgements **xv**

Declaration of Authorship **xvii**

1 Introduction **1**
1.1 Introduction . . . 1
1.2 Swarm Robotics . . . 1
1.2.1 Fault Tolerance for Robot Swarms . . . 5
1.3 Research Question . . . 5
1.4 Structure . . . 5
1.5 Contributions . . . 6

2 Background and Related Work **9**
2.1 Introduction . . . 9
2.2 Robustness in Robot Swarms . . . 9
2.2.1 Fault Tolerance and Scalability in Robot Swarms . . . 11
2.3 Immunity for Swarms . . . 13
2.3.1 Artificial Immune Systems . . . 14
2.3.2 Implementing Artificial Immune Systems . . . 17
Summary . . . 21
2.4 Fault detection . . . 21
2.4.1 Endogenous fault detection . . . 22
2.4.2 Exogeneous fault detection . . . 22
2.4.3 Fault detection in swarms . . . 23
2.4.4 Summary . . . 25
2.5 Fault diagnosis . . . 25
2.6 Summary . . . 26

3 Fault Classification With Feature Vectors **27**
3.1 Introduction . . . 27
3.2 Behaviour Characterisation . . . 27
3.3 Experimental Method . . . 28
3.3.1 Fault Types . . . 30
3.3.2 Deriving Behavioural Features from Robot Behaviours . . . 31
Feature Estimation . . . 36
3.3.3 Experiments . . . 36
3.4 Results and Discussion . . . 38
3.5 Summary . . . 43

4 Fault Diagnosis in Software Simulation: Part I **47**
4.1 Introduction . . . 47
4.2 Fault Diagnosis for Swarm Robots . . . 47
4.3 Experimental Method . . . 48
4.3.1 Deriving Behavioural Feature Vectors . . . 49
4.3.2 Integrated Fault Diagnosis System . . . 50
Fault Detection . . . 50
Fault Diagnosis . . . 52
Fault Recovery . . . 54
4.3.3 Fault Memory . . . 54
4.3.4 Experiments . . . 55
4.4 Results and Discussion . . . 57
4.5 Summary . . . 60

5 Fault Diagnosis in Software Simulation: Part II **63**
5.1 Introduction . . . 63
5.2 Experimental Method . . . 64
5.2.1 Integrated Fault Diagnosis . . . 65
Fault Detection . . . 65
Fault Diagnosis . . . 66
Fault Recovery . . . 67
Fault Memory . . . 67
5.2.2 Experimental Setup . . . 68
5.3 Experiments . . . 69
5.3.1 Scalability & Flexibility . . . 70
Results & Discussion . . . 70
5.3.2 Noise Sensitivity . . . 72
Results & Discussion . . . 73
5.3.3 Collective Photo-taxis . . . 75
Results & Discussion . . . 76
5.3.4 Parameter Sensitivity Analysis . . . 78
Results & Discussion . . . 78
5.4 Summary . . . 85

6 Fault Diagnosis in Hardware **87**
6.1 Introduction . . . 87
6.2 Experimental Methods . . . 87
6.3 Results and Discussion . . . 94
6.4 Summary . . . 96

7 Conclusion **97**
7.1 Introduction . . . 97
7.2 Summary and Contributions . . . 97
7.3 Limitations . . . 99
7.3.1 Communication . . . 99
7.3.2 Fault Types Tested . . . 100
7.3.3 Fault Detection Mechanism . . . 100
7.3.4 Available Hardware & Fault Recovery . . . 100
7.4 Future Work . . . 101
7.4.1 Complex Faults . . . 101
7.4.2 Integration with fault detection . . . 101

7.4.3 Development of more capable swarm hardware 101
7.5 Conclusion 102

Bibliography **103**

List of Figures

2.1 Mean time before failure = 8 hours (Bjerknes and Winfield, 2013) . . . 12
2.2 Mean time before failure = 24 hours (Bjerknes and Winfield, 2013) . . . 13
2.3 An outline conceptual framework for a bio-inspired computational domain (Stepney et al., 2004) . . . 16
2.4 An outline conceptual framework for integrating bio-inspired computational domains (Stepney et al., 2004) . . . 16

3.1 The MarXbot swarm robotic platform (Bonani et al., 2010) . . . 29
3.2 A diagram to illustrate flocking, aggregation and obstacle avoidance behaviours **A:** Flocking behaviour. **B:** Obstacle avoidance behaviour. **C:** Aggregation. . . . 30
3.3 A diagram to illustrate F_1. Red highlight indicates the robot under consideration and its sensing range. **A:** The robot has no neighbours in sensing range. **B:** The robot has one neighbour in sensing range. . . . 32
3.4 A diagram to illustrate F_2. Red highlight indicates the robot under consideration and its close proximity threshold. **A:** The robot has no neighbours within its close proximity threshold. **B:** The robot has one neighbour within its close proximity threshold, and so they both turn to avoid each other. . . . 32
3.5 A diagram to illustrate F_3. Red highlight indicates the robot under consideration. **A:** Between times t_{0-1} the robot was turning, and so moved a relatively short distance ($F_3 = 0$). **B:** Between times t_{0-1} the robot was moving in a straight line at full speed, and so moved a greater distance ($F_3 = 1$). . . . 33
3.6 A diagram to illustrate F_4. Red highlight indicates the robot under consideration. **A:** Between times t_{0-1} the robot did not move at all ($F_4 = 0$). **B:** Between times t_{0-1} the robot did move, albeit a short distance ($F_4 = 1$). . . . 34
3.7 A diagram to illustrate F_5. Red highlight indicates the robot under consideration. **A:** Between times t_{0-1} the robot moved in a straight line and therefore with no angular velocity ($F_5 = 0$). **B:** Between times t_{0-1} the robot was turning, and therefore moved with maximum angular velocity ($F_5 = 1$). . . . 35
3.8 ARGoS simulation of 10 marXbot robots performing a dispersion behaviour. Lines connecting pairs of robots represent mutual neighbour acknowledgement via RAB sensor. Lines protruding from each robot represent IR sensor range. . . . 37
3.9 Visual representation of the decision tree trained on data collected whilst the swarm exhibited aggregation behaviour where p and e indicate proprioceptively and externally estimated features, respectively. 44

3.10 Visual representation of the decision tree trained on data collected whilst the swarm exhibited flocking behaviour where p and e indicate proprioceptively and externally estimated features, respectively. 44
3.11 Visual representation of the decision tree trained on data collected whilst the swarm exhibited obstacle avoidance behaviour where p and e indicate proprioceptively and externally estimated features, respectively. 45

4.1 A flowchart of the proposed fault diagnosis process. 51
4.2 This figure shows the A-Test scores for the proportion of faults the system is able to diagnose from memory (P_1) and the proportion of attempts to diagnose a fault from memory that result in misclassification (P_2). 19 subsets of 100 experimental replicates are tested (1900 total experiments). 57
4.3 The times at which a fault is classified using diagnostic tests or from memory in each experiment replicate (1 hour total run-time). Dashed line indicates the point at which the first fault is injected in each experiment. 60

5.1 **A**: ARGoS simulation of 10 marXbot robots performing obstacle avoidance in an empty arena (O'Keeffe et al., 2017b), **B**: ARGoS simulation of 10 marXbot robots performing obstacle avoidance in a cluttered arena, and **C**: ARGoS simulation of 10 marXbot robots performing collective photo-taxis in an arena with a beacon in the north-west corner. 68
5.2 **A**: The effects of noise on RAB sensed linear velocity on median r-values between diagnoses, and **B**: the effects of noise on RAB sensed linear velocity measurements on the proportion of faults successfully diagnosed from memory 73
5.3 The effects of noise on RAB sensed linear velocity measurements on the proportion unsuccessful of attempts to diagnose faults from memory 74
5.4 **A**: The performance of a robot swarm with no active fault tolerance exhibiting the collective photo-taxis when a varying proportion of robots are subjected to a variety of faults, and **B**: The results of the same experiment with active fault tolerance. Where there is a red box over the condition type indicates that proportional performance is not applicable because the entire swarm reached the beacon in less than 24388 time-steps, meaning that average distances at t $\geq$ 24388 control-steps could not be taken. 77
5.5 The median total number of unresolvable faults, P_4, over one hour of simulated time against the value of the observation period, o. Non-integer values indicate a median value between two integer values, rather than 0.5 unresolvable robots. 79
5.6 **A**: The median proportion of faulty robots the system was successfully able to resolve from memory against the value of the similarity threshold, s, and **B**: the median proportion of attempts by the system to diagnose from memory that failed or were not recognised as successful against the value of the simulation threshold, s. 80
5.7 **A**: The median time taken to detect faults against the value of the detection window, W, and **B**: the median r-value of successful diagnoses against the value of the detection window, W. 81

5.8 The median total number of faulty robots the system was unable to resolve over one hour of simulated time against the value of the detection window, W. Non-integer values indicate a median value between two integer values, rather than 0.5 unresolvable robots. 82
5.9 **A**: The median time taken to detect faulty robots against the proportion of mismatched BFVs in the detection window, ρ, and **B**: the median proportion of attempts by the system to diagnose from memory that failed or were not recognised as successful against the proportion of mismatched BFVs in the detection window, ρ 84
5.10 The median r-value of diagnoses against the proportion of mismatched BFVs in the detection window, ρ 85

6.1 The Psi-Swarm robot used for the experiments in this chapter (Hilder et al., 2016). 88
6.2 Five Psi-swarms performing obstacle avoidance in an arena. The robot with a green hat suffers from power failure (H_1), whilst its assessing robot in this case is circled in blue. 89

List of Tables

2.1 Effects of faults on swarm behaviour (Winfield and Nembrini, 2006) . . 10

3.1 A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing aggregation behaviour where test data is obtained from a swarm performing aggregation behaviour. 38
3.2 A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing aggregation behaviour where test data is obtained from a swarm performing flocking behaviour. 39
3.3 A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing aggregation behaviour where test data is obtained from a swarm performing obstacle avoidance behaviour. 39
3.4 A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing aggregation behaviour where test data is obtained from a swarm performing flocking behaviour. 40
3.5 A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing flocking behaviour where test data is obtained from a swarm performing flocking behaviour. 41
3.6 A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing flocking behaviour where test data is obtained from a swarm performing obstacle avoidance behaviour. 41
3.7 A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing obstacle avoidance behaviour where test data is obtained from a swarm performing aggregation behaviour. 42
3.8 A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing obstacle avoidance behaviour where test data is obtained from a swarm performing flocking behaviour. 42
3.9 A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing obstacle avoidance behaviour where test data is obtained from a swarm performing obstacle avoidance behaviour. . . . 42
3.10 A comparison of the median average proportion of correctly classifed faults for training and test data collected during all combinations of normal swarm behaviour 45

4.1 Total instances of attempts to diagnose from memory that resulted in misclassification (P_3). 58
4.2 Median average correlation coefficients, P_3 ($\pm\sigma$), for diagnoses from memory over 100 replicates. 58
4.3 Average time required to identify faults using diagnostic tests, and the number of additional instances each would occur if the similarity threshold was raised to 0.74. 58
4.4 Total instances of faults requiring diagnostic tests despite previous encounters. 59

5.1 Performance of system for swarms of varying sizes 70
5.2 Performance of system for robot swarms in different environments . . 70
5.3 Proportion of faults successfully diagnosed from memory for swarms of varying sizes 71
5.4 Normally functioning swarm performance of collective photo-taxis . . 76

6.1 Comparison of the performance of the proposed diagnosis system in software simulation and hardware experiments, where $s = 0.56$, $W = 29$ and $\rho = 0.88$. Data for software simulation experiments is taken from O'Keeffe *et al.* (2018). 94
6.2 Comparison of system performance, where $s = 0.56$ and $\rho = 0.88$, for varying values of W. 95
6.3 Comparison of system performance where the system encounters 15 and 25 faults in an experiment ($s = 0.56, W = 58, \rho = 0.88$). 95
6.4 Comparison of system performance, where $W = 0.58$ and $\rho = 0.88$, for varying values of s. 96

Acknowledgements

I would like to thank my supervisors, Jon, Alan and Danesh, for their helpful guidance and support throughout my PhD that enabled me to develop initial research ideas and see them through to maturity. Those ideas, which were formed early in my first year, now account for the bulk of this thesis.

I would like to give special thanks to Jon. It was only because of Jon's last minute advocation on my behalf that I was given the funding I needed to support myself through this PhD. The brief time I have spent in academia has undoubtedly changed me and the trajectory of my life for the better, and without Jon's support it might never have happened. I will always be grateful for the opportunity that Jon gave me.

I would also like to give special thanks to Alan, who has gone above and beyond to help me find a foothold in the world of academic employment. During my final months of funding, and whilst I was frantically trying to secure employment as well as stay on track with my thesis, Alan was going through the CVs and personal statements I had sent him and singing my praises to anyone who might be able to offer me a job in academia. Knowing that I had Alan in my corner during that period was an enormous help, not only to my job search but also to my own sense of security.

I would like to thank my viva assessors, Andy Pomfret and Heiko Hamann, for their constructive criticism of my work, and for making my viva experience an enjoyable one. Andy and Heiko were simultaneously able to challenge the contents of my thesis, as well as my own knowledge, whilst also making me feel at ease – something that I greatly appreciated. Additional thanks to Heiko for making the trip to assess my viva, and to Andy for helping to prepare me during the three years in which I was working at York.

I want to give special thanks to two of my colleagues and friends, James and Naomi, for making my office a place that I wanted to be. James and Naomi joined the lab when I was in my second year, and the effect they had on my morale cannot be overestimated. Knowing that I would get to see them each day in the office made going in to work easy, and enabled me to progress more quickly than perhaps I otherwise would have.

I would like to thank Heather, who was present for the duration of my PhD and who, more so than anybody else, has helped me develop into the person that I am today. Heather shared the weight of every burden and setback I encountered during my PhD, and never stopped trying to find ways to help and support me. Without Heather I do not know where or who I would be, and I am glad that I don't have to find out.

I want to thank my parents, whose belief in me has never faltered, even when it has stood alone. I owe everything to them and their unwavering and unconditional love and support.

I would also like to thank Ayla for having no reservations whatsoever in telling me, during a conversation about the natural immune system, that everything I was saying was wrong and providing me with the references to prove it. Ayla guided me towards the knowledge I needed to make (what I hope are) the sound arguments and justifications for the system I present in this thesis.

I will always remember the time I spent in York as pivotal, with a clear before and after in terms of my own personal development. I feel that I became the person that I am whilst at York and, after several miserable years, the time between starting my PhD and time of writing has been the happiest period of my life so far. A significant factor in making that time so special were the people I met and the friends I made.

There are too many to list each by name, but I hope that they know who they are and, if they ever read this, know how grateful I am to have known them. Having said all that, I still feel the need to acknowledge Ben, Laura, Jo, Dan and Harry, in particular, for making the good times great and the bad times okay.

Finally, I would like to thank Mike Nolan for providing me with the best advice I have ever received, and for being a role model for a bigger and better life.

Declaration of Authorship

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.

List of Abbreviations

FMEA	**F**ailure **M**ode (and) **E**ffect **A**nalaysis
AIS	**A**rtificial **I**mmune **S**ystems
FDDR	**F**ault **D**etection **D**iagnosis (and) **R**ecovery
CRM	**C**ross **R**egulation **M**odel
APC	**A**ntigen **P**resenting **C**ell
BFV	**B**ehavioural **F**eature **V**ector
ARGoS	**A**utonomous **R**obots **G**o **S**warming
RAB	**R**ange **A**nd **B**earing

Chapter 1

Introduction

1.1 Introduction

There is evidence to suggest that the synchronised operation of social insects is decentralised (Camazine et al., 2001), yet they exhibit robust, flexible and scalable behaviour as a system (Şahin, 2005). These are desirable features for autonomous systems, and motivate the use of, and research into robot swarms (Şahin, 2005).

Robustness – the ability of a system to withstand faults and failures – as an inherent feature of robot swarms was initially highlighted by Şahin (2005), and there is evidence to support this in the case of complete failures (Winfield and Nembrini, 2006). However, it is also observed that the performance of a swarm decreases where partial failures are present (Winfield and Nembrini, 2006). It was later demonstrated that the reliability of a robot swarm significantly decreases as a consequence of partial failures as the number of agents in the swarm increases (Bjerknes and Winfield, 2013).

The finding that the reliability and robustness of robot swarms is compromised in some scenarios prompted Bjerknes and Winfield (2013) to advocate for an active approach to improving the fault tolerance of robot swarms. Fault tolerance typically consists of fault detection, fault diagnosis and fault recovery (Bayar et al., 2015). Researchers have already begun investigating ways in which this could be achieved, for example; there have been a number of approaches towards autonomous fault detection in robot swarms (see work by Millard (2016), Khadidos, Crowder and Chappell (2015), Tarapore, Christensen and Timmis (2017) and Christensen, O'Grady and Dorigo (2009)).

This thesis proposes a novel approach to fault diagnosis in robot swarms. The approach proposed in this thesis allows robots to characterise different types of fault in real time such that recurring faults can be diagnosed more efficiently, mimicking the observed functions of the natural immune system at a high level.

The first chapter of this thesis provides an overview of swarm robotic systems, highlights the need for fault diagnosis as part of an approach to active fault tolerance in robot swarms, and describes the hypothesis, structure and contributions of this thesis.

1.2 Swarm Robotics

Swarm robotics can be described as the application of swarm intelligence to multi-robot systems (Şahin, 2005). Swarm intelligence, as a field of research, stems from observations on the abilities of social insects to behave co-operatively and efficiently (Garnier, Gautrais, and Theraulaz, 2007), with seemingly no centralised method of coordination (Şahin, 2005).

A swarm robotic system typically consists of multiple individual robots that are relatively simple, but can collectively exhibit complex behaviour or achieve greater performance as a system (Şahin, 2005). The collective behaviour of a swarm robotic system typically reflects the task or purpose for which it is being used. Collective behaviour is sometimes *emergent*. An emergent behaviour is the distinct collective product of individual robot behaviours, for example the flocking behaviour seen in some species of birds and fish.

Swarm robotic systems are distinct from traditional multi-robot systems, which may feature centralised or hierarchical control structures. Şahin (2005) proposes a number of features that distinguish swarm robotic systems from other robotic systems. Although the definitions used by Şahin (2005) were published over a decade ago at the time of writing, and the field of swarm robotics has developed since then (more recent descriptions of swarm robotic systems are given by Brambilla *et al.* (2013) and Bayindir (2016)), it is useful to revisit them in this chapter for the purposes of outlining the motivation for this thesis.

1. **Consist of autonomous robots**
   Individual robots in a swarm should have a physical embodiment, be able to physically interact with their environment, and be able to do so autonomously.

2. **Feature large numbers of robots**
   Swarm robot systems should be scalable and thus functional for a large range of swarm sizes.
   There is no hard threshold, in terms of the number of individuals, at which a multi-robot system becomes a swarm robot system, but between 10 and 20 individuals is generally acceptable. The cost of hardware may represent an obstacle when trying to realise greater swarm sizes. Studies carried out with smaller group sizes with the promise of scalability in sight can also be considered swarm research.

3. **Feature few homogeneous groups of robots**
   Swarm robotic systems should comprise of a large number of homogeneous robots, or a few groups thereof. Highly heterogeneous robot groups, however large, are less swarm-like.

4. **Consist of robots with local sensing and communication capabilities** To maintain distributed coordination in the swarm, individual robots should only possess local sensing and communication abilities.

5. **Consist of relatively incapable or inefficient robots**
   Şahin (2005) defines this in two ways, depending on the task at hand. Either; in cases where collective behaviour is emergent, individual robots be unable to perform the desired collective behaviour, or; in cases where collective behaviour is not emergent, and so can be performed by an individual, increasing the number of individuals that constitute the swarm should markedly improve the overall performance of the system.
   This criterion should not restrict the complexity of individual robot hardware and/or software. Rather, incapability/inefficiency should be considered in relative terms with respect to the swarm's task.

With the exception of point 5, all of the above criteria persist in the more recent swarm definitions used by either Brambilla *et al.* (2013) or Bayindir (2016).

Şahin (2005) proposes that swarm robotic systems, by virtue of meeting these criteria, will possess an inherent degree of robustness, flexibility and scalability. These desirable properties of robot swarms form the basis of the motivation for their use and research, and are be defined as follows:

1. **Robustness**

   Robustness can be defined as the degree to which a swarm system can tolerate the presence of faults or failures that occur in individuals within the swarm. If a swarm is tolerant to a given type of fault, the presence of that fault in an individual should not significantly reduce the overall performance of the swarm. Şahin (2005) attributes the high level of robustness observed in swarms of social insects to several factors:

   (a) **Redundancy**
   If an individual fails, another can assume its place.

   (b) **Decentralised coordination**
   Failures occurring in an individual or individuals will not disable the systems emergent behaviour. This relies on the assumption that the number of failed individuals does not cause the number of working individuals to drop below the minimum needed to perform the swarms task (Bjerknes and Winfield, 2013).

   (c) **Simplicity of individuals**
   The behaviour of an individual in a swarm will be relatively simple compared to the swarms collective behavior, or alternative system that would be required to equivalently perform it. There will therefore be fewer opportunities for faults and failures to manifest in each individual, and so they can be considered functionally reliable by comparison.

   (d) **Multiplicity of sensing**
   This particular characteristic is not concerned with faults *per se*. However, any physical robot that can respond to its environment in real time, as has previously been expressed to be desirable, will have to make decisions based on information it receives from its sensors. There will be some degree of noise present in whatever signal hardware sensors produce. If signal-to-noise ratio is too low for a robots sensors, it may be unable to perform its task effectively. Distributed sensing can increase signal-to-noise ratio by providing multiple observations of a point of interest.

2. **Flexibility**

   Bayindir (2016) defines flexibility as:

   > "the capability to adapt to new, different, or changing requirements of the environment."

   This is, to an extent, at odds with the requirement for robustness. If a problem or task changes, the system should not attempt to continue exhibiting a behaviour that is unsuitable for the new problem, but rather be flexible enough to adapt to the new problem appropriately. Şahin (2005) highlights that swarm robotic systems should be able to utilise different coordination strategies in response to changes in the environment.

3. **Scalability**
   Scalability can be defined as a swarms ability to increase or decrease the number of individuals that constitute it without significant consequences to overall performance.

The definitions of flexibility and scalability in robot swarms do not differ between those used by Şahin (2005) and Brambilla *et al.* (2013). However, Brambilla *et al.* (2013) updates the definition of robust to *"the ability to cope with the loss of individuals", rather than the ability to tolerate failures*. The reasons for this will be detailed later in this chapter and in the chapter following.

Şahin (2005) outlines a series of task domains in which there could be some advantage in taking a swarm robotic approach:

1. **Tasks that cover a region**

   The distributed nature of swarm robotic systems means that they can typically cover a larger area than a non-distributed system performing the same task in a given period of time. Examples of such tasks include search and rescue or foraging for resources, as was mentioned earlier in this chapter. The work by Sugawara and Watanabe (2002) examines the example of foraging robot swarms in more detail.

2. **Tasks that are dangerous**

   In tasks where it is likely that the performer will receive damage, a swarm robotic approach could possess an advantage in terms of cost efficiency, as the individual robots within the swarm are often dispensable. Cassinis *et al.* (1999) propose swarm robotic systems could be applied to the detection of land mines, for example.

3. **Tasks that scale up or down in time**

   Şahin (2005) gives the example of oil leaking from a sinking ship, whereby the scale of the leakage can increase as the tanks of the ship break down. A swarm robotic system containing the initial spillage in a bounded area could adapt to the increasing oil leakage by adding more robots, allowing the bounded area to expand in sympathy with the amount oil needing to be contained. This type of scenario is examined in more detail in the work by Kakalis and Ventikos (2008).

4. **Tasks that require redundancy**

   Redundancy is defined here as a system that includes surplus components that are non-essential to its functioning. An example of where a swarm robotic approach may give a significant advantage over other approaches is in the case of tasks that may require redundancy. Tasks that require redundancy may also fall under the category of dangerous tasks, or may involve the reconfiguration of a proportion of the swarm whilst other robots continue performing their task. An example of this would be the establishment of *ad hoc* wireless networks, as seen in the work by Correll *et al.* (2009), featuring mobile nodes.

1.2.1 Fault Tolerance for Robot Swarms

A significant part of the motivation for using robot swarms hinges on their presumed robustness. However, it was shown by Winfield and Nembrini (2006) and Bjerknes and Winfield (2013) that robot swarms are not always robust to faults or failures in individuals – particularly in the case of partial failures. In light of this, Bjerknes and Winfield (2013) call for an active approach to fault tolerance in autonomous robot swarms.

The work by Christensen *et al.* (2007) supports the proposal for active fault tolerance by highlighting the infeasibility of implicit approaches to fault tolerance. Christensen *et al.* (2007) describe fault tolerance as a multi-faceted objective, but proposes that it will generally consist of an approach to fault detection, and possibly approaches to fault diagnosis and fault recovery.

Previous work by other researchers towards achieving fault tolerance in robot swarms has so far examined fault detection (Tarapore, Christensen, and Timmis, 2017) (Millard, 2016), or fault detection and recovery (Khadidos, Crowder, and Chappell, 2015) (Christensen, O'Grady, and Dorigo, 2009). Some swarm behaviours will require a minimum number of functioning robots. If faults are detected but not diagnosed, the only available method for fault recovery is to replace the entire robot. Dangerous environments are highlighted as a potential application area for robot swarms. In such environments, or in any environment that is for any reason inaccessible, it will not always be possible to replace faulty robots. This thesis therefore argues that fault diagnosis is a critical step in achieving active fault tolerance if swarm robots are to retain long-term autonomy.

1.3 Research Question

This thesis presents a novel method for fault diagnosis in swarm robotic systems. The research question that underpins this thesis can be broadly described as:

> Can faults in swarm robotic systems be diagnosed by examining specifically where discrepancies between observed and expected behaviours originate from?

1.4 Structure

This thesis is structured as follows:

Chapter 2

This chapter provides review and discussion on the motivation for active fault tolerance in swarms, artificial immunity, previous approaches to fault detection and other relevant work for consideration.

Chapter 3

This chapter details the initial work undertaken for this thesis, aimed at establishing whether or not different types of fault could be classified by encoding robot behaviours as a series of boolean behavioural features.

Chapter 4

This chapter proposes a novel method for immune-inspired fault diagnosis in robot swarms, and details the work completed in order to develop and test a prototype system.

Chapter 5

This chapter further develops upon the immune-inspired fault diagnosis system for robot swarms described in chapter 4, and details a more thorough assessment of the system.

Chapter 6

This chapter describes the transposition of the immune inspired fault diagnosis system from software simulation to actual robot hardware as faithfully as was possible within the parameters of this thesis.

Chapter 7

This chapter concludes the thesis by comparing the results and achievements described in chapters 3-6 against the hypothesis stated in the first chapter. Chapter 7 also describes the limitations of the work produced for this thesis, as well as outlining possible avenues for future work.

1.5 Contributions

The contributions of this thesis are as follows:

Chapter 3

Chapter 3 investigates whether or not common electro-mechanical faults occurring in individual robots can be classified by encoding robot behaviours as behavioural feature vectors. The majority of the work described in this chapter was published in the proceedings of TAROS 2017 (O'Keeffe et al., 2017b).

Chapter 4

Chapter 4 proposes a novel fault diagnosis system for robot swarms. This is built and tested in software simulation. The majority of the work described in this chapter was published in the proceedings of IEEE SSCI ALIFE 2017 (O'Keeffe et al., 2017a).

Chapter 5

Chapter 5 develops the novel fault diagnosis system proposed in chapter 4, and subjects it to more rigorous testing in software simulation. The majority of the work described in this chapter was published as an article in the Frontiers in Robotics and AI journal (O'Keeffe et al., 2018).

Chapter 6

Chapter 6 implements a modified version of the diagnostic system described in chapter 5 in actual robot hardware. This chapter demonstrates that the trends observed in software simulation are also observed for a system in robot hardware.

Chapter 2

Background and Related Work

2.1 Introduction

Swarm robotic systems are highlighted to have applicability to a number of task domains (Şahin, 2005) (Bayındır, 2016). In particular, Şahin (2005) proposes robot swarms will be suited to tasks in dangerous environments, on account of the innate robustness of swarm systems. The findings of Winfield and Nembrini (2006), which are discussed in the next section, that robot swarms are susceptible to partially failed robots, challenges the universality of the swarm robotic properties described by Şahin (2005).

This chapter outlines the problems with fault tolerance in swarm robotic systems, which provide the motivation for this thesis. There is then a discussion on the ways other researchers have proposed these problems could be addressed, principally by creating artificial immunity, and the various approaches to this sub-field. Finally, this chapter provides discussion on the previous approaches towards active fault tolerance in robot swarms by other researchers.

2.2 Robustness in Robot Swarms

Robustness as an inherent property of swarm systems, as described by Şahin (2005) and discussed in the previous chapter, is based on the observed behaviour of social insects. Using a definition of robustness comparable to that given by Şahin (2005), Winfield and Nembrini (2006) perform a Failure Mode and Effect Analysis (FMEA) (Dailey, 2004) for a wireless connected robot swarm exhibiting the following behaviours.

- **Swarm aggregation**
  In this behaviour the robots in a swarm try to stay together in unbounded space
- **Coherent *ad hoc* network**
  Each robot in the swarm broadcasts a message which is received and acknowledged by its neighbours within wireless range. If a robot receives below a defined threshold of messages, it is assumed to be moving out of the swarm and will double back on itself.
- **Beacon taxis**
  Robots in the swarm can sense and will move toward the location of a beacon
- **Obstacle avoidance**
  Robots in the swarm are able to sense their proximity to neighbours and other

Swarm Behaviour	H_1	H_2	H_3	H_4	H_5	H_6
Aggregation	–	e_2	–	–	e_2	–
Ad hoc network	–	e_2	–	–	e_2	–
Beacon taxis	E_1	e_2	–	–	E_1	–
Obstacle avoidance	E_1	e_2	e_3	–	E_1	–
Encapsulation	E_1	e_2	e_3	–	E_1	–

TABLE 2.1: Effects of faults on swarm behaviour (Winfield and Nembrini, 2006)

objects. When robots detect that they are within a certain range of an object or neighbour, they will adjust their paths accordingly in order to avoid collisions.

- **Beacon encapsulation**
  An extension of the former behaviours, when approaching the beacon, the requirement for coherence and obstacle avoidance will result in the beacon being encapsulated by the swarm.

Utilising FMEA, Winfield and Nembrini (2006) define a list of the possible faults that may occur in individual robots within the swarm:

- H_1: Motor Failure
- H_2: Communications Failure
- H_3: Avoidance Sensor Failure
- H_4: Beacon Sensor Failure
- H_5: Control Systems Failure
- H_6: Total Systems Failure

Some of these faults are benign, however the presence of others may adversely affect the swarm's collective behaviour. The potential fault effects are classified as being serious or non-serious, indicated by an E (upper-case) or e (lower-case), respectively:

- E_1: Motor failure anchoring the swarm
- e_2: Lost robot(s) loose in the environment
- e_3: Robot collisions with obstacles or target

The results of the various faults on the swarm as it performs different tasks are displayed in Table 2.1.

Table 2.1 indicates that, although there are some circumstances in which a swarm may be able to tolerate partial or total failures in individual robots, there are others in which partial failures in individual robots present a serious threat to swarm performance. In particular, motor failures which occur in individual robots can anchor the swarm about that robot's location. If the swarm has been assigned a task that requires it to move in a specific direction or towards a point of interest, such as photo-taxis, this type of failure can prevent the swarm from completing its task.

Winfield and Nembrini (2006) conclude that:

> "Analysis of fault tolerance in swarms critically needs to consider the consequence of partial robot failures, and future safety-critical swarms would need designed-in measures to counter the effect of such partial failures."

2.2.1 Fault Tolerance and Scalability in Robot Swarms

That swarm robotic systems may not be robust to partial faults also has implications for another swarm property: Scalability.

Bjerknes and Winfield (2013) conduct an investigation into how the effects of partial faults on swarm robotic systems differ as the scale of the swarm varies. As a case study, a swarm of e-puck robots are considered (Mondada et al., 2009) performing a modified version of the α-algorithm (Winfield et al., 2008), defined by Bjerknes and Winfield (2013) as the ω-algorithm. The ω-algorithm is a combination of flocking and beacon taxis, which results in a single coherent swarm moving towards a beacon. The following failure modes and their effects to the overall swarm are considered by Bjerknes and Winfield (2013):

1. **Complete failure of individual robots:**
   This type of failure has been shown to be relatively benign (Winfield and Nembrini, 2006). Bjerknes and Winfield (2013) state that the only instances in which complete failures could prevent a swarm from realising its task is if they were to reduce the number of working robots in the swarm below the minimum threshold required for the self-organising team work to function.

2. **Failure of a robot's infra-red (IR) sensors:**
   Each e-puck robot has 8 IR sensors, and so it is unlikely that a complete sensor failure would occur. More likely would be a subset of the IR sensors failing, resulting in one or more directional 'blind spots' for the robot. In any case, the failed robot could become lost. In such cases the robot would act as a moving obstacle to the swarm. This type of failure could also act to lower the number of working robots below the minimum required by the swarm.

3. **Failure of a robot's motors only:**
   Established by WInfield and Nembrini (2006) to be the most serious type of fault, if sensing and signalling functions remain operational, a motor failure may cause the faulty robot to anchor the swarm. This is the case for failures in one or both motors.

Results obtained by Bjerknes and Winfield (2013), displayed in Figure 2.1 and Figure 2.2, show that the reliability of a robot swarm depends on the number of robots in the swarm and the probable rate at which failures occur in individual robots.

Bjerknes and Winfield (2013) use the *k-out-of-N* model to characterise the reliability of the swarm. In this model, reliability is the probability that at least k out of N robots are working at a given time t. This is described mathematically by Equation 2.1.

$$P(k, N, t) = \sum_{i=k}^{N} \binom{N}{i} (e^{-t\lambda})^i (1 - e^{-t\lambda})^{N-i} \tag{2.1}$$

where λ is described by Equation 2.2.

$$\lambda = \frac{1}{MTBF} \tag{2.2}$$

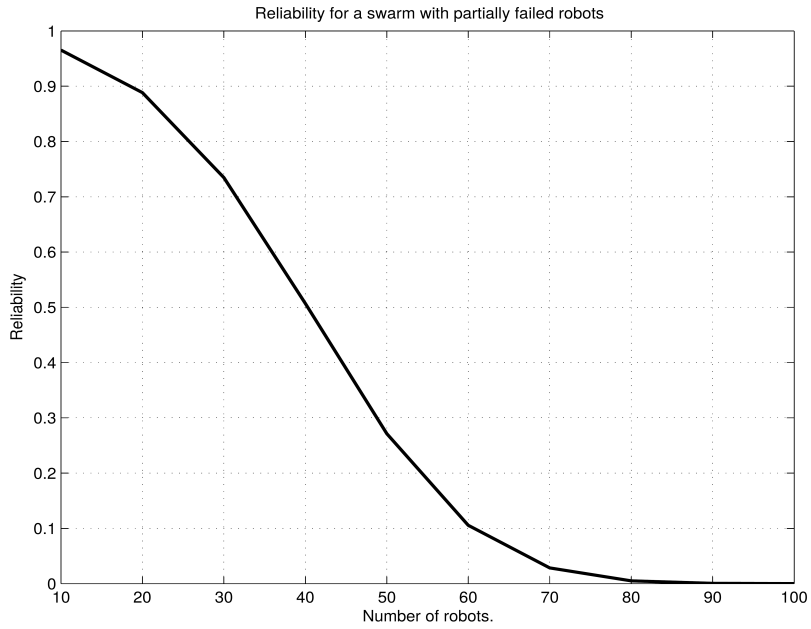

FIGURE 2.1: Mean time before failure = 8 hours (Bjerknes and Winfield, 2013)

MTBF is the mean time before failure for an individual robot.

Although the results obtained by Bjerknes and Winfield (2013) show that swarm systems consisting of robots with a greater *MTBF* value are generally more reliable over a greater range of swarm sizes, there is a critical point in Figure 2.1 and Figure 2.2 at which the reliability of the swarm drastically decreases with increasing swarm size. This would suggest that if a swarm robotic system cannot tolerate faults in individual robots, it cannot be considered truly scalable. These results indicate that, by only making individual robots *less likely* to develop faults, one is just deferring the point at which swarms stop being reliable as more robots are added if the faults that occur are not appropriately handled.

Bjerknes and Winfield (2013) conclude that larger swarm sizes require active measures to improve fault tolerance. Swarm systems are decentralised, so it falls to individual robots to be able to detect and respond to failures in themselves or their neighbours. Bjerknes and Winfield (2013) propose that it is necessary to introduce new behaviours at an individual level that will allow for this, and highlight two points for consideration, firstly; How could an individual robot reliably detect a fault or failure in another? And secondly; Upon detection, what would be an appropriate response? In their concluding sentences, Bjerknes and Winfield (2013) state that:

> "[large-scale self-organising systems] cannot function without an active approach to dealing with a failed or rogue unit, i.e. an immune response. What is perhaps surprising is that such an approach will be needed in swarm systems with relatively few individuals, i.e. less than a hundred."

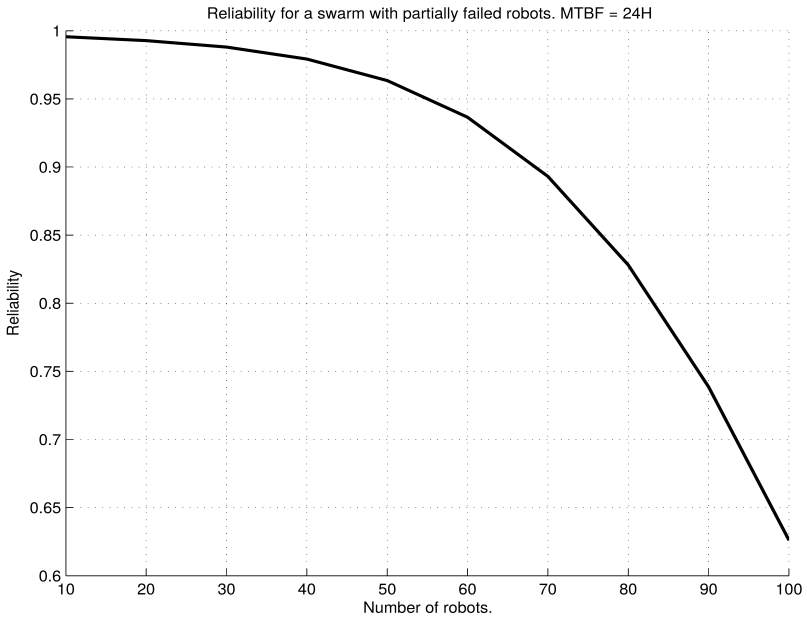

FIGURE 2.2: Mean time before failure = 24 hours (Bjerknes and Winfield, 2013)

2.3 Immunity for Swarms

Bjerknes and Winfield (2013) specifically advocate for an active *immune response* to deal with failed individuals in swarms. Current understanding of what an immune response consists of is derived from observations of the natural immune system in vertebrates. In their review paper, Timmis *et al.* (2008) use the following definition of the natural immune system as:

> "a defence system that has evolved to protect its host from pathogens (harmful micro-organisms such as bacteria and viruses)"

Having evolved over millions of years, the natural immune system provides rapid and effective defence for its host (De Castro and Timmis, 2002). The immune system is described as consisting of two parts: The innate immune system and the adaptive immune system (Owen, Punt, and Stranford, 2013).

The innate immune system can be thought of as a series of defence mechanisms that an individual is born with, such as phagocytosis or the inflammatory response, that generally do not change over the course of the individual's life (Owen, Punt, and Stranford, 2013).

The adaptive immune system consists of defence mechanisms that will change and develop over the course of an individual's lifetime. It is these mechanisms that allow the immune system to learn how to respond to previously unknown pathogens, as well as remember them such that future encounters are dealt with more efficiently (Owen, Punt, and Stranford, 2013).

Consisting of trillions of cells, distributed throughout the host organism, Segal and Cohen (2001) argue that the immune system can be thought of as a decentralised system. Timmis *et al.* (2010) further propose that the immune system can be considered as a swarm system.

Timmis, Andrews and Hart (2010) make the comparison of swarm systems to the immune system by outlining the parallels between each across the five principles of swarm systems, as defined by Millonas (1993), which are:

1. **The principle of proximity**
   Swarm systems and the immune system both perform computation in response to environmental stimuli such that the collective utility of each system is maximised for a given task or function (Millonas, 1993) (Timmis, Andrews, and Hart, 2010).

2. **The principle of quality**
   Swarm systems and the immune system respond to environmental quality factors, such as the quality of food (Millonas, 1993) or the severity of an attack (Timmis, Andrews, and Hart, 2010).

3. **The principle of diverse response**
   Swarm systems and the immune system are physically and functionally distributed, which provides them each with a degree of robustness to sudden environmental changes (Millonas, 1993) (Timmis, Andrews, and Hart, 2010).

4. **The principle of stability**
   Swarm systems do not significantly change their behaviour with slight variations in their environment (Millonas, 1993). The immune system is regulated by a series of networks in order to prevent large uncontrolled changes in behaviour, although it is possible to observe major shifts in behaviour under certain circumstances (Timmis et al., 2010).

5. **The principle of adaptability**
   Swarm systems will change their behaviour when the reward is worth the energy cost (Millonas, 1993). Similarly, the immune system has the ability to make collective decisions on whether or not to launch an immune response, and will only do so if it ascertains that the host's normal function is compromised (Timmis, Andrews, and Hart, 2010).

The implication of the work by Bjerknes and Winfield (2013) is that faults and failures that occur in engineered systems could be considered analogous to pathogens, insofar as pathogens prompt an immune response in biological systems as the authors propose faults and failures should do in robotic systems. Identifying the ways in which the defence mechanisms observed in the natural immune system be utilised in engineered systems is non-trivial, due to the vast complexity of the natural immune system. However, work towards this transposition has birthed the field of *Artificial Immune Systems* (AIS).

The functional parallels, identified by Timmis, Andrews and Hart (2010), between swarm systems and the immune system lead the authors to propose that AIS may be well suited for use with robot swarms. Consequently, Bjerknes and Winfield (2013 cite AIS as a promising potential solution for addressing the problems they and Winfield and Nembrini (2006) highlight on improving fault tolerance in swarm robotic systems.

2.3.1 Artificial Immune Systems

AIS attempt to bridge the gap between immunology and engineering (Timmis et al., 2008). They are defined by DeCastro and Timmis (2002) as:

> "adaptive systems, inspired by theoretical immunology and observed immune functions, principle and models, which are applied to problem solving."

Although the purpose of the natural immune system is host-defence, it is not the only area of application for AIS (Floreano and Mattiussi, 2008). Hart and Timmis (2008) broadly identify the major areas of application for AIS as:

- **Learning**
  Includes clustering, classification and pattern recognition (see Watkins and Timmis (2004) and De Lemos *et al.* (2007)).

- **Anomaly detection**
  Includes fault detection and computer and network security applications (see Stibor, Timmis and Eckert (2005) and Aickelin and Cayzer (2008)).

- **Optimisation**
  (see Cruz-Cortés, Trejo-Pérez and Coello (2005) and Freschi and Repetto (2005))

Applying AIS to these types of problems, there are a number of different approaches that exist. Cohen (2007) outlines three schools of research within AIS:

- **The literal school**:
  Researchers in the literal school try to faithfully replicate *in silico* the observed mechanisms of the natural immune system *in vivo* and *in vitro*.

- **The metaphorical school**:
  Researchers in the metaphorical school do not attempt to replicate the natural immune system, but rather use it as inspiration when building their own algorithms. This is the most common school.

- **The modelling school**:
  The modeling school, simply referred to as "the third school" by Cohen (2007), consists of researchers that attempt to better our own understanding of immunity by developing computer models of immune systems.

The problem of improving fault tolerance in robot swarms is not directly concerned with improving the general understanding of immunity among researchers. Therefore, during the work described later in this thesis, the modelling school was not considered as a practical approach.

Freitas and Timmis (2007) assert that the dominant approach to AIS development is the *algorithm-oriented approach*. The algorithm-oriented approach refers to the development of AIS in the absence of a defined problem, or where the the choice of algorithm employed in an AIS informs the problem it is applied to, as opposed to the problem informing the choice of algorithm. For example, the authors (Freitas and Timmis, 2007) highlighted a quote from Anchor *et al.* (2002):

> "The binary string representation is employed to allow for easy manipulation by a genetic algorithm in the affinity maturation "

It is noted that there is nothing in Anchor *et al.* (2002) to suggest why binary representation is needed in this scenario, as genetic algorithms are able to use real value data (Freitas and Timmis, 2007).

Whilst there is merit in the algorithm-oriented approach from an academic point of view, it is not able to produce practical AIS that can compete with state of the art

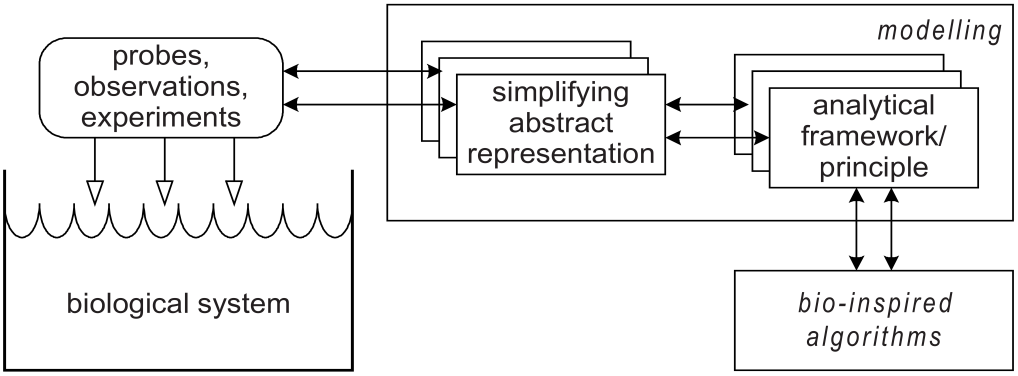

FIGURE 2.3: An outline conceptual framework for a bio-inspired computational domain (Stepney et al., 2004)

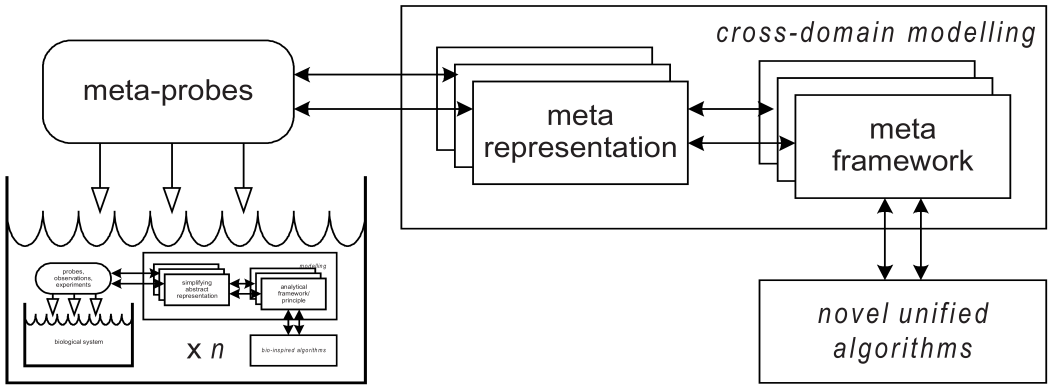

FIGURE 2.4: An outline conceptual framework for integrating bio-inspired computational domains (Stepney et al., 2004)

alternative techniques (Freitas and Timmis, 2007). Consequently, the *problem-oriented approach* is proposed by Freitas and Timmis (2007). The problem-oriented approach proposes that AIS be tailored to their application domain.

The problem-oriented approach lends itself to the metaphorical school of research, which is subject to criticism on the use of 'naive' metaphors. Stepney *et al.* (2004) state that:

> " bio-inspired computational algorithms usually proceed directly from a (naive) biological model to an algorithm, with little analytical framing of the representation's properties"

In order to avoid these weak analogies, stemming from "reason by metaphor", Stepney *et al.* (2004) propose a conceptual framework for AIS, illustrated in Figure 2.3 and Figure 2.4.

Figure 2.3 proposes a framework whereby observations and experiments can be used to form a view of the complex biological system (Stepney et al., 2004). Using the framework described by Figure 2.3, one can potentially produce many representations of the same system from the same observation, each providing different insights (Stepney et al., 2004). It is common to have distinct representations for the

same system, however these representations are rarely compared for unifying properties (Hart and Timmis, 2008). Hart and Timmis (2008) claim that, with a conceptual framework, one can begin to make such comparisons.

Stepney *et al.* (2004) go on to propose that different conceptual, mathematical and computational frameworks can be compared by using more integrated and generic frameworks. Figure 2.4 applies the same conceptual model as seen in Figure 2.3, but at a higher level (Stepney et al., 2004). The probes in Figure 2.4 are meta-questions. The type of meta-model one develops will depend on the meta-questions one asks (Hart and Timmis, 2008).

Hart and Timmis (2008) identify some initial areas to probe with meta-questions. These are known as ODISS questions (Timmis et al., 2010), and refer to:

1. **Openness**
   To what extent must the system be able to continuously grow and develop?
2. **Diversity**
   How many different types of agent are needed for the system?
3. **Interaction**
   To what extent and at what level must individual agents be able to communicate with one another?
4. **Structure**
   How is the system organised?
5. **Scale**
   How many agents are needed for the system?

The advantage of adopting this approach will be the development of immune algorithms that are grounded in immunology, more so than those that adopt a simple *observe and implement* approach (Timmis et al., 2010).

Some examples of meta-questions are given by Hart and Timmis (2008) for complex systems, population models and network models. However, most of the example meta-questions are not directly applicable to swarm robotic systems in their given context, for example; how scale affects diversity in a complex system. Some initial meta-questions for fault tolerant swarm robotic systems could be:

1. How do individual robot behaviours relate to collective swarm behaviour?
2. How and when do faults appear?
3. How do faults affect individual robot behaviours?
4. How do faulty individual behaviours affect collective behaviour?

2.3.2 Implementing Artificial Immune Systems

Millard (2016) highlights that maintenance in biological systems is comparable to fault tolerance in engineered systems, as both pertain to systems that can continue to operate under unusual or unexpected circumstances. Specifically, Millard (2016) highlights that the definitions used by Cohen (2000) for recognition, cognition and action map neatly to the control engineering sub-field of fault detection, fault diagnosis, and recovery (FDDR).

The review paper by Bayar *et al.* (2015) details the application of AIS with respect to FDDR. *et al.* (2015) conclude, unsurprisingly, that there are many different

AIS approaches for FDDR. Bayar *et al.* (2015) divide work on AIS into groups corresponding to their approach, for example; AIS based on self-non-self discrimination.

Self-non-self discrimination is one of many theories put forward to explain how the natural immune system distinguishes between what it needs to attack and what should be tolerated. This discrimination is important as an immune system that conflates the normal cells present in a host with pathogens, known as *autoimmunity*, will have the potential to kill its host (Langman and Cohn, 2000).

The self-non-self theory, as originally outlined by Burnet *et al.* (1959), proposes that the immune system can distinguish between cells that originate from and belong to the host, and those that do not not. Cells that are foreign will be targeted for destruction whilst native cells will be tolerated. The self-non-self model has evolved over time in order to accommodate new findings (Matzinger, 2002). Whilst this theory plausibly explains some aspects of how the immune system can defend its host from pathogens whilst remaining tolerant of the cells that constitute the host, other researchers have not always been satisfied that the self-non-self theory could provide comprehensive explanations for their observations, for example; the proposal by Lafferty and Cunningham (1975) that T-Cells needed 'co-stimulation' prompted Janeway (1992) to propose the *infectious non-self model*.

Each immune theory provides a different perspective or interpretation of the observed functions of the immune system.

Immune network theory

Provided are the definitions, as used by Timmis *et al.* (2008), for the immune terminology used in the following discussion:

- **Antigen presenting cells (APC)**: Portions of ingested pathogen are displayed on cell membrane as antigens to be recognised by T-cells
- **T-cells** and **B-cells**: The main actors in the adaptive immune system, both are lymphocytes
- **Lymphocytes**: The only cells that produce antigen receptors. Each lymphocyte produces a unique antigen receptor.
- Antigen receptors on T-cells are called **T-cell receptors (TCR)**
- Antigen receptors of B-cells are called **antibodies**

Jerne (1974) presents the *immune network theory*, an attempt to explain immune characteristics such as the ability to learn and remember. The immune network theory states that any lymphocyte receptor can be recognised by some subset of the total receptor repertoire – which themselves have their own recognising subset and so on. The product of this is an immune network, also referred to as an *idiotypic network*. Jerne (1974) proposes that emergent properties of the immune system, such as self-tolerance, are the result of the immune systems interactions with itself where there is no foreign antigen present.

Farmer, Packard and Perelson (1986) build on the hypothesis presented by Jerne (1974), using it as the basis for their simplified immune model. The simplified immune model, which consists of a set of differential equations (Equation 2.3) and is notably analogous to the classifier system described by Holland (1983), ignores the effects of T-cells and macrophages (a phagocytic immune cell) and focuses on B-cell population dynamics with respect to levels of B-cell stimulation (Timmis et al., 2008).

Farmer, Packard and Perelson (1986) give the rate of change for antibody concentration as:

$$\dot{x_i} = c\big[\sum_{j=1}^{N} m_{ji}x_i x_j - k_1 \sum_{j=1}^{N} m_{ij}x_i x_j + \sum_{j=1}^{M} m_{ji}x_i y_j\big] - k_2 x_i \tag{2.3}$$

The authors (Farmer, Packard, and Perelson, 1986) elaborate on the terms that make up equation 3.

- The first term represents the stimulation of the antigen-binding site of an antibody type i by the epitope of an antibody type j.
- The second term represents the suppression of antibody type i when its epitope is recognised by paratope type j
- Parameter c is a rate constant dependent on the frequency of collisions and the rate of antibody production stimulated by a collision.
- Match specificities m_{ij} take into account what reactions occur and how strongly.
- constant k_1 represents possible inequality between stimulation and suppression.

Equation 2.3 contains the primary influences, as proposed by Farmer, Packard and Perelson (1986), on B-cell stimulation level, namely; contributions of antigen binding, contributions of neighbouring B-cells and suppression of neighbouring B-cells. Included in the model are mechanisms for removing useless antibodies (a B-cell duplicates proportionally to its degree of stimulation) and maintaining diversity by introducing new ones via mutation (see Davis (1991) on genetic algorithms) (Farmer, Packard, and Perelson, 1986).

To implement an idiotypic network for an immune response in an engineered system would require an autonomous method for simulating antigen binding, which could be considered analogous to fault detection in this context. Furthermore, it would not be enough to simply detect the presence of a fault, in order to simulate binding specificities an autonomous method of isolating the fault type would be required, which could be considered analogous to fault diagnosis. Therefore an appropriate approach to autonomous fault detection and diagnosis is prerequisite in order to apply the immune network model to robot swarms. Of great interest, however, is the immune network theory's ability to learn and remember. This is clearly a desirable characteristic as it is a waste of time and computational power for an engineered system to solve problems it has previously found solutions to.

T-cells and the Crossregulation model

Although Farmer, Packard and Perelson (1986) ignore the presence of T-cells in their immune network model, T-cell population modelling appears in work towards fault tolerance in swarm robots, such as that by Tarapore, Christensen and Timmis (2017), which builds on the *Crossregulation model* (CRM) (Carneiro et al., 2007).

The CRM describes the population dynamics of three mutually interacting cell types found in the adaptive immune system (Tarapore, Christensen, and Timmis, 2017):

1. Antigen presenting cells (APC). Individual APC's have a fixed number of conjugation sites on which effector and regulatory cells can form conjugates.

2. Effector cells (T_E) that can potentially mount immune responses which may be directed to foreign pathogens or to body-antigens, depending on specificity.

3. Regulatory cells (T_R) that suppress proliferation of T_E cells with similar specificities

All discussion here refers to the example given by Carneiro *et al.* (2007) in which the CRM is described in one of its simplest, albeit unrealistic, mathematical forms. In this scenario APC's can form conjugates with, at most, two T-cells. Crossregulatory interactions between T_E and T_R cells can only occur on these multicellular conjugates. This can be characterised mathematically using differential equations describing the densities of T_E and T_R cell populations over time, with the same specificities (E and R).

$$\frac{dE}{dT} = p_e E_A - d_e E \tag{2.4}$$

$$\frac{dR}{dT} = p_r R_A - d_r R \tag{2.5}$$

where

- E_A is the density of activated T_E cells in multicellular conjugates
- R_A is the density of activated T_R cells in multicellular conjugates
- p_E is the proliferation rate of activated T_E cells
- p_R is the proliferation rate of activated T_R cells
- d_e is the death rate of T_E cells
- d_r is the death rate of T_R cells

The densities of activated T_E and T_R cells are calculated in sequence. The equilibrium density of T cells in conjugates, C, is a function of the total density of T cells, T, and the total density of APC conjugation sites, A. This is described in Equation 2.6.

$$C = \frac{1 + K(A+T) - \sqrt{(1+K(A+T))^2 - 4ATK^2}}{2K} \tag{2.6}$$

Where K is a constant and T is given by Equation 2.7

$$T = E + R \tag{2.7}$$

The densities of T_E and T_R conjugates, E_C and R_C, respectively, are given by the corresponding portions of the conjugate densities (see Equation 2.8 and Equation 2.9).

$$E_c = \frac{E}{T}C \tag{2.8}$$

$$R_c = \frac{R}{T}C \tag{2.9}$$

The fractions of conjugation sites per APC that are occupied by T_E and T_R cells at equilibrium, ϵ and ρ, are given by Equation 2.10 and Equation 2.11, respectively.

$$\epsilon = \frac{E_c}{A} \tag{2.10}$$

$$\rho = \frac{R_c}{A} \tag{2.11}$$

Lastly, the density of activated cells can be calculated using Equation 2.12 and Equation 2.12.

$$E_a = E_c\left(1 - \frac{2\epsilon}{2-\rho}\right) \tag{2.12}$$

$$R_a = R_c\left(\frac{2\rho}{2-\epsilon}\right) \tag{2.13}$$

The CRMs application to fault detection problems in swarms has already been demonstrated by Tarapore, Christensen and Timmis (2017), who devised a system that was able to reliably detect the presence of abnormal robot behaviours whilst remaining tolerant to normally behaving robots.

The danger model

Matzinger (2002) proposes the *danger theory*, which re-frames the self-non-self discrimination problem as dangerous vs. non-dangerous such that the immune system will only attack cells that it perceives as dangerous. This means that foreign bodies will only be attacked by the immune system if they are identified as being harmful to the host. Although this theory has its own shortfalls – its inability to adequately explain immune responses to grafts and tumors, for example – the principle of only responding to danger signals lends itself well to engineered systems (Pradeu and Cooper, 2012). From a practical point of view, applying this theory to swarm robotic systems represents a more efficient approach than trying to detect and handle every single abnormality that might occur in a system irrespective of whether or not the systems collective behaviour is being disrupted.

Summary

This section has discussed a number of different immune theories. Some immune theories, such as the CRM model (Carneiro et al., 2007), have been demonstrated to be directly applicable to FDDR problems (Tarapore, Christensen, and Timmis, 2017). Other immune theories, such as the immune network theory (Jerne, 1974) and the simplified immune model (Farmer, Packard, and Perelson, 1986), require an approach to FDDR or something analogous before they can be realised. The following section will discuss the previous attempts by other researchers towards implementing FDDR, or some part thereof, in robotic systems.

2.4 Fault detection

Although there has been a large amount of research on FDDR in engineered systems, comparatively little has been conducted with swarm robot platforms (Zhang and Jiang, 2008).

Reliable fault detection is the first step in any active FDDR approach. There are two different approaches to fault detection; *endogenous* and *exogenous*.

2.4.1 Endogenous fault detection

A robot that detects faults in itself detects them endogenously (Christensen, O'Grady, and Dorigo, 2009). Within endogenous fault detection, there are generally two approaches; *model-based* methods also known as analytic method, and *model-free* methods, also known as data driven methods (Christensen, 2008).

Model-based fault detection compares a systems actual behaviour to a predefined model of how the system should behave. Deviations from this model can then be interpreted as faults or symptoms thereof (Christensen, 2008). A more detailed description and discussion of model-based fault detection approaches can be found in the review paper by Isermann (2005). Christensen (2008) acknowledges the difficulty associated with producing accurate analytical models on account of environmental uncertainties, noisy sensors and imperfect actuators.

The data-driven approach similarly detects faults based on the differences between predicted and observed data. However, predicted data is based on observations of the system during its operation (Christensen, 2008), for example; Marsland, Nehmzow and Shapiro (2005) use unsupervised learning to train a filter which ignores data similar to that which it has already seen. Alternatively, AIS based fault detection would also fall under the category of data-driven, such as the danger-model based AIS for fault detection proposed by Laurentys, Palhares and Caminhas (2010).

Christensen (2008) acknowledges that there are some scenarios in which a robot may not be able to detect faults in itself, or, even it can, may be unable or unwilling to communicate the presence of faults to the rest of the swarm. Such scenarios would make purely endogenous approaches to fault detection unreliable. Christensen (2008) identifies some examples of these scenarios as battery failures, main board short circuiting and bugs that cause software hang, concluding that it is necessary for multi-robot systems to exploit exogenous fault detection if they are to realise their fault tolerant potential.

2.4.2 Exogeneous fault detection

Exogenous fault detection occurs where one robot is able to detect a fault occurring in another, physically separate robot (Christensen, 2008).

Parker (1998) presents ALLIANCE, a fault tolerant control architecture for small-medium sized teams of heterogeneous robots.

Using a hazardous waste cleanup mission as a case-study, Parker (1998) applies the ALLIANCE architecture to a team of three R-2 robots, purchased commercially from IS Robotics. To complete such a task co-operatively, a series of individual subtasks must be performed, such as locating the site of hazardous waste (represented in this experiment by a movable object), removing hazardous waste upon location and communicating with other team members. ALLIANCE achieves fault tolerance by allowing individual robots to recognise what task other team members should be performing and, if it is decided that another robot is not performing its task quickly enough, the individual can assume the task itself. Strictly speaking, although both are approaches to fault tolerance, ALLIANCE does not represent an approach to FDDR. Furthermore, the test case could not be considered swarm robotic (Şahin, 2005). However, the act of an individual acknowledging that another robot is not performing a task effectively and the individuals subsumption of that task has a clear applicability to active fault tolerance in swarm robot systems.

Gerkey and Mataric (2002) present MURDOCH, which employs an auction system for allocating different tasks among multi-robot systems. In this system, one robot acts as an 'auctioneer' and announces a task to the other robots in the group. The announcement will only be received by individuals that are capable of performing the task. Each capable individual can then 'bid' on the task. The strength of the bid will be determined by a fitness function for the individual. The fittest individual(s) will then be given a 'contract' whereby they are allocated the task and given a time to complete it in. The auctioneer robot then monitors the progress of the winning robot(s) and, assuming significant progress is made, 'renews' the contract. MURDOCH achieves fault tolerance by the fact that an individual, or group of individuals, that do not perform their task well enough will not have their contracts renewed. The task will then be reallocated to a different robot, and the under–performing individual(s) will not be considered for subsequent tasks.

MURDOCH is validated on a team of 3 AcitvMedia Pioneer 2-DX mobile robots, the specifications of which can be found in Gerkey and Mataric (2002).

There are several parallels we can draw between MURDOCH and ALLIANCE.

- Both approaches increase the fault tolerance of their respective systems, despite neither one representing an approach to FDDR
- Both achieve fault tolerance by reassigning tasks from underperforming individuals
- Both approaches use multi-robot systems as test cases which could not be considered as 'swarm robotic' because of the small number of robots used (3 in each case) (Şahin, 2005). Furthermore, the robots used rely on external sensors and complex communication protocols, for example; ALLIANCE relies on predefined system-knowledge of specific task ordering dependencies; MURDOCHs use of a single 'auctioneer' robots means the system can be considered centralised, and therefore subject to the associated drawbacks.

Although some of the fault tolerance principles employed by ALLIANCE and MURDOCH may be useful for active FDDR, because the systems to which they are applied cannot be considered swarm robotic systems, they do not, as they are, represent a solution to the problem of fault-tolerant swarms as highlighted by Winfield and Nembrini (2006) and Bjerknes and Winfield (2013).

2.4.3 Fault detection in swarms

Fault detection in robot swarms has previously been approached in a number of ways in software simulation and in hardware.

Firefly fault detection

Christensen, O'Grady and Dorigo (2009) propose that once can achieve fault tolerant robot swarms by the removal of faulty robots from the swarm. In this work, faults are exogeneously detected using a firefly-inspired system whereby robots are each equipped with periodically flashing light-emitting diodes (LEDs) which they try to synchronise with one–another. A fault is detected where an individual robot fails to flash its LED at suitable intervals.

This method of fault detection is shown to be effective, being able to reliably detect faults in a comparatively short space of time. However, the case-study considered by Christensen, O'Grady and Dorigo (2009) is limited. In order to use the

fault detection method proposed in Christensen, O'Grady and Dorigo (2009) for the type of scenarios described by Şahin (2005), an individual robot would need to have its own reliable endogenous fault detection method so that it knew to stop flashing its LED if, for example, one of its motors failed.

Internal simulator approach

The work by Millard (2016) presents an exogeneous method of detecting faults in swarm systems at run-time. This work proposes that faults in individual robots can be detected by comparing their real behaviour with their simulated behaviour. This is achieved in two ways, firstly; through predicting the future behaviour of a robot, and secondly; by analysing a robot's past behaviour. In any case, each robot has a simulated copy of the controller for every other robot in the swarm. A fault is detected where there is discrepancy between the information observed by a robot about its neighbours and the internally simulated information about that neighbour. Millard (2016) ultimately concludes that fault detection by predicting future behaviour is infeasible, and favours fault detection by analysis of past behaviour.

By comparing a robots observed position with it's expected position after a period of time, Millard (2016) is able to detect faults correctly for a large portion of the time they are present, whilst maintaining a general tolerance to normal behaviours.

Neighbour comparison approach

Khadidos, Crowder and Chappell (2015) describe an approach to exogeneous fault detection and recovery in swarm robots. The experiments described within are all carried out in simulation and use simulated e-puck robots.

Each robot in Khadidos, Crowder and Chappell (2015) broadcasts information from its sensors, position coordinates and motor values to its neighbour, and vice-versa. Based on this information, each robot can decide where it believes its neighbour should be relative to itself by calculating the distance and angle between each others coordinates. If there are discrepancies between where neighbours believe each other are then both will be reported as suspicious. At this point, a third robot will be called to the disagreeing neighbours, which will act as an independent adjudicator. After the adjudicator robot is subject to the same fault detection process with each of the original neighbours, the faulty robot should be identifiable. Once a faulty robot is detected it has its power turned off, effectively making it an object in the arena.

The fault detection method described by Khadidos, Crowder and Chappell (2015) works well in simulation. The decentralised manner of the fault detection mechanism, combined with the relatively small proportion of the swarm required to perform it, means that the approach can be considered scalable. Furthermore, the comparison of multiple data sets should theoretically make the approach robust and flexible.

Crossregulation model approach

Tarapore, Christensen and Timmis (2017) use a stochastic, spatial, discrete-time multi-agent system simulator to demonstrate abnormality detection using a system based on the CRM (Carneiro et al., 2007). This considers a swarm of simulated ground-based mobile agents that can exhibit a number of behaviours, into which faults are injected.

The CRM in this instance is implemented such that behaviours that are persistent and abundant among individuals are tolerated, whilst rare behaviours are detected as abnormal.

This work interprets agent behaviours as a series of Boolean features that form a binary string - where 0 and 1 indicate an absent or present behavioural feature, respectively. This is referred to as the *feature vector*. The CRM provides a system of differential equations that consider the distribution of feature vectors as inputs before deciding whether or not an agents feature vector should be tolerated or detected as abnormal.

The CRM-based approach is shown to be effective at detecting abnormalities in swarm behaviour. Encoding robot behaviours as feature vectors also provides this approach with an innate degree of flexibility – features can be added or taken away in sympathy with the requirements of a system, and the CRM-based approach should, in theory, still provide effective fault detection. A feature vector-based approach is therefore one that might lend itself well to approaches to active fault tolerance in robot swarms in the long-term.

2.4.4 Summary

This section has provided an overview of approaches to exogenous fault detection in multi-robot systems and in swarm robotic systems. In each approach, the fundamental mechanism that enables a fault to be detected in a robot is the comparison of observed robot behaviour with expected robot behaviour, where a significant discrepancy indicates the presence of a fault.

2.5 Fault diagnosis

It was highlighted in Chapter 1 of this thesis that fault diagnosis is a necessary step of active fault tolerance if robot swarms are to retain long-term autonomy. Additionally, discussion in section 2.3.2 of this chapter highlighted that a process analogous to fault diagnosis is required for implementation of some immune theories. Any model that attempts to capture the desirable characteristics of the natural immune system, specifically learning and memory, must have some degree of ability to identify a particular fault type and it's associated recovery procedure as distinct from other fault types with alternative recovery procedures.

There have been a number of approaches to fault diagnosis in multi-robot systems. Although the fault diagnosis approaches discussed in this section are not explicitly designed for swarms, the described diagnostic techniques inform the work towards fault diagnosis in robot swarms.

Winfield and Nembrini (2006) demonstrated that different types of fault cause robot behaviour to deviate from what is expected in different ways. As faults can be detected by observing the discrepancies between expected and observed robot states, faults can be diagnosed by examining what specific states or features are discrepant. For example, Daigle, Koutsoukos and Biswas (2007) use the discrepancies between model-predicted behaviour and observed robot behaviours to create a 'fault signature' for different types of faults, which can then be used to diagnose faults that occur in real-time. Similarly, Cassasco, Núñez and Cipriano (2011) model the behaviour of normally functioning and faulty robots offline. Faults are detected based on discrepancies between the model-predicted normal robot state and measured robot states during operation. Faults are then isolated by which modelled

state for faulty behaviour most closely resembles the measured robot state. One limitation of such approaches is their lack of capability to learn or adapt to dynamic fault signatures, and their reliance on comprehensive prior modelling in order to be effective.

Faults can also be diagnosed through more explicit assessment. Kutzer *et al.* (2008) implement diagnosis such that faulty robots perform diagnostic manoeuvres, consisting of various tests designed to isolate the root-cause of a fault. A trained probabilistic model is then used to estimate the state of the faulty robot based on its observed performance of the diagnostic manoeuvres.

To the best of the author's knowledge, no attempt has yet been made towards fault diagnosis with explicit regard to swarm robotic systems, with the exception of the author's previously published work (O'Keeffe et al., 2017b) (O'Keeffe et al., 2017a) (O'Keeffe et al., 2018).

2.6 Summary

This chapter has highlighted the motivation for active fault tolerance in swarm robotic systems, and has provided discussion the attempts towards addressing this problem by other researchers. The relatively small body of work discussed in this chapter towards active fault-tolerance for robot swarms, to the best of the author's knowledge, is comprehensive. This chapter has highlighted the necessity of a fault diagnosis mechanism for enabling robot swarms to retain long-term autonomy. The absence of previous work towards this particular problem means that, until it is addressed, swarm robotic systems will be unable to realise the potential utility and applications listed by Şahin (2005).

Chapter 3

Fault Classification With Feature Vectors

3.1 Introduction

The previous chapter discussed the approaches adopted by other researchers towards improving fault tolerance in robot swarms. Of particular relevance to this work is the behavioural feature vector (BFV) approach taken by Tarapore, Christensen and Timmis (2017). An advantage of this approach, highlighted by Tarapore, Christensen and Timmis (2017), is that faults can be reliably detected without needing an *a priori* model of normal behaviour. In this way, the BFV-based approach possesses an innate degree of flexibility and adaptability.

As Winfield and Nembrini (2006) demonstrated that different faults cause robots to deviate from normal function in different ways, this chapter tests the hypothesis that, by encoding robot behaviour as BFVs, these deviations from normal behaviour will be reflected in a faulty robot's BFVs, and that these will be distinct for different fault types. If this is the case, then there are a variety of machine learning approaches that could be employed to classify different types of faults occurring in robots based on the faulty robot's BFV. This could conceivably be performed autonomously by other robots in the swarm in a distributed fashion, as is desired.

The following chapter describes the initial work conducted for this thesis, which investigated the feasibility of using BFVs for fault diagnosis in robot swarms by training a decision tree on faulty robot BFVs, which could then be used to classify different fault types. The purpose of the work described in this chapter was to obtain proof of principle for BFV-based fault diagnosis, rather than propose a developed method for fault diagnosis.

3.2 Behaviour Characterisation

To detect and classify faults in a robot based on its BFVs relies on the BFVs adequately reflecting the specific deviations from normal robot behaviour caused by different faults. Assuming access to the relevant hardware and software documentation, it would conceivably be possible to reduce each robot functionality to a series of base behaviours. For example, for ground based vehicles, individual features could indicate whether or not its wheels are turning and, if so, which wheel(s) and their speed(s).

To exhaustively reduce robot functionalities to their most basic components may prove to be a time consuming process and, for the purposes of fault tolerance, may not be necessary if it does not produce *discriminating features*. Discriminating features allow a system to distinguish different fault types based on the features presence,

absence, or in more complex systems, its intensity. To illustrate, a complete sensor failure and power failure would both result in an individual robot becoming unresponsive to its neighbours. However, a sensor failure would not necessarily prevent the robot from moving, whereas a power failure would. A feature describing the motion of the robot could therefore be considered discriminatory in this case.

For an individual robot's BFV to be discriminatory, it is necessary that it is designed in sympathy with the behaviour(s) that the robot will exhibit. The initial approach taken for this work was to attempt to atomise high-level robot behaviours into low-level sub-behaviours. By reducing known behaviours into sub-behaviours of an appropriate granularity and assigning features to each, the resultant repertoire of BFVs should then be representative of every behaviour the system can exhibit.

3.3 Experimental Method

The initial work towards validation of BFV-based methods for fault classification was conducted using Autonomous Robots Go Swarming (ARGoS) (Pinciroli et al., 2011), a discrete-time physics-based robot swarm simulator. Use of a robot simulator was considered to be the fastest route to obtaining a proof-of-concept for this approach to fault classification, but with the intention of demonstrating any such system in hardware at the soonest reasonable opportunity.

This work uses simulated models of the marXbot (Bonani et al., 2010) (see Figure 3.1) swarm robotic platform. These are two-wheeled, $17cm$ diameter swarm robots, with on-board proximity and range and bearing (RAB) sensors.

To be useful in swarm robotics systems, it is important that any fault classification technique is compatible with the swarm's requirement to be flexible, and should therefore be applicable to any range of behaviours one might expect a given swarm system to exhibit. Flocking, aggregation, and dispersion behaviours are examined as a case study (see Algorithm 1, Algorithm 2, and Algorithm 3), as the behaviours are widely used in swarm robotics research. These three behaviours can each be considered similar at an individual level, sharing simple functionalities, yet produce significantly different collective behaviours – satisfying the behavioural conditions for swarm robotics systems described by Şahin (2005).

Algorithm 1 Flocking

1: **while** Running **do**
2: **if** Distance to object $<$ close-proximity threshold (C) **then** avoid object
3: **if** Average neighbour distance $<$ flocking threshold (k) **then** calculate target heading from the average bearings of all neighbours in range
4: **else** Calculate target heading from the average positions of all neighbours in range
5: **if** Robot heading **not in range** target heading $\pm$ 15 $^\circ$ **then** turn toward target heading
6: **else** Move forward

The flocking threshold, k, in Algorithm 1 is a user-defined threshold that indicates the point at which a robot's desire to be close to the swarm is over-ridden by its desire to be travelling in the same direction as the swarm (approximately 8cm from the average neighbour position). The close-proximity threshold, C, which appears in Algorithm 1, Algorithm 2 and Algorithm 3 sets the distance at which a robot will avert its course to avoid collision (approximately 3cm). The precise values of k

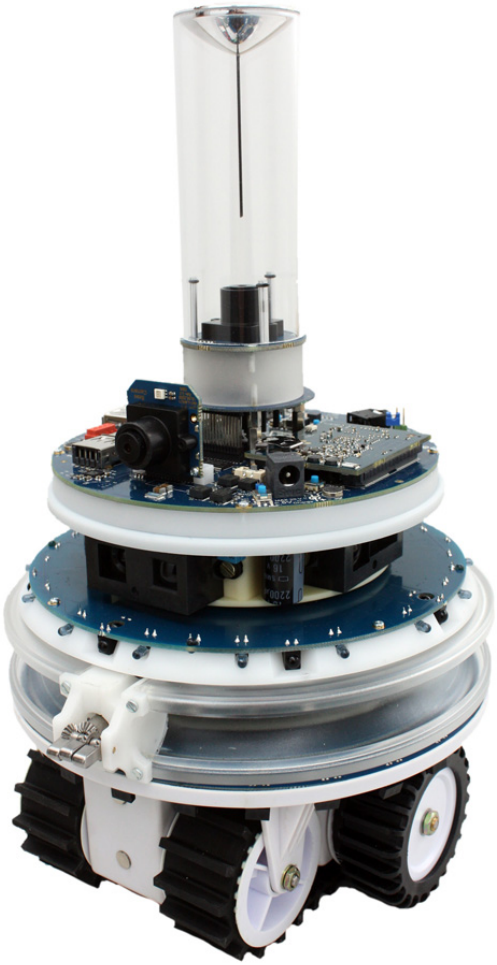

FIGURE 3.1: The MarXbot swarm robotic platform (Bonani et al., 2010)

Algorithm 2 Aggregation

1: **while** Running **do**
2: **if** Distance to object $<$ close-proximity threshold (C) **then** avoid object
3: Calculate target heading from the average positions of all neighbours in range
4: **if** Robot heading **not in range** target heading $\pm$ 15 $^\circ$ **then** turn toward target heading
5: **else** Move forward

Algorithm 3 Dispersion

1: **while** Running **do**
2: **if** Distance to object $<$ close-proximity threshold (C) **then** avoid object
3: **else** Move forward

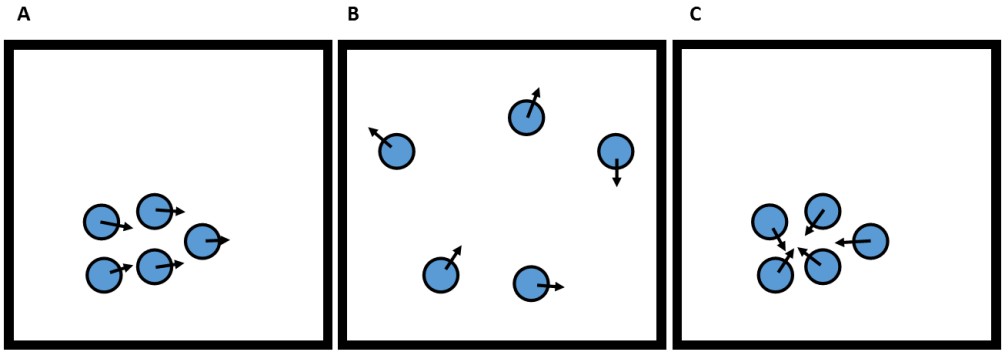

FIGURE 3.2: A diagram to illustrate flocking, aggregation and obstacle avoidance behaviours **A:** Flocking behaviour. **B:** Obstacle avoidance behaviour. **C:** Aggregation.

and C were decided upon with consideration to the scales of the particular robots and arena used for this work. Both k and C are estimated by each robot using its RAB sensor.

Each behaviour is illustrated in Figure 3.2.

3.3.1 Fault Types

Techniques for analysing the types of faults that may occur in engineered systems, such as Failure Mode and Effect Analysis (Dailey, 2004) or Fault Tree Analysis (Ericson and Ll, 1999), are not directly applicable to simulated models of robots in the absence of actual robot hardware. The fault types considered for a simulated model of a marXbot robot (Bonani et al., 2010) are therefore based on a cross-section of the literature that informs this work. Consequently, some of the fault types are not necessarily chosen with explicit regard to the marXbot platform, but with respect to an as-yet undefined swarm robotic system that could conceivably be deployed in the circumstances described by Şahin (2005).

- **Software hang (H_1)**:
  Software hang is given as an example of a fault that necessitates exogenous approaches to fault detection (Christensen et al., 2008). A software hang fault will cause a robot to become stuck performing whatever action it was performing at the last moment it was normally functioning. For this implementation, the robot is able to continue to broadcast its own BFV to the swarm (although this BFV will also be stuck and unresponsive to changes in the robot's behaviour from external factors).

- **Power failure (H_2)**:
  A recurring example in fault tolerant literature (Winfield and Nembrini, 2006) (Carlson, 2004). A robot that suffers a power failure will completely stop moving and remain unresponsive to its surroundings. Neighbouring robots in the swarm will still be able to detect its presence. This assumes that each robot is able to detect the presence of its neighbours independently of whether or not that neighbour is responsive. This may not be the case for all systems, however, in cases where robots are unable to communicate their presence for one

reason or another, they will go unacknowledged by the swarm and become objects in an arena for all practical purposes. Such cases have been shown to have little to no detriment to collective swarm behaviour (Winfield and Nembrini, 2006), and so are therefore not a priority for active approaches to fault tolerance.

- **Complete Sensor failure (H_3**:
  Sensor failure is another recurring example used in fault tolerant swarm research (Winfield and Nembrini, 2006) (Carlson, 2004). For a complete sensor failure an individual robot's RAB sensor, as well its IR sensors, will completely fail and return a value of 0. The robot will be unable to detect the presence of neighbouring robots or objects.

- **Complete Motor failure (H_4)**:
  The study by Winfield and Nembrini (2006) demonstrates motor failure in individual robots to be the most damaging to collective swarm behaviour. It is therefore an obvious choice for inclusion in this work. For a complete failure the motor will stop and henceforth become unresponsive to its controller.

- **Partial Motor Failure (H_5)**:
  For a partial motor failure, the motor will remain responsive but will only cause its associated wheel to turn at half speed.

- **Partial Sensor Failure (H_6)**:
  Similar to the definition used in Milalrd (2016), a partial sensor failure affects a robot such that it will only be able to detect the presence of neighbours and objects within $\pm 45\,^{\circ}$ of its current heading.

3.3.2 Deriving Behavioural Features from Robot Behaviours

The individual robot BFV is defined by considering every instance in Algorithm 1, Algorithm 2 and Algorithm 3 in which a robot executes an action, and then assigning one feature to reflect that action and one feature to reflect the conditions that necessitate it. Following this process leads to the following BFV that is proposed to be sufficiently representative of the aforementioned behaviours. Each feature is binary, where a returned value of 1 or 0 indicates the presence or absence of the feature, respectively. The robot BFV used in this work, where the binary vector $BFV(t) = [F_1(t), F_2(t), F_3(t), F_4(t), F_5(t)]$, can be described as follows:

$$F_1(t) = 1 \text{ if } N_R(t) > 0, \text{ otherwise } F_1(t) = 0 \tag{3.1}$$

where N_R is the total number of neighbours in sensing range of the robot at time t. F_1 is illustrated in Figure 3.3.

$$F_2(t) = 1 \text{ if } \ N_C(t) > 0, \text{ otherwise } F_2(t) = 0 \tag{3.2}$$

where $N_C(t)$ is the total number of neighbours at a distance less than the close proximity threshold, C, to the robot at time t. F_2 is illustrated in Figure 3.4.

$$F_3(t) = 1 \text{ if } |v(t)| > 0.8|v_{max}|, \text{ otherwise } F_3(t) = 0 \tag{3.3}$$

where $|v|(t)$ is the magnitude of linear velocity at time t. F_3 is illustrated in Figure 3.5.

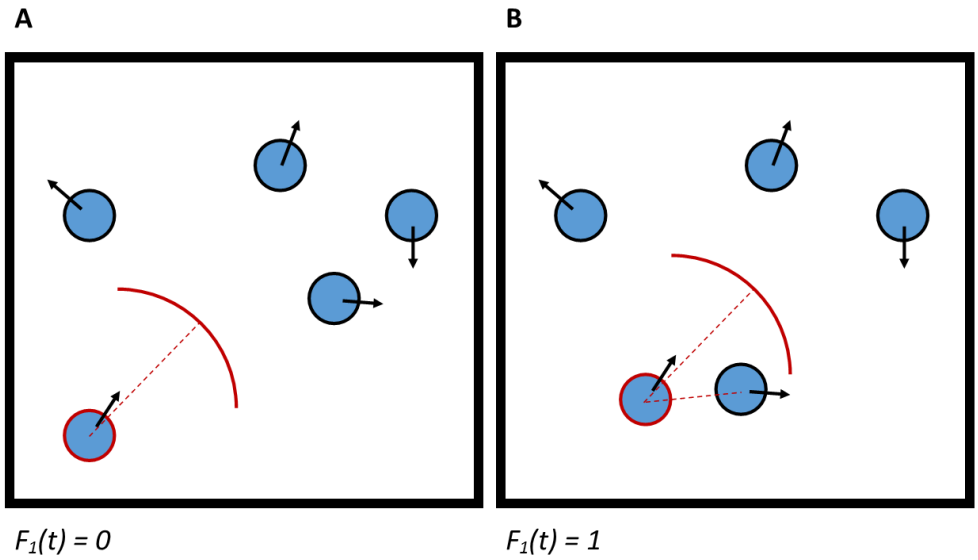

FIGURE 3.3: A diagram to illustrate F_1. Red highlight indicates the robot under consideration and its sensing range. **A:** The robot has no neighbours in sensing range. **B:** The robot has one neighbour in sensing range.

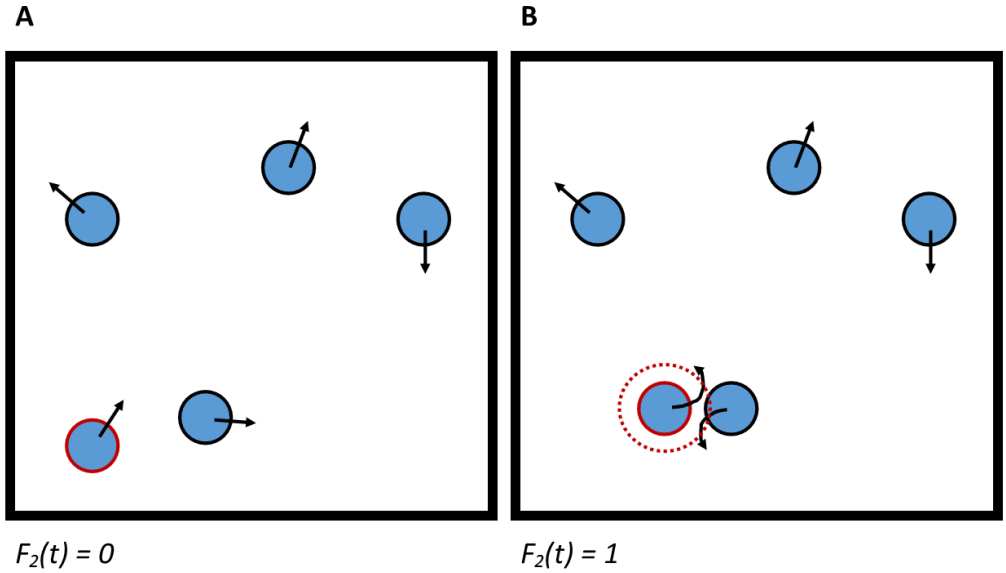

FIGURE 3.4: A diagram to illustrate F_2. Red highlight indicates the robot under consideration and its close proximity threshold. **A:** The robot has no neighbours within its close proximity threshold. **B:** The robot has one neighbour within its close proximity threshold, and so they both turn to avoid each other.

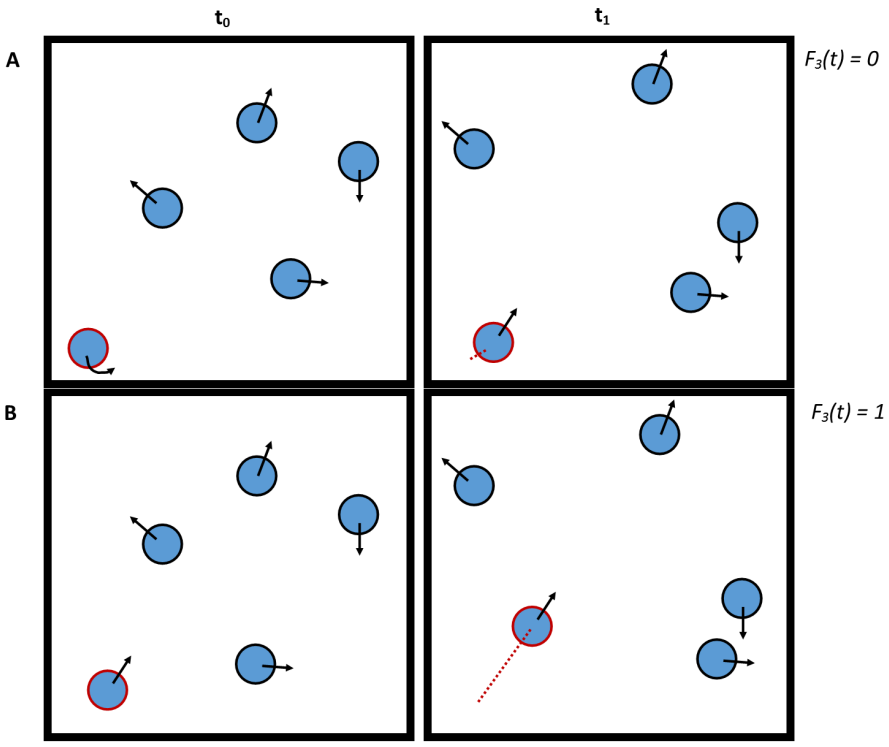

FIGURE 3.5: A diagram to illustrate F_3. Red highlight indicates the robot under consideration. **A:** Between times t_{0-1} the robot was turning, and so moved a relatively short distance ($F_3 = 0$). **B:** Between times t_{0-1} the robot was moving in a straight line at full speed, and so moved a greater distance ($F_3 = 1$).

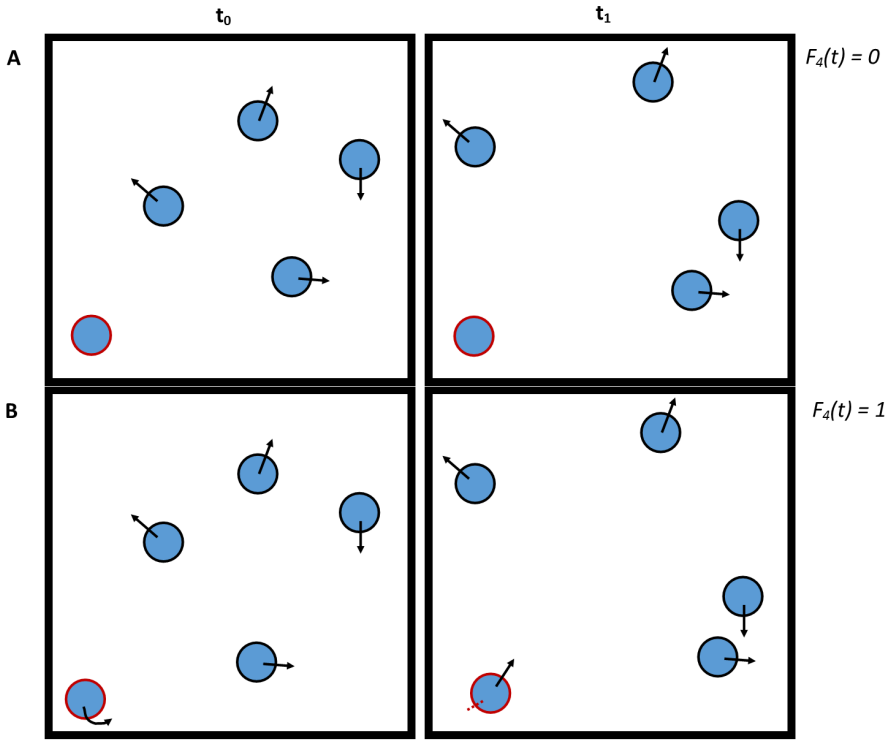

FIGURE 3.6: A diagram to illustrate F_4. Red highlight indicates the robot under consideration. **A:** Between times t_{0-1} the robot did not move at all ($F_4 = 0$). **B:** Between times t_{0-1} the robot did move, albeit a short distance ($F_4 = 1$).

$$F_4(t) = 1 \text{ if } |v(t)| > 0.2|v_{max}|, \text{ otherwise } F_4(t) = 0 \tag{3.4}$$

F_4 is illustrated in Figure 3.6.

$$F_5(t) = 1 \text{ if } |\omega(t)| > 0.4|\omega_{max}|, \text{ otherwise } F_5(t) = 0 \tag{3.5}$$

where $|\omega(t)|$ is the magnitude of the robots angular velocity. F_5 is illustrated in Figure 3.7.

The combination of features $F_3(t)$, $F_4(t)$ and $F_5(t)$ allow for a distinction to be made between a robot that has completely stopped, one that is turning and one that is moving in a straight line. The thresholds for these features were chosen with consideration to robot behaviours and system noise. Given that a normally behaving robots wheels can only be in one of two states (moving at maximum speed or not moving), a normally behaving robot is either moving forward with maximum linear velocity, turning with maximum angular velocity (and approximately half maximum linear velocity), or stationary.

The threshold for F_3 is defined to be 80% of v_{max}. In a noiseless system it could have been set to v_{max}, however, even with noise, there is no scenario in which a normally behaving robot would ever reach this velocity if it were not moving in a straight line. Therefore the presence of F_3 is the indicator for that particular behavioural state.

The threshold for F_4 is defined to be 20% of v_{max}. Similarly to F_3, this could have been set to 0 in a noiseless system. The purpose of this feature is to distinguish a

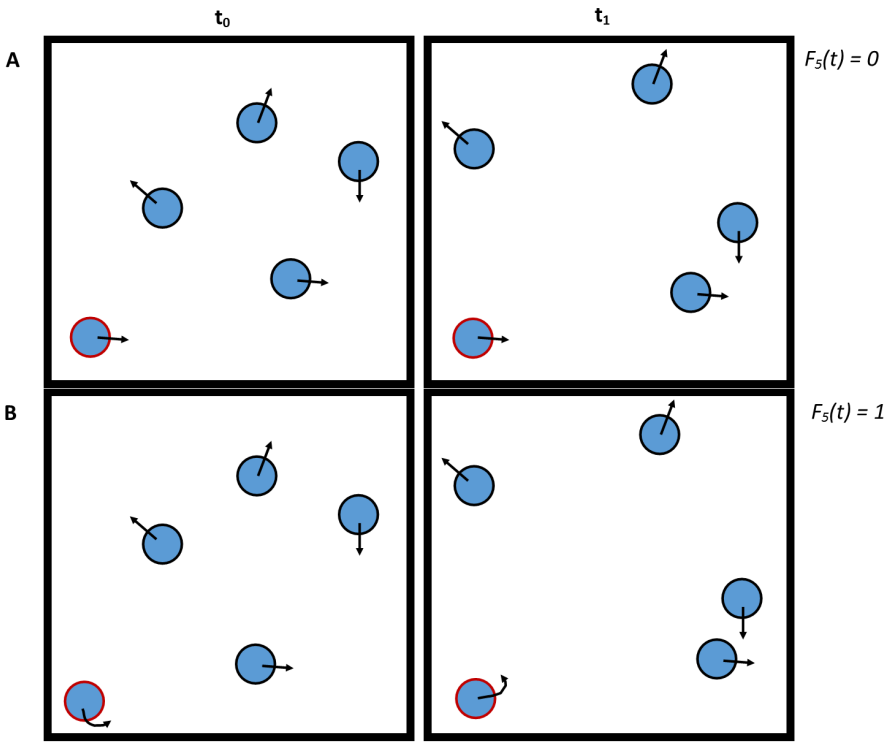

FIGURE 3.7: A diagram to illustrate F_5. Red highlight indicates the robot under consideration. **A:** Between times t_{0-1} the robot moved in a straight line and therefore with no angular velocity ($F_5 = 0$). **B:** Between times t_{0-1} the robot was turning, and therefore moved with maximum angular velocity ($F_5 = 1$).

robot which has completely stopped. Although a robot with $0 < v < 0.2 * v_{max}$ is obviously not stationary in absolute terms, for this experiment there is no scenario where a normally behaving robot should ever have a velocity in this range once its behaviour has stabilised. Therefore, if a robots velocity does lie in this range, it can be attributed to sensor noise about a stationary robot. Consequently, the absence of F_4 is the sole indicator that a robot has stopped moving. F_4 was defined in anticipation of the effects of faults on robot behaviour (particularly power failure), which makes it unique in the BFV for the work described in this chapter, as all other features were defined with respect to normal robot behaviour. Although it was unknown at the time F_4 was originally defined, the results of the work in this chapter demonstrated that consideration of the effects of faults on robot behaviour is a necessary aspect of effective BFV design.

The purpose of F_5 is to distinguish a robot that is turning. Given that a normally behaving robot is either turning with maximum angular velocity or not turning, this threshold could lie anywhere between 0 and ω_{max} in a noiseless system. In this work the threshold was set to be as low as possible whilst retaining a reliable reading.

Notably, there is not a feature to reflect the presence of a non-neighbour object. The reason for this is that every feature must be reliably measured internally by an individual, as well as externally by an observing neighbour. It is not immediately clear how this feature could be measured by an observing neighbour in a distributed manner, and so it has been omitted from this particular BFV.

Feature Estimation

The problems associated with endogenous fault detection, such as a robot being unable to communicate critical information to the swarm (Christensen et al., 2008), also applies to fault diagnosis. Using the example of a software hang fault, the faulty robot would report a BFV that did not reflect its true behaviour, potentially leading to misclassification. The same could also be said of purely exogenous approaches when applied to fault diagnosis. It would not always be possible for an independent observer to reliably distinguish between a robot with a software hang fault that renders it unresponsive to its neighbours and a complete sensor failure, as both could have the same net effect on the robot's behaviour.

This work proposes that a potential solution to this problem is by estimating robot BFVs proprioceptively *and* exteroceptively, and combining the two estimations. To achieve this, the proprioceptive BFV is estimated such that it reflects what an individual robot *believes* its state is, and the exteroceptive BFV such that it reflects what its neighbours ascertain its state to *actually* be. This work anticipated that multiple methods of feature estimation will not only improve the reliability of reported behaviour, but also represent a useful starting point for a classification process, as some faults will create discrepancies in features directly relating to the ways in which they manifest. For example, motor failure will a cause discrepancy in features relating to motion where one feature reflects the inclination of the robot's controller and the other reflects the robot's true movement.

3.3.3 Experiments

The following experiments were conducted to ascertain how useful BFVs are for classifying fault types, where usefulness is indicated by the proportion of correct fault classifications made by a decision tree trained on BFV data.

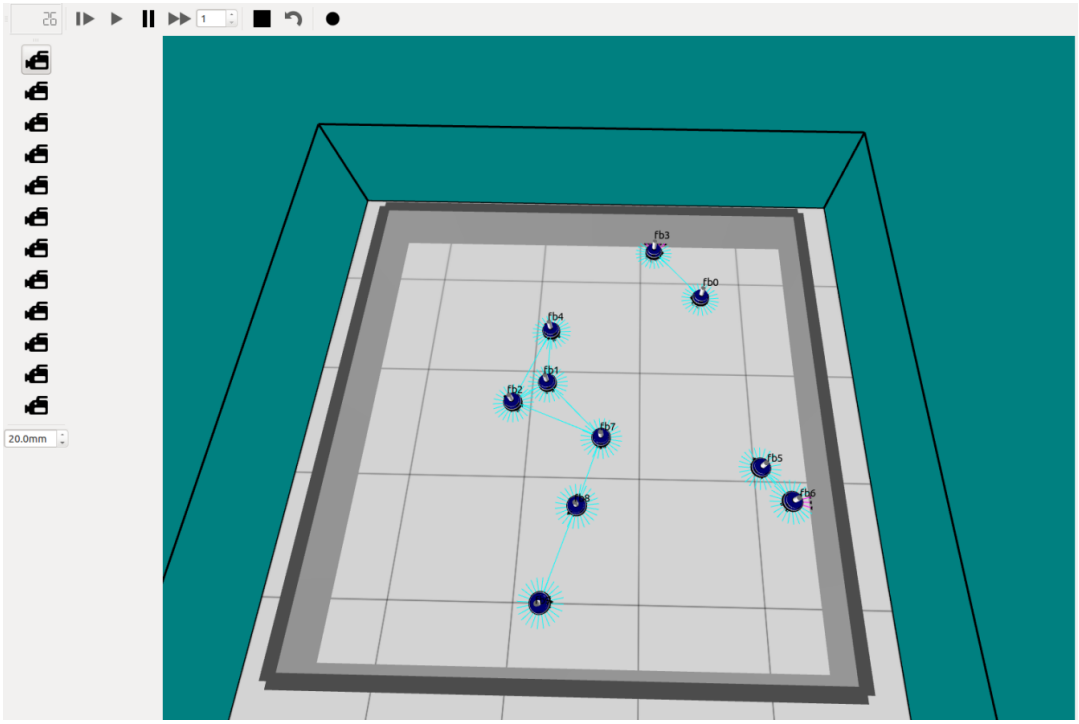

FIGURE 3.8: ARGoS simulation of 10 marXbot robots performing a dispersion behaviour. Lines connecting pairs of robots represent mutual neighbour acknowledgement via RAB sensor. Lines protruding from each robot represent IR sensor range.

A swarm of 10 marXbots (Bonani et al., 2010) collectively perform one of the three behaviours described by Algorithm 1, Algorithm 2 and Algorithm 3 in an otherwise empty enclosed arena measuring 16 square metres (see Figure 3.8).

BFVs are estimated proprioceptively by each robot using its RAB sensor (for features relating to object/neighbour proximity) or its controller (for features relating to a robot's movement). BFVs are estimated exteroceptively using a virtual sensor to obtain co-ordinate information for each neighbouring robot at each time step, allowing angular and linear velocities for those neighbours to be estimated. Using robot co-ordinate information is not traditionally 'swarm-like', and is not proposed as a long term solution. Rather, the use of a virtual sensor facilitates experimentation by providing the data that a robot could obtain locally with the appropriate hardware and sensing capabilities. As the main objective of this work is to highlight the merit of a BFV-based approach to fault classification, and there is intent to develop the system described here towards something more swarm-like at the soonest opportunity, this work meets the criteria for swarm robotics systems described by Şahin (2005).

Robot BFVs are updated once per control-step, at a frequency of $10Hz$. Simulated Gaussian noise is added to each robot's estimation of its neighbour's coordinates (μ=0, $\sigma = 2\%v_{max}$) and its RAB sensor (μ=0, σ=1 °). The effects of varying levels of noise were investigated in later experiments. The system noise applied in this work was selected on a preliminary basis, intended to ensure that any proof-of-concept obtained for BFV-based fault classification was robust to imperfect robot information.

In each experiment, the swarm performs one of the three behaviours (Algorithm 1, Algorithm 2, Algorithm 3) for five minutes of simulated time. One of the six

fault types (H_{1-6}) is injected into a single robot after 50 seconds of simulated time, as this is approximately when collective swarm behaviour stabilised on average for each type of behaviour. The BFV of each robot is recorded for the duration of the experiment and written to an external file.

This process is iterated for every type of fault and behaviour combination. The set of data, consisting of the faulty robot's BFV for each of the 6 fault types whilst it performs (or attempts to perform) one of the behaviours is then used to train a decision tree (Quinlan, 1986). The decision tree is trained using 100 replicates of each behaviour-fault combination. In the initial published version of this work (O'Keeffe et al., 2017b), only 1 replicate was used for each behaviour-fault combination. The change to 100 was prompted by the consistency analysis in the subsequently published work (O'Keeffe et al., 2017a), which is described in Chapter 4 of this thesis. The advantage of using a decision tree for this process over alternative machine learning approaches – neural networks, for example – is that a decision tree allows a user to easily ascertain whether or not a feature is discriminatory.

Subsequent experiments are performed for every behaviour-fault combination. The recorded BFVs of the faulty robot are then used as inputs to the trained decision tree so that the effectiveness of BFVs as a medium for fault classification can be observed.

Consideration was not given to the behaviour of non-faulty robots in the swarm, as noticeable differences in non-faulty robot behaviour were not observed in the cases tested.

3.4 Results and Discussion

Table 3.1, Table 3.2 and Table 3.3 show the average classification distribution of each fault type for a decision tree trained on aggregation behaviour when it is applied to the BFVs produced by faulty robots attempting to perform aggregation, flocking and obstacle avoidance behaviours, respectively. Table 3.4, Table 3.5 and Table 3.6 display the equivalent information for a decision tree trained on flocking behaviour, whilst Table 3.7, Table 3.8 and Table 3.9 display the equivalent information for a decision tree trained on obstacle avoidance behaviour. 300 replicate experiments were performed for each behaviour–fault combination, from which the median average is taken. Note that both training and test sets of data do not include the first 50 seconds in which no faults are present.

classified as	H_1	H_2	H_3	H_4	H_5	H_6
H_1	42%	0%	0%	0%	56%	0%
H_2	0%	100%	0%	0%	0%	0%
H_3	0%	0%	90%	0%	0%	9%
H_4	0%	0%	0%	56%	44%	0%
H_5	0%	0%	0%	0%	99%	0%
H_6	3%	0%	1%	0%	35%	56%

TABLE 3.1: A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing aggregation behaviour where test data is obtained from a swarm performing aggregation behaviour.

Table 3.1, Table 3.2 and Table 3.3 show that for training data obtained during aggregation behaviour, a decision tree is able to appropriately classify most faults in

classified as	H_1	H_2	H_3	H_4	H_5	H_6
H_1	42%	0%	0%	0%	54%	0%
H_2	0%	100%	0%	0%	0%	0%
H_3	0%	0%	91%	0%	0%	9%
H_4	0%	0%	0%	90%	9%	0%
H_5	0%	0%	0%	0%	100%	0%
H_6	9%	0%	1%	0%	60%	19%

TABLE 3.2: A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing aggregation behaviour where test data is obtained from a swarm performing flocking behaviour.

classified as	H_1	H_2	H_3	H_4	H_5	H_6
H_1	5%	0%	0%	60%	0%	10%
H_2	0%	100%	0%	0%	0%	0%
H_3	0%	0%	88%	0%	0%	12%
H_4	0%	0%	0%	19%	80%	0%
H_5	0%	0%	0%	15%	82%	0%
H_6	2%	0%	1%	0%	6%	91%

TABLE 3.3: A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing aggregation behaviour where test data is obtained from a swarm performing obstacle avoidance behaviour.

a majority of cases, irrespective of whether or not the normal swarm behaviour has changed. Power failure faults (H_2) are correctly classified 100% of the time across all normal behaviours. Complete sensor failure (H_3) is also correctly classified for the majority of the time it is present across all behaviours. Partial motor failure (H_5) is correctly classified in the majority of cases for all normal swarm behaviours, however there is a substantial misclassification as complete motor failure (H_4) as partial motor failure during aggregation and obstacle avoidance behaviour (44% and 80% of the time on average, respectively). Although this is not ideal, for practical purposes, this does not pose a significant problem for two reasons:

1. Feature thresholds could be altered in order to distinguish between partial and complete motor failures more effectively. For example, complete motor failure would produce circular motion with a smaller diameter than partial motor failure, essentially making this a calibration problem.

2. Complete and partial motor failures will likely have very similar, if not identical, associated recovery actions. Therefore discriminating between the two from a practical perspective may be redundant in many scenarios.

For these reasons, the fact that complete motor failure (H_4) is misclassified as being only partial motor failure (H_5) during aggregation and obstacle avoidance behaviours was not considered to be a problem when conducting this work. It is also worth noticing that for all instances of complete sensor failure being misclassified for any normal behaviour, it is classified as partial sensor failure (H_6). It follows

that this should share the same pardon as misclassifications between complete and partial motor failures.

Where Table 3.1, Table 3.2 and Table 3.3 highlight a potential problem with this system is in the misclassification of partial sensor failure (H_6) as partial motor failure (H_5) for a substantial portion of time during aggregation and flocking behaviours (35% and 60% of cases on average, respectively). The reason for this is that the effects of partial motor failure (H_5) are most prominently distinct from the effects of other faults where the swarm is performing obstacle avoidance behaviour. For flocking and aggregation behaviours, each individual robot in the swarm spends the majority of it's time turning, rather than moving in a straight line. This obscures the effects of partial motor failure, which are most obvious when a robot is attempting to move in a straight line. Additionally, an individual robot that is flocking or aggregating is likely to have a neighbouring robot within 45° of its bearing at a given time, and therefore will appear to be responding to its neighbour's positions as normal. These two factors combined mean that flocking and aggregating robots which suffer from partial motor failure (H_5) or partial sensor failure (H_6) will both appear to be behaving normally, and thus similarly, for large periods of time.

The most serious shortfall of using a decision tree trained on data collected during aggregation behaviour is the persistent misclassification of software hang (H_6), most prominently as partial motor failure (H_5) for normal aggregation and flocking behaviour, or complete motor failure (H_4) for normal obstacle avoidance behaviour. A high rate of misclassification of software hang faults (H_1) is common to almost every combination of swarm behaviour and trained decision tree used for these experiments, and is discussed in detail at the end of this section.

classified as	H_1	H_2	H_3	H_4	H_5	H_6
H_1	27%	0%	0%	0%	54%	0%
H_2	0%	100%	0%	0%	0%	0%
H_3	0%	0%	90%	0%	0%	10%
H_4	0%	0%	0%	56%	33%	9%
H_5	0%	0%	0%	1%	73 %	25%
H_6	0%	0%	1 %	0%	9%	88%

TABLE 3.4: A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing aggregation behaviour where test data is obtained from a swarm performing flocking behaviour.

Some similar trends are observed when using a decision tree trained on data collected whilst the swarm exhibits flocking behaviour. Table 3.4, Table 3.5 and Table 3.6 show that power failure H_2 is again correctly classified for 100% of the time it is present for all normal behaviours. Complete sensor failure (H_3) is again correctly classified in the majority of cases for all normal behaviours, with the exception of a minority of cases where it is classified as partial sensor failure (H_6). Software hang (H_6) is again persistently misdiagnosed, this time as partial motor failure (H_5), partial sensor failure (H_6) or complete sensor failure (H_3) according to whether normal robot behaviour is aggregation, flocking or obstacle avoidance, respectively.

There are also some differences for a decision tree trained on data collected whilst the swarm exhibits flocking behaviour. For example, partial sensor failure (H_6) is correctly classified for a substantially larger proportion of time, on average 32% and 74% more for test data taken during aggregation and flocking normal behaviour,

classified as	H_1	H_2	H_3	H_4	H_5	H_6
H_1	31%	0%	0%	0%	0%	62%
H_2	0%	100%	0%	0%	0%	0%
H_3	0%	0%	91%	0%	0%	9%
H_4	0%	0%	0%	90%	4%	5%
H_5	0%	0%	0%	0%	49%	49%
H_6	0%	0%	1%	0%	4%	93%

TABLE 3.5: A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing flocking behaviour where test data is obtained from a swarm performing flocking behaviour.

classified as	H_1	H_2	H_3	H_4	H_5	H_6
H_1	0%	0%	84%	0%	0%	12%
H_2	0%	100%	0%	0%	0%	0%
H_3	0%	0%	88%	0%	0%	12%
H_4	0%	0%	4%	19%	4%	72%
H_5	0%	0%	3%	15%	8%	73%
H_6	0%	0%	3%	0%	2%	95%

TABLE 3.6: A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing flocking behaviour where test data is obtained from a swarm performing obstacle avoidance behaviour.

respectively. This result would suggest that the ambiguity between partial sensor failure (H_6) and partial motor failure (H_5) that was observed when the decision tree was trained on data collected during aggregation behaviour had been reduced. However, partial motor failure (H_5) is frequently misclassified as partial sensor failure (H_6) for all normal behaviours, demonstrating that ambiguity between the two faults remains for for a decision tree trained on data collected during flocking behaviour, but that the shared characteristics are now assigned preferentially to partial sensor failure (H_6) by the decision tree. Complete motor failure (H_4) is misclassified as partial sensor failure (H_6) for the majority of the time it is present where the swarm is performing obstacle avoidance behaviour. This is unsurprising given the shared characteristics of complete motor failure (H_4) and partial motor failure (H_5).

Table 3.7, Table 3.8 and Table 3.9 show that for training data obtained whilst the swarm exhibits obstacle avoidance behaviour, power failure (H_2) is once again classified correctly in 100% of cases for all normal behaviours, and complete sensor failure (H_3) is either correctly classified or classified as partial sensor failure (H_6) for all normal behaviours. Software hang (H_1) is again misclassified for a majority of the time during aggregation and flocking behaviours. However, the decision tree trained on data collected whilst the swarm exhibits obstacle avoidance behaviour is uniquely able to classify software hang (H_1) appropriately for the majority of the time it is present during normal obstacle avoidance behaviour.

Table 3.7, Table 3.8 and Table 3.9 display a number of similarities to the experiments where the decision tree was trained on data collected whilst the swarm exhibits aggregation behaviour (Table 3.1, Table 3.2 and Table 3.3). For example, complete motor failure (H_4) is correctly classified or classified as partial motor failure

classified as	H_1	H_2	H_3	H_4	H_5	H_6
H_1	27%	0%	0%	0%	57%	0%
H_2	0%	100%	0%	0%	0%	0%
H_3	0%	0%	90%	0%	0%	10%
H_4	0%	0%	0%	56%	44%	0%
H_5	0%	0%	0%	0%	99%	0%
H_6	0%	0%	1%	0%	40%	56%

TABLE 3.7: A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing obstacle avoidance behaviour where test data is obtained from a swarm performing aggregation behaviour.

classified as	H_1	H_2	H_3	H_4	H_5	H_6
H_1	31%	0%	0%	0%	54%	4%
H_2	0%	100%	0%	0%	0%	0%
H_3	0%	0%	91%	0%	0%	9%
H_4	0%	0%	0%	90%	9%	0%
H_5	0%	0%	0%	0%	100%	0%
H_6	0%	0%	1%	0%	61%	37%

TABLE 3.8: A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing obstacle avoidance behaviour where test data is obtained from a swarm performing flocking behaviour.

classified as	H_1	H_2	H_3	H_4	H_5	H_6
H_1	85%	0%	0%	0%	0%	9%
H_2	0%	100%	0%	0%	0%	0%
H_3	0%	0%	88%	0%	0%	12%
H_4	0%	0%	0%	19%	80%	0%
H_5	0%	0%	0%	15%	85%	0%
H_6	0%	0%	1%	0%	8%	91%

TABLE 3.9: A table to show each type of fault (row) and the median average distribution of classifications (column) by a decision tree trained on a swarm performing obstacle avoidance behaviour where test data is obtained from a swarm performing obstacle avoidance behaviour.

(H_5) in the vast majority of cases for all normal behaviours. Likewise, partial motor failure (H_5) is either correctly classified or classified as complete motor failure (H_4). Furthermore, there is a high rate of misclassification of partial sensor failure (H_6), which is frequently misclassified as partial motor failure (H_5) during normal aggregation and flocking behaviours.

For all experiments, a notably high rate of correct classification across all behaviours for a robot exhibiting power failure (H_2) – 100% of all cases tested. The reason for this is that a robot exhibiting power failure will produce a feature vector that is not only distinct from that of any of the other fault types, but also one that

will remain largely constant. The only features that will change once a robot experiences a power failure will be externally sensed neighbour features. All other features will return 0 for the entire duration of the failure. It can be observed in Figure 3.9, Figure 3.10 and Figure 3.11 that a robot will be classified with a power failure when feature $F_4^p(t)$, corresponding to proprioceptively estimated motion, returns 0. This is a characteristic unique to a power failure fault, as for all other faults the robot controller will still attempt to keep the robot moving, making it immediately recognisable.

For all experiments, with the exception of those which inform Table 3.9, a persistently high rate of misclassification of software hang (H_1) is observed. The reason for this is that software hang can produce entirely different behaviours depending on what the robot was doing at the point of fault injection. For example, if a decision tree is trained on data software hang was injected whilst the robot was moving in a straight line and then applied to a new instance of software hang that is injected whilst the robot is turning, two very different BFVs are produced over time for the same type of fault. Furthermore, if the robot is moving in a straight line and has no neighbours in sensing range at the point of fault injection, the resultant faulty behaviour is indistinguishable from complete sensor failure (H_3) by its BFV.

How an individual fault might produce several different behaviours, dependent on external factors, is something that was not anticipated in the work described in this chapter. However, consideration to this may be very important, if not essential, to a reliable fault diagnosis mechanism. Another absent consideration from this work, and one that could have decreased misclassification, is how different fault types may produce distinct time-dependent BFV patterns. Observing software hang over a period of time would make the fault easily identifiable, as a static proprioceptive BFV would be observed for a changing externally estimated BFV.

Visualising the trained decision trees in Figure 3.9, Figure 3.10 and Figure 3.11 shows that all 10 estimated features were used in the classification process for each tree, making them discriminatory. The bounds used for each decision are all set to 0.5, as this represents the midpoint between the potential values each feature can assume (0 and 1). As each feature is Boolean, the decision tree would give an identical output for $0 < \forall b < 1$.

Table 3.10 shows that overall performance of each decision tree is greatest when swarms perform the same behaviour in training and test data (with the exception of the decision tree trained on data collected whilst the swarm exhibits flocking behaviour, which performs marginally better on test data obtained whilst the swarm exhibits aggregation behaviour). This is unsurprising, as it follows logically that the BFVs produced by faults will be most similar between two robots which, along with the rest of the swarm, are attempting to perform the same normal behaviours. Table 3.10 also shows that for any combination of normal behaviours exhibited during test data and training data, each decision tree is able to appropriately classify the six fault types in a majority of cases.

3.5 Summary

The work described in this chapter has shown that by characterising robot behaviour as BFVs, different fault types can be classified with a generally high reliability. However, there are still a number of issues to address, such as how a system can reliably diagnose fault types which can manifest in different ways, or consideration of how faults can be diagnosed online and in real-time using robot BFVs without relying on

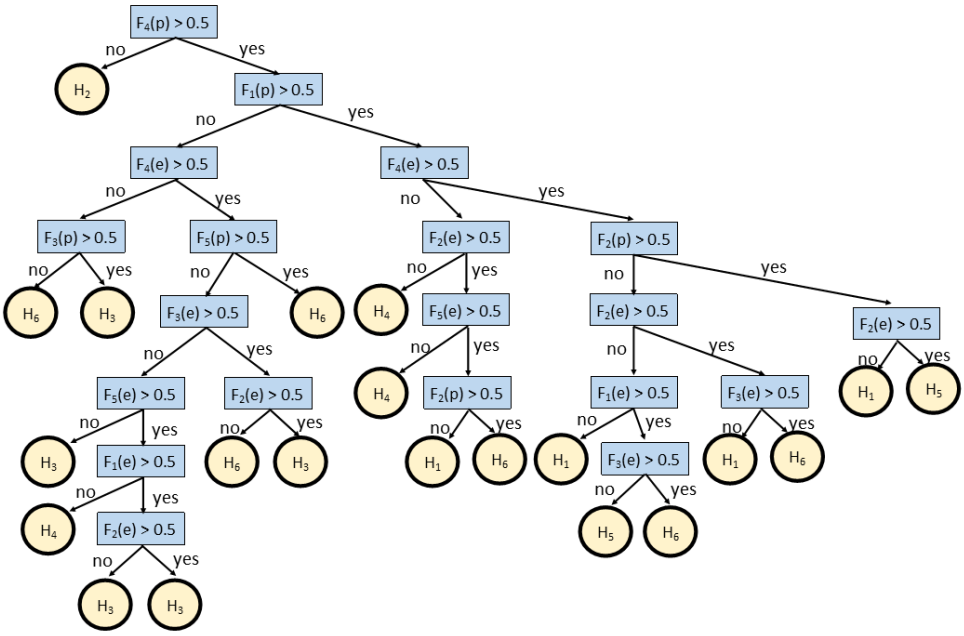

FIGURE 3.9: Visual representation of the decision tree trained on data collected whilst the swarm exhibited aggregation behaviour where *p* and *e* indicate proprioceptively and externally estimated features, respectively.

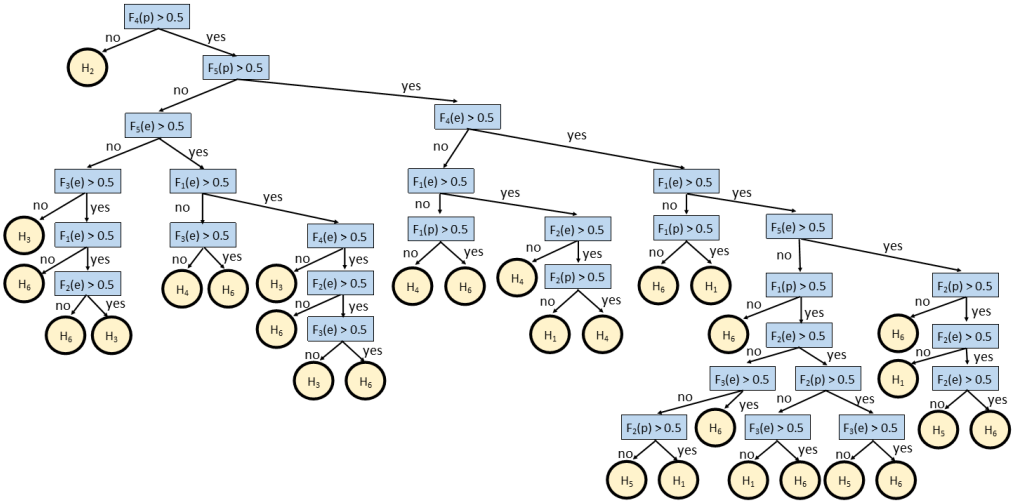

FIGURE 3.10: Visual representation of the decision tree trained on data collected whilst the swarm exhibited flocking behaviour where *p* and *e* indicate proprioceptively and externally estimated features, respectively.

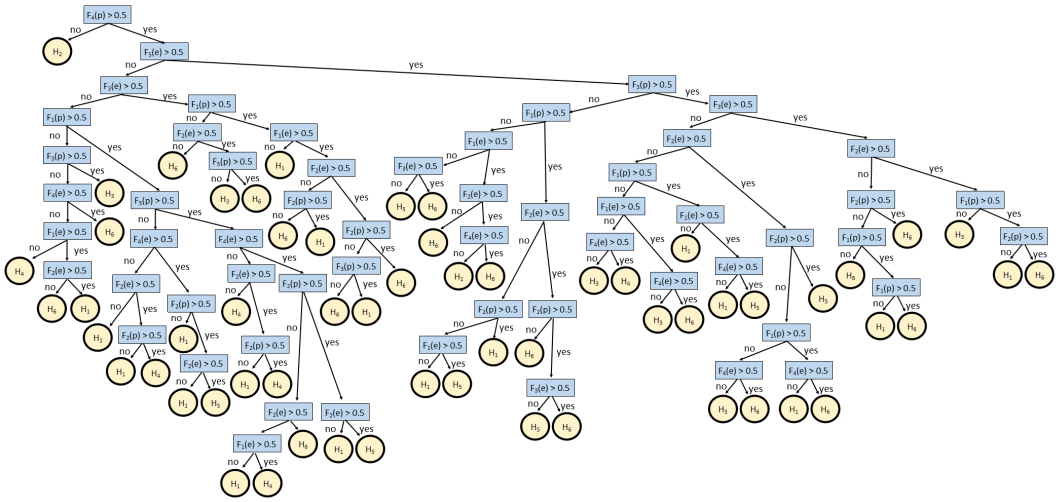

FIGURE 3.11: Visual representation of the decision tree trained on data collected whilst the swarm exhibited obstacle avoidance behaviour where p and e indicate proprioceptively and externally estimated features, respectively.

	(Test Data)		
(Training data)	Aggregation	Flocking	Obstacle avoidance
Aggregation	83%	77%	82%
Flocking	80%	77%	57%
Obstacle	81%	78%	96 %

TABLE 3.10: A comparison of the median average proportion of correctly classifed faults for training and test data collected during all combinations of normal swarm behaviour

a priori models of robot behaviour – although decision tree algorithms can be useful for identifying discriminating features, their inflexibility makes them unsuitable for the fault tolerant approach envisaged by this work.

The following chapters detail the work towards a practical fault diagnosis system for robot swarms, whereby faults can be associated with a recovery option via a series of diagnostic tests and a process of trial and error, after which the BFVs of subsequent faults can be tested for similarities to known faults. The results described in this chapter support the hypothesis that, if this can be accomplished, it is possible to diagnose faults by the similarities observed between faulty robot BFVs of a given type.

Chapter 4

Fault Diagnosis in Software Simulation: Part I

4.1 Introduction

The previous chapter detailed the initial investigation into the suitability of Behavioural Feature Vectors (BFVs) for classifying different types of faults. The results obtained showed promise, and the decision was made to continue working with BFVs (albeit with some modifications).

The work discussed in this chapter used BFVs to encode robot behaviour, allowing for BFV-signatures to be obtained for different types of faults. These were then used as the basis for an immune-inspired fault diagnosis mechanism, whereby faults could be diagnosed and resolved more efficiently if the swarm had already been exposed to them.

This chapter describes the conception of a novel fault diagnosis algorithm, and the design and testing of an initial prototype in software simulation, using the system described and results obtained in the previous chapter as an experimental basis.

4.2 Fault Diagnosis for Swarm Robots

To conceive of a novel approach to fault diagnosis in robot swarms, the vertebrate immune system is examined at a high level as a benchmark for what artificial immunity should encompass and be able to achieve.

The vertebrate immune system is observed to be able to recover its host body from harm and illness with which its host was previously unfamiliar (Capra et al., 1999). This is a characteristic that any approach towards artificial immunity will need to reproduce to some degree. Of equal importance is the natural immune system's ability to *remember* what it has previously learned, allowing it to recover from known illnesses in a more efficient manner (Capra et al., 1999).

When the vertebrate immune system encounters an infection, the lymphocytes that recognise pathogens proliferate and differentiate (Owen, Punt, and Stranford, 2013). Once the pathogens have been destroyed, most of the immune cells that were involved in the immune response are eliminated, except for a small population of long-lasting elements made up of the immune cells which most effectively fought the pathogens (Floreano and Mattiussi, 2008), known as *memory* cells (Kindt et al., 2007). These cells have an increased sensitivity to antigens and will prompt a faster immune response should their host encounter the same pathogen again (Floreano and Mattiussi, 2008). As this process is invaluable in maintaining a host organism for a natural immune system, so too will it be for maintaining a robot swarm – an Artificial Immune System (AIS) should seek to minimise any down-time within a

swarm as this is where the system will be most vulnerable to further faults. For example, for emergent behaviours requiring n functioning robots, the swarm system will be less vulnerable to faults, in terms of task performance, if it consists of $n + 1$ functioning robots than the bare minimum n. AIS should not only be able to diagnose faults that are occurring for the first time, but also be able to characterise these faults for comparison against any other faults that might occur in the future. When subsequent faults do occur, the system should be able to recognise if that fault is similar to one it has previously encountered and diagnose it more efficiently.

Recalling that artificial immunity consists of fault detection, fault diagnosis and recovery (FDDR) (Millard, 2016), fault diagnosis is situated between fault detection and fault recovery. Each of the sub-processes are inter-related, meaning that fault diagnosis cannot be thoroughly examined without consideration of how it integrates with fault detection and recovery. In the early experiments for the work in this chapter it became evident that reliably classifying or diagnosing faults in real time necessarily required a means of fault detection. To illustrate, if a robot was trying to diagnose a fault in its neighbour based on the previous t seconds of its neighbour's behaviour, a reliable diagnoses could only be made if the neighbour's behaviour adequately reflected its fault type. If, on the other hand, the neighbour had a fault with its left motor but was turning on its left wheel, it would appear to be behaving normally. The work described in this chapter towards fault diagnosis, and in all subsequent chapters in this thesis, therefore includes an approach to fault detection (although fault detection is not examined to the same degree in this thesis).

In Chapter 3 it was demonstrated that BFVs could be used to classify different types of fault in a robot swarm using a decision tree. A drawback to this approach, that was outlined in the previous chapter, is that supervised learning techniques are inflexible in this context; they require a comprehensive set of training data (which it may not always be possible to produce, particularly in hardware systems) and are unable to recognise a lack of shared characteristics. A recurrent problem in the work described in chapter 3 was that a trained model would categorise a fault in line with the training data to which it was most similar – even if this similarity was not significant – sometimes resulting in misclassification. It was therefore speculated for the work in this chapter that an unsupervised-learning technique could provide better solutions to fault diagnosis based on BFVs – this would also allow the system to retain one of the advantages of BFVs, highlighted by Tarapore, Christensen and Timmis (2017) in their work on fault detection, which is that a trained model is not required in order to gain valuable insight into robot behaviour. Adopting this approach facilitates the integration of fault detection and diagnosis, as is desired, with a common medium for behaviour representation and in an unsupervised manner.

4.3 Experimental Method

Work towards novel fault diagnosis for robot swarms began with the same system used in the previous chapter. Namely, using Autonomous Robots Go Swarming (ARGoS) (Pinciroli et al., 2011) to generate a swarm of 10 simulated models of the marXbot (Bonani et al., 2010) swarm robotic platform. These robots again exhibit flocking, aggregation, and dispersion swarm behaviours. In all of these behaviours, each robot moves in response to the location and proximity of its neighbours, so a normally functioning robot will always be motional.

The same faults types as described in the previous chapter were used again for this work, namely:

The faults considered in this work are:

- **Software hang (H_1)**: This fault will cause a robot to get stuck performing whatever action it was performing at the last moment it was normally functioning.
- **Power failure (H_2)**: A robot that suffers a power failure will completely stop moving and remain unresponsive to its surroundings.
- **Complete Sensor failure (H_3)**: For a complete sensor failure an individual robot will be unable to detect the presence of any neighbours, light sources or obstacles.
- **Complete Motor failure (H_4)**: A complete motor failure will cause one of the afflicted individual's motors to stop and henceforth become unresponsive to its controller. The other motor will remain functional, meaning that the afflicted robot will either stop completely if it is trying to turn using its faulty motor, or otherwise be limited to turning on the spot.
- **Partial Motor failure (H_5)**: For partial motor failure the individual's faulty motor will remain responsive, however it will only allow its associated wheel to turn at half speed.
- **Partial Sensor failure (H_6)**: For a partial sensor failure a robot will only be able to detect the presence of neighbours, light sources and obstacles within $\pm 60\ ^\circ$ of its current heading. This angle is increased from the value of $\pm 45\ ^\circ$ used in the previous chapter in order to try and make H_6 more separable from H_4, however all subsequent experiments suggested that this change was of negligible consequence to any performance metric.

4.3.1 Deriving Behavioural Feature Vectors

Although BFVs remove the need for *a priori* information on robot behaviour for the purposes of producing a classification model, *a priori* knowledge of the SRS is still required to define BFVs that can be used to discriminate between fault types. This knowledge will include information on individual robot hardware and robot behaviours. The work described in the previous chapter showed that deriving features exclusively from robot hardware and normal behaviours did not produce a comprehensive enough faulty behaviour representation to reliably diagnose every type of fault considered. This was largely because some faults, depending on environmental circumstances, could cause the afflicted robot to deviate from normal behaviour in a number of distinct ways. In order to effectively utilise BFVs for the purpose of fault diagnosis, the types of faults that are likely to occur in individual robots should also be considered. Although not applicable to this work, which uses predetermined fault types, there exists techniques for identifying and anticipating the types of faults robot hardware could be susceptible to, such as Failure Mode and Effect Analysis (FMEA) (Dailey, 2004) or Fault Tree Analysis (FTA) (Ericson and Ll, 1999).

This work initially follows the process of atomising swarm behaviours into individual robot sub-behaviours and conditions described in the previous chapter, initially arriving at the same set of Boolean features:

$$F_1(t) = 1 \text{ if } N_R(t) > 0, \text{ otherwise } F_1(t) = 0 \tag{4.1}$$

where $N_R(t)$ is the total number of neighbours in sensing range (approximately $1m$) of the robot at time t.

$$F_2(t) = 1 \text{ if } N_C(t) > 0, \text{ otherwise } F_2(t) = 0 \tag{4.2}$$

where $N_C(t)$ is the total number of neighbours at a distance less than the close proximity threshold, C, (approximately $0.3m$) from the robot at time t.

$$F_3(t) = 1 \text{ if } |v(t)| > 0.8|v_{max}|, \text{ otherwise } F_3(t) = 0 \tag{4.3}$$

where $|v(t)|$ is the magnitude of linear velocity at time t and $|v_{max}|$ is the maximum linear velocity a robot can achieve (approximately $5cms^{-1}$ in this work).

$$F_4(t) = 1 \text{ if } |v(t)| > 0.2|v_{max}|, \text{ otherwise } F_4(t) = 0 \tag{4.4}$$

$$F_5(t) = 1 \text{ if } |\theta(t)| > 0.4|\theta_{max}|, \text{ otherwise } F_5(t) = 0 \tag{4.5}$$

where $|\theta(t)|$ is the magnitude of angular velocity at time t and $|\theta_{max}|$ is the maximum angular velocity a robot can achieve (approximately $24^{\circ}s^{-1}$ in this work).

In the work described in the previous chapter, software hang, H_1, was by far the most difficult fault to reliably diagnose, and was often misclassified. The reason for this was that, depending on environmental circumstances, the afflicted robot could deviate from normal behaviour in a number of very different ways. To compensate for this, a final feature, $F_6(t)$, inspired by the *watchdog timer* (Huang and Selman, 1986) commonly found in embedded systems, was added to the BFV. If H_1 is present in a robot, its feature F_6 will return a value of 1 – providing a common characteristic for every instance of software hang.

$$F_6(t) = 1 \text{ if } H_1 = True, \text{ otherwise } F_6(t) = 0 \tag{4.6}$$

The close proximity threshold, C, is the distance at which robots will turn away from each other in order to avoid collision (approximately 30cm again).

Whilst the feature vector $BFV(t) = [F_1(t), F_2(t), F_3(t), F_4(t), F_5(t), F_6(t)]$ is only one possible imagining of a BFV for this system, F_{1-5} were demonstrated to be each individually discriminatory when a decision tree was used to classify the same six fault types, and the utility of F_6 (or some alternative) was highlighted in the previous chapter.

As in Chapter 3, each robot estimates its own BFVs proprioceptively as well as exteroceptively estimating the BFVs of any of its neighbours in range.

4.3.2 Integrated Fault Diagnosis System

Detailed in this section is the initially proposed system for fault diagnosis, and how it is integrated with provisional fault detection and recovery mechanisms for an autonomous approach to FDDR. An overview of the system is given by Figure 4.1.

Fault Detection

Each of the faults described here and in the previous chapter will cause some discrepancy between proprioceptively estimated features, F^p_{1-6}, and exteroceptively estimated features, F^e_{1-6}. Using this as a fault detection mechanism ensures that faults are represented in the set of BFVs that are then analysed for diagnosis. For the work

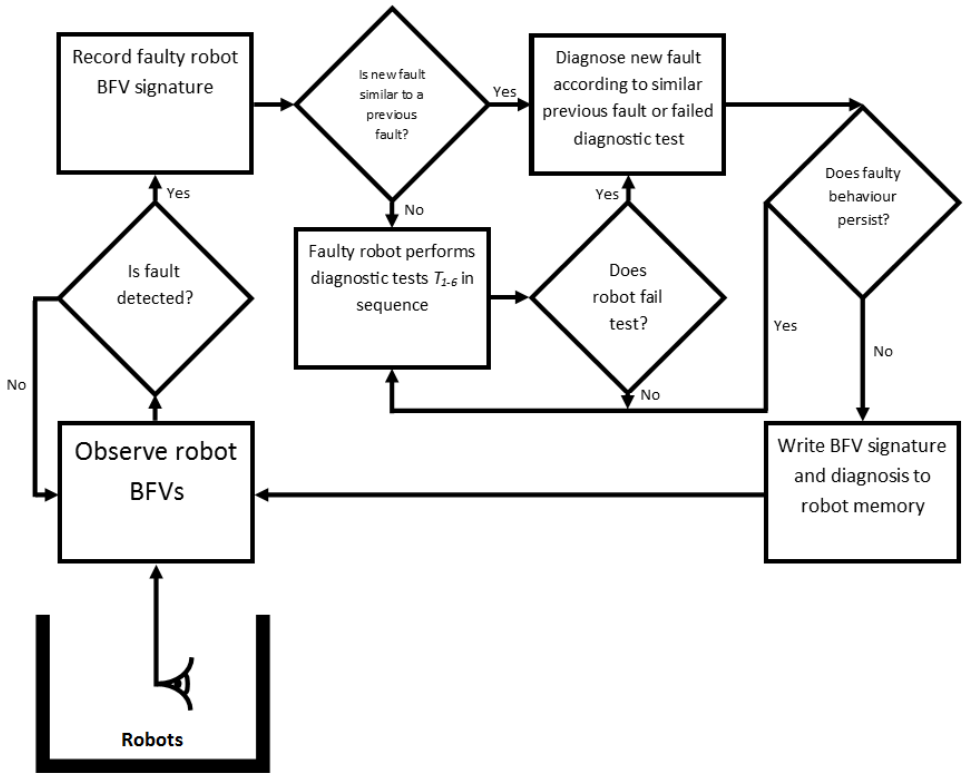

FIGURE 4.1: A flowchart of the proposed fault diagnosis process.

described in this chapter and in those following, faults are detected according to Equation 4.7.

$$\text{fault detected at time } t = \textit{true} \text{ if } \sum_{t=T-W}^{T} d(t) \geq \rho W \text{ for } T > W \tag{4.7}$$

Where T indicates the time elapsed since a robot started being observed by a neighbour. The binary value d is set according to Equation 4.8, where $d = 0$ if there is agreement between exteroceptively and proprioceptively estimated robot features. A robot will make W observations of its neighbour, once per control-step, at discrete times, t, before a fault can be detected, where t indicates the time elapsed since the start of an experiment. The value of ρ indicates what proportion of the W observations must prompt the value of d to be equal to 1 in order for a fault to be detected.

$$d(t) = \begin{cases} 0 \text{ if } F_i^p(t) = F_i^e(t) \text{ for } i = 1,..,6 \\ 1 \text{ otherwise} \end{cases} \tag{4.8}$$

The variables in Equation 4.7 are investigated in the following chapter. However, for the work described in this chapter, the values were set after a process of trial and error as follows:

- $W = 100$
- $\rho = 1$

This means that faults are detected when a persistent discrepancy is observed between a robot's proprioceptively and exteroceptively estimated BFV states over

100 consecutive control steps. This approach to BFV-based fault detection, whilst far simpler than that presented by Tarapore, Christensen and Timmis (2017), shares the benefit of not needing an *a priori* model of faulty behaviour.

When a fault is detected in a robot by another member of the swarm, the set of BFVs, comprising the faulty robot's proprioceptively estimated features and the detecting robot's exteroceptively estimated features for the faulty robot, which prompted the fault to be detected, can be used as the BFV signature for that particular fault.

Fault Diagnosis

The previous chapter showed that BFVs are an effective medium for comparing fault signatures where there is a known reference for each type of fault. However, constructing *a priori* models of faulty robot behaviour negates the advantages of BFV-based fault detection, as described by Tarapore, Christensen and Timmis (2017). To retain these advantages, there needs to be a way of associating fault categories with their respective BFV signatures online.

As the types of fault that may occur in robots can be anticipated using techniques such as FMEA and FTA, it is a realistic possibility to design a series of tests that can isolate the root cause of the fault. These tests can be assessed in real-time by another robot in the swarm, in a similar fashion to the diagnostic manoeuvres used by Kutzer *et al.* (2008), which declares itself as the assessing robot once it detects a fault in its neighbour. These tests are designed in sympathy with each type of fault such that a robot will only be able to pass the test if it is not affected by its corresponding fault type. A robot performs every test in sequence – a brute-force approach to fault diagnosis – until it passes them all or it fails one, which indicates the presence of an associated fault. In this manner, different categories of fault can be identified and thus associated with their BFV signatures in the first instance.

When an assessing robot initiates a diagnostic test routine, it approaches the faulty robot and instructs it to stop moving so that it can begin its assessment. Whilst it is performing its diagnostic routine, the assessing robot will also act as a beacon, instructing other approaching robots in the swarm to turn away so as not to interfere.

The diagnostic tests are arranged in order such that the faults which most severely inhibit robot behaviour are tested for first, as some faults could obscure the test results for others. For example, a robot suffering power failure would also fail motor and sensor tests. The diagnostic tests used for the work in this chapter, and the respective faults they test for, are as follows:

- **Software hang (T_1)**: This occurs immediately after the assessing robot moves into testing range. In order for the faulty robot to perform later diagnostic tests, it must be able to communicate with the assessing robot. To establish this communication, the assessing robot first pings the faulty robot and instructs it to stop moving. If the faulty robot is suffering from software hang, it will disregard this request. If the faulty robot is persistently unreachable and ignores the assessing robot's stop request, it will be diagnosed as having a software hang fault. This assumes that communication between normally operating robots is reliable.

- **Power failure (T_2)**: Almost identical to T_1, however in cases where the robot is suffering from power failure it will also stop moving. The assessing robot therefore diagnoses power failure in instances where the faulty robot is unreachable but stationary. It is acknowledged that this would only work for normal behaviours where robots should always be in motion. For a swarm

exhibiting behaviour that necessitated periods where the robot was stationary, the relationship between T_1 and T_2 would have to be revised.

- **Complete Sensor failure (T_3)**: Once the faulty robot has been established as responsive to the assessing robot, the two robots should have settled within range of each other – something that will be reflected in their BFV. A complete sensor failure is diagnosed if the faulty robot's feature F_1^p returns a value of 0.

- **Complete Motor failure (T_4)**: The faulty robot is asked to spin on its right and left wheels. If the faulty robot receives and processes this request but is unable to perform one or both of the turns, it is decided that the robot is suffering from complete motor failure. This is ascertained by monitoring the faulty robot's feature F_4^e, where a returned value of 0 indicates the robot has failed the test.

- **Partial Motor failure (T_5)**: Having established that both of its motors are responsive, the faulty robot is then asked to demonstrate that both motors work together as they should. The faulty robot is asked to move forward in a straight line. If the faulty robot is observed to move in an arc or otherwise deviate from moving in a straight line, it is decided that the faulty robot is suffering from partial motor failure. This is achieved by monitoring the faulty robot's feature F_5^e, where a returned value of 1 indicates that the robot has failed the test.

- **Partial Sensor failure (T_6)**: To test for partial sensor failure, the faulty robot performs a 360° turn whilst acknowledging the presence of the assessing robot. If the faulty robot fails to acknowledge the assessing robot for any period of this turn, the robot is decided to be suffering from partial sensor failure. Partial sensor failure is sometimes diagnosed as complete sensor failure if the assessing robot initially approaches the faulty robot from an angle that is beyond its now impaired line of sight. It is not clear how this could be resolved without significantly increasing the time taken to perform the diagnostic tests. Furthermore, as both faults share the same recovery option, it does not truly represent a problem for the system.

In each of these tests, the condition for failure must be met for 10 consecutive control steps by the robot before it is considered to have failed the test.

The use of a finite number of pre-written diagnostic tests will only allow for the diagnosis of a finite number of distinct categories – something that could be considered inflexible. However, it is proposed that there is an equivalence between this approach and a high level understanding of the vertebrate immune system. The vertebrate immune system is reliant on an exceedingly large lymphocyte repertoire that has the ability to bind to an exceedingly large number of potential pathogens (Owen, Punt, and Stranford, 2013). This property is the product of cross-generational evolution rather than something learned in the lifespan of a single host body (although the specific composition and distribution of the repertoire does change during host life span). For a swarm robotic system with finite functionality, there will, at a high level, be a finite number of discrete fault types a robot can be affected by, such as actuator failure, sensor failure or power failure. Each of these faults can theoretically then be identified by a corresponding diagnostic test. It is proposed that the resulting repertoire of diagnostic tests that resolve these fault types can be considered analogous to the lymphocyte repertoires observed in the natural immune system – and the use of techniques like FMEA to identify a series of fault types analogous to natural evolutions spurring of the production of lymphocyte repertoires in individual host bodies (albeit an accelerated version). In the event that a fault occurred in a robot that did

not fall into one of the identifiable categories, or for any reason was not recognisable as such, the effect would be the same as when the natural immune system is unable to recognise or remove harmful pathogens; the problem persists and, if the host is not tolerant, it eventually perishes.

Fault Recovery

When a fault is diagnosed, the system must be able to ascertain whether or not the diagnosis was appropriate. This is done by carrying out a corresponding recovery action and then subsequently observing the faulty robot to check whether the fault persists. All six fault types used for this work are resolved by one of three recovery actions:

- **Power Cycle (R_1)**: It is assumed for this work that, when a robot has its power cycled, it is turned back on with a new or replenished power source. This recovery action resolves software hang (H_1) and power failure (H_2) faults.
- **Sensor Replacement (R_2)**: Replacement of robot sensors resolves complete and partial sensor failures (H_3 and H_6, respectively).
- **Motor Replacement (R_3)**: Replacement of robot motors resolves complete and partial motor failures (H_4 and H_5, respectively).

It is acknowledged that carrying out these proposed recovery actions autonomously with the marXbot platform is not realistic at the time of writing, and all recovery actions are simulated for this work. However, it is anticipate that future swarm robotic platforms will possess the ability to perform similar actions and, as with the selection of faults H_{1-6}, the diagnosis system presented here is designed with intention of being implemented on such a platform in the future.

When a recovery action is executed, the faulty robot continues to be observed by the assessing robot. If the faulty robot's exteroceptively and proprioceptively estimated BFVs match after a recovery action has been performed, the fault is considered to be resolved.

The work performed for this chapter was intended to provide a proof-of-principle for the proposed method of fault diagnosis. As such, the tests T_{1-6} were coded such that they would always successfully arrive at the correct diagnosis. More realistic scenarios, such as where the diagnostic tests failed to identify a fault, or misdiagnosed a fault type, were investigated in later work which is detailed in the subsequent chapters of this thesis.

4.3.3 Fault Memory

The diagnostic tests (T_{1-6}) take a relatively long time to run reliably and, in a swarm system, will require at least one additional robot to postpone its primary task in order to ascertain which tests the supposedly faulty robot passes or fails (outcome of diagnostic tests must be concluded exogenously so that they are not subject to the problems associated with endogenous fault detection (Christensen et al., 2008)). Repeatedly removing robots from the swarm will jeopardise long-term autonomy. Furthermore, the natural immune system is observed to recover more quickly from illness after its host's initial exposure to it. The system proposed here attempts to mimic this capability as follows:

Each time a fault is successfully resolved, the BFV signature for that fault and the respective diagnosis are stored in the doctor robot's memory, which is then shared

with other members of the swarm via local communication. When subsequent faults occur, the BFV representations of those faults can be checked for similarity against the BFV representation of previous faults.

Similarity between two faults is established by finding the Pearson correlation coefficient (Benesty et al., 2009) between their BFV signatures (Equation 6.3):

$$r = \frac{\sum_m \sum_n (F_{mn} - \overline{F})(F_{0_{mn}} - \overline{F_0})}{\sqrt{(\sum_m \sum_n F_{mn} - \overline{F})^2 (\sum_m \sum_n F_{0_{mn}} - \overline{F_0})^2}} \tag{4.9}$$

Where r is the correlation coefficient between a current and previous fault signature, F and F_0, respectively, where F and F_0 are two dimensional binary data sets. $\overline{F}$ and $\overline{F_0}$ are the mean values of sets F and F_0, respectively, and m and n indicate positional index within the 2D binary data set (the specific orientation of m and n is arbitrary for this equation).

When attempting to diagnose a fault from memory, the previously stored fault which produces the greatest r-value (Equation 6.3) is considered the most likely fault type and recovery is initiated in sympathy with that fault. The minimum r-value required for two faults to be considered similar, and thus eligible for this process, was set to 0.66 for this work (however this value is investigated thoroughly in the subsequent chapters of this thesis).

The system proposed here is adaptive; the previous chapter discussed how individual fault types could affect robots in different ways, according to the robot's circumstances. This is similar to the evolution of pathogens which makes it harder for the vertebrate immune system to detect and destroy them. In this system, as long as a fault is detectable and diagnosable by tests T_{1-6}, its BFV representation will be committed to memory. In this way the system's representation of each fault type is dynamic and can evolve in real time, as the vertebrate immune system is able to adapt to evolving pathogens (Floreano and Mattiussi, 2008). It is therefore argued that this system, at a high level, mirrors the desirable characteristics of the vertebrate immune system described earlier in this chapter.

4.3.4 Experiments

In each experiment a swarm of 10 simulated marXbots collectively perform flocking (Algorithm 1), aggregation (Algorithm 2) or dispersion (Algorithm 3) behaviour in an otherwise empty enclosed arena of approximately 16 square metres (see Figure 3.8). Simulated Gaussian noise is added to robot orientation sensors (μ=0, σ=1 $^\circ$), range and bearing sensors (μ=0, σ=3cm) and the overhead sensor (μ=0, σ= 5% d_{max}) where d_{max} is the maximum distance that a robot can travel in a single control step (approximately 0.5cm). The noise on the overhead sensor is applied to the co-ordinate values for each robot.

The experiments run for 36000 control steps, representing one hour of simulated time. As in the experiments in the previous chapter, the swarm's normal behaviour is randomly selected from one of the three predefined swarm behaviours, and is updated randomly after every 5000 control steps (500 seconds). A fault is injected at random into a robot after 500 control steps of normal behaviour. Only one robot can be faulty at a time. Faults are detected according to Equation 4.7. When a fault is resolved, another 500 control steps elapse before a new fault is injected again. This approach means that every time an experiment is run, the swarm exhibits a variety of behaviours and is subjected to a variety of faults – very often exhausting

all possible combinations. For this reason it is proposed that the results obtained in this work satisfy the conditions for swarm flexibility (Şahin, 2005).

The experiments described in this chapter, and throughout the remaining chapter of this thesis, do not investigate instances of false-positive fault detection. Therefore, if a fault is detected in a given robot it is because there is a fault is present in that robot. The reason that instances of false-positive fault detection are omitted from this study is that it was initially assumed that, whilst a nuisance, a robot that was incorrectly detected as faulty would comfortably be able to pass each diagnostic test (T_{1-6}), and therefore be recognised as normally functioning by the swarm. It was also assumed that fault diagnosis and fault detection would be generally separable processes and that by focusing on fault diagnosis in this thesis, integration with fault detection could be safely relegated to future work until such a time as detection and diagnostic processes had independently developed to a sufficient level. Both of these assumptions are challenged by the results of experiments that were later conducted for this thesis, and this is discussed in Chapter 7.

Proprioceptively sensed features are estimated by each individual robot, using its on-board sensors for F_1^p and F_2^p, its controller for F_3^p, F_4^p and F_5^p or a watchdog-inspired internal counter for F_6^p. BFVs are estimated exteroceptively using a virtual sensor to obtain co-ordinate information for each neighbouring robot at each time step. As with the experiments in the previous chapter, the use of an overhead sensor can be thought of as equipping each robot with its own virtual range and bearing sensor – similar to the approach taken by O'Dowd, Winfield and Studley (2011). The proposed system therefore still meets the criteria for swarm research, as specified by Şahin (2005), and is conceptually distributed.

Each robot writes fault information to its own circular buffer with 18 places (three times the number of distinct fault categories) as it was considered unlikely in this case that a fault category would be passed out of memory once it had been encountered. The exact number of places will vary for different systems depending on the hardware used and memory space available. Storing fault memory to a circular buffer makes the system scalable by preventing the gradual accumulation of fault data to the software/hardware limitation of whatever platform the system is implemented on, and thus does not inhibit long-term autonomy.

When a fault type and its associated BFV signature is stored in an assessing robot's memory, the assessing robot then passes the information to any other swarm robots it comes within range of, these robots then do likewise and so on. This simulates the swarm robots sharing and updating fault memory via an *ad-hoc* network.

The criteria used to characterise our system's performance during these experiments are as follows:

- P_1: The proportion of faults the system is able to diagnose from memory
- P_2: The proportion of attempts to diagnose a fault from memory that result in misclassification
- P_3: The average r-value between faults, using Equation 6.3, when diagnosing from memory

For all experiments 100 replicates are performed. This number of replicates was selected after performing consistency analysis (Read et al., 2012) to find the point at which performing additional replicates does not alter the median value of the entire set. This was achieved by using the SPARTAN package (Alden et al., 2013) on P_1, as this is considered this the most indicative measure of overall system performance.

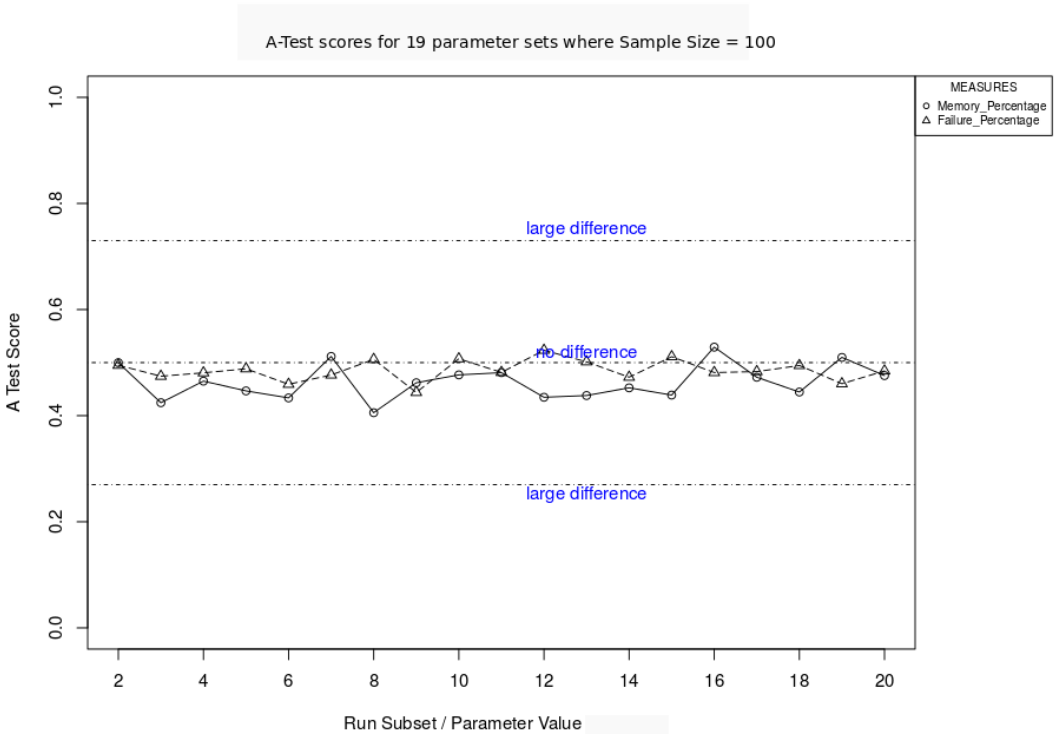

FIGURE 4.2: This figure shows the A-Test scores for the proportion of faults the system is able to diagnose from memory (P_1) and the proportion of attempts to diagnose a fault from memory that result in misclassification (P_2). 19 subsets of 100 experimental replicates are tested (1900 total experiments).

Figure 4.2 shows the A-Test scores for the proportion of faults the system is able to diagnose from memory (P_1) and the proportion of attempts to diagnose a fault from memory that result in misclassification (P_2). For 19 sets of 100 experimental replicates, the A-Test scores for P_1 and P_2 tend to fall around $a = 0.5$, which means that the median averages of P_1 and P_2 converge for 100 experimental replicates (although P_2 converges with a lower number of experimental replicates, which is indicated in Figure 4.2 by the A-Test score falling closer to $a = 0.5$).

4.4 Results and Discussion

For the experiments described in this chapter, the median average proportion of faults that were successfully diagnosed and resolved by their similarity to previously encountered faults, P_1, was 0.71

Across all experiments, where the system attempted to diagnose a fault using Equation 6.3, it arrived at an appropriate recovery option R_{1-3} in almost all cases (median average 100%). Therefore, the median average proportion of diagnostic attempts resulting in misclassification made by the system, P_2, is a remarkably low 0. To be clear, the system *does* occasionally misclassify a fault from memory, however these instances represent a tiny minority of cases.

Of the small number of cases where the system does misclassify a fault when attempting to diagnose from memory, a significant proportion occur in instances where normal swarm behaviour had changed since the last instance of the fault was

	H_1	H_2	H_3	H_4	H_5	H_6
Total	18	0	1	3	3	2
Behavioural Inconsistency	5	0	1	1	0	0
Fault Forgotten	0	0	0	1	3	0

TABLE 4.1: Total instances of attempts to diagnose from memory that resulted in misclassification (P_3).

H_1	H_2	H_3	H_4	H_5	H_6	H_m
0.95	0.92	0.93	0.97	1.00	0.92	0.71

TABLE 4.2: Median average correlation coefficients, P_3 ($\pm\sigma$), for diagnoses from memory over 100 replicates.

committed to memory (behavioural inconsistency), or where the last instance of the fault had simply been passed out of memory (fault forgotten). Table 4.1 shows that 41% of all misclassifications occurred in such cases. Unsurprisingly, the most commonly misclassified fault was software hang, H_1 – the variable nature of which was discussed in chapter 3.

Interestingly, in all cases where faults were misclassified, the misclassification occurred with a relatively low associated correlation coefficient, r. Table 4.2 details the average r-values with which each fault is diagnosed from memory in the 100 experiment replicates, where H_m indicates a misclassification.

Table 4.2 shows that, whilst all correct diagnoses from memory typically occur with an r-value between $0.92 - 1.00$, instances of misclassification occur with an r-value at around 0.71. Interestingly, all diagnoses of partial motor failure, H_5, occur with a very high r-value. This is because H_5 is only ever detected during obstacle avoidance behaviour. The reason for this is that H_5 does not cause a robot to deviate from normal behaviour significantly enough to be detected during flocking or aggregation behaviours. This means that every subsequent instance of H_5 occurs under very similar circumstances.

In all 100 experiments, there was no instance of misclassification that occurred with an r-value greater than 0.74. An obvious way to reduce instances of misclassification would therefore be to increase the required threshold for a fault to be diagnosed from memory from 0.66 to equal to or greater than 0.74. This would remove all instances of misclassification from the data, which would now simply have the diagnostic tests run on them. Given that this has to occur anyway, this would essentially be saving the system the time and resources required to carry out the incorrect recovery option. However, this would be at the expense of any successful diagnoses that occur with an r-value of less 0.74.

Until thorough recovery options are implemented as part of an integrated FDDR

	T_1	T_2	T_3	T_4	T_5	T_6
Control Steps	12	11	13	116	165	500
Instances	35	35	36	31	2	32

TABLE 4.3: Average time required to identify faults using diagnostic tests, and the number of additional instances each would occur if the similarity threshold was raised to 0.74.

	H_1	H_2	H_3	H_4	H_5	H_6
Total Instances	263	6	0	72	3	97
Behavioural Inconsistency	130	3	0	57	0	38
Fault Forgotten	2	2	0	1	3	0

TABLE 4.4: Total instances of faults requiring diagnostic tests despite previous encounters.

system, it is difficult to say exactly how time and resource efficiency would be impacted by changing the similarity threshold. From a purely diagnostic point of view, however, the average time taken to correctly and reliably identify a fault type using the diagnostic tests can be considered.

Table 4.3 reveals that, although increasing the similarity threshold would remove 27 instances of misclassification, it would also require an additional 171 diagnostic tests to be performed. It is worth noting here that the 32 instances of partial sensor failure would not add to the total time spent performing diagnostic tests as drastically as Table 4.3 would seem to imply, as partial sensor failure is diagnosed as complete sensor failure in 91% of cases. Nonetheless, these additional tests would still increase the average number of control steps the system had to spend performing diagnostic tests by approximately 2416 – equivalent to approximately 4 minutes of real time. In order for this to be a worthwhile trade, average recovery time for the system would have to be no shorter than 5.2 seconds. This is almost certainly unrealistic and so optimisation of the similarity threshold with respect to misclassification rate, time taken to perform diagnostic tests and time taken to recover is a problem that will need to be properly addressed in a fully integrated FDDR system of this kind once implemented in hardware.

That the median average proportion of faults diagnosable from memory, P_1, is 0.71, whilst the median average proportion of attempts to diagnose from memory that result in misclassification, P_2, is 0, means that there is a substantial number of instances in which the BFV signature of a fault does not produce a correlation coefficient with Equation 6.3 that is great enough for it to be recognised by the system and successfully diagnosed from memory.

In each experiment there will be a period of learning, where the system encounters faults for the first time, that necessitates use of diagnostic tests T_{1-6}. However, removing these scenarios from consideration, across all experiments there were a total of 441 instances where the diagnostic tests had to be run for a fault despite the system having previously encountered its type (not including cases where diagnostic tests had to be run because the system misclassified a fault when attempting to diagnose from memory with Equation 6.3). As with attempts to diagnose from memory resulting in misclassification, P_2, a significant portion of these instances arose in scenarios where there was behavioural inconsistency, or where the fault type had been forgotten by the system. A breakdown of the results can be seen in Table 4.4.

The results in Table 4.4 show that, of the 441 diagnostic tests run on faults after initial exposure, 53.51% occurred in cases where there was behavioural inconsistency, or where sufficient time had passed since the fault was last encountered that it had been passed out of the memory buffer. Furthermore, 59.64% of all cases were for instances of software hang, H_1. Notably, complete sensor failure, H_3, never had to be re-diagnosed after initial exposure, and the only reason the same cannot be said of partial motor failure, H_5, is because there were 3 instances in which it was passed out of the fault memory buffer. All other fault types in Table 4.4 take

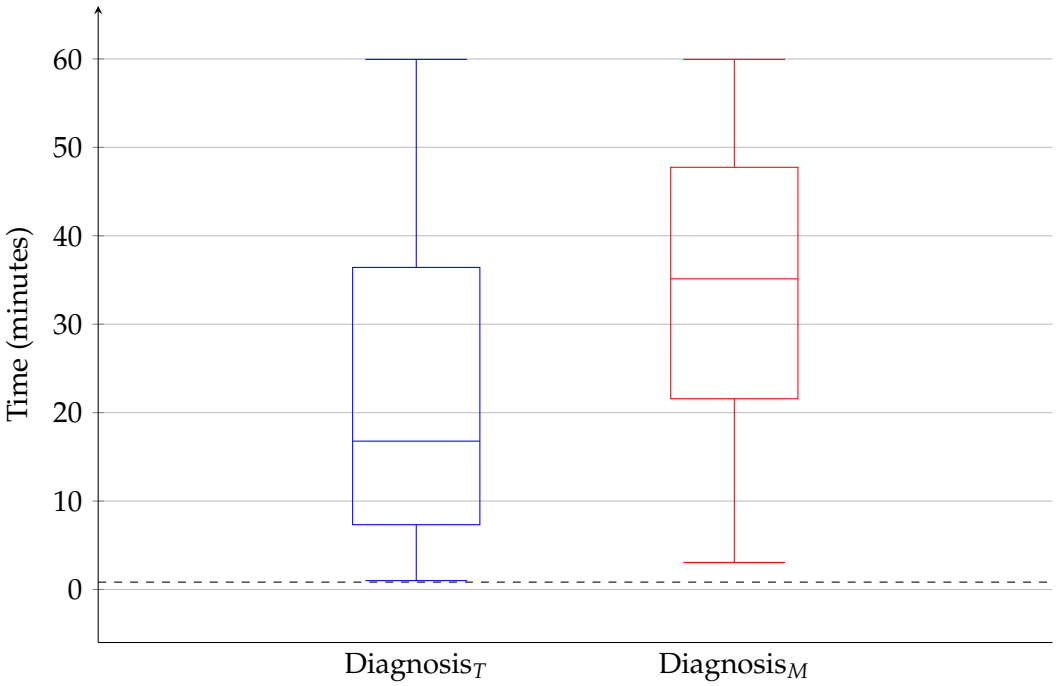

FIGURE 4.3: The times at which a fault is classified using diagnostic tests or from memory in each experiment replicate (1 hour total run-time). Dashed line indicates the point at which the first fault is injected in each experiment.

a significant contribution to their total number of occurrences from instances where swarm normal behaviour changes. A change in normal swarm behaviour significantly increases the probability of a given fault type exerting a different effect on an individual robot e.g. a software hang fault is far more likely to make a robot turn in circles if the robot was previously performing aggregation behaviour than if it was performing dispersion behaviour.

Figure 4.3 shows the time spread for faults classified from the diagnostic tests (Diagnosis$_T$) and from memory (Diagnosis$_M$). The results in Figure 4.3 clearly show a tendency for diagnostic tests to be used earlier in each experiment replicate and increasingly infrequently as time elapses. In contrast, the spread of times for diagnosis from memory appear to be, in general, evenly distributed with a slight offset from the first instance of fault injection. This means that, after a short initial period of each experiment where faults are being encountered for the first time, diagnosis from memory occurs with a fairly even spread for the rest of the experiment.

The results from Table 4.4, Figure 4.3 and the overall proportion of faults diagnosable from memory are evidence of the diagnosis system gradually learning and eventually establishing a multi-faceted characterisation of individual fault types online and in real-time.

4.5 Summary

This work described in this chapter has demonstrated a novel, adaptive and distributed approach to fault diagnosis in simulated robot swarms. The system presented here was able to diagnose fault types adequately in a majority of cases through a process of learning and memorising the fault's characteristics. In 71% of cases, faults were diagnosed based on their similarity to previous faults the system had

encountered and resolved. Furthermore, there was a very low rate of misclassification, a median average of 0% (and a mean average of $\leq$ 1%). The advantages of using BFVs have been retained, as this system exhibits unsupervised learning in real time.

Although the work described in this chapter showed promise for the initially proposed fault diagnosis method, several areas of improvement/further investigation in simulation were identified:

1. System parameters, such as the similarity threshold, s, were defined after a brief process of trail and error, but were not subjected to thorough scrutiny.

2. The method of fault detection used in this chapter was simplistic and served as a place holder until a more sophisticated fault detection algorithm could be integrated with the proposed method of fault diagnosis. The ways in which modifications to fault detection algorithms can affect the performance of fault diagnosis algorithms was not investigated in this chapter.

3. In real world scenarios it is possible that the diagnostic tests T_{1-6} could fail to appropriately diagnose a fault. This is something that was not investigated in this chapter, but will need to be addressed before this system can be deployed in the swarm robotic scenarios described by şahin (2005).

4. In real world scenarios it is possible that multiple robots may develop faults at the same time. The system's ability to deal with faults in parallel was not examined in this chapter.

5. Although not predicted to have any significant effect on overall performance, a useful modification identified during this work was the inclusion of active memory allocation. If a very high correlation coefficient is observed between two faults, it may be wasteful to store both in a finite memory buffer as opposed to relocating the older instance to the front of the buffer.

6. A swarm exhibiting the behaviours used in this work, flocking, aggregation and obstacle avoidance, cannot have its performance quantified in the way a swarm performing beacon-taxis can. To understand the benefits of the proposed fault diagnosis method, it should be implemented on a swarm where the performance of faulty and non-faulty swarms are quantifiable.

The work proceeding that described in this chapter aimed to address the six points listed above, and is detailed in the following chapter.

Chapter 5

Fault Diagnosis in Software Simulation: Part II

5.1 Introduction

Chapter 1 of this thesis proposes that faults can be diagnosed by examining the ways in which various faulty robot behaviours deviate from their normal behaviours. The previous chapter presented a novel method for immune-inspired fault diagnosis in autonomous robot swarms, this is achieved by encoding robot behaviour as behavioural feature vectors (BFVs) and associating the BFVs produced by faulty robots with the respective fault present in that robot. The experimental data collected and analysed suggested that this method of fault diagnosis could work effectively in real world scenarios and directly address the concerns highlighted by Winfield and Nembrini (2006) on fault tolerance in robot swarms. However, it was considered that the work described in the previous chapter was incomplete in terms of the cases tested and insofar as it made the critical assumption that autonomously assessed diagnostic tests would always arrive at an appropriate classification, which may not hold true for all systems and scenarios.

The work detailed in this chapter directly builds on that described in the previous chapter, by presenting what is largely the same fault diagnosis mechanism (albeit with some modifications) and subjecting it to more rigorous experimentation. Specifically, the system presented in this chapter is modified in the following ways:

1. The diagnostic tests (T_{1-6}) described in the previous chapter now have the potential to incorrectly classify a faulty robot
2. Multiple robots can be faulty at any one time
3. When writing fault BFV signatures to memory, a BFV signature will only be remembered if it is sufficiently different to previously encountered faults i.e. multiple very similar recollections of the same type of fault will not be retained in memory.

This chapter also tests the proposed system in the following new ways:

1. Variable system parameters, such as the time faulty behaviour has to be observed over before a fault is detected, are subjected to sensitivity analysis. Noise sensitivity analysis is also performed.
2. A swarm performing the beacon phototaxis algorithm described by Winfield and Nembrini (2006) is examined, and the performance of a faulty swarm with and without the proposed diagnostic system is compared.

This chapter aims to demonstrate and quantify how the proposed immune-inspired fault diagnosis mechanism addresses the problems on fault tolerant robot swarms that are discussed by Winfield and Nembrini (2006) and Bjerknes and Winfield (2013), as well as highlighting its limitations.

5.2 Experimental Method

As in the previous chapter, the work described here is conducted in Autonomous Robots Go Swarming (ARGoS) (Pinciroli et al., 2011) and uses simulated models of the marXbot (Bonani et al., 2010). In the experiments described in this chapter, all robots exhibit flocking (Algorithm 1), aggregation (Algorithm 2), obstacle avoidance (Algorithm 3) or collective photo-taxis swarm behaviours (Algorithm 4). In each of these behaviours, individual robots move in response to the position of neighbouring robots, obstacles or a point of interest. Normally functioning robots, that is robots that are not experiencing power failure, complete motor failure or in some instances complete sensor failure or software hang, will always be motional.

Algorithm 4 Collective Phototaxis

1: **while** Running **do**
2: **if** Line of sight to point of interest **then** close-proximity threshold = 3.5cm
3: **else** close-proximity threshold = 3cm
4: **if** Distance to neighbour < close-proximity threshold **then** avoid neighbour
5: **if** Robot heading **not in range** average bearing of neighbours $\pm$ 15 $^\circ$ **then** turn towards average bearing of neighbours
6: **else** Move forward

The fault types examined here are carried over from the previous two chapters, but listed again for convenience as follows:

- **Software hang (H_1)**: This fault will cause a robot to get stuck performing whatever action it was performing at the last moment it was normally functioning.
- **Power failure (H_2)**: A robot that suffers a power failure will completely stop moving and remain unresponsive to its surroundings.
- **Complete Sensor failure (H_3)**: For a complete sensor failure an individual robot will be unable to detect the presence of any neighbours, light sources or obstacles.
- **Complete Motor failure (H_4)**: A complete motor failure will cause one of the afflicted individual's motors to stop and henceforth become unresponsive to its controller. The other motor will remain functional, meaning that the afflicted robot will either stop completely if it is trying to turn using its faulty motor, or otherwise be limited to turning on the spot.
- **Partial Motor failure (H_5)**: For partial motor failure the individual's faulty motor will remain responsive, however it will only allow its associated wheel to turn at half speed.
- **Partial Sensor failure (H_6)**: For a partial sensor failure a robot will only be able to detect the presence of neighbours, light sources and obstacles within $\pm$60 $^\circ$ of its current heading.

The robot BFV used in this work, where the binary vector $BFV(t) = [F_1(t), F_2(t), F_3(t), F_4(t), F_5(t), F_6(t)]$, is also carried over from the work described in the previous chapter and given again here for convenience:

$$F_1(t) = 1 \text{ if } N_R(t) > 0, \text{ otherwise } F_1(t) = 0 \tag{5.1}$$

where $N_R(t)$ is the total number of neighbours in sensing range (approximately $1m$) of the robot at time t.

$$F_2(t) = 1 \text{ if } N_C(t) > 0, \text{ otherwise } F_2(t) = 0 \tag{5.2}$$

where $N_C(t)$ is the total number of neighbours at a distance less than the close proximity threshold, C, (approximately $0.3m$) from the robot at time t.

$$F_3(t) = 1 \text{ if } |v(t)| > 0.8|v_{max}|, \text{ otherwise } F_3(t) = 0 \tag{5.3}$$

where $|v(t)|$ is the magnitude of linear velocity at time t and $|v_{max}|$ is the maximum linear velocity a robot can achieve (approximately $5cms^{-1}$ in this work).

$$F_4(t) = 1 \text{ if } |v(t)| > 0.2|v_{max}|, \text{ otherwise } F_4(t) = 0 \tag{5.4}$$

$$F_5 = 1 \text{ if } |\theta(t)| > 0.4|\theta_{max}|, \text{ otherwise } F_5(t) = 0 \tag{5.5}$$

where $|\theta(t)|$ is the magnitude of angular velocity at time t and $|\theta_{max}|$ is the maximum angular velocity a robot can achieve (approximately $24^{\circ}s^{-1}$ in this work).

$$F_6(t) = 1 \text{ if } H_1 = True \text{ at time } t, \text{ otherwise } F_6(t) = 0 \tag{5.6}$$

Features are updated at each control-step with threshold values for F_3, F_4 and F_5 set to 80% of v_{max} (Equation 5.3), 20% of v_{max} (Equation 5.4) and 40% of ω_{max} (Equation 5.5), respectively.

It was demonstrated in Chapter 3 of this thesis that $F_1 - F_5$ were each individually discriminatory when a decision tree was used to classify the same six fault types, and the merits of F_6 for classifying software hang faults were clearly demonstrated in Chapter 4 of this thesis.

As in the previous two chapters, each robot estimates its own BFVs proprioceptively as well as exteroceptively estimating the BFVs of any of its neighbours in range.

5.2.1 Integrated Fault Diagnosis

The fault diagnosis mechanism proposed in the previous section is restated here for convenience, and the modifications made in this work highlighted.

Fault Detection

Faults are detected according to Equation 5.7. This is the same equation as that used for fault detection in the previous chapter, however the variables as these will be investigated in the experiments described later in this chapter.

$$\text{fault detected at time } t = true \text{ if } \sum_{t=T-W}^{T} d(t) \geq \rho W \text{ for } T > W \tag{5.7}$$

Where T indicates the time elapsed since a robot started being observed by a neighbour. The binary value d is set according to Equation 5.8, where $d = 0$ if there is agreement between exteroceptively and proprioceptively estimated robot features. A robot will make W observations of its neighbour, once per control-step, at discrete times, t, before a fault can be detected, where t indicates the time elapsed since the start of an experiment. The value of ρ indicates what proportion of the W observations must prompt the value of d to be equal to 1 in order for a fault to be detected.

$$d(t) = \begin{cases} 0 \text{ if } F_i^p(t) = F_i^e(t) \text{ for } i = 1,..,6 \\ 1 \text{ otherwise} \end{cases} \tag{5.8}$$

As in the previous chapter, whenever a fault is detected in a robot by another member of the swarm, the set of BFVs which prompted the fault to be detected are then used as the BFV signature for that particular fault.

Fault Diagnosis

The diagnostic tests described in the previous chapter are carried over for this work, and restated here for reader convenience:

- **Software hang (T_1)**: This occurs immediately after the assessing robot moves into testing range. In order for the faulty robot to perform later diagnostic tests, it must be able to communicate with the assessing robot. To establish this communication, the assessing robot first pings the faulty robot and instructs it to stop moving. If the faulty robot is suffering from software hang, it will disregard this request. If the faulty robot is persistently unreachable and ignores the assessing robot's stop request, it will be diagnosed as having software hang. This relies on the assumption that communication between normally operating robots is reliable.

- **Power failure (T_2)**: Almost identical to T_1, however in cases where the robot is suffering from power failure it will also stop moving. The assessing robot therefore diagnoses power failure in instances where the faulty robot is unreachable but stationary. It is acknowledged that this would only work for normal behaviours where robots should always be in motion. For a swarm exhibiting behaviour that necessitated periods where the robot was stationary, the relationship between T_1 and T_2 would have to be revised.

- **Complete Sensor failure (T_3)**: Once the faulty robot has been established as responsive to the assessing robot, the two robots should have settled within range of each other – something that will be reflected in their BFV. A complete sensor failure is diagnosed if the faulty robot's feature F_1^p returns a value of 0.

- **Complete Motor failure (T_4)**: The faulty robot is asked to spin on its right and left wheels. If the faulty robot receives and processes this request but is unable to perform one or both of the turns, it is decided that the robot is suffering from complete motor failure. This is ascertained by monitoring the faulty robot's feature F_4^e, where a returned value of 0 indicates the robot has failed the test.

- **Partial Motor failure (T_5)**: Having established that both of its motors are responsive, the faulty robot is then asked to demonstrate that both motors work together as they should. The faulty robot is asked to move forward in a straight

line. If the faulty robot is observed to move in an arc or otherwise deviate from moving in a straight line, it is decided that the faulty robot is suffering from partial motor failure. This is achieved by monitoring the faulty robot's feature F_5^e, where a returned value of 1 indicates that the robot has failed the test.

- **Partial Sensor failure (T_6)**: To test for partial sensor failure, the faulty robot performs a 360° turn whilst acknowledging the presence of the assessing robot. If the faulty robot fails to acknowledge the assessing robot for any period of this turn, the robot is decided to be suffering from partial sensor failure.

Fault Recovery

The recovery options for faults H_{1-6} used in this work are the same as described in the previous chapter, but are restated for reader convenience:

- **Power Cycle (R_1)**: It is assumed that when a robot has its power cycled, it is turned back on with a new or replenished power source. This recovery action resolves software hang (H_1) and power failure (H_2) faults.
- **Sensor Replacement (R_2)**: Replacement of robot sensors resolves complete and partial sensor failures (H_3 and H_6, respectively).
- **Motor Replacement (R_3)**: Replacement of robot motors resolves complete and partial motor failures (H_4 and H_5, respectively).

As in the previous chapter, when a recovery action is executed the faulty robot continues to be observed by the assessing robot. If the faulty robot's exteroceptively and proprioceptively estimated BFVs agree after a recovery action has been performed, the fault is considered to be resolved.

An extension made by the work described in this chapter is that the success of diagnostic tests T_{1-6} in identifying a robot's fault type is also evaluated. For the vast majority of cases, diagnostic tests T_{1-6} will classify a fault correctly, however there are some circumstances in which they misclassify faults or the newly-resolved robot is unable to convince the assessing robot that it is no longer faulty. If a robot's fault persists after having attempted diagnostic tests T_{1-6}, the faulty robot is declared to be beyond the swarms ability to recover it, and so it is shut down. The faulty robot will now be, for all intents and purposes, an inanimate object in the environment. On the other hand, if a robot is subjected to diagnostic tests T_{1-6} and passes them all, the faulty robot is treated as a false-positive and is allowed to continue operating normally. The fault signature is considered void in this case and not used for any subsequent comparison.

Fault Memory

As in the previous chapter, each time a fault is successfully resolved, the BFV signature for that fault and the respective diagnosis are stored in the assessing robot's memory, which is then shared with other members of the swarm via local communication. When subsequent faults occur, the BFV representations of those faults can be checked for similarity against the BFV representation of previous faults.

Similarity between two faults is again established by finding the Pearson correlation coefficient (Benesty et al., 2009) between their BFV signatures (Equation 6.3):

$$r = \frac{\sum_m \sum_n (F_{mn} - \overline{F})(F_{0_{mn}} - \overline{F_0})}{\sqrt{(\sum_m \sum_n F_{mn} - \overline{F})^2 (\sum_m \sum_n F_{0_{mn}} - \overline{F_0})^2}} \tag{5.9}$$

Where r is the correlation coefficient between a current and previous fault signature, F and F_0, respectively, where F and F_0 are two dimensional binary data sets. $\overline{F}$ and $\overline{F_0}$ are the mean values of sets F and F_0, respectively, and m and n indicate positional index within the 2D binary data set (the specific orientation of m and n is arbitrary for this equation).

As in the previous chapter, when attempting to diagnose a fault from memory, the previously stored fault which produces the greatest r-value (Equation 6.3) is considered the most likely fault type and recovery is initiated in sympathy with that fault. The minimum r-value required for two faults to be considered similar, and thus eligible for this process is investigated at the end of this chapter.

5.2.2 Experimental Setup

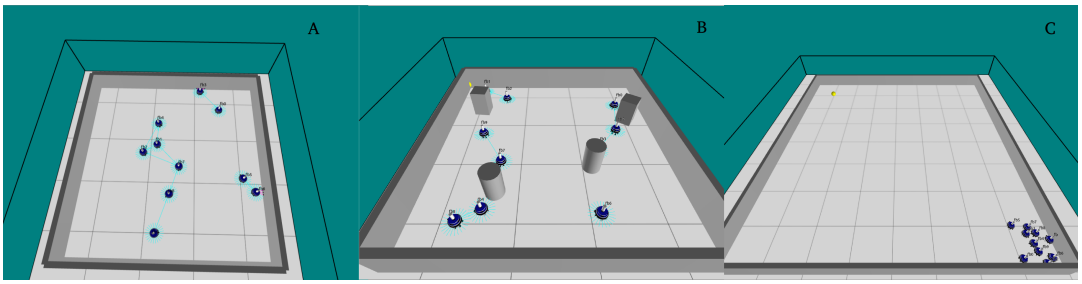

FIGURE 5.1: **A**: ARGoS simulation of 10 marXbot robots performing obstacle avoidance in an empty arena (O'Keeffe et al., 2017b), **B**: ARGoS simulation of 10 marXbot robots performing obstacle avoidance in a cluttered arena, and **C**: ARGoS simulation of 10 marXbot robots performing collective photo-taxis in an arena with a beacon in the north-west corner.

In all experiments described in this chapter, features are updated at every control-cycle ($10ms$). Gaussian noise is applied to the raw linear and angular velocity readings obtained by robot's RAB sensors and its compass. These are the only sensor-obtained values that the proposed system uses to estimate robot BFVs, and thus relies on for overall performance. It is acknowledged that system noise in hardware will not necessarily correspond to noise added in simulation, this effect is known as the reality gap (Jakobi, Husbands, and Harvey, 1995). However, the purpose of implementing noise in this manner is to demonstrate that the proposed system is able to retain performance with imperfect information on robot states. The noise applied to robot RAB sensor readings during the experiments described in this chapter (with the exception of those in section 5.3.2) is as follows:

- Linear velocity, simulated Gaussian noise ($\mu = 0, \sigma = 5\%\ d_{max}$)
- Angular velocity, simulated Gaussian noise ($\mu = 0, \sigma = 5\%\ \omega_{max}$)

Where d_{max} is the maximum distance that a robot can travel in a single control step (approximately $0.5cm$) and ω_{max} is the maximum angular distance a robot can travel in a single control-step (approximately 2.4°).

The system parameters used in these experiments are as follows:

- The length of observation period, o, during which the assessing robot decides post-diagnosis whether or not the associated recovery action is successful in resolving faulty behaviour. $o = 29$.
- The similarity threshold, s. When two faults produce an r-value using Equation 6.3 that is greater than s they are considered similar for the purpose of diagnosis from memory. $s = 0.56$
- The detection window, W. The positive integer W represents the number of control-steps over which a faulty robot must be observed to be producing mismatched BFVs before it is declared faulty. This is the accumulation of the data the system has with which to characterise a given fault and attribute to it a signature. $W = 29$
- The proportion of BFVs in W, ρ which must be mismatched before a fault is declared. $\rho = 0.88$

Details of how these parameter values were chosen are given in section 5.3.4 of this chapter.

Fault memory is written to a circular buffer with capacity for 18 discrete fault signatures, or three times the number of unique fault types. 18 places typically gives a high probability that each type of fault has at least one BFV representation in system memory (assuming all fault types have been encountered).

When a fault type and its associated BFVs are stored in the assessing robot's memory, the assessing robot then passes the information to any other robots it comes within range of, these robots then do likewise and so on. This simulates the swarm of robots sharing and updating fault memory via an *ad-hoc* network. To avoid thrashing, particularly as swarm size increases, robots record which neighbours they have shared memory with and will not attempt to do so again until at least 1000 control-steps have passed.

5.3 Experiments

The criteria used to characterise the proposed system's performance during the experiments described in this chapter (with the exception of those performing collective photo-taxis) are as follows:

- P_1: The proportion of faults the system is able to diagnose from memory
- P_2: The proportion of attempts to diagnose a fault from memory that result in misclassification or are otherwise unrecognised as by the system as being correct.
- P_3: The average r-value between faults, using Equation 6.3, when diagnosing from memory
- P_4: The total number of faulty robots the system is unable to resolve and are shut down in an hour of simulated time
- P_5: The average time (in control-steps) it takes the system to detect a fault
- P_6: The total number of faults the system is able to detect

As in the previous chapter, 100 replicates are performed for all experiments, from which a median average is taken for P_{1-6}.

5.3.1 Scalability & Flexibility

To investigate the scalability and flexibility of the proposed system, a swarm of simulated marXbots collectively perform flocking, aggregation or obstacle avoidance behaviour in a proportionally scaled enclosed square arena. The ratio of robots to square metres is set at 1 : 1.6 (see Figure 5.1A).

As in the previous chapter, these experiments run for one hour of simulated time (36000 control-steps). The swarm's normal behaviour is updated randomly between flocking, aggregation or dispersion at 5000 control step intervals. A maximum of 50% of the robots in a swarm can be faulty at one time, so that there are never more faulty robots than the swarm can provide assessors for. Each robot has a 0.1% probability of having a random fault injected at each control-step, which was chosen after a process of trial and error. A 0.1% probability of having a fault injected was found on average to prompt new faults to be injected into robots with a regularity such that it was common for more than one robot to be faulty at a time, but very rarely reached the maximum 50% of robots with faults as they would be resolved before this could occur. As in the previous chapter of this thesis, instances of false-positive fault detection are not examined (see section 4.3.4 for details).

Results & Discussion

The performance of the proposed diagnostic system for robot swarms of varying sizes is displayed in Table 5.1. The total number of faults detected, P_6, is removed from the table as this will increase proportionally as swarm size increases, and thus is not an indicator of the system's scalability.

Swarm size	P_1	P_2	P_3	P_4	P_5 (control-steps)
10	0.68	0.04	0.87	0	444
40	0.76	0.03	0.91	0	572
90	0.76	0.04	0.90	0	474
160	0.74	0.05	0.90	1	592

TABLE 5.1: Performance of system for swarms of varying sizes

Table 5.2 shows how the performance of the swarm varies when obstacles are added to the arena, and when fault memory gathered from the swarm in an empty arena is imported into a swarm in an arena with obstacles (see Figure 5.1B). The purpose of the latter is to demonstrate that the proposed system is flexible to varying or dynamic environments.

Arena type	P_1	P_2	P_3	P_4	P_5	P_6 (control-steps)
No obstacles	0.68	0.04	0.87	0	444	30
Obstacles	0.68	0.04	0.87	0	403	30
Obstacles (imported memory)	0.93	0.03	0.91	0	413	30

TABLE 5.2: Performance of system for robot swarms in different environments

Table 5.1 shows the proportion of faults successfully diagnosed from memory, P_1, increases slightly between swarm size 10 and 40. P_1 then stabilises between swarm sizes of 40 and 90, in which no significant trend is observed. Performing a Wilcoxon

rank-sum test (Haynes, 2013) on the data distributions of P_2 for swarm sizes of 40 and 90 produced a value $p = 0.98$, where $p > 0.05$ indicates that two data sets are drawn from the same distribution. The small drop in P_1 for swarm sizes of 160 is discussed later in this section. The reason for the increase in P_2, from $P_2 = 0.68$ to $P_2 = 0.76$ between swarm sizes of 10 and 40, respectively, is because of the tendency to diagnose from memory more frequently as each experiment run goes on. In each experiment the system reaches a point of familiarity with the faults it can be subjected to, after which it can diagnose subsequent faults from memory in a majority of cases (approximately 76% of cases for swarms consisting of 40 and 90 robots, and 74% of cases for swarms consisting of 160 robots). In the previous chapter it was demonstrated that the majority of diagnoses occur in the latter two thirds of the hour-long experiments for a swarm with 10 robots – this is approximately where the system reaches its point of familiarity. The rate at which the system is able reach this point will be affected by how often the system encounters faults and how quickly it can resolve them. Reaching the point of familiarity will be accelerated as swarm size increases (as this will naturally increase the frequency of faults occurring if the probability of occurrence is constant) and, combined with the increased overall amount of faults encountered by the system, contributes to the increase in P_1 observed between swarm sizes 10 and 40. This would suggest that the saturation point of the system reaching the point of familiarity with respect to increasing swarm size lies between 10 and 40. Table 5.3 reveals that this rise in actually occurs over a comparatively small subsection of the range – between 15 and 21 robots – before stabilising for swarm sizes of up to 90 robots.

Swarm size	P_1
15	0.69
16	0.73
17	0.73
18	0.74
19	0.75
20	0.75
21	0.76

TABLE 5.3: Proportion of faults successfully diagnosed from memory for swarms of varying sizes

Table 5.1 does not display a clear trend between swarm size and the proportion of unsuccessful attempts to diagnose faults from memory, P_2. The distribution of P_2 does not change significantly for swarm sizes of 10 and 90, performing a Wilcoxon rank-sum test on these distributions produces a value $p = 0.152$. P_2 is at its highest for a swarm size of 160. This, coupled with the slight drop in the proportion of faults successfully diagnosed from memory, P_1, could be attributed to the increased probability that a faulty robot will find itself near the centre of a robot cluster at the time it is being observed by an assessing robot. If the robots neighbouring the faulty robot are unable to disperse or otherwise move out of the faulty robot's path it may disrupt the assessment process.

The total number of faults that are unresolvable, P_4, is 0 between swarm sizes of 10 and 90. P_4 increases from 0 to 1 for swarm size 160, which could be attributed to the vastly increased swarm size increasing the probability of the circumstances in which a faulty robot is unresolvable occurring within one hour. The average r-value

between faults, P_3, and the average time taken to detect faults, P_5, do not display a clear trend with increasing swarm size.

Table 5.1 reveals slight fluctuation in performance with varying swarm size. These fluctuations are observed over an order of magnitude, whilst P_1, the most indicative measure of performance, remains within approximately 90% of its observed maximum. Based on this, it is proposed that the proposed diagnostic system is generally stable – and thus scalable – for practical swarms of between 10 and 160 robots.

Table 5.2 shows that the only difference in median average performance between a swarm in an empty arena and a swarm in a cluttered arena is a reduction in the average time taken to detect faults, P_5, of approximately 9%. For swarms in an arena with and without obstacles, there is no significant difference between the distributions of performance criteria $P_{1-4,6}$ when subjected to a Wilcoxon rank-sum test (p-values $0.33, 0.33, 0.31, 0.81, 0.55$, respectively). When fault memory from the swarm in an empty arena is imported to a swarm in an arena with obstacles, there is a large improvement in the proportion of faults successfully diagnosed from memory, (P_1), an additional 25% of all faults. There is also an improvement in the proportion of attempts to diagnose faults from memory that were unsuccessful, P_2, which falls from 4% to 3% (representing a reduction of 25%). This is because the system begins the experiment already at the point of familiarity with the faults that will be injected and so is able to diagnose a greater proportion from memory – even though the system's gathered memory originates from the empty arena. This suggests that there is little difference between the fault data gathered from robots performing in empty arenas and those in cluttered arenas. The data in Table 5.2 suggests that the proposed system is flexible and robust to environments with varying degrees of complexity, so long as they do not inhibit the swarm from performing normally.

5.3.2 Noise Sensitivity

These experiments again use an identical setup to that described in the flexibility and scalability experiments (described in section 5.3.1), with the exception that the Gaussian noise applied to the calculations of robot linear and angular velocity is varied.

Latin Hypercube Analysis is used to test the proposed system's performance for varying levels of noise on the linear and angular velocity readings obtained by robot RAB sensors. Latin Hypercube Analysis is a statistical technique for sampling multi-dimensional data distributions, two dimensional in this instance. The SPARTAN R package (Alden et al., 2013) is used to perform the sampling for Latin Hypercube Analysis, which allows 500 parameter sets to be randomly generated, where each parameter value is unique in the set, in the following ranges:

- RAB sensor linear velocity reading, simulated Gaussian noise ($\mu = 0, 0 < \sigma < d_{max}$)
- RAB sensor angular velocity reading, simulated Gaussian noise ($\mu = 0, 0 < \sigma < \omega_{max}$)

Again, 100 experimental replicates are performed for each parameter set from which the median value is taken for performance criteria P_{1-6}.

Results & Discussion

The results of these experiments are plotted where there is a visible trend to be observed between the standard deviation of applied Gaussian noise and the performance criteria (P_{1-6}). This is displayed in Figure 5.2 and Figure 5.3.

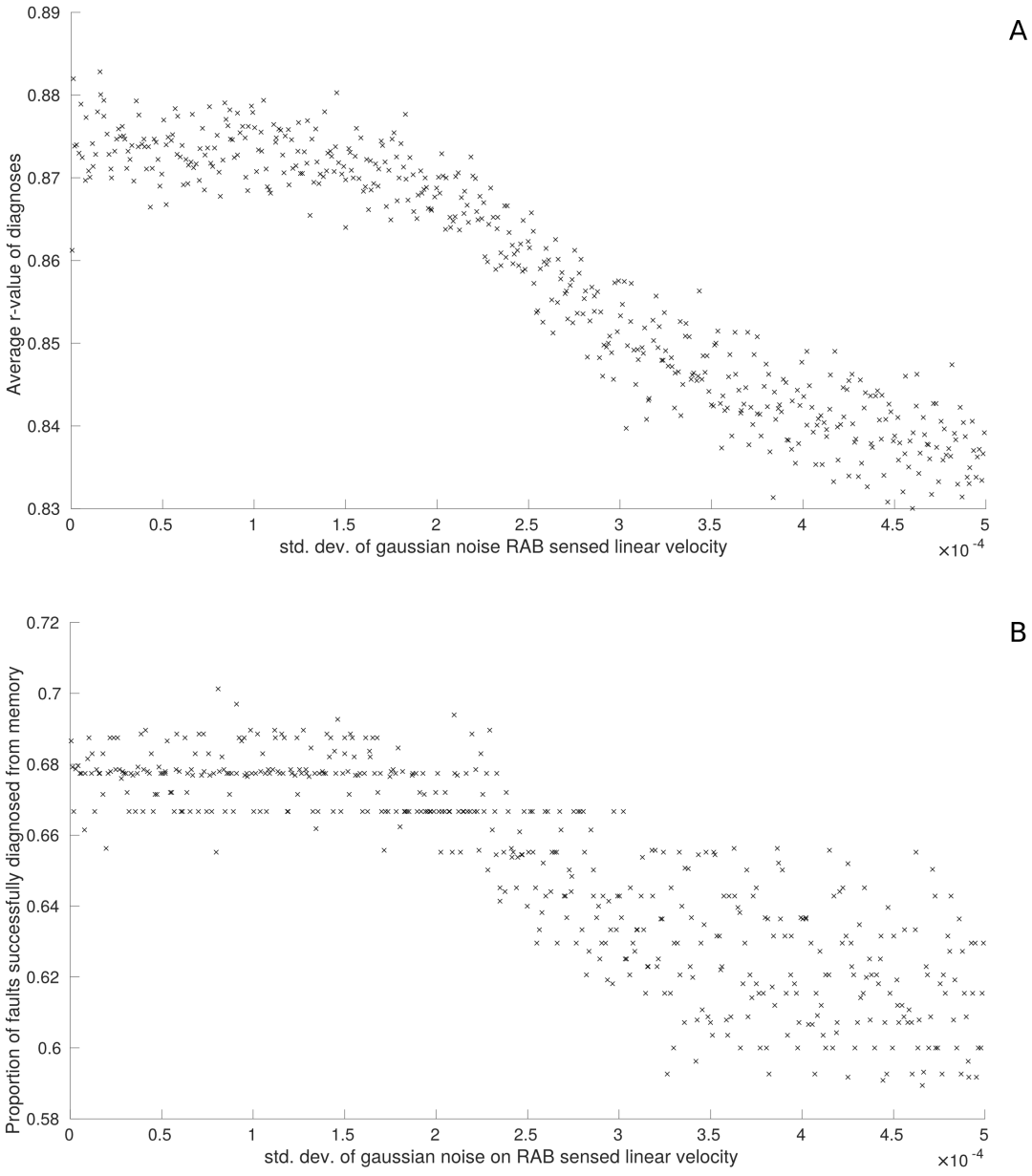

FIGURE 5.2: **A**: The effects of noise on RAB sensed linear velocity on median r-values between diagnoses, and **B**: the effects of noise on RAB sensed linear velocity measurements on the proportion of faults successfully diagnosed from memory

Unsurprisingly, no correlation was observed between noise applied to RAB sensor readings and the average time taken to detect faults, P_5.

The average r-value between faults, P_3, was noticeably affected by noise applied to linear velocity estimations obtained from RAB sensors (see Figure 5.2A), although this is observed over a comparatively small scale. Interestingly, there was no obvious

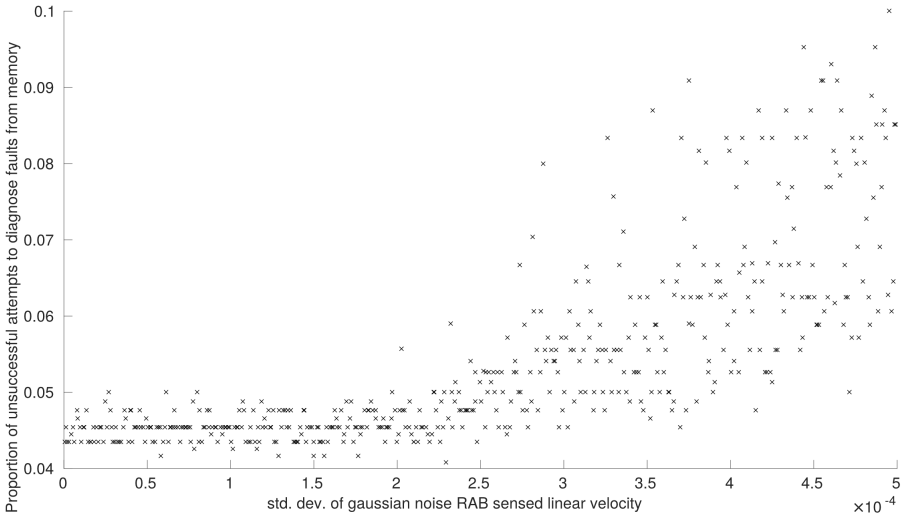

FIGURE 5.3: The effects of noise on RAB sensed linear velocity measurements on the proportion unsuccessful of attempts to diagnose faults from memory

correlation between P_5 and the noise applied to RAB estimated angular velocity. The reason for this observed robustness is that angular velocity is only able to affect a single robot feature F_5^e. The maximum effect that noise can have on a feature is setting it to an incorrect state. Because the features used in this chapter are set by thresholds, in order for the true state to be altered by system noise, the noise needs to be of a significant magnitude *and* of the appropriate sign. As the simulated noise generated on sensor readings is normally distributed, this means that the probability that a feature state is correct will always be greater than that of it being incorrect, no matter how large the standard deviation of noise is set to be. These factors combined mean that the overall change in robot BFVs is comparatively small and therefore the r-values produced between faults represented by these BFVs is not significantly reduced.

The same trend is observed for the proportion of faults successfully diagnosed from memory, P_1, and proportion of attempts to diagnose faults from memory that are unsuccessful, P_2. Noise applied to RAB sensed linear velocity has a clear correlation with system performance (see Figure 5.2B and Figure 5.3, respectively) whilst noise applied to RAB sensed angular velocity was not observed to have any noticeable effect by comparison.

The total number of unresolvable faults, P_4, was unaffected by noise applied to sensor readings. This is primarily because a robot is only declared faulty when it is consistently unable to pass diagnostic tests $T_1 - T_6$. Of these diagnostic tests, the only tests reliant on feature observations that could be disrupted by sensor noise are T_4 and T_5 (constituting a third of all faults encountered by the system on average) and, even with maximum sensor noise applied, there is a greater probability that features will remain in the correct state at any given point in time. Although increasing sensor noise will increase the probability that diagnostic tests need to be run (Figure 5.3, Figure 5.2B), there is still only a slim increase in the overall probability that a faulty robot will be deemed unresolvable.

The work in this chapter does not examine how sensor noise would impact fault detection. Based on the effects of noise on RAB sensor readings, (Figure 5.3), it is

predicted that increasing sensor noise (at least on the virtual range sensors) would drastically increase the amount of false-positives encountered by the system in a scenario where fault detection was more realistically implemented. Although the results in this section demonstrate that the system would generally be capable of resolving these scenarios, they would correlate with an increase in the amount of time spent with one or more robots performing diagnostic routines, which may cause problems in time-critical scenarios or where a certain number of functioning robots are required.

5.3.3 Collective Photo-taxis

These experiments again consider a swarm of simulated marXbots, this time performing collective photo-taxis behaviour. The implementation of this behaviour here is similar to that used by Bjerknes and Winfield (2013), whereby robots in the swarm will attempt to aggregate whilst avoiding collisions with one another. The behaviour is described in Algorithm 4. Robots that are closer to a beacon will avoid neighbouring robots at greater distances than those that are furthest away (robots usually turn to avoid collision at distances less than $0.3m$, for these experiments the robots closest to the beacon will turn away from neighbours at distances less than $0.35m$). The emergent effect of this behaviour is that the swarm gravitates towards the beacon.

Here the performance of a normally functioning swarm, a faulty swarm without active fault tolerance and a faulty swarm with active fault tolerance are compared. The experiments in this section use a swarm of 10 marXbots in an enclosed arena ($6m \times 6m$), where robots begin in the south-east corner of the arena and are drawn to a light source at the north-west corner (see Figure 5.1C).

The diagnosis system is implemented in the same manner as previously. Robots that are engaged in the diagnostic process are unable to participate in collective photo-taxis, however the assessing robot will still repel nearby neighbours, preventing the diagnostic process from anchoring the swarm. Faults are injected into between 1 and 5 robots after the first 500 control-steps of each experiment, as this was found to be the point at which swarm behaviour had typically stabilised (one or more robot clusters will have formed after 500 control-steps have passed).

The performance criteria examined in these experiments are the times taken for the first member of the swarm to reach the beacon, the time taken for half the members of swarm to reach the beacon, and the time taken for all members of the swarm to reach the beacon, and the average distance of all members of the swarm at these times, respectively. An individual robot is considered to have reached the beacon when the distance between the two is less than $0.8m$.

As with all other experiments, 100 replicates are performed from which a median average is taken for all performance criteria.

The performance of a normally functioning swarm is used as a point of reference, which is displayed in Table 5.4.

The performance of a swarm where $1-5$ robots are faulty is examined as a proportion of the performance of a normally functioning swarm, where the value for proportional performance is given by Equation 5.10.

$$\text{Proportional Performance } (P_p) = \frac{1}{3}\sum_{i=1}^{3}\frac{dt_{i_0} - dt_i}{dt_{i_0}} \tag{5.10}$$

	Time (control-steps)	Avg. dist. from beacon (metres)
1st Robot	24388	1.4275
Half Robots	25548	0.8517
All Robots	26909	0.4746

TABLE 5.4: Normally functioning swarm performance of collective photo-taxis

Where dt_{1-3_0} and dt_{1-3} are the average distances between a normally functioning swarm and the beacon and the average distances between a faulty swarm and the beacon after 24388, 25548 and 26909 control-steps, respectively (these values are taken from Table 5.4). Equation 5.10 is such that a swarm that takes longer to reach the beacon than the normally functioning swarm described in Table 5.4 will produce a proportional performance value $P_p < 0$. A swarm that takes the same amount of time to reach the beacon as the normally functioning swarm will produce a proportional performance value $P_p = 0$. A swarm that reaches the beacon in a shorter amount of time than the normally functioning swarm will produce a proportional performance value $0 < P_p < 1$.

The proportional performance of robot swarms with and without the proposed diagnosis system are then compared.

Results & Discussion

Figure 5.4A displays the proportional performance of the robot swarm performing the ω-algorithm with up to 5 faulty robots where the proposed diagnosis system is not implemented. Figure 5.4B displays the proportional performance of the same swarm in the same conditions where the proposed diagnosis system is implemented.

For an output of Equation 5.10 that returns $P_p > 0$, the test swarm has performed better than the control swarm. In some cases, the test swarm outperforms the control swarm by such a large margin that comparing the distances at the times of interest listed in Table 5.4 loses applicability. In this case the value of some or all parts of dt_{1-3} go to 0, returning a proportional performance value of 1. As Equation 5.10 should only ever approach 1 asymptotically, performance plots in these instances are removed from Figure 5.4, but it should be noted that the time taken for all members of the swarm to reach the beacon was less than that of a normally behaving swarm.

With the exception of some cases of partial sensor failure, H_6, the performance of the test swarm with any number of faulty robots is worse than for the control swarm, which corresponds with the findings of Winfield and Nembrini (2006).

Interestingly, Figure 5.4A shows that the presence of some faults, most notably partial sensor failure, H_6, actually improves overall performance from that of the control swarm. Depending on how many robots are faulty, this ranges from an average of $+0.274$ proportional performance (given by Equation 5.10) to such an improvement that all robots had reached the beacon in less than 24388 control steps. This is true for any number of faulty robots with partial sensor failure. The reason for this is that robots afflicted with partial sensor failure will not be inclined to aggregate towards the centre of a swarm if the centre of mass is behind them. This means that the faulty robots will press ahead, whilst normally functioning robots are consequently able to move toward the beacon at a faster rate.

The performance of the swarm under the same conditions with the proposed diagnostic system implemented is shown in Figure 5.4B.

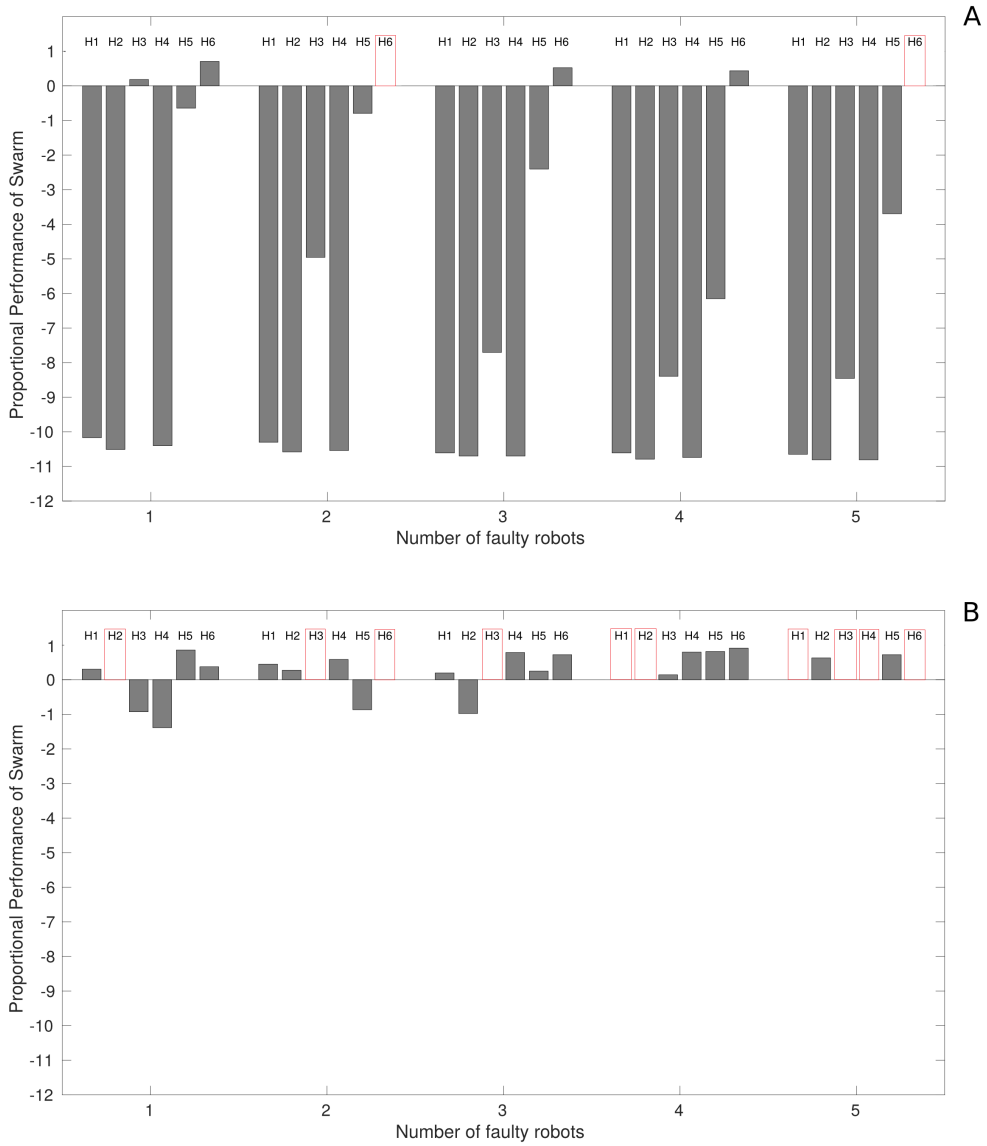

FIGURE 5.4: **A**: The performance of a robot swarm with no active fault tolerance exhibiting the collective photo-taxis when a varying proportion of robots are subjected to a variety of faults, and **B**: The results of the same experiment with active fault tolerance. Where there is a red box over the condition type indicates that proportional performance is not applicable because the entire swarm reached the beacon in less than 24388 time-steps, meaning that average distances at t $\geq$ 24388 control-steps could not be taken.

Figure 5.4B shows that, again with exception of some cases of partial sensor failure, H_6, the presence the proposed diagnosis mechanism results in a large improvement in overall system performance where faulty robots are present. Depending on the type of fault and the number of faulty robots, this can be as much as a $+11.8$ improvement in proportional performance (given by Equation 5.10). Additionally, in a number of cases overall performance is again better than that of a normally functioning swarm. This is because when two robots are involved in a diagnostic routine, they will repel other normally functioning robots in the swarm, allowing them to continue toward the beacon unhindered and at a faster rate brought about

by a smaller local swarm size (this is only true for local swarm sizes with 3 or more robots, however this is most likely the case in these experiments given that all robots start experiments within local sensing range of each other Figure 5.1C).

Figure 5.4 validates the proposed diagnostic system in the context of the original proposal for active fault tolerance in swarms by Winfield and Nembrini (2006). The results in this section demonstrate a clear improvement in swarm performance in most cases where active fault tolerance mechanisms are present.

5.3.4 Parameter Sensitivity Analysis

Using an identical setup to that detailed in the scalability and flexibility experiments in section 5.3.1 (with the exception that swarm size is kept at 10 and the arena clear of obstacles for all experiments), the sensitivity of overall system performance to variable system parameters is tested.

The purpose of these experiments is twofold; to identify a baseline parameter configuration for future work, and to shed a general light on where certain parameter configurations become unacceptable.

Latin Hypercube Analysis was used again to test the proposed system's performance for varying parameter values. The SPARTAN R package (Alden et al., 2013) was again used to randomly generate 500 parameter sets in the following ranges:

- Observation period, $1 < o < 100$ (control-steps)
- Similarity threshold, $0 < s < 1$.
- Detection window, $1 < W < 500$. (control-steps)
- The proportion of mismatched BFVs in W, $0.01 < \rho < 1$.

Once again, 100 replicates are performed for each parameter combination, from which a median is taken for performance criteria P_{1-6}.

Results & Discussion

Results from these experiments are again only plotted where there is a correlation to be observed between a parameter value and performance criteria (P_{1-6}).

Sensitivity to observation period The value of the observation period, o, predictably, did not effect the average time taken for the system to detect faults, P_5. More surprisingly, the value of o did not effect the total number of faults detected by the system, P_6. This is believed to be because of two factors; the decentralised nature of the proposed diagnostic system means that faults can be recovered in parallel i.e. the system does not need to wait until one fault is resolved before it can begin working on another, and secondly; the 100 control-step maximum value of o is a short amount of time compared to the 36000 control-step experiment length.

The value of o most prominently affected the total number of faulty robots that the system is unable to resolve, P_4 (see Figure 5.5). In the experiments performed, diagnostic tests T_{1-6} failing to resolve the fault occurred under one scenario; where a faulty robot became stuck in a corner or against a wall or object. In these circumstances the fault could have been diagnosed correctly, either from memory or the diagnostic tests, and the fault resolved. However, the subsequent observations of the robot would lead the assessing to believe the fault had persisted. This is because

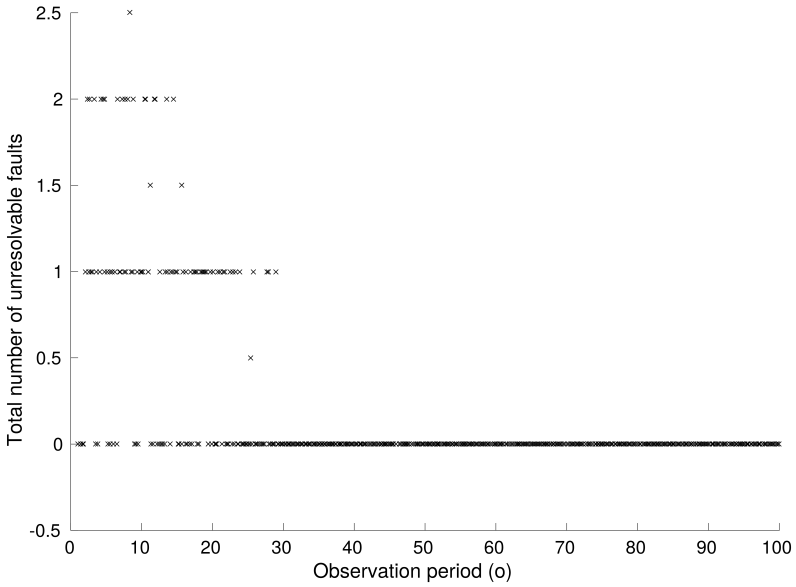

FIGURE 5.5: The median total number of unresolvable faults, P_4, over one hour of simulated time against the value of the observation period, o. Non-integer values indicate a median value between two integer values, rather than 0.5 unresolvable robots.

the faulty robot would try to move normally but be obstructed by the walls of the arena. How long the robot is observed for directly corresponds to the likelihood of this scenario occurring; the larger the value of o, the more opportunity a stuck robot has to become unstuck and demonstrate its normal function.

The the value of o also has a somewhat less pronounced effect on the proportion of faults successfully diagnosed from memory, P_1, the proportion of unsuccessful attempts to diagnose faults from memory, P_2, and the average r-value produced between faults P_3. For all three properties, values of $o \leq 5$ tend to severely reduce performance. Above this value, however, there is no observable correlation as o increases. To minimise the time spent in the diagnostic process and the number of robots lost, and to maximise reliability of diagnoses from memory, the value of o is set at $o = 29$.

Sensitivity to similarity threshold In these experiments the similarity threshold, s, was observed to have no effect on the total number of faults detected, P_6, or the average time taken to detect them, P_5. The value of s was not observed to have any obvious correlation with the total number of unresolvable faults, P_4, as this is caused by the circumstances in which the diagnostic tests are run, rather than how well two faults correlate. As s increases, the average r-value between faults, P_5, also increases. However, this is simply because lower r-values are removed from the set of values which are averaged.

As s approaches 1, the proportion of faults successfully diagnosed from memory, P_1, remains largely unresponsive until $s > 0.55$, at which point it rapidly declines (suggesting that the average r-value for all fault, behaviour and parameter combinations is approximately at this point) Figure 5.6A. Therefore, to maximise the number of faults the system can diagnose from memory, and minimise the time spent in the diagnostic process, the value s should be minimised. However, as s approaches 1,

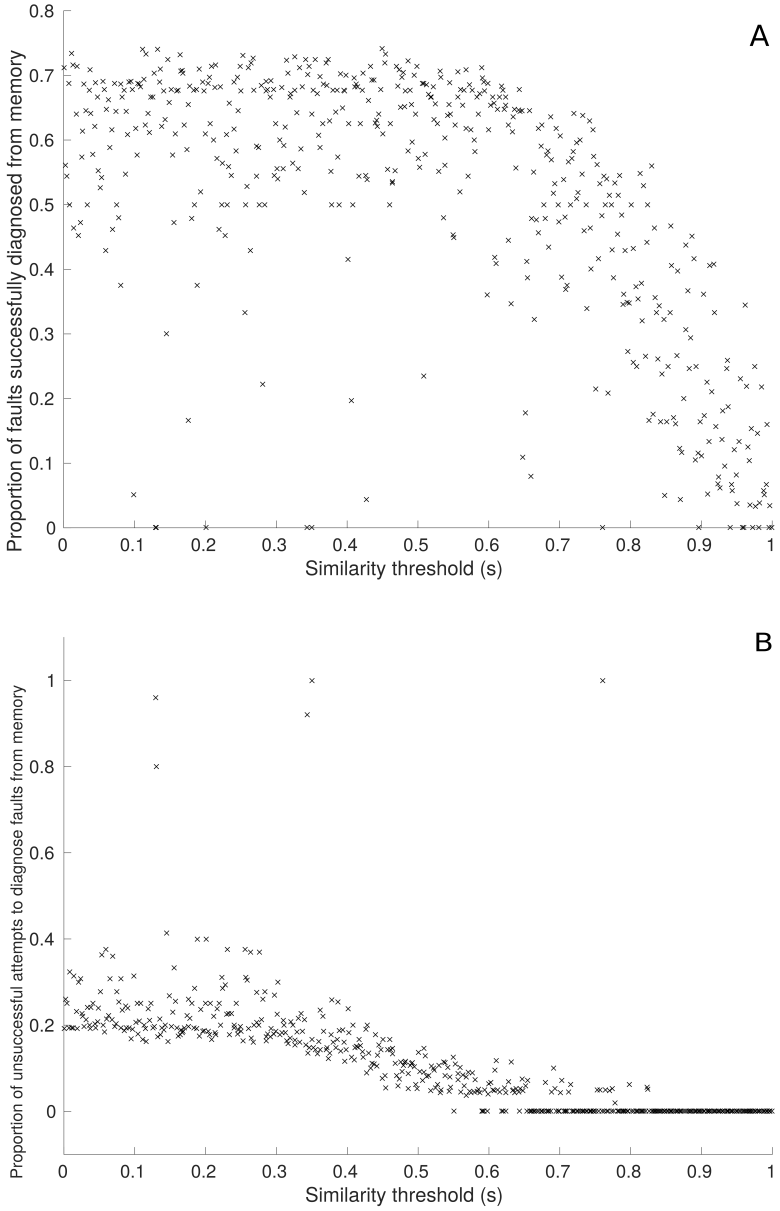

FIGURE 5.6: **A**: The median proportion of faulty robots the system was successfully able to resolve from memory against the value of the similarity threshold, s, and **B**: the median proportion of attempts by the system to diagnose from memory that failed or were not recognised as successful against the value of the simulation threshold, s.

the proportion of unsuccessful attempts to diagnose faults from memory, P_2, decreases in sympathy, averaging 0 for all other parameter combinations at $s > 0.825$ Figure 5.6B. The value s will therefore be greater than 0.825 in order to minimise time wasted on performing incorrect recovery actions.

The goal here is to find the value of s that will provide optimal cost efficiency between running more diagnostic tests and performing wasteful recovery actions. For a real world scenario, this will depend on a number of circumstances such as the time-sensitivity of a particular task, and the number of replacement parts available to the swarm. If it is assumed that there is an ample stock of replacement robot

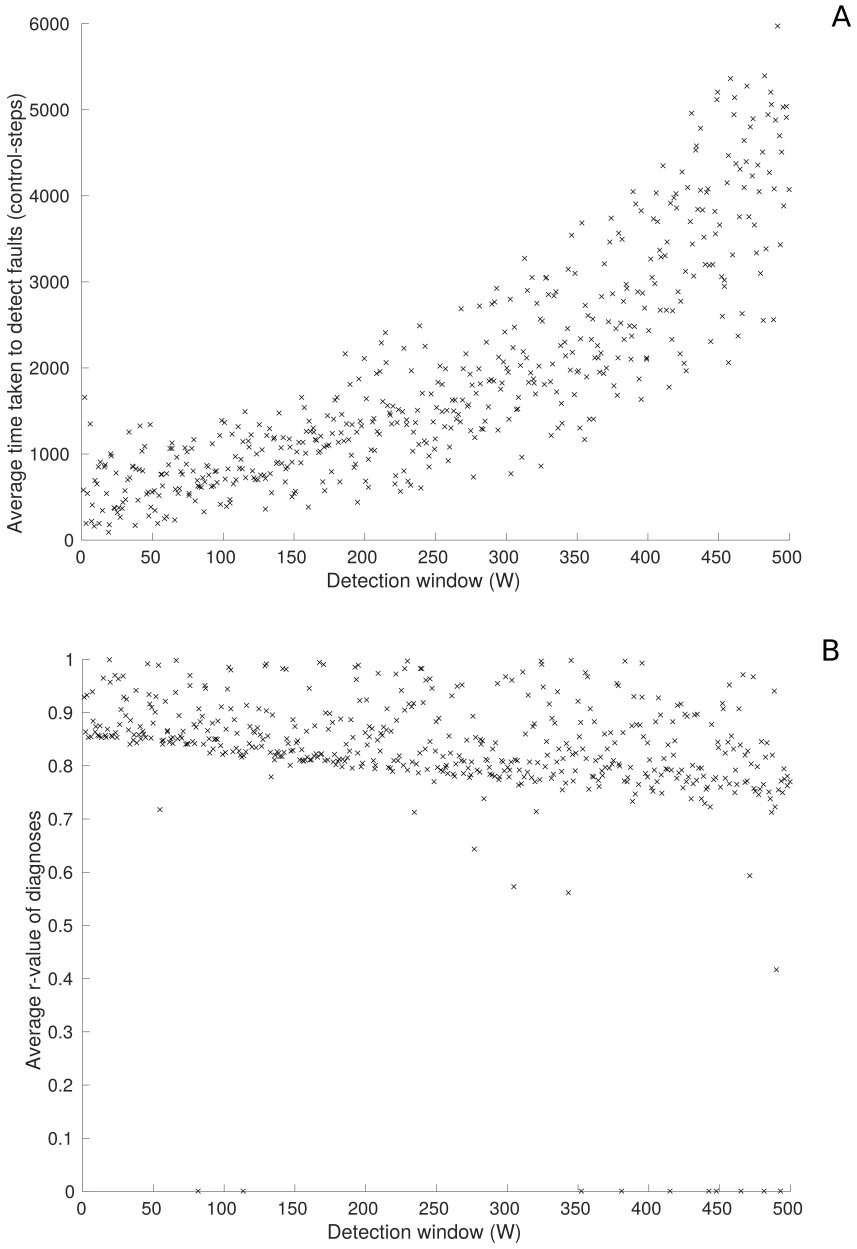

FIGURE 5.7: **A**: The median time taken to detect faults against the value of the detection window, W, and **B**: the median r-value of successful diagnoses against the value of the detection window, W.

parts, this becomes a time-optimisation problem, specifically between the time taken to perform recovery actions and the time taken to perform the diagnostic tests. The time taken to perform diagnostic tests and recovery actions will vary for different swarm robot platforms, and cannot be known in the context of this work until the specific recovery actions described in section 5.2.4 have been investigated for an appropriate robot platform. It is argued that the best that can be done at this stage is to minimise the number of attempts to diagnose a fault from memory that fail to resolve the fault. Figure 5.6B shows that for $s \geq 0.56$ there are parameter combinations that result in an average of $P_3 = 0$, suggesting that for combinations with $s \geq 0.56$ where there are failed attempts, it is because of an alternative parameter

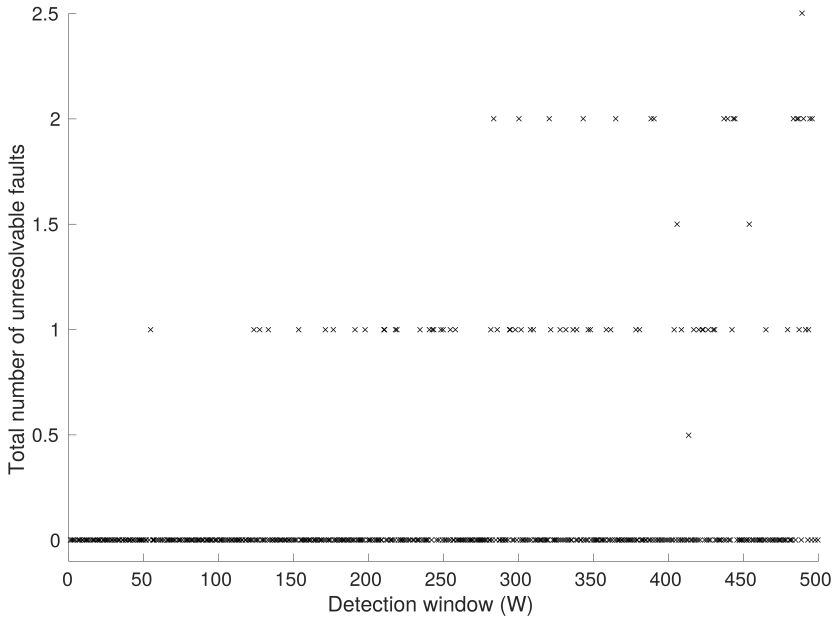

FIGURE 5.8: The median total number of faulty robots the system was unable to resolve over one hour of simulated time against the value of the detection window, W. Non-integer values indicate a median value between two integer values, rather than 0.5 unresolvable robots.

value. The value of s is therefore set to 0.56. Conveniently, this allows the system to theoretically retain a performance score of P_1 that is close to the maximum potential value.

Sensitivity to detection window size The experiments in this section showed that as the detection window size, W, increases, so too does the time taken to detect faults, P_5, (see Figure 5.7A), which consequently decreases the total number of faults detected by the system, P_6. In order to maximise how quickly the diagnostic system is able to resolve the faults, the time spent observing the fault prior to investigating should be minimised.

It was observed that as the detection window size, W, increases, so too does the proportion of unsuccessful attempts to diagnose faults from memory, P_2, and that the proportion of faults successfully diagnosed from memory ,P_1, decreases. This trend arises from the tendency of a larger detection window to reduce the average r-value of diagnoses, P_3 (see Figure 5.7B).

The reduced average r-value of diagnoses, P_3, with increasing values of W is caused by greater opportunity for what could be considered outliers in robot behaviours to be observed in the fault's characterisation. To illustrate; consider two instances of a robot with a partial motor failure. The robot is detected as faulty when it is observed to be moving in an arc whilst it is attempting to move in a straight line. If the faulty robot is observed for a very short period of time, it is likely that this effect will be observed for the entire duration each time, resulting in a very high r-value between the two faults. If the effect is not observed, the fault will not be detected. If it is only observed in one instance, the two faults will then have such low r-values

that they will not be recognised as the same fault and therefore not eligible for diagnosis from memory. If the faulty robots are observed for a long period of time, it is far more likely that the arcing effect will be observed for a majority of the time, but not the entirety – resulting in the lower r-values we observe in Figure 5.7B.

There is also a positive correlation between W and the total number of unresolvable faults, P_4, (see Figure 5.8). Because this effect arises from the environmental circumstances of the faulty robot, this correlation is attributed to the increased time a robot will spend being faulty before being detected as such – which increases the probability that these circumstances arise.

All analysis of W suggests that its value should be minimised. The value of W could theoretically be set to 1 in a noiseless system, and the results in this section reveal no obvious drawback to using the smallest possible value of W. However, this work has not thoroughly examined how the value of W will affect fault detection, particularly false-positive fault detection. Using the method of fault detection described in this chapter, a low W value combined with any system noise or delay whatsoever would certainly result in large numbers of false-positive detections – resulting in time and resources being wasted by the system on performing diagnostic tests and wasteful recovery actions. Although it is impossible to know precisely how the value of W will effect system efficiency until fault detection and diagnosis are properly integrated, it can be said that the optimal value will strike a balance between maximising the r-values between faults and minimising the time taken to detect faults and the frequency of false-positives. In the analysis of the observation period, o, 29 control-steps was ascertained to be the minimum amount of time required for consistently reliable analysis of robot behaviour, and so the value of W is therefore set to 29. Figure 5.8 shows that this value is comfortably in the range that, on average, completely mitigates unresolvable faults in robots.

Sensitivity to proportion of BFV mismatches In the initial experiments conducted for this section, the proportion of mismatched BFVs, ρ, had almost no correlation with any of the performance criteria. Given the counter-intuitive nature of this observation, it was theorised that this may be because the magnitude of contributions of s and W to the performance criteria were obscuring the contributions made by the value of ρ. The effect ρ had on performance criteria whilst s and W were held at their optimised values was then examined. These additional experiments, which produced Figure 5.9 and Figure 5.10, confirmed these suspicions.

The step-like behaviour observed in Figure 5.9A, Figure 5.9B and Figure 5.10 arises from the fact that, as $W = 29$, the value of ρ must increase by approximately 0.035 before it can represent an additional control-step where BFVs were matched or mismatched. As 500 unique parameter values between 0.01 and 1 are used, ρ is incremented at a finer granularity than that required to cause any observable change.

There is a positive correlation between the value of ρ and the average time taken to detect faults, P_5, as one would expect, meaning ρ should be minimised in order to minimise the time faulty robots go undetected. Figure 5.9A shows that this effect is observed over a considerably smaller scale for ρ than W, which would explain why this correlation was not observed when plotting the effects of all s, W, ρ and o on a single graph. This is true of all relations between ρ and system performance.

There was no visible trend observed between the value of ρ and the proportion of faults successfully diagnosed from memory, P_1, however there is a clear negative correlation between increasing values of ρ and the proportion of attempts to diagnose faults from memory that were unsuccessful, P_2 (see Figure 5.9B). Figure 5.9B

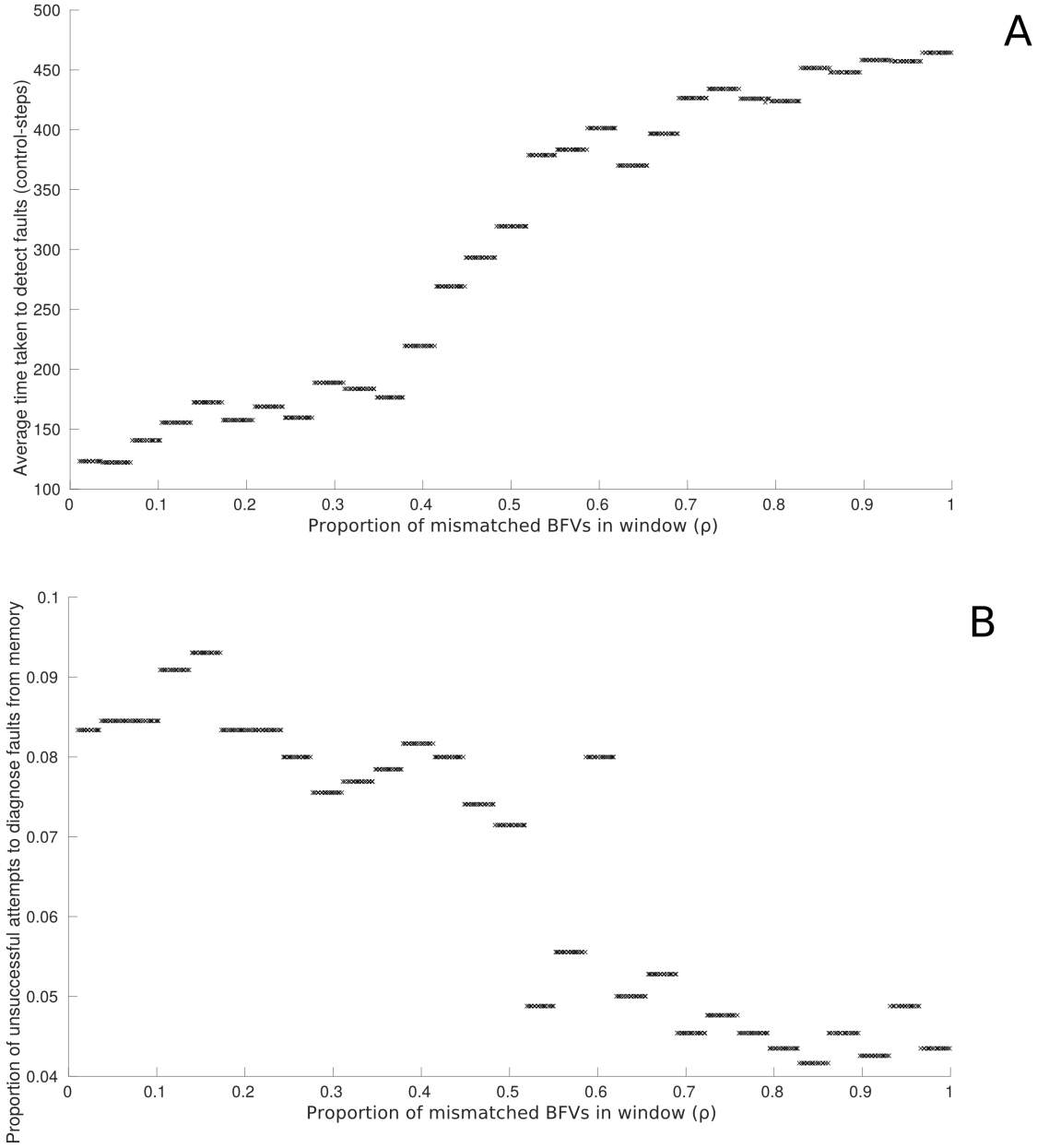

FIGURE 5.9: **A**: The median time taken to detect faulty robots against the proportion of mismatched BFVs in the detection window, ρ, and **B**: the median proportion of attempts by the system to diagnose from memory that failed or were not recognised as successful against the proportion of mismatched BFVs in the detection window, ρ

suggests that the optimal value of ρ lies between 0.83 and 0.86 (the precise value of ρ in between this range makes no difference).

Figure 5.10 shows that the highest average r-values between faults, P_3, are observed as ρ approaches 1 (for $\rho > 0.55$). Interestingly, a quadratic relationship is observed between the value ρ and P_3, with the trough centred in the approximate range $0.5 < \rho < 0.6$ (see Figure 5.10). Obviously, when ρ is a high value, it necessitates that the BFV must comprise of instances where a fault is explicitly demonstrated in robot behaviour. Instances of each fault category are therefore more likely to be similar, resulting in high r-values between faults. However, even when the value ρ is very low,

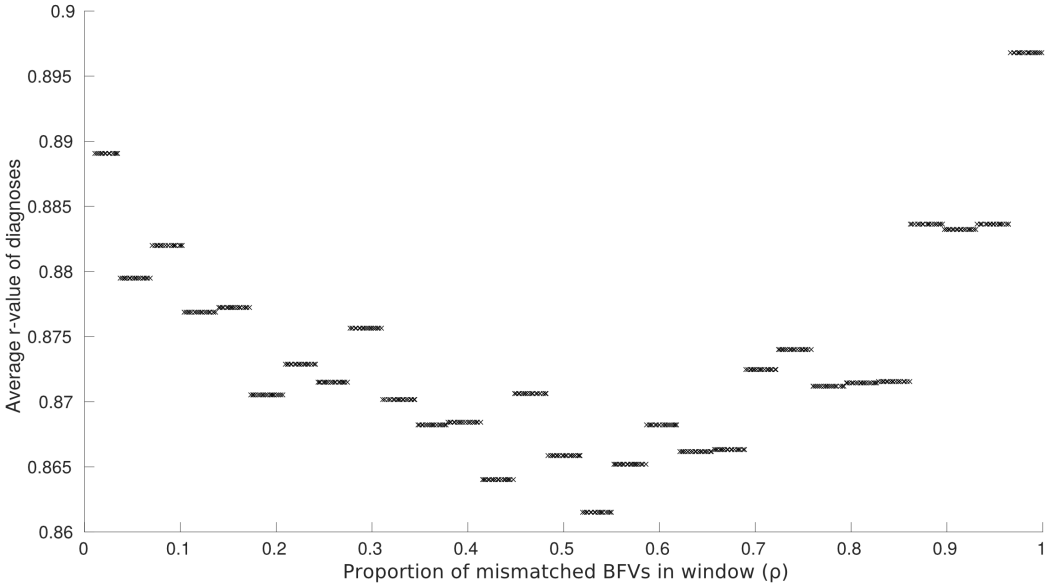

FIGURE 5.10: The median r-value of diagnoses against the proportion of mismatched BFVs in the detection window, ρ

robots still need to make observations of the faulty robot for W control-steps (this is another contributing factor to the comparatively small effect of ρ compared to other parameters). If the fault type heavily effects robot behaviour, this will be represented in the BFV irrespective of the value ρ. If the fault type is more tolerable, there is an increased chance the robot will exhibit normal behaviour for a greater proportion of the period W, which will also result in an increased r-value compared to instances where ρ is in a mid-range. This is because instances where faults are more tolerable will necessarily then be observed as a combination of normal and faulty behaviour with a random distribution. However, this effect is relatively insignificant on overall performance, as demonstrated by the small scale of Figure 5.10. The reason the relationship between P_3 and ρ is stronger for $\rho > 0.55$ is believed to be because, where the value ρ is very low, there is an element of luck present in the BFV representations produced by robots, whereas higher ρ values will produce BFVs that reliably represent the fault present and so produce higher r-values.

The value of ρ was observed to have no effect on the total number of unresolvable faults, P_4.

Based on these observations, ρ is set to 0.86 . This value gives an average detection time of approximately 450 control-steps – an increase of 335 from the theoretic minimum of 115. Given that the previous chapter demonstrates that the average time taken to run diagnostic tests is approximately 180 control-steps (not including the additional period of observation, o), and that these may have to be run multiple times if the proposed system does not diagnose faults correctly the first time, the compromise on detection time necessitated by choosing a high value for ρ is considered to be worthwhile.

5.4 Summary

This chapter set out to modify the diagnostic system proposed in the previous chapter, with the aim of making it closer to a system that could conceivably be deployed

on an actual swarm robotic system. This chapter also set out to demonstrate the proposed diagnostic system's flexibility, scalability and ability to improve the fault tolerance of swarm robotic systems.

This chapter has presented a modified version of the novel approach towards autonomous fault diagnosis in robot swarms which was proposed in the previous chapter. The new diagnostic system with specifically chosen parameter values was demonstrated to be flexible and scalable, being able to diagnose faults from memory in $68\% - 76\%$ of cases on average, depending on the size of the robot swarm. The proposed system also displayed a low rate of misclassification ($3\% - 5\%$ on average) and a general tolerance to imperfect robot state information.

A clear improvement in performance was observed when the proposed diagnosis system was implemented on a swarm performing collective photo-taxis and subjected to partial faults. This improvement supports the case for active fault tolerance proposed by Bjerknes and Winfield (2013) and also represents a viable solution for artificial immunity in robot swarms, as the proposed method for autonomous fault diagnosis should be compatible with existing methods for autonomous fault detection and, eventually, autonomous fault recovery.

This chapter also performed parameter sensitivity analysis on the proposed diagnostic system. It is acknowledged that truly optimal system parameters, and the extent of some system limitations – such as dependence on reliable communication – cannot be known until fully integrated FDDR is implemented in a hardware robot swarm. The work described in the following chapter attempts to take steps towards addressing this problem by implementing the diagnosis system proposed here as faithfully as possible on a hardware robot swarm.

Chapter 6

Fault Diagnosis in Hardware

6.1 Introduction

The previous chapter presented a novel immune-inspired fault diagnosis system for robot swarms in software simulation. The system was demonstrated to be generally scalable for swarms of between 10 and 160 robots, as well as flexible. The benefits of the system were demonstrated by the increase in the overall performance of a simulated swarm where a number of individual robots were injected with various simulated sensor and actuator faults. This was performed in a very similar scenario to that described by Winfield and Nembrini (2006) (albeit a simulated version), which provides much of the motivation for the work conducted for this thesis.

The results detailed in the previous chapter were a promising indicator that the proposed fault diagnosis system (or some version thereof) could be suitable for deployment on actual robot hardware in real world scenarios. Actually achieving this was beyond the scope of this thesis, however the work described in this chapter aims to provide a proof-of-concept that the proposed diagnosis system could be effectively implemented in robot hardware.

To demonstrate an improvement in robot performance as was accomplished in the previous chapter, would require experimental equipment and infrastructure that was not readily available during the completion of this thesis. Instead, this chapter proposes that if similar trends as were observed in software simulation can also be observed in a comparable hardware system, then it suggests that the desirable features of the proposed diagnostic system, such as increasing overall swarm performance, may also hold true.

The swarm system used for the experiments described in this chapter was designed to be as faithful as possible to that used in software simulation in previous chapters, but it was not possible to reproduce all elements of the simulated swarm, such as each robot being equipped with a range and bearing (RAB) sensor, which required workarounds (all of which are detailed in the following chapter). In all cases where the hardware swarm necessarily deviates from the software swarm, the deviation was implemented to be as conceptually similar to the software swarm as was possible.

6.2 Experimental Methods

The experiments for this chapter were conducted using the psi-swarm robot (Hilder et al., 2016), a two-wheeled, 10.8cm diameter swarm robot with on-board proximity sensors. The lack of a RAB sensor on the psi-swarm meant that the proposed diagnosis system could not be faithfully reproduced in hardware. To work around this, the ARDebug tool (Millard et al., 2018) was used, which tracks an arUco tag mounted

on top of each psi-swarm, allowing distances between and linear and angular velocities for each robot to be calculated. Whilst not traditionally swarm-like, the use of ARDebug to provide robots with the information they require on their neighbours does not interfere with the performance of the proposed diagnosis system. Rather, it allows this research to progress until such a time as there are readily available robot platforms that can acquire this information reliably and autonomously.

FIGURE 6.1: The Psi-Swarm robot used for the experiments in this chapter (Hilder et al., 2016).

A swarm of 5 robots performed obstacle avoidance behaviour in a 2.2 square metre enclosed arena (see Figure 6.2). These robots were subjected to a subset of the faults examined in previous chapters, namely; Power failure (H_2), Complete Sensor failure (H_3) and Complete Motor failure (H_4). Each of these faults are implemented in the same way as in previous chapters, but are described again here for convenience:

- **Power failure (H_2)**: The robot completely stops moving and is unresponsive to its surroundings.
- **Complete Sensor failure (H_3)**: The robot is unable to detect the presence of neighbours or obstacles.
- **Complete Motor failure (H_4)**: One of the robot's motors stops and becomes unresponsive to its controller.

It was impractical to allow experiments to run for one hour of real time due to to power constraints. The Psi-Swarm robot is stated to have a battery life of approximately 30 minutes where all LEDs are on, IR sensors active at 10Hz, Bluetooth active and motors at 100% on a smooth surface. This is a generally accurate reflection of the Psi-Swarm behaviour used in the experiments described in this chapter. However, during the work conducted for this chapter, it was also found that the Psi-Swarms became increasingly unreliable, particularly in their ability to send and receive commands over Bluetooth, as their power was depleted (typically after approximately

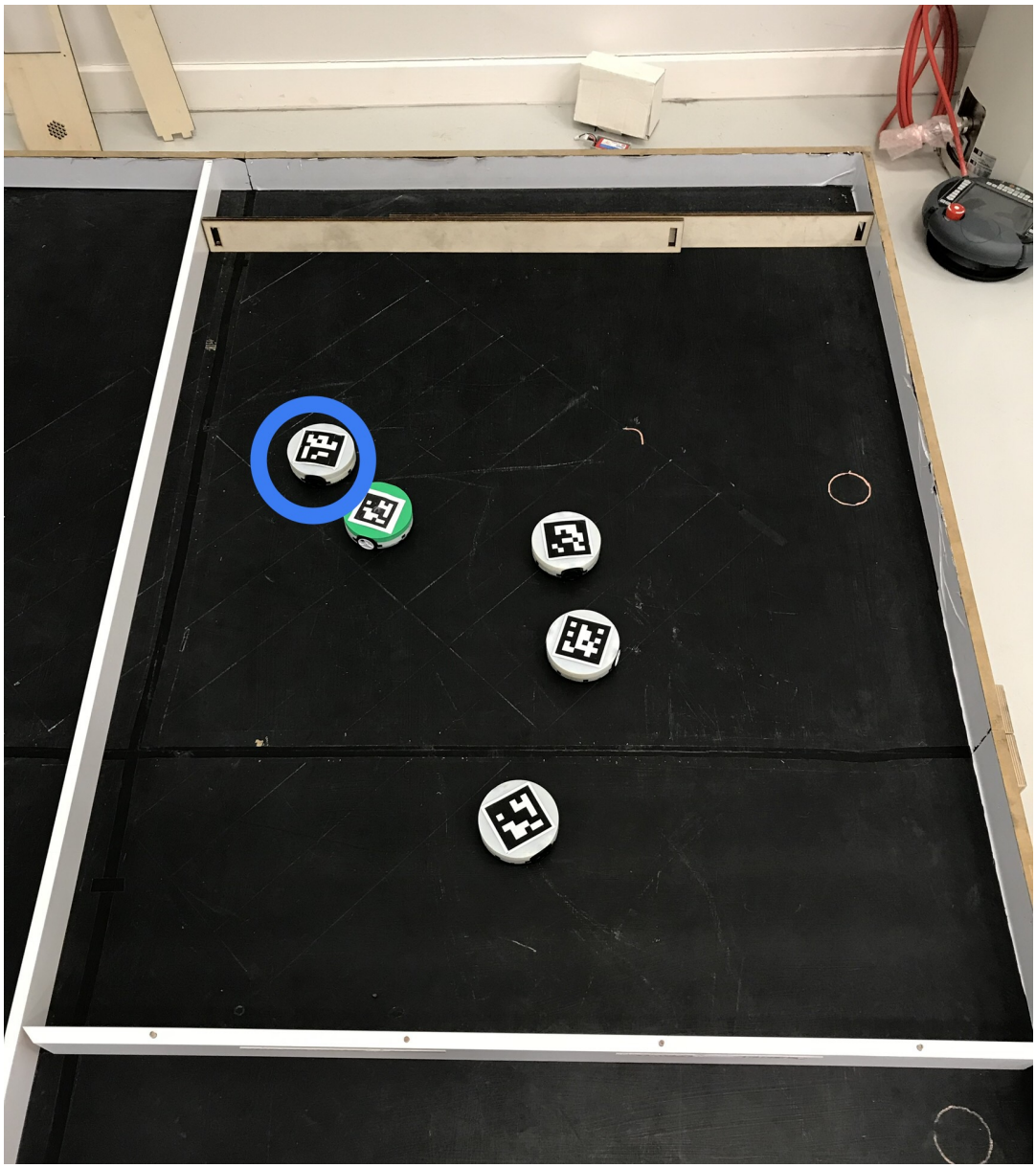

FIGURE 6.2: Five Psi-swarms performing obstacle avoidance in an arena. The robot with a green hat suffers from power failure (H_1), whilst its assessing robot in this case is circled in blue.

15-20 minutes of use). In the simulated experiments described in the previous chapter, an average of 30 faults were encountered by a swarm of 10 robots during one hour of simulated time. Therefore each hardware experiment was set to last for as long as it took the system to resolve an equivalent amount of faults. As the experiments in this chapter consider half as many robots (5) that can be subjected to half as many fault types (H_{2-4}), the first hardware experiments ran for as long as it took the system to encounter and resolve 15 faults. In these experiments, each robot had a 1% probability of being injected with a random fault. As with the previous chapters of this thesis, instances of false-positive fault detection are not examined (see section 4.3.4 for details). As these experiments are using 5 robots, a maximum of 2 robots can be faulty at one time so that there are never more faulty robots than the

remaining swarm can assess at a given time.

Robot behaviour is encoded as BFVs, where the binary vector $BFV(t) = [F_1(t), F_2(t), F_3(t), F_4(t), F_5(t)]$, is defined as the following set of features:

- $F_1(t) = 1$ if $N_R(t) > 0$, otherwise $F_1(t) = 0$
  where $N_R(t)$ is the total number of neighbours in sensing range (approximately $1m$) of the robot at time t.
- $F_2(t) = 1$ if $N_C(t) > 0$, otherwise $F_2(t) = 0$
  where $N_C(t)$ is the total number of neighbours at a distance less than the close proximity threshold, C, (approximately $0.3m$) from the robot at time t.
- $F_3(t) = 1$ if $|v(t)| > 0.8|v_{max}|$, otherwise $F_3(t) = 0$
  where $|v(t)|$ is the magnitude of linear velocity at time t and $|v_{max}|$ is the maximum linear velocity a robot can achieve (approximately $5cms^{-1}$ in this work).
- $F_4(t) = 1$ if $|v(t)| > 0.2|v_{max}|$, otherwise $F_4(t) = 0$
- $F_5(t) = 1$ if $|\theta(t)| > 0.4|\theta_{max}|$, otherwise $F_5(t) = 0$
  where $|\theta(t)|$ is the magnitude of angular velocity at time t and $|\theta_{max}|$ is the maximum angular velocity a robot can achieve (approximately $24^\circ s^{-1}$ in this work).

As these experiments do not consider software hang faults, the watchdog timer inspired feature used in the previous chapter, $F_6(t)$, which appeared in the Chapters 4 and 5, is omitted from the BFV here. It was shown in Chapter 3 that the BFV given above was sufficient for classifying H_{2-4} in a majority of cases. The parameters used in feature calculations for hardware experiments vary slightly:

- The range of the virtual RAB sensor is $60cm$
- Close proximity threshold $C = 3cm$
- Threshold values to set F_{3-5} high are $7.5cms^{-1}$, $1.5cms^{-1}$ and $75^\circ s^{-1}$, respectively.

$F^{e,p}_{1-2}$ are conceptually estimated for each robot using the ARDebug virtual RAB, where faults will only affect the proprioceptively sensed features. F_{3-5} are estimated proprioceptively by the robot monitoring its own controller output, and exteroceptively by the rate of change in readings from the ARDebug virtual RAB. Features are updated at an approximate frequency of 1.5Hz. Faults are detected according to Equation 6.1.

Faults are detected when discrepancy is observed between proprioceptively estimated features, F^p_{1-5}, and exteroceptively estimated features, F^e_{1-5}, according to Equation 6.1. This is the same method of fault detection as described in the previous chapter, but is given again here for convenience.

$$\text{fault detected at time } t = true \text{ if } \sum_{t=T-W}^{T} d(t) \geq \rho W \text{ for } T > W \tag{6.1}$$

Where T indicates the time elapsed since a robot started being observed by a neighbour. The binary value d is set according to Equation 6.2, where $d = 0$ if there is agreement between exteroceptively and proprioceptively estimated robot features. A robot will make W observations of its neighbour, once per control-step, at discrete times, t, before a fault can be detected, where t indicates the time elapsed

since the start of an experiment. The value of ρ indicates what proportion of the W observations must prompt the value of d to be equal to 1 in order for a fault to be detected.

$$d(t) = \begin{cases} 0 \text{ if } F_i^p(t) = F_i^e(t) \text{ for } i = 1, .., 6 \\ 1 \text{ otherwise} \end{cases} \tag{6.2}$$

When a fault is detected in a robot by another member of the swarm, the set of W proprioceptively and exteroceptively estimated BFVs which prompted the fault to be detected are used as the BFV signature for that particular fault. The assessing robot – the robot that first detected the fault – then initiates a diagnostic test routine. The remainder of the swarm will continue to perform their normal behaviours, as per the simulated experiments in previous chapters. Over the course of setting up and conducting these experiments, writing a routine for the assessing robot to follow the faulty robot, as in the simulated experiments described in previous chapters, was found to be infeasible. This was because the controller for that routine would have had to run on the ARDebug system and communicate commands to the robots over Bluetooth, the latency on the robots receiving and processing these commands was unacceptable for such a task. Instead, the assessing robot stops while the faulty robot performs the diagnostic tests. Each robot still only has access to the same information as in the previously detailed simulated experiments – for which there was a conceptually fully distributed system. Again, this implementation is intended to allow this work to progress until appropriate robotic platforms become available.

The selection of diagnostic tests used for these experiments are arranged as follows, where failing the test indicates the presence of its associated fault:

- **Power failure (T_2)**: Almost identical to T_1. H_1 and H_2 are distinguished in an unreachable robot by whether it has stopped moving.
- **Complete Sensor failure (T_3)**: The assessing robot, whilst in sensing range, observes the faulty robot's feature F_1^p, which should return 1.
- **Complete Motor failure (T_5)**: The faulty robot is instructed to move forward in a straight line whilst its feature F_5^e, which should return 0 is monitored.

These diagnostic tests are implemented in the same way as in the simulated experiments described in the previous chapter, with the exception of testing for complete motor failure (H_4). Instead of using T_4 for this, T_5 is used instead. The psi-swarms were able to perform T_5 in a much more fluid manner, primarily because of the latency incurred by using bluetooth communication to initiate each part of T_4. As partial motor failure, H_5, was not examined in these experiments, the use of T_5 to test for H_4 did not cause any problems.

When a fault is diagnosed, the system then carries out a corresponding recovery action. These are the same as those used in the previous chapter, but are given again here for convenience:

- **Power Cycle (R_1)**: This recovery action resolves power failure (H_2) faults.
- **Sensor Replacement (R_2)**: Replacement of robot sensors resolves complete sensor failures (H_3).
- **Motor Replacement (R_3)**: Replacement of robot motors resolves complete motor failures (H_4).

Carrying out these recovery actions autonomously with the marXbot platform is not realistic at the time of writing, and all recovery actions are simulated for this work.

The assessing robot then makes another W observations of the faulty robot. If the faulty robot's exteroceptively and proprioceptively estimated BFVs match during this period, the fault is considered to be resolved. This period was described as a separate parameter, o, in the previous chapter. However, as W and o both pertain to the period over which faults are detected, they are set to be equal to one another.

If a robot's fault persists after having attempted diagnostic tests $T_{2,3,5}$, the faulty robot is shut down, at which point it becomes an inanimate object in the environment.

When a fault is successfully resolved the BFV signature for that fault and the respective diagnosis are stored in the assessing robot's memory, which is then shared with other members of the swarm via an *ad-hoc* network, which is facilitated by the ARDebug system i.e. each robot reports its state to the ARDebug system, which also monitors the robot's state independently, and this information is then shared with other robots in the swarm via bluetooth communication. When subsequent faults occur, the BFV representations of those faults are checked for similarity against the BFV representation of previous faults by finding the Pearson correlation coefficient (Benesty et al., 2009) between their BFV signatures, given again here for convenience (Equation 6.3):

$$r = \frac{\sum_m \sum_n (F_{mn} - \overline{F})(F_{0_{mn}} - \overline{F_0})}{\sqrt{(\sum_m \sum_n F_{mn} - \overline{F})^2 (\sum_m \sum_n F_{0_{mn}} - \overline{F_0})^2}} \tag{6.3}$$

Where r is the correlation coefficient between a current and previous fault signature, F and F_0, respectively, where F and F_0 are two dimensional binary data sets. $\overline{F}$ and $\overline{F_0}$ are the mean values of sets F and F_0, respectively, and m and n indicate positional index within the 2D binary data set.

When attempting to diagnose a new fault from memory, the previously stored fault which produces the greatest r-value (Equation 6.3) is considered the most likely fault type and recovery is initiated in sympathy with that fault. To be eligible for this process $|r| > s$ is required, where s represents the threshold above which two BFV-signatures are considered similar.

The parameter sensitivity analysis performed in the previous chapter suggests that optimum parameter values for experiments in software simulation were $s = 0.56, W = 29$ and $\rho = 0.88$. These were therefore used as the starting point for the experiments described in this chapter.

The proposed system's performance during these experiments was assessed on the following criteria:

- P_1: The proportion of faults the system is able to diagnose from memory
- P_2: The proportion of attempts to diagnose a fault from memory that fail or are unrecognised by the system
- P_3: The average r-value between faults, using Equation 6.3, when diagnosing from memory
- P_4: The total number of faulty robots the system is unable to resolve and are shut down in an hour of simulated time

To summarise:

- 5 Psi-Swarm robots perform a random walk/collision avoidance behaviour in an otherwise empty enclosed arena
- Once the experiment has begins, each robot has a 1% probability of being injected with a fault. When a fault is injected, the chosen robot will receive a bluetooth command from the ARDebug system that will alter the robot's current behavioural program to reflect that fault type.
- The ARDebug system estimates co-ordinate information, and therefore BFV states, for each robot at each control-step. These BFV states are then used to detect faults according to Equation 6.1. This is implemented in such a way as to simulate decentralised fault detection by other robots i.e. a fault will only be detected if a neighbouring robot has been within sensing range for the duration of the process, and all robots in sensing range of a faulty robot are able to detect the fault.
- A fault is detected when Equation 6.1 returns true. If there were multiple robots in sensing range of the faulty robot for the duration of the detection process, there will be multiple detections of the same fault at the same control-step. In this case, when assigning a robot to assess the fault, the first robot in the list of paired devices will be selected.
- If the fault is similar to a previous fault in the assessing robot's memory, it will use the remembered diagnosis. Otherwise, it will initiate diagnostic tests $T_{2,3,5}$. The commands in these tests are communicated via bluetooth from the ARDebug system. If one of the tests is failed, the system will make the corresponding diagnosis. If no tests are failed, the detected fault is treated as a false-positive and normal behaviour resumes for faulty and assessing robots.
- Once a robot is diagnosed, it will resume normal behaviour under observation. If the ARDebug system estimates abnormal BFVs form the robot under observation, the diagnosis is treated as incorrect. If the diagnosis was made from memory, the system will now initiate diagnostic tests $T_{2,3,5}$. If diagnostic tests $T_{2,3,5}$ have already been performed, the faulty robot is shut down via bluetooth command and becomes an obstacle in the arena.
- If the diagnosed robot is able to demonstrate normal BFVs, as estimated by the ARDebug system, the presumed-to-be-correct diagnosis is stored in memory, as well as the faulty robots BFV sequence that initially caused Equation 6.1 to return true. There is then a simulated process of sharing this information via local communication. Initially, only the assessing robot is able to access the information. As that robot comes within range of other robots in the swarm, they too will be allowed to access the information, who in turn will allow their neighbours to access it and so on.
- Throughout the experiment, performance criteria P_{1-3} are recorded in real time by the system.
- The experiment ends once 15 faults have been resolved by the system in one way or another.
- 10 replicates are performed for each experiment.

	P_1	P_2	P_3	P_4
Software	0.68	0.04	0.87	0
Hardware	0.27	0.41	0.75	1.5

TABLE 6.1: Comparison of the performance of the proposed diagnosis system in software simulation and hardware experiments, where $s = 0.56$, $W = 29$ and $\rho = 0.88$. Data for software simulation experiments is taken from O'Keeffe *et al.* (2018).

6.3 Results and Discussion

The performance of the proposed diagnosis system implemented in robot hardware, where $s = 0.56, W = 29$ and $\rho = 0.88$, is displayed in Table 6.1, as well as the results obtained in software simulation in the previous chapter for comparison.

Table 6.1 shows that, using the parameters that gave optimal performance in software simulation and applying them to hardware experiments, performance suffers considerably on all metrics. The primary cause of this is that setting $W = 29$ does not allow enough time for the faulty robot to react, this is especially true for T_5. To illustrate, the assessing robot will instruct the faulty robot to demonstrate its motor function. The faulty robot receives this request over Bluetooth and returns a message to the assessing robot, instructing it to begin its observations. Between the assessing robot making its first and twenty-ninth observation, the faulty robot is often unable to accelerate quickly enough and maintain velocity for long enough to satisfy the assessing robot that the fault is resolved.

A general shortcoming of this system in hardware is that there is latency between proprioceptively and exteroceptively sensed features updating. When a robot turns to avoid an obstacle or neighbour, it is able to communicate this change in controller state almost immediately. However, it takes longer for the ARDebug system to observe and report this change in linear and angular velocity. This is often to the extent that, by the time the exteroceptively sensed features reflect the robot turning, the turn has already occurred and the robot is back to moving in a straight line, which is reflected by the proprioceptive features in close to real-time. This means that for short turning periods, which account for the vast majority of occasions in which a robot turns during these experiments, there is discrepancy between proprioceptively and exteroceptively sensed features. As instances of false-positive fault detection are beyond the scope of this thesis, this does not pose a significant problem to detecting faults or their BFV signatures. However, when observing a faulty robot post-diagnosis, the greater the proportion of time it spends exhibiting avoidance behaviour, the greater the likelihood that it will be assessed as having its fault persisting, even if the fault has been resolved. The most immediate way around this, given that avoidance is commonly exhibited but generally only for short periods of time, is to observe how increasing the value of W affects the proportion of faults diagnosable from memory. The same experiments were repeated for integer multiples of W. The results are shown in Table 6.2.

The trends observed in Table 6.2 are very similar to observed in simulation. A step-like relationship between W and $P_{1,2}$ was observed in the previous chapter, where values below a threshold ($W < 5$ in simulation) severely reduced performance. Above this threshold, however, there was no observable correlation as W increased. Table 6.2 demonstrates that increasing the value of W from 29 to 58 improves overall performance on metrics P_{1-4}, after which point there is relatively little

W	P_1	P_2	P_3	P_4
29	0.27	0.41	0.75	1.5
58	0.5	0.1	0.78	0
87	0.53	0.15	0.79	0.5

TABLE 6.2: Comparison of system performance, where $s = 0.56$ and $\rho = 0.88$, for varying values of W.

Num. Faults	P_1	P_2	P_3	P_4
15	0.5	0.1	0	0.78
25	0.6	0.14	1	0.75

TABLE 6.3: Comparison of system performance where the system encounters 15 and 25 faults in an experiment ($s = 0.56, W = 58, \rho = 0.88$).

change in performance.

It was also observed in the previous chapter that for low values of W, P_4 is initially at optimum $P_4 = 0$. However, increasing W eventually causes P_4 to increase proportionately (Figure 5.8), a trend that can be seen in the data in Table 6.2 between $W = 58$ and $W = 87$. The fact that P_4 is at its highest value for low values of W ($W = 29$) in Table 6.2 but not in Figure 5.8, arises because the previous chapter separated the detection window, W, and the observation period, o, whereas in this chapter the two variables are merged i.e. the window over which faults are detected is equal to the period over which a diagnosis is assessed to be correct. Figure 5.5 demonstrated that using a short period to assess the validity of a diagnosis also results in a high number of unresolvable faults, P_4, during an experiment.

Table 6.2 shows that the best overall performance was obtained for $W = 58$. This is the value therefore used for all remaining experiments.

In chapter 4 it was observed that, the longer an experiment went on, the more frequently the system was able to diagnose faults from memory. This effect can be seen in Figure 4.3, and is a result of the system establishing a more comprehensive repertoire of BFV signatures for different types of faults such that it eventually becomes very probable when a fault is encountered that the system will have already diagnosed and resolved the fault under similar circumstances. Table 6.3 compares performance where the system encounters 15 faults and 25 faults during an experiment.

The results obtained in hardware obey the trend observed in Figure 4.3: As the system is subjected to more faults, it becomes increasingly likely that future instances will be diagnosable from memory.

The last trend investigated in these hardware experiments is how altering the value of s affects performance, this is displayed in Table 6.4.

The trends between s and system performance observed in Table 6.4 are very similar to those we observed in software simulations. In the previous chapter it was observed that P_1 is largely unresponsive to s until a critical point, presumed to be the average r-value between faults, after which P_1 begins to decline with increasing s. The data in Table 6.4 generally fits this trend, but suggests that the critical point for s is lower in hardware, which is supported by the lower value of P_4 observed.

It was also observed in the software simulation experiments described in the previous chapter that P_2 decreases for increasing values of s, which can also be observed

s	P_1	P_2	P_3	P_4
0.28	0.6	0.34	1	0.72
0.42	0.63	0.19	0	0.72
0.56	0.5	0.1	0	0.78

TABLE 6.4: Comparison of system performance, where $W = 0.58$ and $\rho = 0.88$, for varying values of s.

in Table 6.4.

It has been argued throughout this thesis that P_1 is the most indicative measure of our system's performance. The hardware experiments described in this chapter were able to achieve $P_1 = 0.63$, which is proposed to be comparable to the value obtained by our system in software simulation, $P_1 = 0.68$. A reduction in performance is to be expected when implementing a system in hardware (Jakobi, Husbands, and Harvey, 1995). The fact that the proposed system in hardware was able to achieve 93% performance on what is considered the most fundamental performance metric, combined with the fact that the same general trends are observed in hardware experiments as in software, is evidence of the proposed system's feasibility for deployment in real-world scenarios, and that the results obtained and trends observed in experiments conducted in software simulation should generally translate proportionately into real-world systems.

6.4 Summary

The work described in this chapter has demonstrated a novel approach to fault diagnosis in a hardware robot swarm. The system presented here performed comparably in hardware and software simulation. Although performance was reduced on all metrics in hardware, the proposed system was able to diagnose faults from memory in the majority of cases ($P_1 = 0.63$) when set with appropriate parameter values, and was also able to maintain a comparatively low rate of misclassification ($P_2 = 0.19$ for the same parameter values, although by adjusting the parameter values the system was able to get this as low as $P_2 = 0.1$). The proposed system also maintained a very low rate of instances where it was unable to diagnose a fault (P_3).

Although the results discussed in this chapter demonstrate the proposed systems viability as a solution to improving fault tolerance in hardware SRS, they also highlight some of the present limitations. Most prominent of these is the proposed system's dependence on reliable fault detection. That low values of W result in such degradation in performance is indicative of the fact that, without a reliable means of assessing the presence or absence of faults and failures autonomously, the proposed system cannot be expected to demonstrate learning and memory functionality as desired. This is largely unsurprising, and previous chapters highlight that fault detection is necessary for effective fault diagnosis. However, the results presented in this chapter suggest that work on this particular method of fault diagnosis is now mature enough that it is no longer practical to separate the process of diagnosis from detection, as it has been demonstrated in this chapter and in previous chapters that the two processes are intrinsically linked.

Chapter 7

Conclusion

7.1 Introduction

The concluding chapter of this thesis summarises the work detailed in the previous chapters, as well as outlining the contributions made by this thesis. This chapter then discusses the limitations of the work described in the previous chapters of this thesis, before suggesting what steps might be taken to carry this work forward. This chapter closes with a comparison of the conclusions of this thesis with the initial research hypothesis described in the first chapter.

7.2 Summary and Contributions

Chapter 2

Chapter 2 highlights the motivation for providing swarm robotic systems with active fault tolerance capabilities. It then goes on to discuss a variety of the approaches towards active fault tolerance in multi-robot systems and swarm robotic systems devised by other researchers. Chapter 2 also details a number of immune-inspired approaches to fault tolerance, and broadly describes the natural biology those approaches draw from. Chapter 2 demonstrates the body of work towards active fault-tolerance for robot swarms to be relatively small, and in doing so highlights the large space for novel approaches towards active fault tolerance in swarm robotic systems. Chapter also highlights the necessity of a fault diagnosis mechanism for enabling robot swarms to retain long-term autonomy. The absence of previous work towards this particular problem justifies the work undertaken in this thesis.

Contributions

1. An up to date review of existing work towards fault tolerant swarm robotic systems.
2. An argument that fault diagnosis is a necessary component of active fault tolerance for swarm robotic systems.
3. The identification of ways in which previous work on fault detection could be carried forward for a novel approach to fault diagnosis.

Chapter 3

Chapter 3 carries forward the Behavioural Feature Vector (BFV) approach, demonstrated to be effective in fault detection by Tarapore, Christensen and Timmis (2017), and applies it to fault classification. By encoding robot behavioural states as BFVs,

Chapter 3 demonstrates that a number of different electromechanical fault types can be classified offline by training a decision tree on the BFVs produced by robots with a variety of different fault types operating within a swarm. The results obtained in Chapter 3 showed promise for a BFV-based approach to fault diagnosis for robot swarms.

Contributions

1. A demonstration that BFVs can be used effectively to classify a variety of electromechanical fault types in individual robots within a simulated swarm.

Chapter 4

Chapter 4 is informed by the results obtained in Chapter 3, and uses a BFV-based approach to devise and present an immune-inspired novel, adaptive and distributed approach to fault diagnosis in simulated robot swarms. Chapter 4 demonstrates the proposed fault diagnosis system to diagnose fault types appropriately in the majority of the cases tested through a process of learning and memorising a given fault's characteristics online and in real time. The proposed diagnostic system method was also demonstrated to give a very low rate of misclassification.

Contributions

1. The proposal of a novel method for fault diagnosis in swarm robotic systems.
2. A demonstration that the proposed method of fault diagnosis can diagnose a variety of fault types appropriately in the majority of cases tested, and that the proposed system mimics some of the desirable characteristics of the natural immune system.

Chapter 5

Chapter 5 builds on the work described in Chapter 4, and makes adjustments to the proposed fault diagnosis system to bring it closer to a system which could conceivably be deployed in a real world scenario. The fault diagnosis system presented in this chapter is novel, and is demonstrated to be flexible and scalable when implemented on a simulated autonomous robot swarm. The proposed diagnostic system is demonstrated to be able to appropriately diagnose a variety of electromechanical faults in the majority of the cases tested, as well as maintaining a low rate of misclassification. The system is also subjected to parameter and noise sensitivity analysis.

The proposed diagnostic system is also tested on a simulated swarm performing collective photo-taxis and subjected to partial faults, in a scenario very similar to that used by Winfield and Nembrini (2006) which significantly motivates this thesis. A clear improvement in performance was observed when the proposed diagnosis system was implemented, compared with a simulated swarm without active fault tolerance. This improvement supports the case for active fault tolerance proposed by Bjerknes and Winfield (2013) and also indicates that the proposed diagnostic system is a viable solution for achieving long term autonomy in robot swarms.

Contributions

1. The proposal of a novel method for fault diagnosis in swarm robotic systems.

2. A demonstration that the proposed method of fault diagnosis can diagnose a variety of fault types appropriately in the majority of cases tested, and that the proposed system mimics some of the desirable characteristics of the natural immune system.

3. A demonstration that the proposed diagnostic system is flexible, scalable and robust to varying degrees of system noise under the conditions tested.

4. A demonstration that the proposed diagnostic system improves the overall performance of a simulated robot swarm by providing it with active fault tolerance under similar conditions to those used by Winfield and Nembrini (2006).

Chapter 6

Chapter 6 attempts to implement Chapter 5's novel fault diagnosis approach in hardware as faithfully as possible. Chapter 6 demonstrates a novel approach to fault diagnosis in a hardware robot swarm. The system presented in Chapter 6 performs comparably with that presented in software simulation in Chapter 5. Although performance was reduced in hardware, the proposed system was able to diagnose faults appropriately in the majority of cases tested, and was also able to maintain a comparatively low rate of misclassification.

Chapter 6 revealed that the proposed diagnostic system, when implemented in hardware, displayed the same general trends as the software simulated version when the same parameter values were varied. This suggests that the vast improvement to swarm performance observed in Chapter 5 may also hold true for a hardware system. If this is the case, then the proposed diagnostic system described in this thesis takes a direct step towards addressing the problems associated with fault tolerant robot swarms that are outlined by Bjerknes and Winfield (2013).

Contributions

1. The implementation and demonstration of a modified but conceptually similar version of the proposed diagnostic system in actual robot hardware.

2. A demonstration that the proposed diagnostic system in hardware generally obeys the same trends as in software simulation when system parameters are varied.

7.3 Limitations

This section highlights the limitations of the work presented in this thesis that were identified during its undertaking.

7.3.1 Communication

Each variation of the fault diagnosis system, described in Chapters 4, 5 and 6, is heavily dependent on reliable communication between robots. Each robot needs to be able to communicate its own BFV states with neighbouring robots, as well as being able to share its memory of previous fault signatures. If this data is lost, delayed, sent or received in the wrong order or otherwise not as it should be, this will affect system performance. When robots communicate their BFV states it must be done in real time with as little delay as is possible. If this is hindered, two possible outcomes

are predicted; a robot will be unable to make a comparison of proprioceptively and exteroceptively estimated BFVs at that point in time, in which case any active diagnostic processes will have to pause until up-to-date BFV information becomes available again; or, robots will have to use the most recent BFV data irrespective of whether or not it accurately describes a robot's present state – which is tantamount to system noise in this case, the effects of which have already been shown to be detrimental to overall performance. Similarly, if memory data is lost or corrupted, the system will either have non-representative fault characterisations – which will lead to an increase in failed diagnosis attempts – or it will simply have no representation at all in these scenarios, in which case it will require the time-consuming diagnostic tests to be run more frequently.

7.3.2 Fault Types Tested

The experiments conducted for this thesis only consider individual robot faults that can be considered orthogonal to each other. In real world scenarios it is possible that multiple faults may occur at once or in sequence, or that the effects of one type of fault may be indistinguishable from the effects of a different type of fault by their BFV signatures – this was observed in Chapter 3, however it might not always be possible to get around these problems with the addition of new features. Additional problems may occur if faults do not explicitly manifest in robot software/hardware, but in the interaction space between robots. Although it is beyond the scope of this thesis, as AIS in robots develop, it is probable that Trusted Autonomy (Abbass, Leu, and Merrick, 2016) will become an increasingly prevalent consideration for fault tolerant systems. The proposed fault diagnosis system described in the previous chapters of this thesis is not equipped to examine or address this problem space, and to modify it do so would require an additional dimension of fault detection, diagnosis and recovery to be integrated.

7.3.3 Fault Detection Mechanism

In the wider context of achieving fault tolerant swarms, another limitation of the proposed fault diagnosis system, and perhaps of the process of fault diagnosis in general, is the dependence on reliable fault detection. This is indicated by the results of the noise sensitivity analysis, and in the parameter sensitivity analysis in Chapter 5. Without a reliable means of autonomous fault detection, the proposed fault diagnosis system in software simulation is impaired in its ability to demonstrate learning and memory functionality as is desirable. This was even more prominent when the fault diagnosis system was moved into hardware in Chapter 6, where a large degradation in performance was observed when using low values of W, which determines the window of time a fault is detected over. To reiterate what was proposed in the summary of Chapter 6, a general conclusion of this thesis is that the process of fault diagnosis is not practically separable from fault detection.

7.3.4 Available Hardware & Fault Recovery

Fault recovery at a finer scale than the replacement of an entire robot is a problem that is beyond the capabilities of any swarm robotic platforms known to the author at the time of writing. Until the functionality of swarm robotic hardware increases such that basic recovery actions – such as those described in this thesis or similar

– are possible, the path to achieving fault tolerant swarms will be confined to the realm of concept.

7.4 Future Work

This section discusses some potential and/or necessary avenues for carrying the work described in this thesis forward.

7.4.1 Complex Faults

A natural avenue for future work is the inclusion of multiple faults in parallel, or faults occurring in series where one is the consequence of another e.g. a software fault that causes a sensor or motor fault. It is possible that such faults may have unique fault signatures, or that they may be indistinguishable from single faults. In the case of the former, the proposed system should be able to learn the signature of the new type of fault, however it may be desirable for the system to be able to initiate multiple recovery actions in parallel rather than dealing with each fault individually in a brute force style approach. To implement this additional layer of complexity to the proposed system will likely be non-trivial, but the ability to isolate the individual faulty components that comprise a complex fault signature would certainly be desirable for autonomous systems.

7.4.2 Integration with fault detection

Future work should seek to integrate the proposed method of fault diagnosis with a mature approach to fault detection. The CRM-based approach (Tarapore, Christensen, and Timmis, 2017), or something similar, is an obvious candidate for this integration. The results obtained for this thesis suggest that improved fault detection mechanisms would allow the proposed system to achieve competitive performance when implemented in hardware as was demonstrated in software simulation. If this can be achieved, available swarm robotic hardware will be one of the last remaining barriers between research into improving fault tolerance in robot swarms and tangible real-world demonstrations of fault tolerant swarms.

7.4.3 Development of more capable swarm hardware

To the best of the author's knowledge, at the time of writing there are no commercially available swarm robotic platforms that have the sensing capabilities required to estimate the behavioural states of neighbouring robots as is described in this thesis. This is a necessary element of the proposed fault diagnosis system in order to avoid the limitations associated with endogenous fault detection. Additionally, the recovery actions proposed in this paper are beyond the capabilities of current swarm robotic platforms. This thesis argues that the ability of the swarm to autonomously perform recovery actions on individual robots is crucial to maintaining long-term autonomy.

Whilst not a direct extension of this particular work, the development of swarm robotic platforms with greater sensing capabilities and a greater range of functionalities is a requirement for implementing the proposed system in actual robot hardware, as well as generally bringing the state of swarm robotic hardware closer to being able to perform the types of task described by Şahin (2005).

7.5 Conclusion

The initial research question set out in the first chapter of this thesis was:

> Can faults in swarm robotic systems be diagnosed by examining specifically where discrepancies between observed and expected behaviours originate from?

This thesis demonstrates in Chapter 3 that different fault types could be classified appropriately and reliably by encoding robot behaviours as behavioural feature vectors (BFVs). In Chapters 4, 5 and 6 this thesis proposes and demonstrates a novel method for fault diagnosis in software simulation and robot hardware, based around the fault signatures produced by different types of faults in robot BFVs. The proposed system was demonstrated to quantifiably improve swarm performance when implemented, and was also demonstrated to be flexible and scalable as is desired for swarm systems.

The initial aims of this thesis have been met, and the motivating factor of improving fault tolerance in swarms has been directly addressed. The research described in this thesis therefore represents a valuable new step towards achieving active fault tolerance in swarm robotic systems.

Bibliography

Abbass, Hussein A, George Leu, and Kathryn E Merrick (2016). "A Review of Theoretical and Practical Challenges of Trusted Autonomy in Big Data." In: *IEEE Access* 4, pp. 2808–2830.

Aickelin, Uwe and Steve Cayzer (2008). "The danger theory and its application to artificial immune systems". In: *arXiv preprint arXiv:0801.3549*.

Alden, Kieran et al. (2013). "Spartan: a comprehensive tool for understanding uncertainty in simulations of biological systems". In: *PLoS Computational Biology* 9.2, e1002916.

Anchor, Kevin P et al. (2002). "Extending the computer defense immune system: Network intrusion detection with a multiobjective evolutionary programming approach". In: *First International Conference on Artificial Immune Systems*. Citeseer, pp. 12–21.

Bayar, Nawel et al. (2015). "Fault detection, diagnosis and recovery using Artificial Immune Systems: A review". In: *Engineering Applications of Artificial Intelligence* 46, pp. 43–57.

Bayındır, Levent (2016). "A review of swarm robotics tasks". In: *Neurocomputing* 172, pp. 292–321.

Benesty, Jacob et al. (2009). "Pearson correlation coefficient". In: *Noise reduction in speech processing*. Springer, pp. 1–4.

Bjerknes, Jan Dyre and Alan F T Winfield (2013). "On fault tolerance and scalability of swarm robotic systems". In: *Distributed Autonomous Robotic Systems*. Springer, pp. 431–444.

Bonani, Michael et al. (2010). "The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4187–4193.

Brambilla, Manuele et al. (2013). "Swarm robotics: a review from the swarm engineering perspective". In: *Swarm Intelligence* 7.1, pp. 1–41.

Burnet, Sir Frank Macfarlane and Others (1959). *The clonal selection theory of acquired immunity*. Vol. 3. Vanderbilt University Press Nashville.

Camazine, S et al. (2001). *Self-Organisation in Biological Systems*. Princeton University Press.

Capra, J Donald et al. (1999). *Inmunobiology: the inmune system in health and disease*. Garland Publishing,

Carlson, Jennifer (2004). "Analysis of how mobile robots fail in the field". PhD thesis. University of South Florida.

Carneiro, Jorge et al. (2007). "When three is not a crowd: a crossregulation model of the dynamics and repertoire selection of regulatory CD4+ T cells". In: *Immunological reviews* 216.1, pp. 48–68.

Carrasco, Rodrigo A, Felipe Núñez, and Aldo Cipriano (2011). "Fault detection and isolation in cooperative mobile robots using multilayer architecture and dynamic observers". In: *Robotica* 29.4, pp. 555–562.

Cassinis, R et al. (1999). "Strategies for navigation of robot swarms to be used in landmines detection". In: *Third European Workshop on Advanced Mobile Robots*. IEEE, pp. 211–218.

Christensen, A. L. et al. (2008). "Fault Detection in Autonomous Robots Based on Fault Injection and Learning". In: *Autonomous Robots* 24.1, pp. 49–67.

Christensen, Anders Lyhne (2008). "Fault detection in autonomous robots". PhD thesis. Université libre de Bruxelles.

Christensen, Anders Lyhne, Rehan O'Grady, and Marco Dorigo (2009). "From fireflies to fault-tolerant swarms of robots". In: *IEEE Transactions on Evolutionary Computation* 13.4, pp. 754–766.

Christensen, Anders Lyhne et al. (2007). "Exogenous fault detection in a collective robotic task". In: *European Conference on Artificial Life*. Springer, pp. 555–564.

Cohen, Irun R (2000). *Tending Adam's Garden: evolving the cognitive immune self*. Academic Press.

— (2007). "Real and artificial immune systems: computing the state of the body". In: *Nature Reviews Immunology* 7.7, pp. 569–574.

Correll, Nikolaus et al. (2009). "Ad-hoc wireless network coverage with networked robots that cannot localize". In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 3878–3885.

Cruz-Cortés, Nareli, Daniel Trejo-Pérez, and Carlos A Coello Coello (2005). "Handling constraints in global optimization using an artificial immune system". In: *International Conference on Artificial Immune Systems*. Springer, pp. 234–247.

Daigle, Matthew J, Xenofon D Koutsoukos, and Gautam Biswas (2007). "Distributed diagnosis in formations of mobile robots". In: *IEEE Transactions on Robotics* 23.2, pp. 353–369.

Dailey, K W (2004). *The FMEA Handbook*.

Davis, Lawrence (1991). "Handbook of genetic algorithms". In:

De Castro, Leandro Nunes and Jonathan Timmis (2002). *Artificial immune systems: a new computational intelligence approach*. Springer Science & Business Media.

De Lemos, Rogério et al. (2007). "Immune-inspired adaptable error detection for automated teller machines". In: *IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS PART C APPLICATIONS AND REVIEWS* 37.5, p. 873.

Ericson, Clifton A and Clifton Ll (1999). "Fault tree analysis". In: *System Safety Conference, Orlando, Florida*, pp. 1–9.

Farmer, J Doyne, Norman H Packard, and Alan S Perelson (1986). "The immune system, adaptation, and machine learning". In: *Physica D: Nonlinear Phenomena* 22.1, pp. 187–204.

Floreano, Dario and Claudio Mattiussi (2008). *Bio-inspired artificial intelligence: theories, methods, and technologies*. MIT press.

Freitas, Alex A and Jon Timmis (2007). "Revisiting the foundations of artificial immune systems for data mining". In: *IEEE Transactions on Evolutionary Computation* 11.4, pp. 521–540.

Freschi, Fabio and Maurizio Repetto (2005). "Multiobjective optimization by a modified artificial immune system algorithm". In: *International Conference on Artificial Immune Systems*. Springer, pp. 248–261.

Garnier, Simon, Jacques Gautrais, and Guy Theraulaz (2007). "The biological principles of swarm intelligence". In: *Swarm Intelligence* 1.1, pp. 3–31.

Gerkey, Brian P and Maja J Mataric (2002). "Sold!: Auction methods for multirobot coordination". In: *IEEE transactions on robotics and automation* 18.5, pp. 758–768.

Hart, Emma and Jon Timmis (2008). "Application areas of AIS: The past, the present and the future". In: *Applied soft computing* 8.1, pp. 191–201.

Haynes, Winston (2013). "Wilcoxon rank sum test". In: *Encyclopedia of Systems Biology*. Springer, pp. 2354–2355.

Hilder, James et al. (2016). "The Psi Swarm: A Low-Cost Robotics Platform and Its Use in an Education Setting". In: *Conference Towards Autonomous Robotic Systems*. Springer, pp. 158–164.

Holland, John H (1983). "Escaping brittleness". In: *Proceedings Second International Workshop on Machine Learning*. Citeseer, pp. 92–95.

Huang, Leon and John J Selman (1986). *Watchdog timer*. US Patent 4,627,060.

Isermann, Rolf (2005). "Model-based fault-detection and diagnosis–status and applications". In: *Annual Reviews in control* 29.1, pp. 71–85.

Jakobi, Nick, Phil Husbands, and Inman Harvey (1995). "Noise and the reality gap: The use of simulation in evolutionary robotics". In: *European Conference on Artificial Life*. Springer, pp. 704–720.

Janeway, Charles A (1992). "The immune system evolved to discriminate infectious nonself from noninfectious self". In: *Immunology today* 13.1, pp. 11–16.

Jerne, Niels K (1974). "Towards a network theory of the immune system." In: *Annales d'immunologie*. Vol. 125. 1-2, pp. 373–389.

Kakalis, Nikolaos M P and Yiannis Ventikos (2008). "Robotic swarm concept for efficient oil spill confrontation". In: *Journal of hazardous materials* 154.1, pp. 880–887.

Khadidos, Adil, Richard M Crowder, and Paul H Chappell (2015). "Exogenous Fault Detection and Recovery for Swarm Robotics". In: *IFAC-PapersOnLine* 48.3, pp. 2405–2410.

Kindt, Thomas J et al. (2007). *Kuby immunology*. Macmillan.

Kutzer, Michael DM et al. (2008). "Toward cooperative team-diagnosis in multi-robot systems". In: *The International Journal of Robotics Research* 27.9, pp. 1069–1090.

Lafferty, Kevin John and A J Cunningham (1975). "A new analysis of allogeneic interactions." In: *Australian Journal of Experimental Biology & Medical Science* 53.1.

Langman, R E and M Cohn (2000). "Third Round-Editorial Summary". In: *Seminars in Immunology*. Vol. 12. 3. [Philadelphia, PA]: WB Saunders,[c1989-, p. 343.

Laurentys, C A, Reinaldo M Palhares, and Walmir M Caminhas (2010). "Design of an artificial immune system based on Danger Model for fault detection". In: *Expert Systems with Applications* 37.7, pp. 5145–5152.

Marsland, Stephen, Ulrich Nehmzow, and Jonathan Shapiro (2005). "On-line novelty detection for autonomous mobile robots". In: *Robotics and Autonomous Systems* 51.2, pp. 191–206.

Matzinger, Polly (2002). "The danger model: a renewed sense of self". In: *Science* 296.5566, pp. 301–305.

Millard, Alan G (2016). "Exogenous Fault Detection in Swarm Robotic Systems". PhD thesis. University of York.

Millard, Alan Gregory et al. (2018). "ARDebug: an augmented reality tool for analysing and debugging swarm robotic systems". In: *Frontiers in Robotics and AI* 5, p. 87.

Millonas, Mark M (1993). "Swarms, phase transitions, and collective intelligence". In: *arXiv:adap-org/9306002*.

Mondada, Francesco et al. (2009). "The e-puck, a robot designed for education in engineering". In: *Proceedings of the 9th conference on autonomous robot systems and competitions*. Vol. 1. LIS-CONF-2009-004. IPCB: Instituto Politécnico de Castelo Branco, pp. 59–65.

O'Dowd, Paul J, Alan FT Winfield, and Matthew Studley (2011). "The distributed co-evolution of an embodied simulator and controller for swarm robot behaviours". In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 4995–5000.

Owen, Judith A, Jenni Punt, Sharon A Stranford, et al. (2013). *Kuby immunology*. WH Freeman New York.

O'Keeffe, James et al. (2017a). "Fault Diagnosis in Robot Swarms: An Adaptive Online Behaviour Characterisation Approach". In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, pp. 1–8.

— (2017b). "Towards Fault Diagnosis in Robot Swarms: An Online Behaviour Characterisation Approach". In: *Conference Towards Autonomous Robotic Systems*. Springer, pp. 393–407.

— (2018). "Adaptive Online Fault Diagnosis in Autonomous Robot Swarms". In: *Frontiers in Robotics and AI* 5, p. 131.

Parker, Lynne E (1998). "ALLIANCE: An architecture for fault tolerant multirobot cooperation". In: *IEEE transactions on robotics and automation* 14.2, pp. 220–240.

Pinciroli, Carlo et al. (2011). "ARGoS: a modular, multi-engine simulator for heterogeneous swarm robotics". In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 5027–5034.

Pradeu, Thomas and Edwin L Cooper (2012). "The danger theory: 20 years later". In: *Frontiers in immunology* 3, p. 287.

Quinlan, J. Ross (1986). "Induction of decision trees". In: *Machine learning* 1.1, pp. 81–106.

Read, Mark et al. (2012). "Techniques for grounding agent-based simulations in the real domain: a case study in experimental autoimmune encephalomyelitis". In: *Mathematical and Computer Modelling of Dynamical Systems* 18.1, pp. 67–86.

Şahin, Erol (2005). "Swarm robotics: From sources of inspiration to domains of application". In: *Swarm Robotics*. Springer. Springer Berlin Heidelberg, pp. 10–20.

Segal, L and I Cohen (2001). *Design principles for the immune system and other distributed systems*.

Stepney, Susan et al. (2004). "Towards a conceptual framework for artificial immune systems". In: *International Conference on Artificial Immune Systems*. Springer, pp. 53–64.

Stibor, Thomas, Jonathan Timmis, and Claudia Eckert (2005). "A comparative study of real-valued negative selection to statistical anomaly detection techniques". In: *International Conference on Artificial Immune Systems*. Springer, pp. 262–275.

Sugawara, Ken and Toshinori Watanabe (2002). "Swarming robots-foraging behavior of simple multirobot system". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 3. IEEE, pp. 2702–2707.

Tarapore, Danesh, Anders Lyhne Christensen, and Jon Timmis (2017). "Generic, scalable and decentralized fault detection for robot swarms". In: *PLoS ONE 12(8): e0182058*. https://doi.org/10.1371/journal.pone.0182058.

Timmis, J et al. (2010). "An artificial immune system for robot organisms". In: *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*, pp. 268–288.

Timmis, Jon, Paul Andrews, and Emma Hart (2010). "On artificial immune systems and swarm intelligence". In: *Swarm Intelligence* 4.4, pp. 247–273.

Timmis, Jon et al. (2008). "An interdisciplinary perspective on artificial immune systems". In: *Evolutionary Intelligence* 1, pp. 5–26.

Watkins, Andrew and Jon Timmis (2004). "Exploiting parallelism inherent in AIRS, an artificial immune classifier". In: *International Conference on Artificial Immune Systems*. Springer, pp. 427–438.

Winfield, Alan F T and Julien Nembrini (2006). "Safety in numbers: fault-tolerance in robot swarms". In: *International Journal of Modelling, Identification and Control* 1, pp. 30–37.

Winfield, Alan F T et al. (2008). "Modelling a wireless connected swarm of mobile robots". In: *Swarm Intelligence* 2.2-4, pp. 241–266.

Zhang, Youmin and Jin Jiang (2008). "Bibliographical review on reconfigurable fault-tolerant control systems". In: *Annual reviews in control* 32.2, pp. 229–252.