Towards Fault Diagnosis in Robot Swarms: An Online Behaviour Characterisation Approach

James O'Keeffe[1]*, Danesh Tarapore[2], Alan G. Millard[1], and Jon Timmis[1]

[1] Department of Electronic Engineering
University of York, York, UK
jhok500@york.ac.uk*
[2] School of Electronics and Computer Science
University of Southampton, Southampton, UK

Abstract. Although robustness has been cited as an inherent advantage of swarm robotics systems, it has been been shown that this is not always the case. Fault diagnosis will be critical for future swarm robotics systems if they are to retain their advantages (robustness, flexibility and scalability). In this paper, existing work on fault detection is used as a foundation to propose a novel approach for fault diagnosis in swarms based on a behavioural feature vector approach. Initial results show that behavioural feature vectors can be used to reliably diagnose common electro-mechanical fault types in most cases tested.

Keywords: Fault Diagnosis · Feature Vector · Behaviour Characterisation · Swarm Robotics

1 Introduction

For many years, robustness – the ability of a system to tolerate the presence of faults or failures – was considered to be an inherent property of swarm robotics systems [15]. However, investigations by Winfield and Nembrini [18] revealed that swarm robustness cannot be taken for granted in all scenarios, particularly those in which partial faults or failures are present.

A further study, conducted by Bjerknes and Winfield [3], demonstrated that the scalability of a swarm system may also suffer in scenarios where the system is not robust to the presence of faults. In light of these two supposed advantages of swarm robotics being lost or severely hindered in the presence of partial failures, the authors [3] advocate for an active approach to improving the fault tolerance of swarm systems – specifically Artificial Immune Systems (AIS), which are defined by De Castro and Timmis [9] as:

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

Timmis et al. [17] argue that an AIS can be considered a type of swarm intelligent system. In the context of the natural immune system, Cohen [8] defines

maintenance as the ability of the immune system to maintain its host despite the unpredictable blows that it will undoubtedly encounter over the course of its life. The idea of maintenance in biological systems is comparable to that of fault tolerance in engineered systems – both pertain to systems that continue to operate under unusual or unexpected circumstances. Cohen [8] argues that maintenance comprises three stages:

1. Recognition – distinguishing what is normal from what is abnormal
2. Cognition – making decisions based on available information
3. Action – doing something as a result of any decisions made

Recognition, cognition and action (RCA), taken alone as a three part programme, do not do justice to the complexity of the natural immune system. However, it could be argued that some of the more complex elements of the natural immune system, such as its ability to seemingly learn and remember, exist largely to enhance the process of RCA. Therefore, it is proposed that RCA be considered the core process of an immune response – and that establishing RCA in swarm robotics systems is a step toward solving the problems outlined in [3]. Cohen's definitions of recognition, cognition and action [8] map nicely to the control engineering subfield of fault detection, diagnosis and recovery (FDDR). If RCA is considered to be the core process in the natural immune response then, by extension, FDDR may be considered as the foundation of an artificial immune response in swarm robotics systems.

The first part of any active FDDR approach is fault detection. There are a number of examples where fault detection in swarm systems, by various methods, has been successful in simulation (see the work by Millard et al. [11], Khadidos et al. [10] and Tarapore et al. [16]), and in hardware (see the work by Christensen et al. [7]). In contrast, there has been limited research on fault diagnosis in robots, examples include the immune-inspired work by Bi [1] and Bi et al. [2]. However, these works only consider single-robot systems. To the best of our knowledge, very little research has been conducted with regard to fault diagnosis in swarm robotics systems, as defined by Şahin [15].

Some of the previous work on fault detection in swarms (such as [7], [10]) opts to omit an attempt at fault diagnosis, and, instead, removes or shuts down any robot which is detected as faulty. Although such approaches have shown themselves to be effective in the controlled scenarios in which they are tested, this is an expensive approach, and one which may not always be justifiable - particularly in cases where the cost of replacing a faulty part of an individual robot is significantly less than replacing the whole unit.

It is not unreasonable to assume that, as the field of swarm robotics continues to expand and develop, so too will the complexity of the tasks they are assigned to, as well as the complexity of their constituent robots. If this is the case, omitting fault diagnosis and partial recovery options will become an increasingly inefficient approach – inhibiting a swarm robotics system's scalability and their capability to operate in challenging environments. Furthermore, some emergent behaviours typically require a minimum number of functional robots in a swarm,

so approaches that simply remove faulty individuals from the collective may jeopardise long-term autonomy [3].

For the reasons outlined above, the problem addressed in this paper is that of fault diagnosis in swarm robotics systems. Although fault diagnosis is the focus of this paper, a complete artificial immune response will include fully-integrated FDDR. Therefore, when developing a means of fault diagnosis, there should be some consideration as to how this might bridge the gap between fault detection (which is a prerequisite to any integrated autonomous diagnosis mechanism) and fault recovery.

Winfield and Nembrini [18] show that different fault types will cause a robot to exhibit behaviour that deviates from its expected behaviour in different ways. The hypothesis of this work is that, if individual robot behaviours can be appropriately characterised as a series of features, different fault types will produce unique patterns of these features – thereby making them classifiable. This paper therefore proposes a fault diagnosis approach based on behavioural feature vectors (BFV), which encode a robot's behaviour as a series of binary features. Such feature vectors have been previously used successfully by Tarapore et al. [16] for the purpose of fault detection in swarm robotics systems and, as is shown in this paper, can similarly be used to perform fault diagnosis via the characterisation of robot behaviour.

The contribution of this paper is to propose a novel approach to combine proprioceptively sensed and externally sensed features to diagnose fault types. This is in line with the fault tolerant and swarm robotics perspectives proposed by Winfield and Nembrini [18] and Şahin [15], respectively.

2 Behaviour Characterisation

The problem now is in defining a BFV that will allow for the reliable classification of fault types. Given that a swarm robotics system may exhibit a number of different normal behaviours, this must be taken into account when defining the BFV and selecting an appropriate classification method.

There are a variety of ways to design a feature vector for an individual robot. For example, assuming access to the relevant hardware/software documentation, it should be possible to systematically reduce every possible functionality to a series of base behaviours. For example, for ground based vehicles, individual features could relate to whether or not its wheels are turning and, if so, which wheels and how fast? To do this exhaustively, however, may prove to be a time consuming process and, in the context of fault classification, may not be necessary as it may not exclusively produce *discriminating features*. Discriminating features, which are actively sought, are features that allow a system to distinguish different fault types based on their presence, absence, or in more complex systems, their intensity. By way of example, 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. Hence, a feature describing the motion of the robot could be considered discriminatory in this case.

It may be possible for a system to develop, and even evolve in real-time, its own discriminatory BFV using search-based optimisation techniques. However, the focus in this paper will be on predefined and unchanging BFVs. It is necessary, if it is to be discriminatory, that a BFV for an individual robot be designed in sympathy with the behaviour(s) that the robot will exhibit. It is therefore also necessary for the swarm robotics system to exhibit static user-defined, or at least user-selected, behaviour(s). It should then be possible for the user to systematically atomise these behaviours into sub-behaviours. For example, obstacle/neighbour proximity sensing and linear and angular motion could be considered sub-behaviours of general obstacle avoidance behaviour. If the user atomises a known finite number of behaviours into sub-behaviours of an appropriate granularity (some of which will be common to multiple behaviours) and assigns a feature to each of these, the resultant repertoire of BFVs should then be representative of every behaviour the system can exhibit.

Although this makes a number of assumptions regarding swarm systems and their applications, it is anticipated that, in a vast majority of cases, a swarm system will have been designed or selected for use with a specific application, or applications, in mind. It is therefore reasonable to say that, assuming the systems behaviours are non-evolving, the chosen system will exhibit a finite number of pre-determined behaviours when performing its task(s).

There has been previous mention of work by several researchers toward fault detection in swarm robotics systems [12], [7], [10], [16]. In each piece of work, the fundamental mechanism that enables a fault to be detected can be essentially reduced to the comparison of an individual's observed behaviour to that which the swarm or observer expects it to exhibit. We also acknowledge that the problems associated with endogenous fault detection (fault detection performed proprioceptively), such as a robot suffering controller software hang being unable to communicate this to the swarm [6], also applies to fault diagnosis – using the same example, the faulty robot would report a BFV that did not reflect its true behaviour, potentially leading to misclassification. Interestingly, the same could also be said of exogenous approaches (fault detection performed using external observations) when applied to fault diagnosis; it would be very difficult for an independent observer to distinguish a robot suffering an onboard software hang that renders it unresponsive to its neighbours from a common sensor failure, as both could potentially have the same net effect on robot's behaviour.

It is proposed that one way in which this problem could be solved is by combining proprioceptively and externally estimated BFVs, producing what can be thought of as a behavioural feature quasi-matrix. To do this, the proprioceptive BFV is designed such that it reflects what an individual robot *believes* its state is, and the externally estimated BFV such that it reflects what its neighbours ascertain its state to *actually* be. It is 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.

2.1 Deriving Behavioural Features from Robot Behaviours

For fault diagnosis to be a useful technique 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. This paper chooses to use flocking, aggregation, and dispersion behaviours as a case study (see Algorithms 1, 2, and 3) based on their widespread use in swarm robotics research. These three behaviours can each be considered similar at an individual level, sharing simple and identical functionalities, yet produce significantly different collective behaviours – satisfying the behavioural conditions for swarm robotics systems described in [15]. This is therefore a simple, yet representative, starting point for work towards a fault diagnosis mechanism for swarm systems.

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, after noise). The close-proximity threshold, C, which appears in Algorithm 1, Algorithm 2 and Algorithm 3 dictates the distance at which a robot will avert its course to avoid collision (approximately 3cm, after noise). The precise values of k and C were decided upon with consideration to the scales of the particular robots and arena used for this work. The values for both are liable to vary in proportion to the system under consideration.

This work considers every instance in which a robot executes an action, and then assigns one feature to that action and one feature in sympathy with the conditions that necessitate it. Following this process leads to the following BFV that is proposed to be sufficiently representative of the aforementioned behaviours. Following Tarapore et al. [16], the BFV described here is a concatenation of five individual binary features, where a returned value of 1 or 0 indicates the presence or absence of the feature, respectively:

$$F_1 = 1 \text{ if } N_R > 0, \text{ otherwise } F_1 = 0 \tag{1}$$

where N_R is the total number of neighbours in sensing range of the robot.

$$F_2 = 1 \text{ if } \ N_C > 0, \text{ otherwise } F_2 = 0 \tag{2}$$

where N_C is the total number of neighbours at a distance less than the close proximity threshold, C, to the robot.

$$F_3 = 1 \text{ if } |v| > \frac{4}{5}|v_{max}|, \text{ otherwise } F_3 = 0 \tag{3}$$

where $|v|$ is the magnitude of linear velocity.

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

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

$$F_4 = 1 \text{ if } |v| > \frac{1}{5}|v_{max}|, \text{ otherwise } F_4 = 0 \tag{4}$$

$$F_5 = 1 \text{ if } |\omega| > \frac{2}{5}|\omega_{max}|, \text{ otherwise } F_5 = 0 \tag{5}$$

where $|\omega|$ is the magnitude of the robots angular velocity.

Each feature is updated once per control-cycle (10ms). Because of the nature of the externally sensed features, this means that F_3, F_4 and F_5 are measured over a very small window. Exactly how long a period these features should be measured over is itself an optimisation problem. Measuring them over a longer period will reduce the effects of noise. However, each time these features change there will be some resultant discrepancy. This is because proprioceptively sensed features will update in real-time, whereas the externally sensed features are only able to update, at best, one tick subsequently. Therefore, increasing the window over which these features are measured necessarily increases the amount of time proprioceptive and externally sensed features will be discrepant - possibly producing undesirable results. This is a problem that may have to be revisited in the future, however, for the purposes of this work in simulation, updating features at each control-cycle produced adequate results.

Algorithm 3 Dispersion

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

The proprioceptive BFV is estimated via an individual robot's sensors (for features relating to object/neighbour proximity) or its controller (for features relating to a robot's movement). The externally estimated BFV uses a simulated overhead sensor to obtain coordinate information for each robot at every timestep. It is acknowledged that the use of an overhead sensor cannot be considered traditionally 'swarm-like', nor realistic for multi-robot system in various scenarios, and it is not proposed as a long term solution. Rather, it should be thought of as a virtual sensor, in so far as it provides data that could be obtained locally but, for ease of experimentation, has been obtained via an overhead sensor with consideration given to a decentralised system i.e. robots will not be provided with neighbour information if that neighbour is not in range of a local sensor. Furthermore, the main objective of this work is not to provide a fully integrable fault diagnosis mechanism for swarm systems, but to provide evidence that a feature vector approach has merit in the context of fault diagnosis, and is something that is worth developing towards a more complete system. Therefore this work meets the criteria for swarm robotics systems, as described by Şahin [15].

Feature Thresholds The combination of features F_3, F_4 and F_5 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 in a straight line at maximum speed, turning (which is approximate to moving at half speed, albeit not in a straight line), or stationary. It is desirable that each of these states be distinct from one another.

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 sole 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 robot which has completely stopped. Although a robot with $0 < v < v_{max}$ 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 overhead sensor noise about a stationary robot. Consequently, the absence of F_4 is the sole indicator that a robot has stopped moving.

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.

This is by no means an exhaustive list of discriminatory features for the given behaviours and, similarly to the use of an overhead sensor, should only be viewed as a provisional BFV that can provide an adequate proof-of-principle. 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.

2.2 Fault Types

If a fault or failure occurs in a system that does not cause the system to exhibit a behaviour that deviates from what could be considered normal, then the system can be said to be tolerant to the failure and thus no further action is required. Therefore, research into swarm FDDR need only consider faults or failures that cause disruption to the swarm's collective behaviour. Exactly what these faults are will vary from system to system and depend largely on the behaviour that particular system is exhibiting at a particular time. More important at this stage is obtaining a proof-of-concept that different fault types are indeed distinguishable from one another via the BFVs they produce. Defined below are a number of representative fault types for a simulated marXbot robot [4], based on a cross-section of the literature that informs this work.

Motor failure: The study by Winfield and Nembrini [18], in which they reject the notion of inherent swarm robustness, is among the most significant in motivating this work. In their study, motor failure is, by far, the most damaging to collective behaviour. It is therefore an obvious choice for inclusion in this study. Motor failure is split into two parts for this experiment. The first being a complete failure of an individual's left motor, the second being a partial failure of the same motor. For a complete failure the motor will stop and henceforth become unresponsive to its controller. For a partial failure the motor will remain responsive, however will only cause its associated wheel to turn at half speed.

Sensor failure: Sensor failure is another recurring example used in fault tolerant swarm research [18] [5]. The exact details of a sensor failure will vary from robot to robot, depending on their hardware. In this case, it is once again split into complete sensor failure and partial sensor failure. For a complete sensor failure an individual's range and bearing (RAB) sensor, as well its infra-red (IR) proximity sensor, will completely fail and return 0. That is, the robot will be totally unable to detect the presence of its neighbours or the walls of the arena. For a partial sensor failure, similar to the definition used in [11], a robot will only be able to detect the presence of neighbours within ± 45 ° of its current heading (but still able to detect the arena walls).

Power failure: Again, a recurring example in work on fault tolerance [18] [5]. A robot that suffers a power failure will completely stop moving and remain unresponsive to its surroundings. However, other robots in the swarm will

still be able to detect its presence. This assumes that, in a physical system, each robot will be 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 essentially become nothing more than objects in an arena. Such cases have been shown to have little to no detriment to collective swarm behaviour [18], so are therefore not a priority for FDDR.

Software hang: Software hang is given as an example of a fault that necessitates exogenous, or at least partly exogenous, approaches to fault detection [6]. For this work, software hang will cause a robot to get stuck performing whatever action it was performing at the last moment it was normally functioning. It is worth noting that the robot is allowed to continue broadcasting 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 e.g. encountering a wall). This, again, works on the assumption that robots can sense their neighbours independently, the justification for which is outlined in the previous discussion on power failure.

3 Experimental Setup

To design and test this initial method for a BFV-based approach to fault classification, Autonomous Robots Go Swarming (ARGoS) [13] – a widely-employed robot swarm simulator – is used. The use of a simulator is only a short-term solution, and it is used only with the intention of trialling any fault diagnosis mechanism in a hardware system at the first reasonable opportunity. Given that the work presented in this paper is primarily concerned with developing an approach to fault classification and diagnosis, it is proposed that simulation is sufficient for this proof-of-principle.

A swarm of 10 marXbots [4] are simulated. These robots collectively perform one of the three behaviours described previously in an otherwise empty enclosed arena, representing approximately 16 square metres (see Figure 1). Simulated Gaussian noise is added to each robot's sensor (μ=0, σ=1 °), as well the overhead sensor (μ=0, σ= 2% d_{max}) – where d_{max} is the maximum distance that a robot can travel in one tick. The noise on the overhead sensor is applied to the coordinate values for each robot. Therefore 95% of the estimated linear velocity values of each robot will be within a window equal to 4% of the robots maximum velocity around it's true velocity. The BFV of each robot is recorded over five minutes of simulated time for all behaviour–fault combinations for 10 separate and independent runs. In each run, swarm behaviour stabilised after 50 seconds. This was therefore chosen as the point after which one of the six fault types would be injected into a single robot.

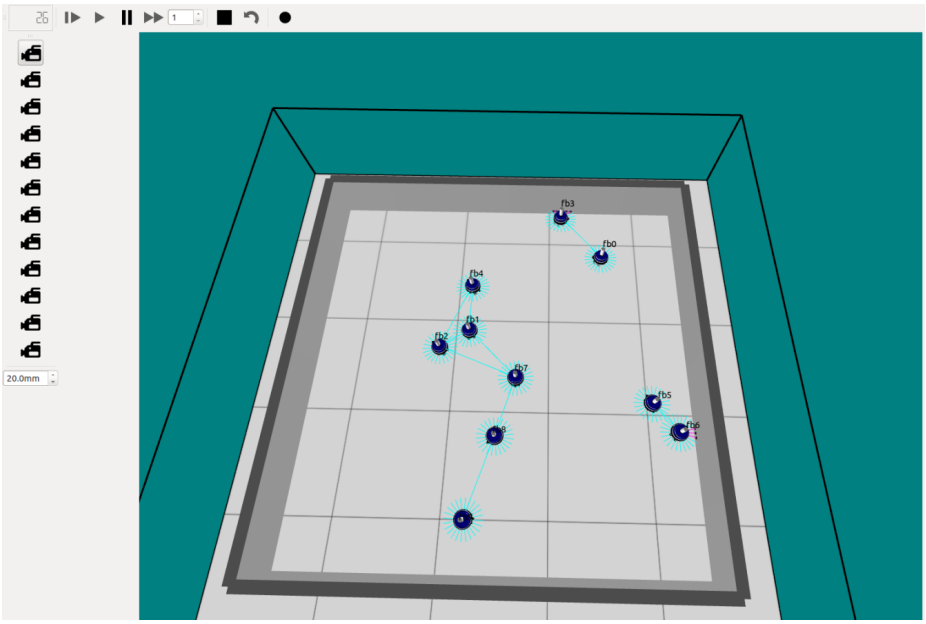

Fig. 1. ARGoS simulation of 10 marXbot robots performing a dispersion behaviour. Lines connecting pairs of robots represent mutual neighbour acknowledgement via RAB sensor. Lines protruding from each robot represent IR sensor range.

4 Results and Discussion

To ascertain how useful BFVs are for classifying fault types a set of data is obtained, consisting of an individual robot's BFV for each of the 6 fault types whilst it performs flocking behaviour. This is then used to train a decision tree [14], after which one can observe how well this tree can classify the same fault types for different behaviours. The advantage of using a decision tree for this process over, for example, a neural network, is that a decision tree allows a user to easily ascertain whether or not a feature is discriminatory. Flocking is used as the normal behaviour for training data because it is the most complex of the behaviours. Consideration is not given to the behaviour of non-faulty robots in the swarm. The reason for this being that significant differences in robot behaviour were not observed unless the robot was itself faulty. It is acknowledged that this may not always be the case, as in [18].

Table 1 shows the average true positive rate for the trained decision tree when it is applied to all three behaviours. These results were obtained by taking an average over all 10 separate and independent runs of this experiment for each behaviour–fault combination. Note that both training and test sets of data do not include the first 50 seconds in which no faults are present. Fault types are indicated in Table 1 as follows:

Fault type	Flocking	Aggregation	Dispersion
H_1	79.5 ±20.4 %	85 ±14.7 %	27.9 ±13.8 %
H_2	98.3 ±2.9 %	88.8 ±31.3 %	80.6 ±11.3 %
H_3	100%	91.8 ±12.3 %	92.1 ±9 %
H_4	86 ±5.8 %	89.3 ±8 %	94 ±3.6 %
H_5	100%	100%	100%
H_6	43.1 ±39 %	8.1 ±25.6 %	40.4 ±42.6 %

Table 1. Average true positive rate ± std.dev. (σ) % for a trained decision tree over 10 replicates.

- H_1 – Complete motor failure
- H_2 – Partial motor failure
- H_3 – Complete sensor failure
- H_4 – Partial sensor failure
- H_5 – Power failure
- H_6 – Software Hang

The results show that, bar a few notable exceptions, a decision tree is able to correctly classify fault types with a high average reliability based on their proprioceptive and externally estimated BFVs. Visualising the trained decision tree confirmed that all 10 features were used in the classification process, and were therefore discriminatory (See Figure 2). The bounds used for each decision, as seen in Figure 2, are all set to 0.5 ($\forall b = 0.5$). This is a result of optimisation, as 0.5 represents the midpoint between 0 and 1. However, as each feature can only take on a value of 0 or 1, the decision tree would give an identical output for $0 < \forall b < 1$.

As the decision tree was trained on a swarm exhibiting flocking behaviour, the best performance is observed when it is applied to a swarm exhibiting flocking behaviour. Although the performance of the decision tree is generally good when applied to swarms exhibiting aggregation or dispersion behaviours, it is expected that the performance observed for each would be even better if the decision tree used were trained on either of these behaviours, respectively.

There is a notably high rate of correct classification across all behaviours for a robot exhibiting power failure. 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 2 that a robot will be classified with a power failure (denoted as 'PF') immediately and exclusively when feature 7 (corresponding to proprioceptively estimated linear velocity) 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.

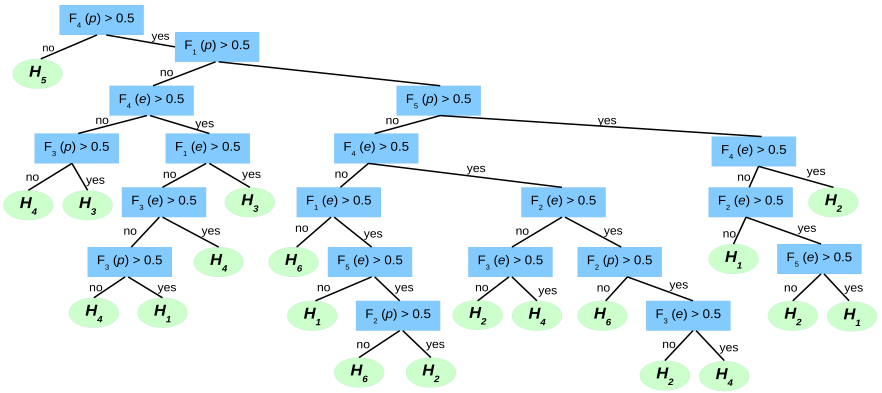

Fig. 2. Visual representation of the trained decision tree where p and e indicate proprioceptively and externally estimated features, respectively.

Interestingly, there is a high rate of misclassification when a complete motor failure is present in a swarm performing dispersion, despite the opposite being true for other behaviours. In these instances, the fault is misclassified as being a partial motor failure (see Table 2). This occurs because complete motor failure in flocking behaviour, on which the decision tree is trained, produces generally different behaviour (robot is static for large periods of time) compared to complete motor failure in dispersion - which far more closely resembles the effects of partial motor failure in flocking (robot spends large periods of time in circular motion). Although this is not ideal, for all practical intents and purposes, this does not pose a significant problem for two reasons:

- Firstly, 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.
- Secondly, 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.

Additionally, there is a persistently high rate of misclassification of software hang across all behaviours. The reason for this is that software hang can produce

Table 2. Mean classification accuracy for 10 replicates of complete motor failure in dispersing robotic swarms.

H_1	H_2	H_3	H_4	H_5	H_6
27.9 ±13.8 %	61.7 ±34.6 %	0%	10.2 ±21.8 %	0%	0.2 ±0.4 %

	H_1	H_2	H_3	H_4	H_5	H_6
Aggregation	1 ±2.8 %	79.9 ±42.1 %	8 ±25.3 %	3 ±6.6 %	0 %	8.1 ±25.6 %
Dispersion	3.8 ±8.3 %	32.1 ±47.1 %	17 ±36.1 %	6.7±6.8 %	0 %	40.4 ±42.6 %

Table 3. Mean classification accuracy for 10 replicates of software hang failure in aggregating and dispersing robotic swarms.

entirely different behaviours depending on what the robot was doing at the point of fault injection. The training data used an example where software hang was injected whilst the robot was moving in a straight line. However, for many subsequent examples, particularly where the swarm behaviour was aggregation, the robot was turning at the point of fault injection - leading to a classification of partial or complete motor failure. Furthermore, even when the robot was moving in a straight line, if the robot had no neighbours in sensing range at the point of fault injection, the subsequent behaviour was indistinguishable from complete sensor failure (see Table 3).

How an individual fault might produce several different behaviours, dependent on external factors, is a consideration that was absent from the work described. However it is one that 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.

5 Conclusion

In this work it has been shown that by characterising robot behaviour as a BFV one is able to diagnose fault types occurring in individuals with a generally high reliability. Having said that, it is acknowledged that the work, described here, towards fault diagnosis in swarms is by no means complete. 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 one can diagnose faults in real-time.

In future work the feature vector approach will be carried forward and developed, for instance by considering time-based patterns, however there will also be a move towards an online and largely unsupervised approach to fault diagnosis. Furthermore, the work will move from simulation to a hardware system. It is acknowledged that working with the sensor noise observed in a hardware system will not be the same as working in simulation with controlled noise, and that the feature vectors described in this paper may not be appropriate for such a system. It is expected that a reduction in performance will be observed when making that transition. However, as each feature has an associated user-defined

threshold to determine it's value, these thresholds (along with other parameters) should be modifiable for compatibility with a hardware system.

Although decision tree algorithms can be useful for identifying discriminating features, they are not envisaged as being a long term solution to a classification problem on account of their inflexibility. Rather, this work will move towards a more complex system whereby a previously unseen fault 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. It is proposed that, based on the results previously described, not only would such a system be possible, but that it could also be built to satisfy the conditions for swarm robustness, flexibility, and scalability.

References

1. Ran Bi. *Immune-inspired fault diagnosis for a robotic system.* PhD thesis, University of York, 2012.
2. Ran Bi, Jon Timmis, and Andy Tyrrell. The diagnostic dendritic cell algorithm for robotic systems. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.
3. Jan Dyre Bjerknes and Alan F T Winfield. On fault tolerance and scalability of swarm robotic systems. In *Distributed Autonomous Robotic Systems*, pages 431–444. Springer, 2013.
4. Michael Bonani, Valentin Longchamp, Stéphane Magnenat, Philippe Rétornaz, Daniel Burnier, Gilles Roulet, Florian Vaussard, Hannes Bleuler, and Francesco Mondada. 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)*, pages 4187–4193. IEEE, 2010.
5. Jennifer Carlson. *Analysis of how mobile robots fail in the field.* PhD thesis, University of South Florida, 2004.
6. A. L. Christensen, R. O'Grady, M. Birattari, and M. Dorigo. Fault detection in autonomous robots based on fault injection and learning. *Autonomous Robots*, 24(1):49–67, 2008.
7. Anders Lyhne Christensen, Rehan O'Grady, and Marco Dorigo. From fireflies to fault-tolerant swarms of robots. *IEEE Transactions on Evolutionary Computation*, 13(4):754–766, 2009.
8. Irun R Cohen. *Tending Adam's Garden: evolving the cognitive immune self.* Academic Press, 2000.
9. Leandro Nunes De Castro and Jonathan Timmis. *Artificial immune systems: a new computational intelligence approach.* Springer Science & Business Media, 2002.
10. Adil Khadidos, Richard M Crowder, and Paul H Chappell. Exogenous Fault Detection and Recovery for Swarm Robotics. *IFAC-PapersOnLine*, 48(3):2405–2410, 2015.
11. Alan G Millard. *Exogenous Fault Detection in Swarm Robotic Systems.* PhD thesis, University of York, 2016.
12. Alan G Millard, Jon Timmis, and Alan F T Winfield. Run-time detection of faults in autonomous mobile robots based on the comparison of simulated and real robot behaviour. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3720–3725. IEEE, 2014.

13. Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, Timothy Stirling, Álvaro Gutíerrez, Luca Maria Gambardella, and Marco Dorigo. ARGoS: a modular, multi-engine simulator for heterogeneous swarm robotics. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 5027–5034. IEEE, 2011.
14. J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.
15. Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *International workshop on swarm robotics*, pages 10–20. Springer, 2004.
16. Danesh Tarapore, Pedro U. Lima, Jorge Carneiro, and Anders Lyhne Christensen. To err is robotic, to tolerate immunological: fault detection in multirobot systems. *Bioinspiration & biomimetics*, 10(1):16014, 2015.
17. Jon Timmis, Paul Andrews, and Emma Hart. On artificial immune systems and swarm intelligence. *Swarm Intelligence*, 4(4):247–273, 2010.
18. Alan F T Winfield and Julien Nembrini. Safety in numbers: fault-tolerance in robot swarms. *International Journal of Modelling, Identification and Control*, 1(1):30–37, 2006.