Journal of King Saud University – Engineering Sciences 33 (2021) 43–53

Contents lists available at ScienceDirect

Journal of King Saud University – Engineering Sciences

journal homepage: www.sciencedirect.com

Review

A survey of modern exogenous fault detection and diagnosis methods for swarm robotics

Olivier Graham Miller, Vaibhav Gandhi *

Faculty of Science and Technology, Middlesex University London, UK

ARTICLE INFO

Article history:
Received 16 April 2019
Accepted 8 December 2019
Available online 14 December 2019

Keywords:
Fault detection
Fault diagnosis
Swarm robotics

ABSTRACT

Swarm robotic systems are heavily inspired by observations of social insects. This often leads to robustness being viewed as an inherent property of them. However, this has been shown to not always be the case. Because of this, fault detection and diagnosis in swarm robotic systems is of the utmost importance for ensuring the continued operation and success of the swarm. This paper provides an overview of recent work in the field of exogenous fault detection and diagnosis in swarm robotics, focusing on the four areas where research is concentrated: immune system, data modelling, and blockchain-based fault detection methods and local-sensing based fault diagnosis methods. Each of these areas have significant advantages and disadvantages which are explored in detail. Though the work presented here represents a significant advancement in the field, there are still large areas that require further research. Specifically, further research is required in testing these methods on real robotic swarms, fault diagnosis methods, and integrating fault detection, diagnosis and recovery methods in order to create robust swarms that can be used for non-trivial tasks.

Contents

1. Introduction

Swarm robotic systems are heavily inspired by observations of social insects such as bees, ants and termites (Barca and Sekercioglu, 2013; Navarro and Matía, 2013; Şahin and Winfield, 2008). These insects act with no centralised coordination mechanisms and can accomplish tasks that are well beyond the skills of an individual in the swarm (Navarro and Matía, 2013; Şahin and Winfield, 2008). The definition of swarm robotics that arises from these social insect systems and is used consistently throughout the literature is of Şahin (2005):

* Corresponding author.
E-mail address: V.Gandhi@mdx.ac.uk (V. Gandhi).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksues.2019.12.005
1018-3639/

'Swarm robotics is the study of how large number [sic] of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from the local interactions among agents and between the agents and the environment'

Şahin (Şahin, 2005) also proposes three main desirable properties of swarm robotic systems:

1 Robustness

The ability of the system to continue to operate despite failures in the individuals.

2 Flexibility

The ability of the system to offer solutions to a variety of tasks by utilizing different coordination strategies.

3 Scalability

The system should be able to operate under a range of different sizes.

These desirable properties of a robot swarm are widely implemented (Ben-Ari and Mondada, 2018; Dorigo et al., 2014; Navarro and Matía, 2013). The robustness of swarm robotic systems is regularly described as being inherent to them, as with social insect swarms. There are three main reasons for this (Winfield and Nembrini, 2006):

1 Inherent redundancy

As with swarms of social insects, a large robotic swarm does not require every member in order to complete a task.

2 Distributed Nature of the System

As robotic swarms do not require a centralised control system which could be susceptible to faults, there is no risk of system-wide failure.

3 Simplicity of Individuals in the Swarm

Individual members of the swarm are simple robots with few components and as such the probability of faults in each individual is greatly reduced when compared to more complex robotic systems.

In spite of these properties, it has been shown that swarm robotic systems are still highly susceptible to system failures resulting from failures in the individuals (Bjerknes and Winfield, 2013). This is more true for partial failure than for complete failure in the individuals of the system (Bjerknes and Winfield, 2013; Winfield and Nembrini, 2006). Additionally, the task that the swarm is meant to perform plays an important role in failure, i.e. if the task involves teamwork, (e.g. moving a large object), then the faults in the system are far more disastrous than if the swarm is only used to speed up a task that a single robot could complete (e.g. exploration of an area) (Bjerknes and Winfield, 2013). This leads to the importance of robust fault detection, diagnosis and recovery methods in swarm robotic systems as being indispensable for completing a variety of tasks.

The term *fault detection* as it used in this paper and those surveyed is defined as the identification of the occurrence of a fault in a swarm robotic system. This can refer either to the general identification of a fault that has occurred somewhere in the system, or the specific identification of the occurrence of a fault in an individual member of the swarm. The term *fault diagnosis* is used to describe the process of identifying the underlying cause of a fault once it has been detected. This involves identifying the system of the robot in which the fault has occurred in the work surveyed but could be as specific as diagnosing more specifically how that subsystem has failed. Once the fault is diagnosed, *fault recovery* can occur. Fault recovery is the process by which a diagnosed fault is repaired or mitigated so as to allow the individual member of the swarm to continue the task, e.g., sharing battery with a member of the swarm whose battery is running low (Carrillo et al., 2018). Fault recovery methods fall outside of the scope of this paper, though they are briefly discussed in relation to the work on fault diagnosis without reference to the practicality of how to achieve them in a real system. These definitions broadly mirror those provided by Chang, Russell and Bratz with regards to industrial process monitoring procedures (Chiang et al., 2000) and of Isermann and Ballé based on the work of IFAC SAFEPROCESS Tech. Committee (Isermann and Ballé, 1997).

Fault detection methods in swarm robotics are broadly categorised as being endogenous or exogenous (Christensen, 2008; Lau, 2012). Endogenous fault detection involves individuals taking on the task of detecting their own faults. This type of fault detection is not unique to swarm robotics, as all robotic systems can make use of internal fault detection mechanisms. However, in the context of a swarm robotic system, endogenous fault detection may be of limited use. For example, if an individual detects a fault in its communication systems, it would then be unable to communicate this fault to the rest of the swarm. Exogenous fault detection involves external detection of faults, usually by another robot. This is highly applicable in case of swarm robotics where individual robots can report on other individuals within the swarm.

This paper will provide an overview of exogenous fault detection and diagnosis methods for swarm robotics with the aim of identifying areas for further research and providing background information on which further research can be built. The three main areas of modern work in exogenous fault detection are data model, immune system and blockchain-based fault detection methods. This paper will also discuss exogenous fault diagnosis methods in swarm robotics. The only research conducted in this area so far is in the area of local-sensing based fault diagnosis. These four areas encompass all modern work in exogenous fault detection & diagnosis in swarm robotics.

This paper is organized as follows: Section 2 provides a brief overview of previous work in exogenous fault detection and diagnosis to provide context for the exploration of the more recent research which follows. The modern research in this area is then presented in four sections. Section 3 details data modelling-based fault detection methods. Section 4 discusses immune system-based fault detection methods. Section 5 gives an overview of the modern research in local sensing-based fault diagnosis. Section 6 focuses on blockchain-based fault detection methods. Section 7 provides a discussion of the advantages and disadvantages of the methods presented to provide a summary of the areas that may require further research. Section 8 concludes the paper with a brief summary and an overview of the main broad areas for further research.

2. Exogenous fault detection and diagnosis

It is useful to preface the discussion of modern research with a brief overview of the previous work in this field. The research described in this section provides the foundations on which much of the modern work on exogenous fault detection and diagnosis is built.

The work of Christensen et al. (2009) in creating a Firefly-based exogenous fault detection method can be seen as the beginning of swarm robotic-specific work in exogenous fault detection. Their work involves synchronising flashing LEDs on each robot in the swarm. If a robot undergoes complete failure, its LED will stop flashing and this will be detected by the swarm. This work was implemented on a real swarm robotic system using the *swarm-bot* platform (Mondada et al., 2004). It is still one of very few pieces of work in this field to be tested on a real robotic swarm.

Other work in exogenous fault detection has been based around simulating a robot's controller (Khadidos et al., 2015; Millard et al., 2014a,b; O'Dowd et al., 2014). Individuals within the swarm simulate the controller of a neighbour and then compare the simulated behaviour to the actual behaviour. This allows the individual running the simulation to detect if another robot is behaving in a faulty manner. This method can detect partial as well as total failures. The simulation requires a fairly high level of complexity in order to accurately determine and predict the behaviour of other robots in the swarm and as such they are computationally expensive. Some of this work has been implemented on a real swarm robotic system, however these methods have not been explored further in modern research.

The idea of using an artificial immune system model for fault detection has also been evolving (Ismail and Timmis, 2010, 2009; Jacob et al., 2006; Lau et al., 2011; Owens et al., 2009; Timmis et al., 2010). The immune system has been suggested to be analogous to a swarm system by Jacob et al. (2006). Some initial work in proposing a fault detection method based on granuloma formation was conducted by Ismail and Timmis (2010, 2009). The potential for immune-system based fault detection methods was then explored further in the work of Timmis et al. (2010) by comparing artificial immune system and swarm intelligence paradigms, arguing that the two are analogous in many ways. In their work, Lau et al. (2011) use a T-cell based algorithm known as the Receptor Density Algorithm (Owens et al., 2009) to measure how effectively the individuals in a robotic swarm complete a foraging task. Work in the field of immune-system based fault detection has been expanded upon significantly and is discussed in detail in Section 4.

Previous work in fault diagnosis for individual robotic systems (e.g. Bi, 2012; Bi et al., 2010; Goel et al., 2000) and multi-robot systems (e.g. Carrasco et al., 2011; Daigle et al., 2007; Kutzer et al., 2008) has been conducted. However, to the best of the authors' knowledge the only research that has been completed on exogenous fault diagnosis in swarm robotics is that of O'Keeffe et al. (2017a,b) which is explored in detail in Section 5.

3. Data model-based fault detection

Khaldi et al. (2017b) have completed work on modelling a swarm robotic system using Principal Component Analysis (PCA). PCA is a commonly used multivariate analysis technique that reduces the dimensionality of a collection of observations by eliminating the less-informative components from the features (Jolliffe, 2011). Input and output data of each robot in the swarm is collected. PCA is then applied to this data, creating linear combinations of various data types to decrease the dimensionality of the observations while still capturing a large amount of the information in the data. All experiments were performed using the ARGoS (Autonomous Robots Go Swarming) simulator (Pinciroli et al., 2012) with 6 foot-bots (Dorigo et al., 2013). The swarm was tested using the Virtual Viscoelastic Control model for circle formation forming (Khaldi and Cherif, 2016). First, a fault-free model of the system was built using two variables from each of the 6 robots, thus giving a total of 12 data points. The two variables used were virtual viscoelastic force length and virtual viscoelastic force angle. An initial fault-free data matrix for the PCA was created from 3000 observations. This PCA model was then used as a comparison for the data collected during further experiments with faults introduced into the system. Three univariate analysis mechanisms were compared to see which provides the best fault-detection capabilities when applied to the key variables created by the PCA model – T2, Q, and Exponentially Weighted Moving Average (EWMA). Three fault types were also tested:

1 Abrupt

The virtual viscoelastic force length of the first individual (i.e. the individual that provides the first two data points) in the system has a small constant deviation of 40% of its variance for a period of time.

2 Intermittent

Both measured variables of the first individual have a small constant deviation for a period of time (40%) and another smaller deviation at a later period of time (10%).

3 Drift

Both measured variables of the first individual are increased at a constant rate after a certain time.

The T^2 statistic was ineffective at detecting all types of faults and created false-positives when no faults were present. The Q statistic was far more effective, detecting abrupt and drift faults with only a small delay, however it struggled when detecting intermittent faults compared to EWMA (cf. Fig. 1). EWMA was able to detect abrupt, drift and intermittent faults in the system with a high level of accuracy and fast speed of detection.

Khaldi et al. further extended their work in (Khaldi et al., 2017a) to compare and test more fault types. The experimental setup used in this work is identical to that of (Khaldi et al., 2017b), but includes another univariate analysis mechanism to compare – Cumulative Sum (CUSUM). The three fault types mentioned previously, as well as an additional three fault types are tested, as detailed below:

1 Abrupt – Minor

The virtual viscoelastic force length of the first individual in the system has a smaller constant deviation (10%) for a period of time.

2 Random Walk

The virtual viscoelastic force length of the first individual is contaminated with random Gaussian noise from a certain time until the end of the experiment.

3 Complete Stop

The virtual viscoelastic force length of the first individual is zeroed from a certain time until the end of the experiment.

The quality of the four methods was quantified based on the false detection rate (detecting an error when none was present) and the missed detection rate (failing to detect an error when it was present). As with their previous work (Khaldi et al., 2017b), Khaldi et al. (2017a) showed that EWMA and CUSUM were significantly better fault detection mechanisms than the T^2 and Q methods, having far lower false detection and missed detection rates over all the 6 faults tested. EWMA was better than CUSUM in detecting faults in 5 of the 6 fault types tested and was only slightly inferior in the case of the random walk fault. EWMA was

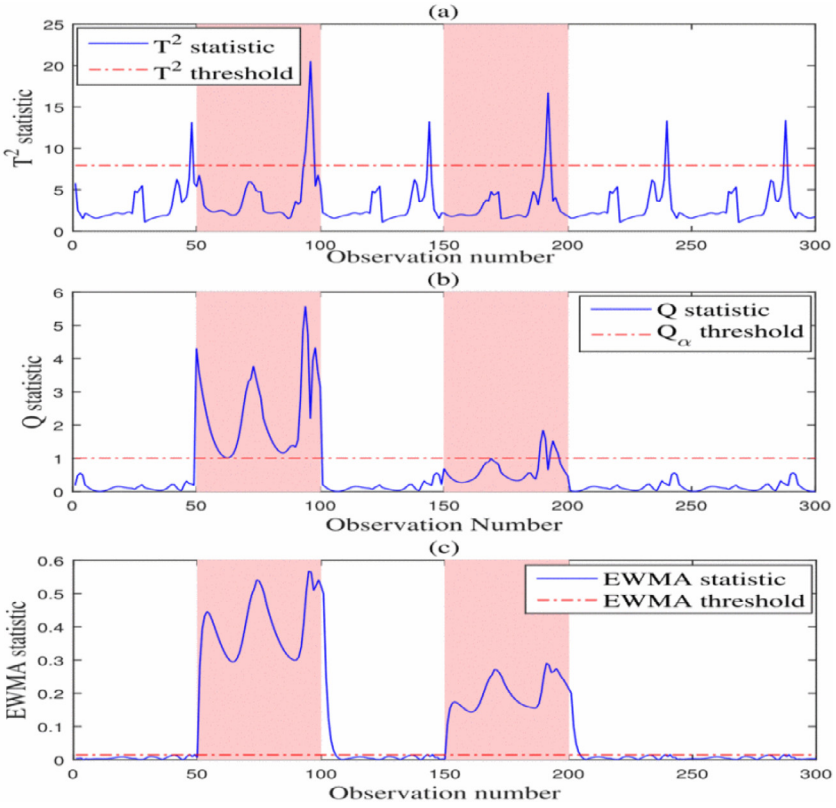

Fig. 1. The efficacy of the three PCA monitoring indices in detecting intermittent faults in the swarm (Source: Khaldi et al., 2017b).

also significantly better than CUSUM in the case of an intermittent fault, with CUSUM detecting a false positive during the period of time between the two fault periods.

The PCA-EWMA model for fault detection was extended by Harrou et al. (2018) by using a Discrete Wavelet Transform (DWT) (Gao and Yan, 2010) model to filter noise from the input data. Harrou *et al* showed that adding DWT for denoising the data of the PCA prior to applying the EWMA analysis led to a significantly lower false detection and missed detection rate than if the data contained any noise.

However, the PCA-based model for fault detection has several serious shortcomings. In a real swarm robotic system, it would be difficult to guarantee that the data being collected to create the fault-free model is actually fault-free. Other fault detection methods would need to be used while the fault-free model is being built and these might interfere with the model if they are not also used during real operation of the system. The second issue is with the centralized nature of the fault detection system. Data has to be collected from every individual in the swarm at regular time intervals, requiring synchronicity of the entire swarm, and then compared against the central fault-free model. This contradicts the swarm robotics paradigm where distributed functionality is key. A solution proposed by Khaldi et al. (2017a) demands that the individuals in the swarm should broadcast the sensor readings to observer robots that would then perform the PCA-based fault detection. This is certainly an improvement in the distribution of the method, however it does require observer robots that are not useful beyond their capacity as observers. Another issue lies in the identification of which individual within the swarm is faulty. Since the PCA model reduces the dimensionality of the raw data, it can only acknowledge that there is a fault in at least one individual in the system and the particular faulty individual cannot be determined. These issues are discussed further in Section 7.

As a fault detection method for a multi-robot system, the work of Khaldi *et al* in PCA-based modelling (Khaldi et al., 2017b,a) may have applications. However, as a fault detection method for real swarm robotic systems, future work into distributing the model and data collection amongst the swarm, as well as research into using the model to identify the faulty individual(s) are required.

4. Immune system-based fault detection

An exogenous fault detection method based on the adaptive immune system of vertebrates has been explored by Tarapore et al. (2017, 2015). These systems can be seen as analogous to the requirements of an effective fault detection method in robotic swarms (Timmis et al., 2010). These adaptive immune systems tolerate persistent and abundant antigens (normal behaviour) while mounting immune responses to foreign pathogens (those which have antigens that are neither persistent nor abundant in the biological system - this is seen to be analogous to faulty behaviour in a robotic swarm). They are also able to learn over time which antigens can be tolerated and which should produce an immune response. This is ideal in a robotic swarm if the tasks of the swarm change over time. Many models for fault detection require an idealized model of the system to which the current system state can be compared. Such models are unable to adapt to dynamic changes in the baseline state of the system, e.g. when the overall goal or process of the system changes. Using an immunological-based fault detection method allows the normal behaviour of the system to easily be reclassified within the model.

Tarapore et al. (2015) demonstrate the effectiveness of using this system as a method of fault detection in robot swarms by implementing a simplified version of the *crossregulation model* (CRM) (Carneiro et al., 2007). CRM is a mathematical model of

immunological tolerance that distinguishes between antigens based on their density and persistence in the system. Each robot in the system runs a local CRM and feeds Binary Feature Vectors (BFVs) that are obtained by taking measurements of the 10 nearest robots in the swarm into this model. These BFVs are comprised of a number of simple features of the observed system that are coded in Boolean form (with either a 0 indicating not present, or a 1 indicating present). Tarapore et al. (2015) use 6 of these features to create a BVF (cf. Table 1). This information is then fed into the CRM model. The state of the model can then be used to determine if the binary feature vectors calculated for the surrounding robots correspond to normal or faulty behaviour. However, the method that Tarapore et al. (2015) use to test their CRM model is idealised. 20 individuals are placed in a 5 × 5 m toroidal arena. The swarm then performs one of four prescribed behaviours: aggregation, dispersion, flocking or homing. One of the 20 robots is prescribed to display a faulty behaviour: moving in a straight line, walking randomly, moving in a circle or stopping altogether. These are not actual system faults of a robot, but rather they are possible outcomes of a system fault. The individuals in the swarm are also assumed to be able to sense their 10 nearest neighbours at all times with no limitations.

Several other tests were also run. These involved varying the size of the swarm, not having any faulty robot to test for false-positives, changing the prescribed behaviour of the swarm during a single test and testing different lengths of the binary feature vectors. It was shown that in this idealised system, most fault types were detected with a high rate of accuracy, though certain fault types were almost never detected during certain behaviours, i.e. a robot that was faulty and moving in a straight line during flocking behaviour was almost never detected. False positives were shown to be reliant on the number of cycles taken to classify the behaviour of the robots. When only one cycle was used to classify behaviour, the incidence of false-positives was high, dropping to a very low number when 100 cycles were used to classify behaviour. This is inversely proportional to the instance of false-negatives and would need to be fine-tuned. Tolerance to changes in the behaviour of the swarm (i.e. from aggregation to dispersion, back to aggregation) was found to be very high. The ideal length for the BFV was shown to be 6, with lower values not providing enough data for accurate error detection, and higher values not providing a significantly higher error detection rate while requiring more computational time to process. There was little difference between the efficacy of fault detection for one normal behaviour or when transitioning between different normal behaviours.

This work was expanded upon (Tarapore et al., 2017) using 20 e-puck robots (Gonçalves et al., 2009) simulated in ARGoS. In this work, Tarapore *et al* use a far less idealised approach than in (Tarapore et al., 2015), where each robot is simulated with real sensing capabilities and noise in the sensor signals. Tarapore *et al*'s system involves a behaviour observation model for generating the BFVs using only local sensing capabilities of the individuals, using the CRM model to classify this behaviour, and then using a swarm coalition algorithm to allow the swarm to make decisions regarding the behaviour of individuals. They also introduce a heterogeneous behaviour for the swarm – foraging. This task requires individuals to be split between 4 separate swarm behaviours: exploration, signalling, resource gathering, and resource transportation. The results mirror and extend those in (Tarapore et al., 2015). False-positives were found to occur in only 1–2% of cases, rising to 2–3% for significant changes in the environment simulated during the foraging behaviour (simulating scarcity of resources, abundance of resources, and a larger distance to resources). The simulated faults were based on the real systems of the robots, i.e. disconnected proximity sensors, obstructed proximity sensors, range and bearing sensor errors, and one or both motors failing. These were all found to be detected reliably in most of the normal behaviour patterns. However, certain faulty behaviours were detected at low rates when the faulty behaviour did not differ significantly from the normal behaviour of the system, e.g. the malfunctioning sensor or motor was not required for the current task of the swarm.

This method of fault detection has shown a high level of robustness and adaptability which will be discussed further in Section 7. There is still future work to be done in testing the method on a real robotic swarm system. Models other than CRM could also be explored and contrasted with the current work to test their efficacy.

Table 1
6 features used to develop the BFVs (Tarapore et al 2015).

Feature	Description
F_1	Proximity to at least one other robot in the swarm: within [0, 30] cm for >50% of the measured time period
F_2	Proximity to at least one other robot in the swarm: within (30, 60] cm for >50% of the measured time period
F_3	Distance travelled >5% of maximum possible distance travelled in the measured time period
F_4	Velocity >5% of maximum velocity at least once during measured time period
F_5	Proximity to at least one other robot in the swarm: [0, 60] cm and angular acceleration >3% of maximum angular acceleration at least once during measured time period
F_6	Angular acceleration >3% of maximum angular acceleration at least once during measured time period

5. Local sensing-based fault diagnosis

Fault diagnosis is an important extension of fault detection and can allow for the possibility of implementing fault recovery methods to increase the robustness of the swarm. In O'Keeffe et al. (2017a), present a proof of concept approach to using BFVs for fault diagnosis in a robotic swarm. They begin by providing a discussion around appropriate feature selection for the BFV. It is important that features are selected such that the combinations of their presence or absence are useful for diagnosing a wide range of faults, e.g., a feature that states whether the robot is currently moving could identify the possibility of a motor fault or a software fault but it would be difficult to diagnose which of these faults is the most likely without additional features that can help to discriminate between the options. It is suggested that these features should be selected based on the specific hardware used and the task(s) that the swarm is applied to. It is also noted that it is necessary to select features that are able to be identified using internal and external sensing capabilities of the members of the swarm. In this paper, O'Keeffe *et al*'s method combines the use of endogenous and exogenous fault detection to allow members of the swarm to compare two versions of their BFVs – one created internally and one generated from an external observer. This comparison adds more depth to the potential data for fault diagnosis, as certain combinations of differences between the two may imply specific fault types. Three behaviours are tested – flocking, aggregation and dispersion. Five features are used in the creation of the BFV. These are detailed in Table 2. Each feature is measured every 10 ms. Features 3, 4 and 5 have thresholds chosen so as to reduce the impact of noise and are meant to represent the robot moving forward, at a complete standstill, or rotating in place. The chosen features are noted not to be ideal but to be adequate for demonstrating the proof of concept. The endogenous BFV is measured using a combination of the individual's sensors and controller. The exogenous BFV is measured using a simulated overhead sensor to track the position of members of the swarm. Though this does break the swarm robotics paradigm, it is noted that this is used only as a

Table 2
5 features used to develop the BFVs (O'Keeffe et al., 2017a).

Feature	Description
F_1	One or more other members of the swarm within sensing distance
F_2	One or more other members of the swarm within the defined close proximity distance
F_3	Linear velocity >80% of maximum linear velocity
F_4	Linear velocity >20% of maximum linear velocity
F_5	Angular velocity >40% of maximum angular velocity

temporary measure for proof of concept and that it would be replaceable in a real system by using only the local sensing capabilities of the individuals. Six fault types were tested as listed below:

1 Complete Left Motor Failure

The left motor stops functioning, and the robot can only rotate using its right wheel.

2 Partial Left Motor Failure

Simulated by the motor's maximum speed dropping to 50% of the initial maximum.

3 Complete Sensor Failure

Simulated by the individual being unable to detect the presence of neighbours.

4 Partial Sensor Failure

Simulated by the individual being unable to detect the presence of neighbours outside ±45° of its current bearing.

5 Complete Power Failure

The individual stops and becomes completely unresponsive.

6 Software Hang

The individual continues whatever behaviour it was exhibiting at the moment of the software hang (e.g. moving forward, turning) and continues broadcasting the same BFV.

This approach was tested in ARGoS using 10 simulated marXbots (Bonani et al., 2010). Simulated Gaussian noise is added to their sensor readings. Once whichever of the three behaviours the swarm is performing has stabilised, one of the 6 fault types is introduced into a single individual. The BFVs of all individuals are recorded over 10 runs for each possible behaviour-fault combination. The endogenous and exogenous BFVs for the faulty individuals in the flocking behaviour are then used to train a decision tree to recognise the various fault types. All 10 features (5 exogenous features and 5 endogenous features) are used in the decision tree's classification, demonstrating that they are discriminating features. Generally, the decision tree is moderately accurate at detecting faults for the flocking behaviour, and somewhat worse for the other two behaviours. If a separate decision tree was trained for each behaviour it is noted that this would likely be more effective. The software hang fault was the most difficult to detect as it is unlikely to present in the same way across each separate test.

This work is significantly extended in (O'Keeffe et al., 2017b) to include real-time fault classification. O'Keeffe *et al*'s method is shown in Fig. 2 and involves individuals detecting faults using BFVs, running diagnostic procedures to diagnose the type of fault, running a recovery procedure to try and fix the fault, and then remembering the presentation of the fault so as not to have to run diagnostic procedures for the same or similar faults in the future. They use 6 fault categories corresponding to the 6 faults tested in their previous work and detailed earlier in this section. It is noted that broad fault categories are generally good enough to allow for fault recovery, e.g. if a sensor is diagnosed to be faulty it does not matter what the specific reason is – the easiest recovery option is just to replace the sensor. This may not always be true in individual swarm robots if they have particularly expensive or difficult-to-replace components, however as most swarm robots are currently quite electromechanically simple this level of fault recovery is acceptable.

Faults are detected via discrepancies between the endogenous and exogenous BFVs. When a discrepancy is detected for a certain period of time, the closest robot to the faulty individual is assigned the role of doctor. If the doctor robot has a record of a similar fault, then it will immediately try the recovery option that was previously successful with that fault. If it does not or if the initial recovery option does not work, the doctor robot will perform a series of diagnostic tests, initially testing for the most serious faults (e.g. software hang, complete power failure) and moving through to testing for the least important faults (e.g. partial motor failure). If one of the tests and the applicable recovery option are successful, the BFVs from the fault are stored in association with the recovery option for future reference. These stored faults and diagnoses are shared between robots in the swarm via an ad-hoc network when individuals are within each other's communication range.

This method was tested in the ARGoS simulation environment. As with their previous work, an overhead sensor is simulated to provide coordinate information for each individual in the swarm. This information is still only available to robots when they are within sensing range of one another and as such could be replaced by purely local sensing capabilities in the future in order to meet the requirements of the swarm robotics paradigm. 10 marXbots are simulated with flocking, aggregation and dispersion behaviours. In this work, the swarm changes between the three behaviours during each single test run, with a random fault injected after each behavioural change once the behaviour has stabilised. A similarity value is calculated between any recorded faulty BFVs and previously BFVs that have been stored along with a recovery option. If the similarity value is over a certain threshold, the stored recovery option is attempted before any diagnosis needs to be undertaken. The fault recovery is purely simulated, with a selection of the right option instantly successfully restoring the

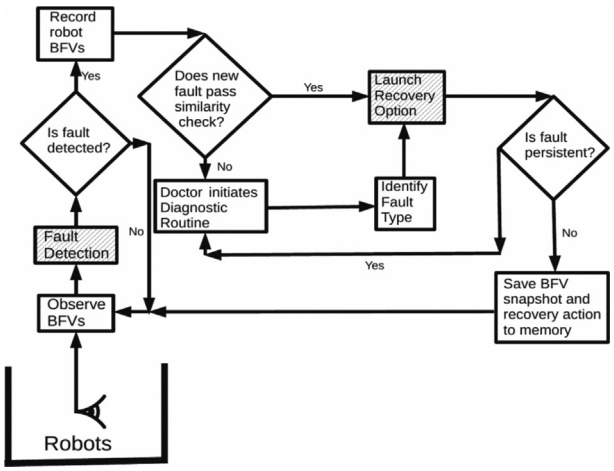

Fig. 2. The fault detection, diagnosis and recovery method used in the work of O'Keeffe et al (Source: O'Keeffe et al., 2017b).

faulty robot to a normal state. The storage of previous detected faults uses a memory buffer so as to keep the most recently successfully recovered faults in the front of the buffer and for the oldest successfully recovered faults to be discarded so as not to continue to take up an increasing amount of memory. Faults that had been previously encountered were able to be diagnosed over 80% of the time with no diagnostic tests required. There were some issues with misdiagnosis due to the similarity value to a previous diagnosis being closer to the threshold value. The threshold for certainty of similarity to a previous diagnosis could be adjusted, however this would be at the cost of an increased requirement to run diagnostic tests for faults under the threshold level of certainty which may have been successfully recovered without this. Changing this threshold value would therefore be a trade-off between the rate of successful re-diagnosis of a previously encountered fault and the need to run diagnostic tests in a larger number of cases.

Overall the work of O'Keeffe et al. is a large advancement in the field of fault diagnosis for swarm robotic systems and provides many avenues for further research to build upon, including how to implement the fault recovery options in a realistic manner, research into optimising feature selection and similarity thresholds for diagnosis, and further work in applying the current methods to a real robotic swarm. Further work could also be conducted to integrate this method with an artificial immune system model such as the crossregulation model discussed in Section 4 or to integrate the memory buffer of previous fault diagnoses with a blockchain-based approach to fault detection (these are discussed in Section 6) in order to increase the reliability of the system.

6. Blockchain-based fault detection

The use of the blockchain in swarm robotics was first suggested by Ferrer (Ferrer, 2016). Strobel *et al* extended this work by examining the possibility of using a blockchain protocol as a distributed way of monitoring the swarm and identifying Byzantine robots (Strobel et al., 2018; Strobel and Dorigo, 2018). Byzantine robots are those which display some form of faulty or malicious behaviour.

The blockchain implementation used in this work is based on Ethereum (Wood, 2018). A blockchain is an append-only immutable data structure where each block has a hash code referencing the block before it. When a transaction is sent between two individuals, a new block may be created. For this block to be verified and added to the chain, a form of proof is required. In their work, Strobel et al. used Proof of Work (PoW). PoW proves that a certain amount of computational power was used to solve a computational puzzle, thereby allowing the block to be added to the chain. This method does mean there is a chance that localized versions of the blockchain may have different transaction records. When this occurs, the chain that is longer (and therefore used more computational power overall in its creation) is taken to be correct once the individuals have broadcast their version of the chain. This means that if a faulty or malicious individual in the system were to try to alter the blockchain, they would need to spend a large portion of the combined computational power used to build the chain to create a whole new chain that overwrites old transactions. This allows the blockchain to be a secure way of storing information in a distributed manner. Though Ethereum was initially created as a cryptocurrency, the ability for it to be used with smart contracts has been added. This allows transactions to act as a decentralized way of calling functions that are part of the contract. Once verified by PoW these calls are made by the individual who verified the transaction. When this version of the blockchain is shared with other individuals, each transaction along the chain is executed again to check that they are all valid.

In their initial experiments (Strobel et al., 2018), Ferrer *et al* compare the classical approach of Valentini et al. (2016) to their blockchain-based approach for reaching swarm-level consensus regarding which colour of tile has a higher occurrence rate on a floor containing only black and white tiles. The classical approach involves two states and the use of a probabilistic finite state machine. The two states are exploration and dissemination. The exploration state lasts for an amount of time determined by a sample from an exponential distribution. During this state, the robot walks randomly while avoiding obstacles and collecting ground sensor readings to identify the colour of the tiles below it. Each individual begins with a random opinion regarding which colour represents the majority of the tiles – 50% begin with the opinion that white is the majority colour and 50% with black. The individual robot updates a quality estimate of its opinion during the exploration phase. This quality estimate is a ratio of the amount of time it detected the colour of its current opinion over the total amount of time spent detecting. At the end of the exploration state, the robot switches to the dissemination state. During this state the robot continues to walk randomly and avoid obstacles while broadcasting its current opinion. During the direct comparison decision-making strategy explained below, the robot also disseminates its current quality estimate. Three decision-making strategies for whether the individual should change its opinion are tested:

1 Voter Model

The individual robot adopts the opinion of a random neighbour.

2 Majority Voting

The individual robot adopts the opinion of the majority of neighbours (including its own opinion).

3 Direct Comparison

The individual adopts the opinion of a random neighbour, but only if that neighbour has higher quality estimate of its opinion.

With the first two strategies, the amount of time an individual robot stays in the dissemination state is selected as a sample of an exponential distribution with the mean value based on its quality estimate. In this way, individuals with higher quality estimates of their opinions disseminate them for a longer period of time. In the third strategy, the amount of time spent in the dissemination state is a sample of an exponential distribution with a standard mean value. During the last three seconds of the dissemination state, the robot receives opinions of other robots. It uses the last two opinions received to decide whether to change its opinion via the strategy used in that experiment. The robot then switches back to the exploration state.

The blockchain-based method is similar, with individuals moving between exploration and dissemination states. Using the Ethereum smart contract protocol, three functions are implemented. The experiment begins after the robots have sent a transaction to call a function which registers their initial opinions (50% initialized with white as their initial opinion and 50% with black) on the Ethereum system and this transaction has been verified via PoW. The experiment then begins with every robot in the exploration state. The exploration state is identical to that used in the classical approach. However, the dissemination state functions quite differently. When using the voter model or majority voting decision-making strategies, a transaction is sent with a call to a *vote* function using the robot's opinion every half second. This leads to the individuals with a higher quality estimate of their opinion creating more *vote* transactions, as they spend more time in the dissemination state

on average. When using the direct comparison decision-making strategy, a single transaction for the *vote* function is created which includes the individual's quality estimate as well as their opinion. During the last three seconds of the dissemination state, the individual connects to the Ethereum processes of nearby members of the swarm and creates a transaction to a function that applies the decision-making strategy to provide the individual with a new opinion. This function is called the *apply strategy* function. It chooses two pseudo-random opinions based on the previous votes stored on the blockchain and then applies the relevant decision-making strategy to these two opinions and the robot's own opinion to decide whether its opinion should change. The individual then has to wait until their transaction is verified via PoW. During this time, the robot walks randomly while avoiding obstacles, continuing to connect to and disconnect from the Ethereum processes of other swarm members depending on proximity while working on verifying other transactions in the system. Once the individual's *apply strategy* function transaction is verified, it then returns to the exploration state.

Three methods are implemented in the blockchain-based method for excluding votes from the blockchain:

1 Outdated Opinion

The individual does not have a verified call to the apply strategy function in the previous 25 blocks. As such, it is likely that their vote is based on outdated information.

2 Vote Limit Exhausted

Each individual is only allowed a maximum of 50 votes in each dissemination state to prevent vote spamming by malicious or faulty individuals.

3 Different Blockchain Versions

If the individual has an entirely different version of the blockchain, their votes are excluded until they obtain the blockchain version on which consensus has been reached.

These strategies allow for the votes of individual robots who may have been away from the swarm for too long or that are faulty to be discarded and not affect the rest of the system.

Byzantine robots are modelled in the experiments as individuals that always vote for black (the minority colour) with a quality estimate of 100%. Experiments were conducted in ARGoS using 20 simulated e-puck robots. Experiments were conducted at varying difficulties, defined by the ratio of white tiles to black tiles (with similar numbers of black and white tiles being difficult and a large amount more white tiles than black being easy). With no Byzantine robots present, the classical approach performs significantly better in terms of both probability of achieving swarm consensus on the correct answer (that is, that more white tiles are present) and in the time taken to achieve this consensus. As the classical method has no way of excluding the Byzantine robots from the consensus process, when Byzantine robots are introduced, sub-swarm consensus is considered. This is achieved when all robots except the Byzantine robots agree on which is the predominant colour. It is also worth noting that with the classical approach, the system will always converge towards deciding that black is the predominant colour even after sub-swarm consensus is reached. Experiments were stopped after sub-swarm consensus was reached for this reason. The blockchain method does exclude the Byzantine robots from its consensus process, and therefore full swarm consensus is still considered for this method. The blockchain method is shown to be significantly better at dealing with Byzantine robots. Though the probability of success is low with high numbers of Byzantine robots, there is still a moderate probability of success. With the classical method, the probability of success drops to single digits with>3–5 Byzantine robots.

Strobel and Dorigo extended this work in (Strobel and Dorigo, 2018) by focusing on improving the blockchain-based shared communication between the swarm and adding a reputation management system for detecting Byzantine robots. Instead of just focusing on which colour of tile is predominant, the swarm tries to reach a consensus within a margin of error regarding the ratio between the two colours. The reputation system works via increasing an individual's reputation if their votes are close to the mean of the votes already stored on the blockchain and decreasing an individual's reputation if they are very different to this. The updated system is found to be highly accurate at estimating the ratio of tiles without the presence of Byzantine robots. The efficacy of the system was then tested with an increasing number of Byzantine robots. The results of these tests are presented in Fig. 3. The system is tested with and without the reputation management system functioning. Without the reputation management system, the mean average error between the swarm estimate of the ratio of black to white tiles and the correct value rises linearly with the number of Byzantine robots in the system. With the reputation management system in place, the mean average error is much lower for a small number of Byzantine robots, however it increases to similar values to the no-reputation-management experiments for large numbers of Byzantine robots.

This work has a lot of potential as both an exogenous decentralized fault detection method and a method of collective communication and decision making. However, it has only been tested in moderately idealised simulations with a limited scope of what malicious or faulty behaviour is tested. Further testing with a wide variety of Byzantine behaviours and a wider variety of swarm behaviours will need to be conducted to prove the full efficacy of this method. Strobel *et al* also propose testing different methods

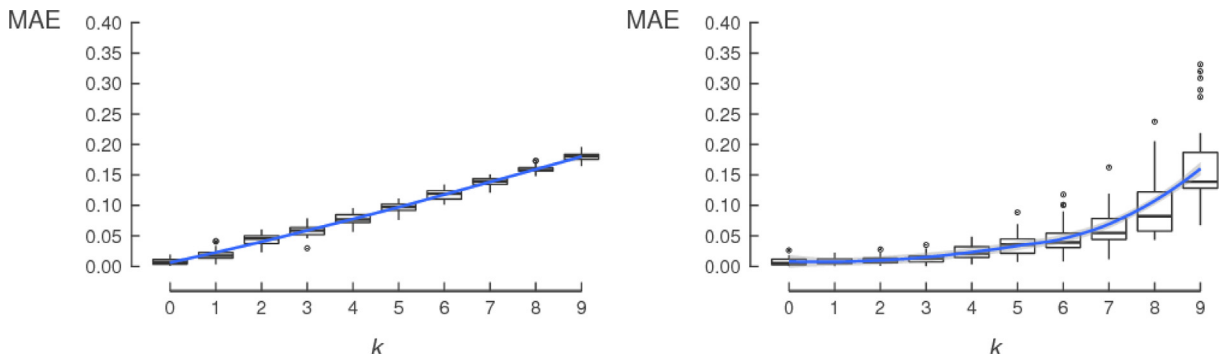

Fig. 3. Mean Average Error (MAE) vs. Number of Byzantine Robots (k) for experiments with no reputation management system and with reputation management system (Source: Strobel and Dorigo, 2018).

of block verification that are more suitable to swarm robotic systems including Proof of Stake (a block is added based on randomized selection and some form of stake such as a high reputation), Proof of Sensing (individuals that can produce a certain sensor output can send and validate transactions) and Proof of Physical Work (the individual must complete a physical task to verify a block).

7. Discussion

Each of the methods presented in this paper has potential advantages and disadvantages. Though these have briefly been discussed in the previous sections they will be expanded upon here and can be used to identify potential areas for further research.

A. Data Model-Based Fault Detection

As discussed in Section 3, this method currently has many serious disadvantages. In its current state, it breaks the swarm robotics paradigm of distributed information by requiring the behaviour of the entire swarm to be analysed at a specific point in time in order to identify potential faults. This requires a central point of data collection and also requires the synchronicity of the entire swarm for data to be collected at the same point in time from every individual. Khaldi *et al* suggest the use of observer robots to collect the data, however this would require the swarm to use more members than necessary to complete a task and these observer robots themselves would not be easily covered by the fault detection method.

As the dimensionality of the data collected from the swarm is purposely reduced for ease of analysis, it is not currently possible to tell which individual in the swarm may be faulty, but only whether there is a potential fault in the swarm. This means that even the most basic form of fault recovery – whereby a faulty robot is taken offline without fault diagnosis – is not possible as the specific faulty individual cannot be identified.

Another major disadvantage of this method is the need for a fault free data set in order to create the model. While getting fault-free data from a simulated swarm is trivial, collecting fault-free data from a real robotic swarm is difficult. There is no simple way of knowing whether the data that has been collected is in fact fault-free, and if it is not then the model created from it will not be an accurate model for comparison of the runtime variables of the swarm. To be certain that the fault-free data being collected is in fact fault-free, another fault detection method would need to be implemented and proper care would need to be taken to ensure that this added fault detection method does not interfere with the data being collected for the PCA model. In collecting the data for the generation of the model, multiple runs could conceivably be required in order to have a perfect fault-free run. This method also has issues with flexibility and scalability, two of the main desirable properties for swarm robotic systems outlined in Section 1. Currently, the method has only been tested using variables that are specific to one swarm behaviour (i.e. circle forming). For the system to be able to exhibit other behaviours, a more generalised set of data for each smarm member would need to be collected to generate the fault-free model.

A similar issue applies to scalability. As the fault-free model is generated from a data set with a specific size (corresponding to a specific swarm size), adding or subtracting members of the swarm would likely render the model unusable. With less swarm members than when the fault-free model was created, the data that the model is expecting from the missing swarm members would have to be dealt with in some way. It might be possible to zero the missing data, however this would likely lead to the model constantly detecting a fault in the system. It could also be possible to simulate the data from the missing swarm members in order to allow the model to operate accurately, however there are two main issues with this. The first relates to the computational power required to use this fault detection method. If simulation of missing swarm members is required, more computational resources will be required to achieve this. If this method is extended to be distributable amongst the swarm, each individual in the swarm will have an extra computational burden for each missing member compared to the number of members that were used to generate the model. The second issue with this relates to swarm behaviours. While simulating the missing individuals would likely be fine for many behaviours, for a more complex behaviour such as the circle forming behaviour explored in the research on this method (where each member of the swarm aims to align themselves on a circle with a maximum distance to each other member of the swarm), the values for the virtual viscoelastic force length and virtual viscoelastic force angle of each individual will necessarily be different than with a larger swarm membership and may again render the initial fault-free model unusable. This issue also applies to having more members in the operating swarm than there were in the fault-free model swarm. While the data from the extra swarm members could be discarded, this would not allow the fault detection to include the extra members and the values collected from the other swarm members may also necessarily be different depending on the behaviour of the swarm. This method will require a large amount of future research as well as novel ideas as to how to apply it more specifically to a real swarm robotic system. It is likely that further work in other areas presented in this paper will yield more practical results.

B. Immune System-Based Fault Detection

The immune system-based fault detection method presented in Section 4 solves many of the issues of the data model-based method. It has been designed to closely follow the swarm robotics paradigm by allowing the swarm to be easily scalable and have the flexibility to complete different tasks and exhibit different behaviours. This method is also fully distributed and has been tested in a partially realistic simulation where individuals only have limited sensing capabilities and all sensor readings contain noise.

It is not clear how this compares to the other methods presented here in terms of the processing power required to implement the method. As each individual is maintaining a CRM at all times, this method is likely to be moderately computationally expensive. As this work uses BFVs, selecting features that prove useful at detecting many different kinds of faults, are applicable for a wide variety of behaviours, and that are simple to measure is imperative to the success of the method. This selection could prove difficult in less idealised contexts with more complex swarm behaviours than those tested. These will also depend on the internal and external sensing capabilities of the swarm robotic platform being used.

Choosing a threshold value at which to detect a fault is also a difficult problem with this method. Setting a higher threshold increases the likelihood of failing to detect a fault and increases the accuracy of any fault detection that is made. Decreasing the threshold value increases the likelihood of detecting a fault but also increases the chance of false-positives. This is a difficult trade-off and would require application-specific fine-tuning.

Further work in comparing the CRM model to other immune system models could prove useful, as well as testing this model on a real swarm robotic system. Further work in extending this method to include fault diagnosis would help to secure this model as one of the best options for use in applications of swarm robotic systems.

C. Local Sensing-Based Fault Diagnosis

The method of fault diagnosis presented in Section 5 details the only method of fault diagnosis being explored in the modern

research in this area. Though the work presented is largely impressive and has high potential for future use, there are some disadvantages that should be considered.

Firstly, the method presented in (O'Keeffe et al., 2017a) requires a model to be created for the run time features of the system to be compared to. As detailed earlier in the discussion of the data model-based fault detection method, this has many serious issues. It should be noted that the issues are somewhat lessened in this work, as once the model is trained, using it only requires exogenous and endogenous BFVs for one individual. The model could therefore be implemented in each individual and it would not need to be centrally managed.

The extended method detailed in (O'Keeffe et al., 2017b) fixes this issue entirely by no longer requiring a model to be trained. Instead, it simply looks for discrepancies between the exogenous and endogenous BFVs recorded for each individual. While this does work moderately well for the categories of faults tested, it is possible to imagine a scenario where there is no discrepancy between the exogenous and endogenous BFVs, but a fault is still present, e.g. if a software hang fault occurs and the number of other individuals present around the faulty individual stays constant. The second main issue with this method is the memory required to store the previous diagnoses. The amount of memory that can be allocated to storing these would be dependent on the swarm robotic platform being used. Without enough memory, useful diagnoses could be lost. Swarm robots are currently fairly simple and are more likely than other robotic systems to have strict memory limitations. This method also suffers from two of the minor issues mentioned for the immune-based fault detection method. These are the feature selection for BFVs and the selection of an appropriate threshold value. This work begins to bridge the gap between fault detection and fault recovery methods. There is still further research to be done in mitigating these issues and in other methods of comparing the BFVs (such as using the CRM method of the immune-based fault detection method). This work has not been tested on a real swarm robotic platform and future work on implementation and testing on this is required.

D. Blockchain-Based Fault Detection

Blockchain-based methods of fault detection are the newest area of research presented in this paper. As such, there are still many issues that need to be explored further. Currently, the main advantage of this method over the others presented is in the use of detecting and isolating malicious robots along with any faulty robots. Blockchains are inherently designed to be very resistant to external attacks and as such could provide a secure method of communication between members of a swarm beyond their use as a fault detection method. The main issue with this method as a method of fault detection is that it does not specifically detect faults, but rather is able to exclude or diminish the opinions of faulty individuals in order to complete the task successfully. While this does greatly increase the robustness of the swarm, it currently provides no avenue for attempting any sort of fault recovery or mitigation. This method has only been tested with one specific swarm behaviour. It would likely be possible to implement this method for other swarm behaviours, but this method is inherently inflexible in its current state as it is tied to task-specific metrics.

Currently, the blockchain-based method requires a large time investment in distributing information among the swarm. This means that each individual of the swarm is spending a significant portion of time broadcasting their views and receiving the views of others while not progressing the task. Any fault detection and diagnosis requires a time commitment, however the time commitment for this method is likely to be the highest of those presented here as every individual in the swarm spends a significant portion of its time failing to progress the task. The use of Proof of Work as the method of block authentication could also prove problematic. Proof of Work relies on the difficulty of performing a higher number of computations than the total computations required to authenticate each block of the chain that is currently constructed. Individual swarm robots are not computationally powerful. As such, an individual or organisation that wishes to hack a swarm using this security measure would likely easily be able to override the previously existing version of the blockchain by solving proof of work problems faster than the swarm is able to.

Overall, the blockchain-based method is still in its infancy and there is a wide array of possible uses for it in the future in terms of increasing the robustness and security of swarm robotic systems. This is the only method presented here that considers security of a system from a fault detection viewpoint.

8. Conclusion and areas for further research

The immune system, data model and blockchain-based fault detection and diagnosis methods presented in this paper represent three significant areas of study, all of which have a large capacity for further research. The research in these methods and in the area of exogenous fault detection and diagnosis as a whole is mostly in its infancy and more testing will be required to prove the robustness of these techniques and to develop new techniques.

It is worth noting that very few of the experiments discussed in this paper were run on real robots, those that were being the work of Christensen *et al* on a firefly-based fault detection method (Christensen et al., 2009) and that of Millard *et al* on simulating a robot's controller (Millard et al., 2014a). All other experiments were run in simulation, with all of the modern research explored being conducted in the ARGoS simulator. There is significant scope for further research in testing these fault detection and diagnosis methods on real swarm robotic systems. It is likely that for this, synchronicity of the system will be required for comparing data accurately, such as that gained by using the firefly fault-detection method (Christensen et al., 2009) or the quorum sensing method from the work of Bechon and Slotine (Bechon and Slotine, 2012).

There is little work currently being completed on exogenous fault diagnosis in robotic swarms beyond that of O'Keeffe et al. (2017a,b) and future research in combining fault diagnosis techniques with fault detection techniques is required. In addition, fault recovery and self-healing methods are needed to create robust, fault-tolerant swarms that can detect, diagnose and recover from faults. Further work will also be required in exploring the efficacy of these fault detection and diagnosis methods and any methods developed in the future in the presence of Byzantine robots and other security threats. These are likely to be an issue in any real-world deployment of swarm robotic systems.

Though robustness is regularly presented as an inherent property of swarm robotic systems, this has been shown to not be the case in many situations (Bjerknes and Winfield, 2013; Winfield and Nembrini, 2006). For real-life applications of swarm robotics, fault detection and diagnosis are key components for the success of the swarm. The work presented here provides interesting avenues for further research while significantly advancing the potential of swarm robotic systems.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Barca, J.C., Sekercioglu, Y.A., 2013. Swarm robotics reviewed. Robotica 31, 345–359. https://doi.org/10.1017/S026357471200032X.

Bechon, P., Slotine, J.-J., 2012. Synchronization and quorum sensing in a swarm of humanoid robots.

Ben-Ari, M., Mondada, F., 2018. Swarm robotics. Elem. Robot. https://doi.org/10.1007/978-3-319-62533-1_15.

Bi, R., 2012. Immune-Inspired Fault Diagnosis for a Robotic System. ProQuest Dissertations Publishing.

Bi, R., Timmis, J., Tyrrell, A., 2010. The Diagnostic dendritic cell algorithm for robotic systems. IEEE Congr. Evol. Comput. https://doi.org/10.1109/CEC.2010.5586499.

Bjerknes, J.D., Winfield, A.F.T., 2013. On fault tolerance and scalability of swarm robotic systems. Distributed Autonomous Robotic Systems. Springer Tracts in Advanced Robotics, Vol. 83. Springer, Berlin, Heidelberg, pp. 431–444.

Bonani, M., Longchamp, V., Magnenat, S., Rétornaz, P., Burnier, D., Roulet, G., Vaussard, F., Bleuler, H., Mondada, F., 2010. The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research IEEE/RS. J. Int. Conf. Intell. Robot. Syst. https://doi.org/10.1109/IROS.2010.5649153.

Carneiro, J., Leon, K., Caramalho, I., van den Dool, C., Gardner, R., Oliveira, V., Bergman, M.-L., Sepúlveda, N., Paixão, T., Faro, J., Demengeot, J., 2007. When three is not a crowd: a Crossregulation model of the dynamics and repertoire selection of regulatory CD4+ T cells. Immunol. Rev. 216, 48–68. https://doi.org/10.1111/j.1600-065X.2007.00487.x.

Carrasco, R.A., Núñez, F., Cipriano, A., 2011. Fault detection and isolation in cooperative mobile robots using multilayer architecture and dynamic observers. Robotica 29, 555–562.

Carrillo, M., Gallardo, I., Ser, J. Del, B, Osaba, E., Sanchez-Cubillo, J., Bilbao, M.N., Gálvez, A., Iglesias, A., 2018. A Bio-inspired Approach for Collaborative Exploration with Mobile Battery Recharging in Swarm Robotics. https://doi.org/10.1007/978-3-319-91641-5_7

Chiang, L.H., Russell, E.L., Braatz, R.D., 2000. Fault Detection and Diagnosis in Industrial Systems. Springer Science & Business Media.

Christensen, A.L., 2008. Fault Detection in Autonomous Robots. Université Libre de Bruxelles.

Christensen, A.L., O'Grady, R., Dorigo, M., 2009. From fireflies to fault-tolerant swarms of robots. IEEE Trans. Evol. Comput. 13, 754–766. https://doi.org/10.1109/TEVC.2009.2017516.

Daigle, M.J., Koutsoukos, X.D., Biswas, G., 2007. Distributed diagnosis in formations of mobile robots. IEEE Trans. Robot. 23, 353–369. https://doi.org/10.1109/TRO.2007.895081.

Dorigo, M., Birattari, M., Brambilla, M., 2014. Swarm robotics. Scholarpedia 9, 1463. https://doi.org/10.4249/scholarpedia.1463.

Dorigo, M., Floreano, D., Gambardella, L.M., Mondada, F., Nolfi, S., Baaboura, T., Birattari, M., Bonani, M., Brambilla, M., Brutschy, A., Burnier, D., Campo, A., Christensen, A.L., Decugniere, A., Di Caro, G., Ducatelle, F., Ferrante, E., Forster, A., Martinez Gonzales, J., Guzzi, J., Longchamp, V., Magnenat, S., Mathews, N., Montes de Oca, M., O'Grady, R., Pinciroli, C., Pini, G., Retornaz, P., Roberts, J., Sperati, V., Stirling, T., Stranieri, A., Stutzle, T., Trianni, V., Tuci, E., Turgut, A.E., Vaussard, F., 2013. Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. IEEE Robot. Autom. Mag. 20, 60–71. https://doi.org/10.1109/MRA.2013.2252996.

Ferrer, E.C., 2016. The blockchain: a new framework for robotic swarm systems. https://doi.org/10.1007/978-3-030-02683-7_77

Gao, R., Yan, R., 2010. Wavelets: Theory and Applications for Manufacturing. doi: 10.1007/978-1-4419-1545-0.

Goel, P., Dedeoglu, G., Roumeliotis, S.I., Sukhatme, G.S., 2000. Fault detection and identification in a mobile robot using multiple model estimation and neural network. Proc. 2000 ICRA. Millenn. Conf. IEEE Int. Conf. Robot. Autom. Symp. Proc. (Cat. No.00CH37065). https://doi.org/10.1109/ROBOT.2000.846370

Gonçalves, P. j S., Torres, P. j D., Alves, C. m O., Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D., Martinoli, A., 2009. The e-puck, a Robot Designed for Education in Engineering.

Harrou, F., Khaldi, B., Sun, Y., Cherif, F., 2018. Monitoring robotic swarm systems under noisy conditions using an effective fault detection strategy. IEEE Sens. J. https://doi.org/10.1109/JSEN.2018.2877183.

Isermann, R., Ballé, P., 1997. Trends in the application of model-based fault detection and diagnosis of technical processes. Control Eng. Pract. https://doi.org/10.1016/S0967-0661(97)00053-1.

Ismail, A.R., Timmis, J., 2010. Towards self-healing swarm robotic systems inspired by granuloma formation. In: 2010 15th IEEE International Conference on Engineering of Complex Computer Systems. IEEE, pp. 313–314. doi: 10.1109/ICECCS.2010.63.

Ismail, A.R., Timmis, J., 2009. Aggregation of Swarms for Fault Tolerance in Swarm Robotics using an Immuno-engineering Approach.

Jacob, C., Steil, S., Bergmann, K., 2006. The swarming body: simulating the decentralized defenses of immunity. In: Bersini, H., Carneiro, J. (Eds.), Artificial Immune Systems. Springer, Berlin, Heidelberg, pp. 52–65.

Jolliffe, I., 2011. Principal component analysis. Int. Encycl. Statist. Sci., 1094–1096 Springer.

Khadidos, A., Crowder, R.M., Chappell, P.H., 2015. Exogenous fault detection and recovery for swarm robotics. IFAC Pap. 48, 2405–2410. https://doi.org/10.1016/j.ifacol.2015.06.448.

Khaldi, B., Cherif, F., 2016. A virtual viscoelastic based aggregation model for self-organization of swarm robots system. In: Alboul, L., Damian, D., Aitken, J.M. (Eds.), Towards Autonomous Robotic Systems. Springer International Publishing, Cham, pp. 202–213.

Khaldi, B., Harrou, F., Cherif, F., Sun, Y., 2017a. Monitoring a robot swarm using a data-driven fault detection approach. Rob. Auton. Syst. 97, 193–203. https://doi.org/10.1016/j.robot.2017.06.002.

Khaldi, B., Harrou, F., Sun, Y., Cherif, F., 2017b. A measurement-based fault detection approach applied to monitor robots swarm, in: 2017 6th International Conference on Systems and Control (ICSC). IEEE, pp. 21–26. https://doi.org/10.1109/ICoSC.2017.7958703

Kutzer, M.D.M., Armand, M., Scheid, D.H., Lin, E., Chirikjian, G.S., 2008. Toward cooperative team-diagnosis in multi-robot systems. Int. J. Rob. Res. 27, 1069–1090. https://doi.org/10.1177/0278364908095700.

Lau, H., Timmis, J., Bate, I., 2011. Collective Self-detection Scheme for Adaptive Error Detection in a Foraging Swarm of Robots. doi: 10.1007/978-3-642-22371-6_23.

Lau, H.K., 2012. Error Detection in Swarm Robotics: A Focus on Adaptivity to Dynamic Environments. University of York.

Millard, A.G., Timmis, J., Winfield, A.F.T., 2014a. Run-time detection of faults in autonomous mobile robots based on the comparison of simulated and real robot behaviour. In: in: 2014 IEEE/RSJ International Conference on Intelligent Robots and System. IEEE, pp. 3720–3725. doi: 10.1109/IROS.2014.6943084.

Millard, A.G., Timmis, J., Winfield, A.F.T., 2014b. Towards Exogenous Fault Detection in Swarm Robotic Systems. Springer, Berlin Heidelberg.

Mondada, F., Pettinaro, G., Guignard, A., Kwee, I., Floreano, D., Deneubourg, J.-L., Nolfi, S., Gambardella, L., Dorigo, M., 2004. Swarm-bot: a new distributed robotic concept. Auton. Robots 17, 193–221. https://doi.org/AURO.0000033972.50769.1c.

Navarro, I., Matía, F., 2013. An introduction to swarm robotics. ISRN Robot. 2013, 1–10. https://doi.org/10.5402/2013/608164.

O'Dowd, P., Studley, M., Winfield, A., 2014. The distributed co-evolution of an on-board simulator and controller for swarm robot behaviours. Evol. Intell. 7, 95–106. https://doi.org/10.1007/s12065-014-0112-8.

O'Keeffe, J., Tarapore, D., Millard, A.G., Timmis, J., 2017a. Towards Fault Diagnosis in Robot Swarms: An Online Behaviour Characterisation Approach. Springer-Verlag.

O'Keeffe, J., Tarapore, D., Millard, A.G., Timmis, J., 2017b. Fault diagnosis in robot swarms: An adaptive online behaviour characterisation approach. Comput. Intell. (SSCI), 2017 IEEE Symp. Ser. https://doi.org/10.1109/SSCI.2017.8280891

Owens, N.D.L., Greensted, A., Timmis, J., Tyrrell, A., 2009. T cell receptor signalling inspired kernel density estimation and anomaly detection. In: Andrews, P.S., Timmis, J., Owens, N.D.L., Aickelin, U., Hart, E., Hone, A., Tyrrell, A.M. (Eds.), Artificial Immune Systems. Springer, Berlin Heidelberg, pp. 122–135.

Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., Birattari, M., Gambardella, L., Dorigo, M., 2012. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. Swarm Intell. 6, 271–295. https://doi.org/10.1007/s11721-012-0072-5.

Şahin, E., 2005. Swarm robotics: from sources of inspiration to domains of application. In: Şahin, E., Spears, W.M. (Eds.), Swarm Robotics. Springer, Berlin Heidelberg, pp. 10–20.

Şahin, E., Winfield, A., 2008. Special issue on swarm robotics. Swarm Intell. 2, 69–72.

Strobel, V., Castelló Ferrer, E., Dorigo, M., 2018. Managing Byzantine Robots via Blockchain Technology in a Swarm Robotics Collective Decision Making Scenario, in: Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '18. International Foundation for Autonomous Agents and Multiagent Systems, pp. 541–549.

Strobel, V., Dorigo, M., 2018. Blockchain technology for robot swarms: A shared knowledge and reputation management system for collective estimation. Tech. Rep. Ser.

Tarapore, D., Christensen, A.L., Timmis, J., 2017. Generic, scalable and decentralized fault detection for robot swarms. PLoS One 12,. https://doi.org/10.1371/journal.pone.0182058 e0182058.

Tarapore, D., Lima, P.U., Carneiro, J., Christensen, A.L., 2015. To err is robotic, to tolerate immunological: fault detection in multirobot systems. Bioinspir. Biomim. 10, 16014. https://doi.org/10.1088/1748-3190/10/1/016014.

Timmis, J., Andrews, P., Hart, E., 2010. On artificial immune systems and swarm intelligence. Swarm Intell. 4, 247–273. https://doi.org/10.1007/s11721-010-0045-5.

Valentini, G., Brambilla, D., Hamann, H., Dorigo, M., 2016. Collective Perception of Environmental Features in a Robot Swarm. https://doi.org/10.1007/978-3-319-44427-7

Winfield, A.F.T., Nembrini, J., 2006. Safety in numbers: fault-tolerance in robot swarms. Int. J. Model. Identif. Control 1, 30–37. https://doi.org/10.1504/IJMIC.2006.008645.

Wood, G., 2018. Ethereum: A secure decentralised generalised transaction ledger – Byzantium Version 2d0661f.