Immune-Inspired Error Detection for Multiple Faulty Robots in Swarm Robotics

HuiKeng Lau[3] and Iain Bate[1,4] and Jon Timmis[1,2]

[1]Department of Computer Science, University of York, UK,
[2]Department of Electronics, Univesity of York, UK,
[3]Applied Computing Group, SKTM, UMS, Malaysia,
[4]Mälardalen Real-Time Research Centre, Mälardalen University, Västerås Sweden
hklau@ums.edu.my, {iain.bate, jtimmis}@york.ac.uk

Abstract

Error detection and recovery are important issues in swarm robotics research, as they are a means by which fault tolerance can be achieved. Our previous work has looked at error detection for single failures in a swarm robotics scenario with the Receptor Density Algorithm. Three modes of failure to the wheels of individual robots was investigated and comparable performance to other statistical methods was achieved. In this paper, we investigate the potential of extending this approach to a robot swarm with multiple faulty robots. Two experiements have been conducted: a swarm of ten robots with 1 to 8 faulty robots, and a swarm of 10 to 20 robots with varying number of faulty robots. Results from the experiments showed that the proposed approach is able to detect errors in multiple faulty robots. The results also suggest the need to further investigate other aspects of the robot swarm that can potentially affect the performance of detection such as the communication range.

Introduction

A robot swarm is robust to failure of individuals has always been a dominant view in swarm robotics research (e.g. (Bayindir and Şahin, 2007; Şahin et al., 2008)). It is expected that when an individual robot fails, the task left behind by the failed robot will be taken over by other robots in the swarm, and thus the swarm is robust. This view of a robust swarm has two underlying assumptions: the number of fault-free robots is significantly greater than the number of faulty robots, and that the failed robots do not interfere with other robots with respect to the operation of the swarm.

However, studies have demonstrated that the assumption that failed robots do not interfere does not hold for all modes of failure (Winfield and Nembrini, 2006). For a partly failed robot in which only some components are faulty whilst other components are still operational, the failed robot *can* and *will* interfere with the operation of the swarm. For example, in a swarm taxis scenario (swarm moving toward a beacon) investigated in Winfield and Nembrini (2006), a fault to the wheels while other components (e.g. wireless communication) are still operational causes physical anchoring of the robot swarm. Therefore, for such cases, there is a need to handle these failures explicitly. One approach that is applicable for such cases is with explicit error detection and recovery. This approach consists of three stages: error detection, fault diagnosis, and recovery.

Error detection is a crucial first step as the activation of subsequent stages only occur when an error is detected. Previous studies on error detection in swarm robotics have looked at this problem for the case of a single faulty robot in the swarm. However, there is little work that directly addresses error detection when there are multiple (simultaneous) faulty robots in a swarm. In Christensen et al. (2009), the detection of faulty robots occur at the system-level for sensor faults that can be visibly detected by other robots. In Li and Parker (2009), fault detection is investigated for tightly-coupled multi-robot teams. In this paper, we investigate multiple faulty robots in the context of a foraging swarm robotic system in which the ability to forage for each robot can be affected by faults as well as the conditions in the operational environment.

Results from earlier work (Lau et al., 2011b,a) for a single faulty robot have demonstrated the potential of adaptive error detection with the collective self-detection (CoDe) scheme. In the CoDe scheme, a robot determines whether itself is faulty by cross-reference its behaviour with other robots within a logically defined neighbourhood. Each robot communicates (broadcast) its data to other robots in the same neighbourhood. The neighbourhood is defined by the communication range. If there are multiple failures in the same neighbourhood, the detection of faulty robots is harder and the CoDe scheme might work less effectively. This is because if there are more faulty robots in a neighbourhood, the CoDe scheme (which is analogous to a majority voting scheme) might (mis-)detect the fault-free robots as faulty, and vice versa. This is unwanted as the likelihood of multiple failures in a large swarm can be high (Carlson et al., 2004). Luckily, since the robots are mobile, the likelihood of a faulty robot in a neighbourhood of more faulty robots can be low. Besides, if the errors can be detected and recovered early, it can also reduces the likelihood of having multiple faulty robots in a neighbourhood.

The main contributions of this paper, therefore, are (1) an extended investigation on an immune-inspied CoDe scheme for multiple faulty robots in a swarm, (2) presentation on the calculation of the performance of detection for the case of multiple faulty robots, (3) an investigation on the correlation between the number of faulty robots and the required swarm size for effective error detection, and (4) the identification of the robot's communication range as a potential influencing factor on the performance of detection.

This paper is structured as follows. Section present the state-of-the-art on error detection in swarm robotics and the motivation of this paper. Section presents details on the set of experiments, and the experimental setup to investigate error detection for the case of multiple simultaneous faulty robots. Results from the experiments are presented in Section whilst Section concludes with the findings from the experiments in this paper.

Background

There are many approaches for detecting errorss in faulty robots. Generally, they can be grouped into model-driven and data-driven approaches. In model-driven approaches, analytical models of how a robot should behave are build and the actual behaviour is then compared to the predicted behaviour of the models. A problem with model-driven approaches is that the development of accurate models is often difficult, if not impossible, especially if the operational environment is not static or controlled (Christensen et al., 2007a). Due to the interactions between robots and the environment, as well as other natural factors, the state of the environment can change and this in turn can affect the behaviour of the robots.

Alternatively, a data-driven approach uses data produced during normal operation as the basis to infer the presence of a fault. This eliminates the need for precise analytical models of the robot's behaviour. In addition, it is possible to deploy the same robot swarm in many different environments. Therefore, data-driven approaches are genereally more preferred.

Previous studies on data-driven error detection in swarm robotics have investigated scenarios of a single faulty robot in the swarm (e.g. (Canham et al., 2003; Christensen et al., 2007a,b; Lau et al., 2011b; Mokhtar et al., 2009)). However, having only one faulty robot in a robot swarm is a best-case scenario because the likelihood of multiple robots failing is high due to a variety of circumstances.

From Single to Multiple Faulty Robots

The reason for the previous focus on a single faulty robot was that the proposed solution should scale to failure on multiple robots, as the detection is only based on data from one robot (e.g. (Canham et al., 2003; Christensen et al., 2007a; Mokhtar et al., 2009)). In other words, it is assumed that the changes in the behaviour of the robots are only caused by faulty components. The environment in which the robot swarm operates has no impact on the behaviour of the robots. In this case, even with multiple faulty robots in the swarm, the detection of errors on each individual robot remained the same and unaffected by the number of faulty robots.

However, for many scenarios especially when the robot swarms are deployed in real-world environment, the operational environment does affect the behaviour of the robots. An example would be a robot foraging scenario in which the performance of foraging of each robot can not only be affected by the presence of faults but also by the amount of objects in the arena or the condition of the terrain. Therefore, instead of using data from a single robot, the CoDe scheme (Lau et al., 2011b,a) utilises data from a collective. The collective is defined over a logical neighbourhood based on the communication radius of an observer robot. In the CoDe scheme, the presence of a error is determined by cross-referencing a robot's behaviour with other robots within a neighbourhood. Results from the studies show that with the CoDe scheme, an adaptive error detection in the presence of time-varying environmental changes can be achieved.

As mentioned earlier, previous studies have only looked at error detection for a single faulty robot in the swarm. In practise, the probability of having multiple faulty robots in a swarm can be high. A survey on mobile robot failures in Carlson et al. (2004) found that the mean time between failures across all robot types surveyed is twenty four hours and the availability was fifty four percent. Indeed, the frequency of failure is very high.

To detect multiple faulty robots in a swarm is a challenging problem, in particular for scenarios in which the behaviour of the robots is affected by the operational environment as well as the presence of faults. First, detection approaches that operate on the basis of a single robot may not be applicable as changes in the environment that affect the behaviour may be detected as errors. This may lead to false positives. In Canham et al. (2003); Christensen et al. (2007a); Mokhtar et al. (2009), the robots are trained with a set of behaviour that is considered fault-free and thus the learning is static. During operation, changes in the environment that can affect the behaviour of the robots are not anticipated, and thus likely to cause the environmental changes to be detected as faults. Second, the assumption that there are more fault-free robots compared to faulty robots as employed in the CoDe scheme, might not be true for all scenarios. This is particularly in scenarios in which there are more faulty robots compared to fault-free robots in the neighbourhood. This leads to false negatives.

However, the fact that the robots are mobile minimises the frequency of such scenarios. The membership of robots in a logical neighbourhood is dynamic as the robots moves about in the arena. In addition, if errors can be detected early, the likelihood of having a neighbourhood with more

faulty robots can also be minimised. Therefore, this paper aims to investigate the following research questions

- Can the CoDe scheme be extended to multiple faulty robots?
- Is there any correlation between the swarm size and the number of faulty robots that can be detected with the CoDe scheme?

To investigate these research questions, the next two sections will provide details on the CoDe scheme, the evaluation metrics, and the set of experiments conducted.

The Detection Framework

The Detection Scheme The detection of errors in faulty robots in this paper is based on the CoDe scheme proposed in Lau et al. (2011b,a). This scheme is analogous to a majority voting scheme (majority wins) which is the basis for social comparison when objective, non-social means are available (Festinger, 1954). However, the CoDe scheme is designed and implemented for self-detection of errors. Self-detection here means that instead of identifying whether other robots are faulty, it detects whether itself is faulty. This is inspired by the observed behaviour of self-isolation, to die alone, in ants. Instead of being actively located and isolated by healthy members, some species of ants infected by parasites tends to isolate themselves to die (Heinze and Walter, 2010). Taking this approach means that, at this stage, the detection can be more robust as it does not requires the identification of other robots in the swarm, as well as storing and keeping record of previous encounters with other robots. If the identification of faulty robots is required, it can be implemented on top of self-detection as proposed in Christensen et al. (2009). The pseudocode for the CoDe scheme is presented in Algorithm 1.

Algorithm 1: Collective Self-Detection Scheme (CoDe)

Input: current data instance v, data instances from neighbours $\mathcal{DN}$, classifier $\mathcal{A}$
Output: report error
foreach *control cycle* t **do**
  if *CalculateNeighbour*($\mathcal{DN}$) < 2 **then**
    err = $\mathcal{A}(v, \mathcal{DN}_{temp})$;
    //$\mathcal{DN}_{temp}$ *is the data from previous control cycle having more than 2 neighbours*
  else
    err = $\mathcal{A}(v, \mathcal{DN})$;
    $\mathcal{DN}_{temp}$=$\mathcal{DN}$;
    //assign current data from neighbours to $\mathcal{DN}_{temp}$
  end
  if err **then**
    Report err;
  end
end

The Classifier The classifier used in this paper is the Receptor Density Algorithm (RDA) Owens et al. (2009) inspired by the T-cell receptor signalling mechanism in the immune system. By extraction of certain features of the generalised T -cell receptor, it was then mapped onto kernel density estimation. The RDA works as follow. The spectrum of input data is divided into s discretised location and a *receptor* $\mathbf{x}_s$ is placed at each of these locations. The input data is the variable values from the robots used for the detection. A receptor has a length $\ell = (\sqrt{2\pi})^{-1}$, a position $r_p \in [0, \ell]$, and a negative feedback barrier $\beta \in (0, \ell)$. At each control cycle step t, each receptor takes input $\mathbf{x}_i$ and performs a binary classification $c_t \in$ 0,1 to determine whether that location is considered anomalous. In general, the observation of one anomalous location is sufficiently representative to indicate the present of an anomaly at t (Owens et al., 2009).

The classification decision is determined by the dynamics of r_p and negative feedback $r_n \in (0, \ell)$. During training or initialisation, after $\mathbf{x}_i$ was presented, if the resulting $r_p \geq \beta$ then a negative feedback r_n is generated which acts to reverse the progression of r_p. If $r_p < \beta$, no negative feedback will be generated, $r_n = 0$.

$$r_p(\mathbf{x}) = \sum_{i=1}^{n} \frac{1}{nh} K(\frac{\mathbf{x} - \mathbf{x}_i}{h}) \tag{1}$$

$$r_n(\mathbf{x}) = \begin{cases} r_p(\mathbf{x}) - \beta, & \text{if } r_p(\mathbf{x}) \geq \beta \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The receptor position and negative feedback decay over time. During testing, for a new data instance $\mathbf{v}$ if $r_p^t > \ell$, then the receptor generates an anomaly classification c_t=1.

$$r_p^t(\mathbf{x}) = b \times r_p^{t-1}(\mathbf{x}) + gb \times K(\frac{\mathbf{x} - \mathbf{v}}{h}) - a \times r_n^{t-1}(\mathbf{x}) \tag{3}$$

where $b \in \mathbb{R}^+$ is receptor position's decay rate,
$gb \in \mathbb{R}^+$ is current input stimulation rate,
$a \in \mathbb{R}^+$ is negative feedback's decay rate.

$$c(\mathbf{v}) = \begin{cases} 1, & \text{if } r_p^t(\mathbf{x}) \geq \ell \\ 0, & otherwise \end{cases} \tag{4}$$

For the experiments in this paper, the K(x) in Eq. 1 and Eq. 3 is Gaussian kernel, the β = 0.01, b = 0.02, gb = 1.1, and a = 1.7.

The Experiments

Two experiments are conducted: 1) an investigation of the potential of detecting errors in a swarm of multiple faulty robots with the CoDe scheme; 2) an investigation on the

correlation between the size of the swarm and the ability to detect errors.

Fixed Swarm Size This experiment investigates the robustness in detecting errors for multiple faulty robots with the CoDe scheme. The aim is to find out whether the approach that has been demonstrated to work well with a single faulty robot Lau et al. (2011b,a) can also be applied to the case of multiple faulty robots. More importantly, if the approach works with multiple failures then how will it degrade as the the number of failures increases.

In this experiment, faults are injected to robots in the system at 2500s (which is at control cycle 10). However, the fault models, number of faulty robots, and the duration for each fault are randomly generated. With a random number of faulty robots and fault durations, the number of faulty robots in a single simulation run can vary from one control cycle to another. For this reason, section describes how the performance is to be evaluated.

Variable Swarm Size This experiment investigates the possible correlation between the swarm size and the number of faulty robots that can be detected. This relates to the reliability of the error detection. The aim is to find a correlation $n = ak + c$, if exists, such that for k faulty robots there needs at least n robots in the swarm to ensure that the errors can be detected reliably with a true positive rate greater than or equals to x, a false positive rate of less than or equals to y, or both. For hardware redundancy, the number of redundant components generally suggested is $2k$ + 1 (Abd-El-Barr, 2006).

The configuration for the time of faults injection, number of faulty robots, duration for each fault in this experiment is the same as previous experiment. However, the swarm size is increased gradually starting from 10 robots. For every successive increment, an additional of two robot will be added.

The Evaluation Metrics

The performance of detection is evaluated based on the performance is based on the true positive rate, false positive rate, and the (`Latency`), as in Lau et al. (2011a). Given:

- True Positive (`TP`) - an error is correctly classified;
- False Positive (`FP`) - a normal instance is incorrectly classified as an error;
- True Negative (`TN`) - a normal instance is correctly classified as normal; and
- False Negative (`FN`) - an error is incorrectly classified as a normal instance.

The True Positive Rate (`TPR`) is the proportion of the number of correctly classified errors over the total number of errorneous instances (Eq. 5).

$$\texttt{TPR} = \frac{\texttt{TP}}{\texttt{TP} + \texttt{FN}} \tag{5}$$

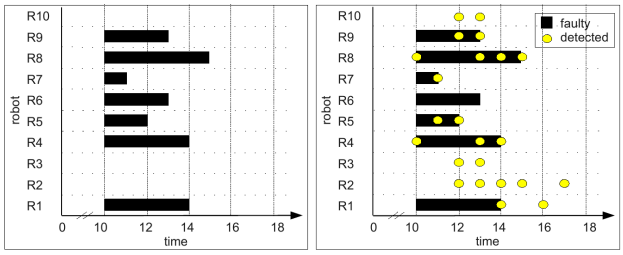
Similarly, the False Positive Rate (`FPR`) is the proportion of the number of incorrectly classified errors over the total number of normal instances (Eq. 6).

$$\texttt{FPR} = \frac{\texttt{FP}}{\texttt{FP} + \texttt{TN}} \tag{6}$$

The (`Latency`) metric evaluates how long the time has elapsed before an error is positively identified (Eq. 7). Given that $\texttt{t}_{\texttt{pd}}$ is the fault detection time, and $\texttt{t}_{\texttt{ft}}$ is the fault injection time, then

$$\texttt{Latency} = \texttt{t}_{\texttt{pd}} - \texttt{t}_{\texttt{ft}} \tag{7}$$

We present how `TPR` and `FPR` can be calculated for multiple faulty robots with reference to Figure 1. In the figure, there are ten robots in the system, labelled R1 to R10. In Figure 1(a), seven robots are faulty from time t=10 onwards. However, the durations of faults between the robots are different as indicated by the black-coloured bar. For example, the fault in R1 lasts for 4 control cycles, from t=10 to t=14. Since the durations faults are different, the number of faulty robots at each control cycle also differs. From t=10 to t=11, there are seven faulty robots whereas from t=14 to t=15 there is only one faulty robot. By analysing the simulation data in this way, the `TPR` and `FPR` for each number of faulty robots can be calculated. For the scenario in Figure 1(a), there are instances for seven, six, five, three, and one faulty robot(s).



(a) fault injection (b) error detection

Figure 1: An illustration to show the calculation of `TPR` and `FPR` given information regarding the fault injection time, duration of fault, and detected errors.

In Figure 1(b), the circles represent the instances of error being detected. It can be seen that there are many instances of false positive (detection of error even when no fault was injected), e.g.with robots R2, R3, and R10. To calculate the `TPR` and the `FPR`, starts at t=11. At t=11, there are seven faulty robots. Therefore, the `TPR` for the case of seven faulty robots is 2/7. Similarly, the `FPR` is 0/3. At t=12, the `TPR` for six faulty robots is 2/6 whilst the `FPR` is 3/4, and so on.

Experimental Setup

The experiments in this paper were carried out in simulation. The context of the work is a robot swarm in a foraging scenario. The source code for the foraging swarm robotic system, data, and scripts used to produce results for this paper are available at http://sites.google.com/site/huikenglau/shared.

Simulation Settings

The software used to implement the foraging robot swarm is the Stage plug-in (Gerkey et al., 2003). The robot swarm is placed in an arena to continuously locate, transport, and deposit objects until the end of simulation. At any time, a random number of robots can fail.

Arena The arena is an octagonal-shaped area of 12m × 12m with a circular base of 3m in diameter in the centre. Objects are placed at random[1] locations but outside of the base at a default object-replenishing-rate (OPR) of 0.10. This means that the probability of adding an object at every second is 0.10.

Robot The physical robot from which the simulation model was based is the Linuxbot from Bristol Robotics Laboratory [2]. Each robot is equipped with an array of sensors and components needed for foraging. The default moving speed of the robots is 0.15 m.s^{-1} with a communication range of 2m.

Fault Each robot in the swarm is subject to a particular fault in the wheels whilst other components are still functioning, as examined in Winfield and Nembrini (2006). Three models of faulty wheels were simulated: complete $\mathtt{P}_{\mathtt{CP}}$, partial $\mathtt{P}_{\mathtt{PT}}$, and gradual $\mathtt{P}_{\mathtt{GR}}$. The $\mathtt{P}_{\mathtt{CP}}$ causes the wheels of a robot to stop responding completely and thus the robot is unable to proceed with foraging. With $\mathtt{P}_{\mathtt{PT}}$, the robot moves with a reduced speed and thus resulting in less objects being collected when compared to a fault-free condition. With $\mathtt{P}_{\mathtt{GR}}$, the robot moves with a gradually reducing speed until eventually it comes to a complete stop. In simulation, $\mathtt{P}_{\mathtt{CP}}$ is simulated by setting the left wheel to left turn by 10° causing the robot to move in circle. For $\mathtt{P}_{\mathtt{PT}}$, the robots move with a reduced speed of 0.45 $\times 10^{-1}$ms^{-1} whilst for the $\mathtt{P}_{\mathtt{GR}}$ the speed of the robot is reduced by 0.10 $\times 10^{-3}$ms^{-2}. The mode of the fault is transient; the fault lasts for a random period of time and then the robot recovers and continue with normal operation.

Environment Two different scenarios in which the robot swarm operates: constant OPR (CST), varying OPR ($\mathtt{V}_{\mathtt{OPR}}$). In a CST scenario, the OPR is fixed at 0.10. On the other hand, in a $\mathtt{V}_{\mathtt{OPR}}$ scenario, the OPR alternates between 0.10 to 0.01 at different intervals.

A Simulation Run A simulation starts with 100 initial objects placed randomly in the arena. A maximum number of objects in the arena at any one time is capped at 400 units to avoid overcrowding. Each object is a small red coloured square box which can be sensed by the camera on each robot and picked up by the grippers on the robot. Robots depart from the base and the heading for each robot R_i is based on the formula $R_i = \frac{i \times 2\pi}{n}$, n is the number of robots in the swarm. The robots will continuously carry out foraging until the end of the simulation. In this paper, each simulation lasts for 10,000s. Periodically (i.e. every control cycle, 250s), data on the number of objects collected obj, energy used eng, and distance travelled dist for each robot are extracted and output as csv files. For each variable, an instance of the CoDe scheme is executed separately and an error is considered detected if it is reported in at least one of the variables. The h in Eq. 1 for obj = 1.0, eng = 12.0, and dist = 2.5

Experimental Results

Fixed Swarm Size

Figure 2 is the result for the TPR and FPR in detecting errors for different number of faulty robots in a CST scenario. Each point in the graphs represent the TPR or FPR calculated over 100 repeated runs. Note that since the number of faulty robots and the duration of each fault are random, the number of instances for each group of faulty robots also differs. For example, out of the 100 runs, there are 1208 instances of eight faulty robots, 777 instances for seven faulty robots, 948 instances of six faulty robots and so on.

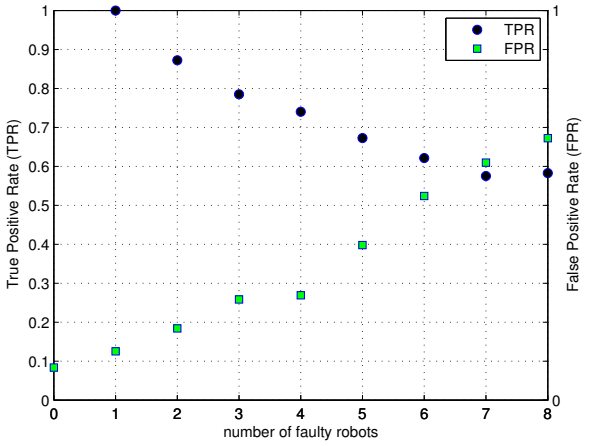


Figure 2: The TPR and the FPR for detecting multiple faulty robots in a CST scenario with the CoDe scheme.

In Figure 2, the results show that as the number of faulty

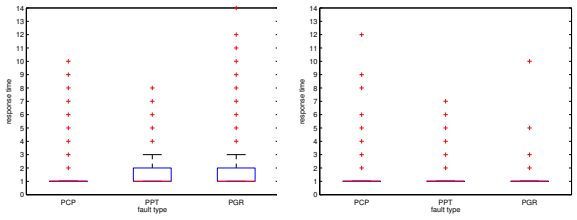
[1]The random number generator used is from GSL-GNU Scientific Library.

[2]http://www.ias.uwe.ac.uk/Robots/linuxbot.htm

robots increases, the performance of detection decreases. Note that in this experiment, no recovery is included. Therefore, a TPR of 1.00 with one faulty robot means that the errors in that faulty robot can always be detected. Similarly, a TPR of 0.55 with eight faulty robots means that there is 55% chance that the errors in all eight faulty robots will be detected. This is possible because there are still two fault-free robots for the faulty robots to cross-referencing their data. In addition, because of the dynamic neighbourhood of the robots from one control cycle to another, a faulty robot might also be cross-referencing its data against others from the previous control cycle (refer line 5 in Algorithm 1).

The results show that with every increase of one faulty robot, the TPR decreases at an approximately constant rate of 0.10. This means the probability of detecting errors in all faulty robots decreases as the number of fault-free robots decreases. This is expected because the likelihood of more than one faulty robot in the logical neighbourhood is increased, and thus resulting in more false negatives. Similarly, the increase of the number of faulty robots also increases the FPR. The increase in the FPR is also approximately 0.10 for each addition of one faulty robot. Due to the same reason for the TPR, each addition of faulty robot increases the likelihood of fault-free robots to classify itself as faulty (false positives).

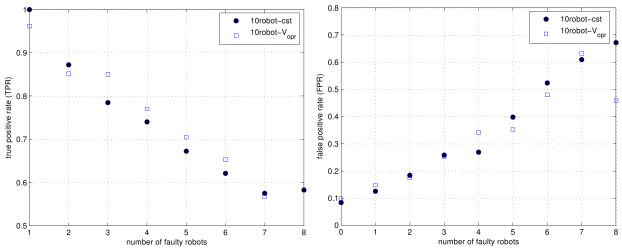
The result for the Latency in detecting the errors is shown in Figure 3. From the 100 runs, the median Latency is 1 control cycle. This means that the errors are detected in the next control cycle after faults were injected. This is a positive result because if recovery measures were implemented, the number of multiple simultaneous faulty robots can be reduced.



(a) CST (b) V_{OPR}

Figure 3: The Latency in detecting the models of fault of the wheels in a CST and a V_{OPR} scenario. The '+' points on the boxplots are the outliers.

The graphs in Figure 4 compares the performance between a CST and a V_{OPR} scenario. Overall, there is no significant difference between the performance in the TPR and the FPR. The drop in the TPR and the the increase in the FPR as the number of fault robots increases are similar to the results for a CST scenario. In fact, some of the results for the V_{OPR} scenario are better than the CST scenario. This is because even with a fixed OPR, the presence of multiple faulty robots can significantly affect the ability to positively identify errors. Nevertheless, this result shows that CoDe scheme works well in a non-dynamic as well as a dynamic environment even with multiple faulty robots in the swarm. This is encouraging as it further supports that the CoDe scheme can be adaptive to dynamic environments.



(a) TPR (b) FPR

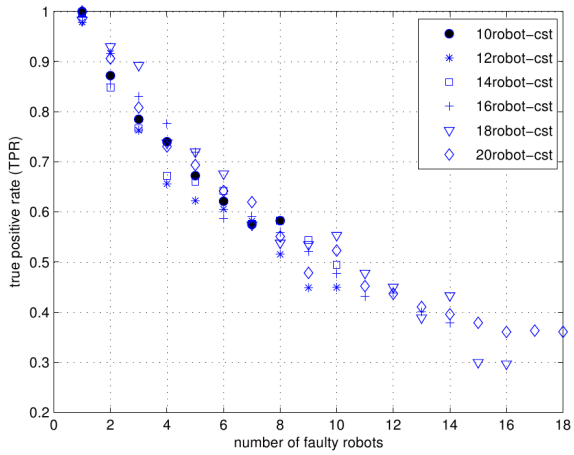
Figure 4: The TPR and FPR in detecting errors for a robot swarm with multiple faulty robots in a CST and a V_{OPR} scenario.

Variable Swarm Size

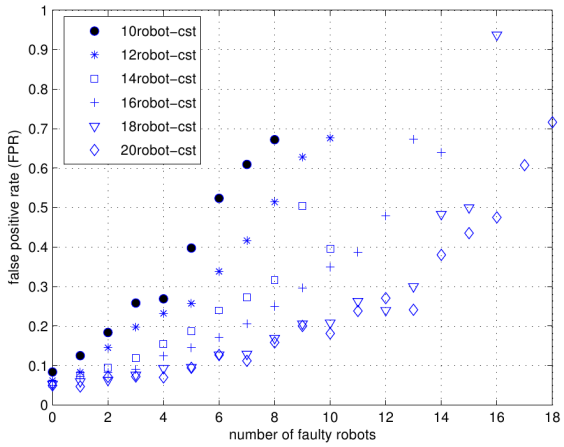
Figure 5 show the TPR and FPR for multiple faulty robots with different swarm sizes. A general observation is that as the swarm size increases, the performance of detection also improves. For example, on the x-axis with two faulty robots in Figure 5(a), as the swarm size is increased from 10 robots to 18 robots, the TPR also increases (from about 0.85 to slightly above 0.90). Similarly in Figure 5(b) with two faulty robots, as the swarm size increases from 10 to 18 robots the FPR decreases from about 0.20 to less than 0.10.

However, note that the increase in the TPR does not occur in all cases. In some cases, rather counter-intuitive. For example with eight faulty robots in Figure 5(a), the TPR decreases from slightly below 0.60 to only above 0.50 when the swarm size increases. This observation is interesting and worth further investigation. One particular factor comes to mind is the communication range of each robot. This parameter influences the size of the logical neighbourhood. Here, it is set to 2m radius. From the results, it appears that an increase in the swarm size does not guarantee an increase in the neighbourhood size (i.e. the number of robots in the neighbourhood) at each control cycle. Therefore, this aspect will be investigated in the near future.

From this result of varying swarm size, the required swarm size for different number of faulty robots can be calculated. For example, in order to not falsely detect errors at 80% of the time in a swarm with four or less faulty robots (i.e. FPR = 0.80) the swarm needs to have at least 12 robots. Similarly, with six or less faulty robots, a swarm of at least 16 robots is needed. From this trend, it seems that a swarm of $n = k + 10$ is required to achieve a FPR less than 0.20 with k faulty robots.



(a) TPR



(b) FPR

Figure 5: The TPR and FPR in detecting errors for different swarm sizes.

Based on the same analysis, to be able to detect four faulty robots or less at 80% of the time (i.e. TPR = 0.80), the swarm needs to have more than 20 robots. Unlike the TPR, based on current results, it is impossible and unrealistic to extrapolate the required swarm size with more than four faulty robots.

A general observation is that the swarm size required is greater than $2k + 1$. This is comparable to the generally used hardware redundancy Abd-El-Barr (2006). The reason is that in swarm robotics the robots are mobile and there is no guarantee that for k faulty robots, there will be at least $2k + 1$ fault-free robots in the same logical neighbourhood. This observation hints that there are other factors involved and one particular parameter that came to mind is the communication range of the robots. For confirmation, this parameter will be investigated in the near future.

Comparison with Q-test

The performance of error detection with multiple faulty robots using the CoDe scheme with the RDA is compared with the Q-test (Gibbons, 1994) (Table 1, Table 2). Dixon's Q test (Gibbons, 1994), or simply the Q-test, is a non-parametric technique that can be used for error detection. It has been applied for error detection in the case of a single faulty robot in Lau et al. (2011a) and shown to produced the best results when compared to other statistical classifiers such as T-test, Quartile-based, and Extreme Studentised Deviate.

In Table 1, the RDA has consistently achieved a higher TPR when compared to the Q-test from all swarm sizes (from 10 to 20). However, a similar result is not obtained for the FPR (Table 2). When the number of faulty robots increases the FPR for the RDA increases. From the perspective of the CoDe scheme (i.e. majority voting), this is expected, in particular when the number of fault-free robots is significantly less than the number of faulty robots. Having said that, a more detailed investigation will be conducted in the near future.

Table 1: The difference of the TPR of the RDA and the Q-test (i.e. RDA-Q-test) in detecting errors with multiple faulty robots. Note that for the TPR, a positive value means a better result.

No. Faulty robots	10	12	14	16	18	20
1	0.18	0.13	0.15	0.15	0.09	0.14
2	0.27	0.18	0.26	0.19	0.23	0.15
3	0.28	0.25	0.31	0.26	0.22	0.25
4	0.33	0.30	0.32	0.29	0.27	0.22
5	0.36	0.28	0.34	0.32	0.27	0.28
6	0.33	0.28	0.30	0.26	0.30	0.27
7	0.21	0.28	0.30	0.31	0.21	0.31
8	0.20	0.27	0.30	0.29	0.27	0.30
9		0.22	0.26	0.29	0.27	0.28
10		0.12	0.26	0.27	0.26	0.28
11			0.22	0.22	0.27	0.24
12			0.13	0.21	0.22	0.23
13				0.19	0.19	0.22
14				0.12	0.20	0.20
15					0.12	0.19
16					0.02	0.18
17						0.15
18						0.09

Conclusion

We have presented our initial investigation on error detection for multiple faulty robots in the swarm. Specifically, we looked at scenarios in which the behaviour of the robots can be affected by both faulty components and changes in the environment. In addition, the way to calculate the performance metrics, namely the true positive rate, false positive rate, and latency for the case of multiple faulty robots are also presented. Revisiting the research questions, results from the first experiment give evidence that the CoDe scheme, which work for a single faulty robot, performs well for multiple faulty robots. In the second experiment, the general results show that as the swarm size is increased, the performance in detecting errors with multiple faulty robots also increases. In particular, the size of the swarm needs to be greater than

Table 2: The difference of the FPR of the RDA and the Q-test (e.e RDA-Q-test)in detecting errors with multiple faulty robots. Note that for the FPR, a negative value means a better result.

No. Fauty Robots	10	12	14	16	18	20
1	-0.04	-0.08	-0.09	-0.10	-0.09	-0.08
2	0.01	-0.01	-0.02	-0.07	-0.08	-0.07
3	0.07	0.04	-0.05	-0.06	-0.07	-0.06
4	0.07	0.05	-0.01	-0.02	-0.07	-0.07
5	0.16	0.07	0.03	0.01	-0.06	-0.06
6	0.19	0.08	0.02	-0.03	-0.04	-0.01
7	0.27	0.09	0.05	0.00	-0.07	-0.03
8	0.30	0.16	0.13	0.04	-0.02	0.02
9		0.17	0.11	0.01	-0.02	0.02
10		0.27	0.14	0.08	0.00	0.01
11			0.24	0.06	0.03	0.03
12			0.26	0.11	0.00	0.04
13				0.19	-0.08	-0.02
14				0.24	0.13	0.08
15					0.25	0.10
16					0.38	0.11
17						0.15
18						0.31

$2k + 1$ where k is the number of faulty robots. The results also suggest the need for further investigation on the correlations between swarm size, communication radius, and the performance of detection.

Acknowledgements

We acknowledge the Swedish Foundation for Strategic Research (SSF) SYNOPSIS Project and Artificial Intelligence Research Unit, Universiti Malaysia Sabah for supporting this work.

References

Abd-El-Barr, M. (2006). *Design And Analysis of Reliable And Fault-Tolerant Computer Systems*. Imperial College Press.

Bayindir, L. and Şahin, E. (2007). A Review of Studies in Swarm Robotics. *Turkish Journal on Electrical Engineering and Computer Sciences*, 15(2):115–147.

Canham, R., Jackson, A., and Tyrrell, A. (2003). Robot Error Detection Using an Artificial Immune System. In *Proce. 2003 NASA/DoD Conf. Evolvable Hardware*, pages 199–207. IEEE Computer Society.

Carlson, J., Murphy, R., and Nelson, A. (2004). Follow-up Analysis of Mobile robot Failures. In *Proc. IEEE Intl. Conf. Robotics and Automation*, pages 4987–4994. IEEE.

Christensen, A., O'Grady, R., Birattari, M., and Dorigo, M. (2007a). Automatic Synthesis of Fault Detection Modules for Mobile Robots. In *Proc. 2nd NASA/ESA Conf. Adaptive Hardware and Systems*, pages 693–700. IEEE Computer Society.

Christensen, A., O'Grady, R., Birattari, M., and Dorigo, M. (2007b). Exogenous fault detection in a collective robotic task. In *Proc. 9th European Conf. Advances in Artificial Life*, LNAI 4648, pages 555–564. Springer.

Christensen, A. L., O'Grady, R., and Dorigo, M. (2009). From Fireflies to Fault-Tolerant Swarms of Robots. *IEEE Trans. Evolutionary Computation*, 13:754–766.

Şahin, E., Girgin, S., Bayindir, L., and Turgut, A. (2008). Swarm robotics. In Blum, C. and Merkle, D., editors, *Swarm Intelligence: Introduction and Applications*, pages 87–100. Springer.

Festinger, L. (1954). A Theory of Social Comparison Processes. *Human Relations*, 7(2):117–140.

Gerkey, B., Vaughan, R. T., and Howard, A. (2003). The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In *Proc. 11th Intl. Conf. Advanced Robotics*, pages 317–323.

Gibbons, R. D. (1994). *Statistical Methods for Groundwater Monitoring*. John Wiley & Sons, Inc.

Heinze, J. and Walter, B. (2010). Moribund Ants Leave Their Nests to Die in Social Isolation. *Current Biology*, 20:249–252.

Lau, H., Bate, I., Cairns, P., and Timmis, J. (2011a). Adaptive Data-Driven Error Detection in Swarm Robotics with Statistical Classifiers. *Robotics and Autonomous Systems*, 59(12):1021–1035.

Lau, H., Timmis, J., and Bate, I. (2011b). Collective Self-detection Scheme for Adaptive Error Detection in a Foraging Swarm of Robots. In *Proc. ICARIS 2011*, LNCS 6825, pages 254–267. Springer.

Li, X. and Parker, L. (2009). Distributed sensor analysis for fault detection in tightly-coupled multi-robot team tasks. In *IEEE Intl. Conf. Robotics and Automation*, pages 3103–3110.

Mokhtar, M., Timmis, J., Tyrrell, A., and Bi, R. (2009). A Modified Dendritic Cell Algorithm for On-Line Error Detection in Robotic System. In *Proc. CEC2009*, pages 2055–2062. IEEE Press.

Owens, N., Greensted, A., Timmis, J., and Tyrell, A. (2009). T Cell Receptor Signalling Inspired Kernel Density Estimation and Anomaly Detection. In *Proc. ICARIS 2009*, LNCS 5666, pages 122–135. Springer.

Winfield, A. and Nembrini, J. (2006). Safety in Numbers: Fault Tolerance in Robot Swarms. *Intl. J. Modelling Identification and Control*, 1(1):30–37.