Robotics and Autonomous Systems 59 (2011) 1021–1035

Contents lists available at SciVerse ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Adaptive data-driven error detection in swarm robotics with statistical classifiers

HuiKeng Lau [a,c,*], Iain Bate [a], Paul Cairns [a], Jon Timmis [a,b]

[a] Department of Computer Science, University of York, UK
[b] Department of Electronics, University of York, UK
[c] School of Engineering & IT, Universiti Malaysia Sabah, Malaysia

ARTICLE INFO

Article history:
Received 23 August 2010
Received in revised form
5 August 2011
Accepted 17 August 2011
Available online 27 August 2011

Keywords:
Swarm robotics
Statistical error detection
Adaptive error detection

ABSTRACT

Swarm robotics is an example of a complex system with interactions among distributed autonomous robots as well with the environment. Within the swarm there is no centralised control, behaviour emerges from interactions between agents within the swarm. Agents within the swarm exhibit time varying behaviour in dynamic environments, and are subject to a variety of possible anomalies. The focus within our work is on specific faults in individual robots that can affect the global performance of the robotic swarm. We argue that classical approaches for achieving tolerance through implicit redundancy is insufficient in some cases and additional measures should be explored. Our contribution is to demonstrate that tolerance through explicit detection with statistical techniques works well and is suitable due to its lightweight computation.

1. Introduction

A swarm robotic system (SRS) consists of a collection of simple and homogenous miniature robots interacting with each other and the environment to perform some tasks without a centralised control [1]. It is inspired by observations in social insects which demonstrate emergent behaviour such as robustness to the lost of individuals, flexibility in carrying out tasks of different nature, and scalability in continuous operation with different group sizes [2,3]. These characteristics, together with distributed autonomy, make swarm robotics particularly useful for a variety of task domains such as tasks with a bounded spatial coverage, tasks that are too dangerous for human operators, tasks with dynamic scales, and tasks that require redundancy [1].

To translate research in swarm robotics from laboratory to real-world implementation, there is a need to unsure the SRSs exhibit a high level of safety and reliability [4]. SRSs in the physical world, as with any physical system, are bound to experience undesirable behaviours because of a variety of reasons such as random hardware failures, design errors, and even deliberate sabotage [4]. Therefore, the SRSs need to be fault-tolerant to ensure continuous operation in the event of failures to some individuals.

One might expect, that implicit fault-tolerance by redundancy with the large number of robots in the system should be sufficient and that tasks not completed by certain robots will be completed by other robots. This indirectly implies the independence of failures in which faulty robots do not interfere with other robots and thus the completion of tasks. However, it has been demonstrated that this is not always the case and that a number of failing robots can significantly affect the swarm and the task [4] and thus should be repaired [5]. For a swarm aggregation task, a (partial) motor failure in a single robot or a small number of robots (simultaneously) causes physical anchoring of the swarm by either impeding or even preventing the swarm from reaching its target [4]. In this case, having redundancy alone is insufficient and it would be useful to have additional fault tolerance measures and explicit error detection and recovery (ERD) is such a measure commonly applied in engineering.

ERD involves three stages: error detection, fault diagnosis, and recovery [6] (Fig. 1). Error detection identifies erroneous states and fault diagnosis determines the causes of an error including the nature and the exact location of the faults. When a fault has been identified, recovery measures can then be carried out to prevent the faults from reoccurring. This can be done by disabling the faulty components from being invoked again. If a fault still persists after recovery measures have been carried out, that information may be used as a form of feedback to the error detection and fault diagnosis mechanism for tuning and maintenance purposes.

Error detection in SRSs is interesting and challenging, particularly when the SRSs are deployed in uncertain environments where the changes in the environment also affect behaviours of the systems. In a robot foraging example, the performance of a robot may be determined by some task-related-measures such as the quantity of objects collected. If there is a fault to the wheels or grippers of a robot, the number of objects collected over a fixed period

* Corresponding author at: Department of Computer Science, University of York, UK. Tel.: +44 1904 325332; fax: +44 1904 325599.
E-mail addresses: hklau@cs.york.ac.uk, kelvinlhk@yahoo.com (H. Lau).

0921-8890/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2011.08.008

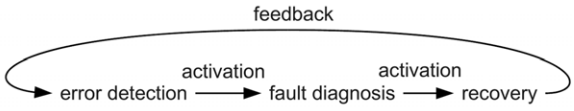


Fig. 1. Stages in an explicit error detection and recovery (EDR) mechanism for fault tolerance.

by that robot will be less and likely to deviate significantly from a fault-free condition. However, in this example, the quantity of objects collected is also influenced by environmental conditions such as the availability of objects and physical distribution of objects in the arena, which likely to change as time progresses. From the perspective of a data-driven approach to error detection [7] in which the presence of a fault is inferred from the operational data, this is challenging as the task is to figure out whether the change in the data is due to the presence of a fault or an artefact of the changes in the environment. Thus any error detection approach needs to be adaptive in order to differentiate between changes in data due to faults and environment. As SRSs are generally deployed to operate over an extended period, and possibly with limited resources (e.g. computational power, memory, and storage) there is a need for the EDR to be lightweight and inexpensive to run. These constraints impose restrictions on the complexity of the error detection method. Our task was to implement and if necessary adapt existing methods to address the aspect of adaptive error detection.

Some of these challenges have been addressed in previous work such as detection capabilities [8], flexible detectors [9], resource usage [10] and data-driven detection [7]. However, the aspect of adapting to dynamically changing environments has largely being ignored. In this paper, we address this through social comparison [11] (which is essentially a neighbourhood scheme) in which a robot cross-referencing its behaviour (data) with others within its communication range (local neighbourhood) to self-detect whether an error has occurred within itself. In this case, there is no central point for the identification of errors within the swarm but rather distributed over all robots. This, in principle, would allow for a greater degree of fault tolerance.

The main contributions of this work, therefore, are (1) an investigation on the detection of errors due to faults on wheels in a foraging SRS with four classical statistical classifiers, (2) deployment of the statistical classifiers for dynamic environments, (3) augmentation and deployment of a novel statistical based technique that has previously not been applied in an practical setting, and (4) two communication strategies to reduce communication overhead in the context of proposed implementation. It should be noted that fault diagnosis and recovery will not be addressed in this paper but interested readers can refer to [12,10].

The rest of the paper is organised as follows: Section 2 describes the foraging SRS in the context of our work. Section 3 describes the statistical classifiers with the application of those classifiers for adaptive error detection in Section 4. Results are presented in Sections 5 and 6 provides suggestions to reduce false positive rate and communication overhead. Conclusions are followed by a discussion of future work in Section 7.

2. Case study: a foraging swarm robotic system

2.1. Foraging

Typically, robot foraging involves a group of robots deployed in an arena to search for specific objects and transport collected objects to a specific location [13]. At any time these robots might experience faults, (i.e. faulty motor) and the task is to detect it and recover if possible. Faults on wheels (as well as grippers) directly affect the ability of robots to forage, and we propose that the presence of these faults can be inferred, e.g. through the number of objects collected ($\texttt{obj} \in \mathbb{Z}_0^+$), energy used ($\texttt{eng} \in \mathbb{R}_0^+$), and distance travelled ($\texttt{dist} \in \mathbb{R}_0^+$) over a duration (control cycle). Unfortunately, these data are also influenced by the environment in which the SRS operates, (e.g. the availability of objects, obstacles, spatial distribution of objects). Thus, the challenge is to determine whether a change in data is due to faults or changes in the environment.

2.2. Forager robot

The foraging behaviour of the robots in our study is coded based on a subsumption architecture [14], by decomposing foraging behaviour into smaller behavioural modules of obstacles avoidance, moving to an object, grabbing an object, moving to a base, follow, deposit, scan arena and random walk. These behaviours are expressed as a finite state machine, as illustrated in Fig. 2. The energy consumption for each decomposed behavioural task per second as in [15]: *avoidance* 0.9, *searching* 0.8, *MovingToObject* 0.8, *grabbing* 1.2, *departing* 0.8, *homing* 1.2, *following* 0.8, and *depositing* 1.2 unit.

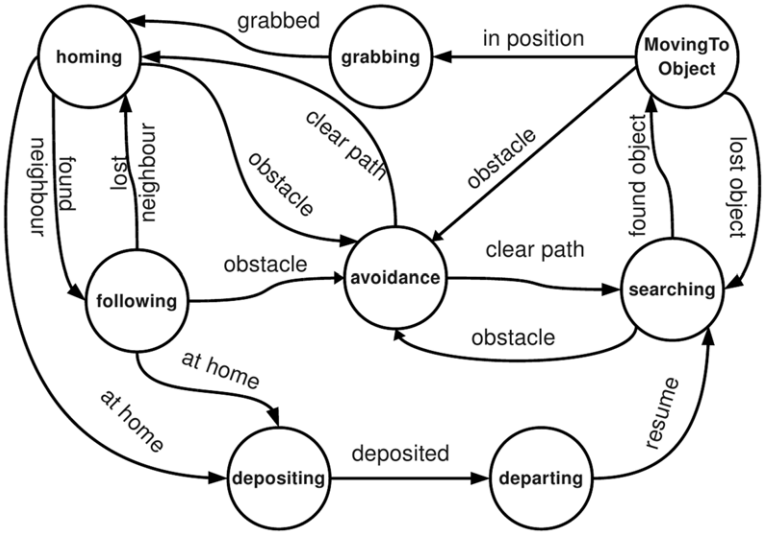


Fig. 2. Finite state machine for robot foraging.
Source: Adapted from [15].

Table 1
Summary of fixed simulation parameters.

Parameter	Value	Parameter	Value
No. of robots	10 units	Component fault	Wheels
Default OPR	0.10 unit/s	No. faulty robot	1 unit
Initial object	100 units	Fault injection at	Control cycle 20
Maximum object	200 units	Default speed	0.15 m/s
Simulation	80 control cycles	Comm. range	2 m radius
Control cycle	250 s		

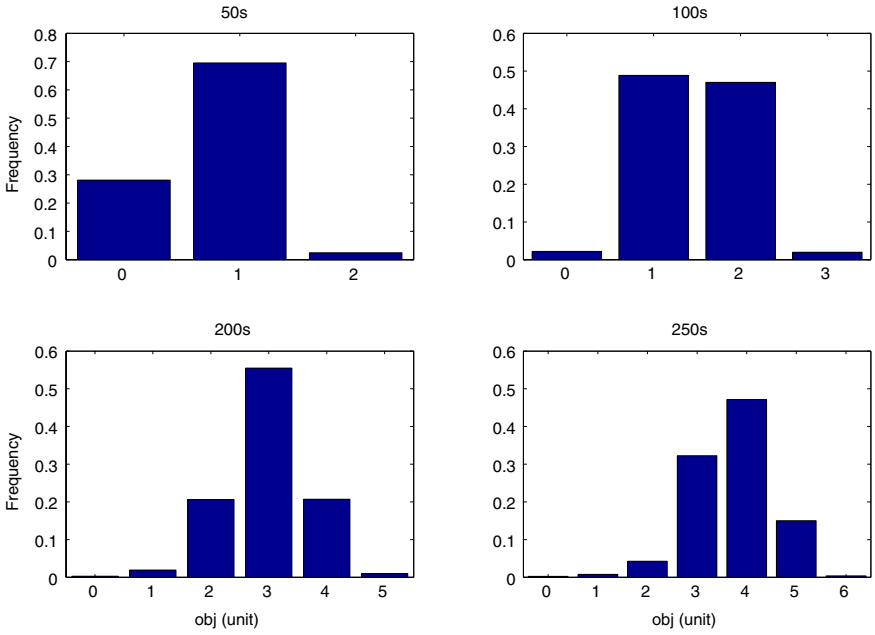


Fig. 3. Histograms on the distribution of obj for different control cycle lengths.

A simulation begins with ten robots dispatched from the base and then follow the state behaviours as shown in Fig. 2. The arena is a 10-m by 10-m bounded octangular shaped with a 1-m radius circular base located in the middle. The robots are dispatched from the base in all direction and the heading for each robot is calculated assuming the robots start from a single point. With n robots, the direction for robot Ri $= i \times \frac{2\times\pi}{n}$. For example, in a SRS with ten robots, the heading for robot R1 is $\frac{\pi}{5}$, robot R2 is $\frac{2\pi}{5}$, and so on. Red-coloured square objects are placed randomly[1] in the arena.

The foraging task is normally completed when all objects in the arena have been collected. However, since we are interested in the detection of errors in the presence of changing environmental conditions, collected objects are replaced continuously at an object replenishing rate (OPR). The OPR is the probability of adding one object in one second, which default value is 0.10. To avoid overcrowding, the maximum number of objects in the arena at one time is set to 200 units. Table 1 is the setting for other parameters in this study.

Our current work focuses on the error detection in the case of a single robot failure for the reason that if the failure modes for a single fault are sufficiently complex that current state of the art approaches do not work well, then they are unlikely to deal with more complex ones. In addition, by only considering a single failure, it is possible to provide a set of results more easily comprehensible such that a more accurate understanding of the problem domain is gained. Therefore, the single failure case is a good place to begin for our investigation on the aspect of dynamic environments for adaptive detection.

2.3. Operational data

Data-driven approach to error detection is preferable for our work instead of model-based approach (i.e. build analytical models of how a system should behave at design-time, and compared with run-time behaviour for the detection of errors) because analytical models are often not feasible due to uncertainties in the environment [7]. Thus, data (in our case obj, eng, and dist) at each control cycle is recorded[2] during the simulations. We then evaluate the error-detection ability of various statistical classifier in terms of the true positive rate (TPR), false positive rate (FPR), and the time it takes to detect the errors (Latency).

The granularity or the length of a control cycle affects the magnitude of data values and also the time to detect the errors. If a control cycle is too short, the data values are likely to be small and not meaningful for the detection. Similarly, if a control cycle is too long, by the time an error is detected it might be too late for any recovery actions. Fig. 3 is an example of obj values for control cycles of 50, 100, 200, and 250 s. For a control cycle of 50 s the value of obj is between 0 and 2 with an arithmetic mean of 1; whereas for a control cycle of 250 s it is between values of 0 and 6 with a arithmetic mean of 4. If the presence of a fault is inferred through the difference between a data instance and the

[1] The random number generator used is gsl_rng_mt19937 in GSL-GNU Scientific Library initialised with seed value 999. The library can be found on www.gnu.org/software/gsl.

[2] The data is available at https://sites.google.com/site/researchmaterialshkl/data.

arithmetic mean, then the difference between a value 0 (faulty) and the arithmetic mean for control cycle of 250 s is more 'significant' for error detection compared with a control cycle of 50 s. However, it also means that up to 250 s has passed before an error is detected. Therefore, there is a tradeoff of `Latency` and the TPR. Results in this paper are based on a control cycle of 250 s.

Summarising the data from a simulation run of the SRS in non-dynamic environment with all robots in fault-free condition, Fig. 3 shows histograms on the distribution of `obj` over the different control cycle lengths. From the histograms, the distribution of the data values is somewhat normal. A similar bell-shape pattern is also seen in the distribution of `eng` (Fig. A.12) and `dist` (Fig. A.13). Hence, error detection with statistical classifiers seems a reasonable starting point.

2.4. Operational environment

This work addresses the aspect of adaptive data-driven error detection in which the operational data used to infer the presence of faults are also influenced by changes in the environment. Three conditions in which the SRS operates: `CST` (non-dynamic), $\mathtt{V}_{\mathtt{OPR}}$ (varying `OPR`), and $\mathtt{V}_{\mathtt{ODS}}$ (varying objects distribution). In `CST`, the `OPR` is constant at 0.100 with homogeneous object distribution; whereas in $\mathtt{V}_{\mathtt{OPR}}$ and $\mathtt{V}_{\mathtt{ODS}}$, the concentration and the spatial distribution of objects changes over time, and this has a direct effect on the operational data. In simulation, the `OPR` changes between 0.100 and 0.025 at specific intervals in $\mathtt{V}_{\mathtt{OPR}}$ whilst in $\mathtt{V}_{\mathtt{ODS}}$ the placement of a new object was biased between top left and bottom right regions in the arena.

The foraging SRS was simulated with Player/Stage [16],[3] a free open source robotic simulation tool. It comprises of Player which communicates with hardware through the source code and a plug-in called Stage which receives instructions from Player and moves robots in a simulated world and passes data to the Player. We work with simulation as it allows us to collect data easily in a controlled manner and allows for large numbers of experiments to be undertaken. Of course, ultimately any system developed should be deployed in an actual robotic system, but insight gained from this simulation process will allow for a more principled design and hence reduce the risk associated with premature deployment in real robots.

3. Statistical error detection

Five statistical techniques were implemented in this work, two of which are parametric and the remainder are non-parametric. The mix of both parametric and non-parametric techniques is for unbiased exploration of solutions. Extreme Studentised Deviate (ESD), student's T-test, box plot rules with quartiles, and Dixon's Q-test [17] are classical statistical techniques for outliers detection, with ESD and T-test being parametric (assuming the data belong to some distributions). In addition to classical non-parametric techniques, a more recent nonparametric kernel-based algorithm called The Receptor Density Algorithm (RDA) [18] was also tested. The RDA was inspired by the T-cells signalling mechanisms in the immune system [18], and it is particularly interesting as it has been demonstrated to be suitable for dynamic anomaly detection of different substances in a spectra [19]. These techniques that have been used in our study are presented in the following sections.

[3] Latest version of Player/Stage can be found on http://playerstage.sourceforge.net/. We used Player 2.1 and customised Stage 2.1.0 from www.brl.uwe.ac.uk/projects/swarm/index.html.

3.1. Classical statistical techniques

ESD, also known as Grubbs' test [20], calculates a Z value as the difference between the mean μ and a suspected outlier x divided by the standard deviation σ. If Z is larger than a certain threshold k, x is classified as an outlier.

$$\frac{|x-\mu|}{\sigma} > k. \tag{1}$$

The value of k is usually taken as 2 or 3 because if the data are normally distributed, they are expected to be within two (three, respectively) standard deviation from the mean [21].

For the T-test, a sliding window technique is used to provide a running mean of a sample. Each robot produces a running mean μ which, if all robots are anomaly free, will come from the same distribution. Hence μ can be compared to the running means of other robots ν to see if it is outlying using a T-test. The formula for the T-test is a ratio of the difference between the two means, and the denominator is a measure of the variability or dispersion of the scores (also referred to as the standard error of the difference) [22].

$$\frac{|\mu-\nu|}{\mathrm{SE}(\mu-\nu)} \leq T(p, \mathrm{d}f) \tag{2}$$

where $\mathrm{SE}(\mu-\nu) = \sqrt{\frac{\sigma_\nu^2}{N}}$, σ_ν = standard deviation of the means, p is the statistical significant threshold (usually 0.05 for a 95% confidence level), $\mathrm{d}f = N-1$ is the degrees of freedom, and $T(p, \mathrm{d}f)$ is the standard t-value that corresponds to given values of p and $\mathrm{d}f$ in the T-table.

In the box plot rule, an outlier is defined as an instance falling outside 1.5 interquartile range (`IQR`) of either the first quartile, 25th percentile ($x_{0.25}$), or the third quartile, 75th percentile ($x_{0.75}$) [23]. The `IQR` is the range between the upper quartile and lower quartile, i.e. $\mathtt{IQR} = x_{0.75} - x_{0.25}$. The multiplier of 1.5 is the *de facto* value in statistics to define a mild outlier and a multiplier of 3 is for extreme outlier [24].

$$x_{0.75} + k\mathtt{IQR} \geq x \geq x_{0.25} - k\mathtt{IQR}, \quad \mathtt{IQR} = x_{0.75} - x_{0.25}. \tag{3}$$

Dixon's Q test, or simply the Q test, calculates a Q value as the ratio between the absolute difference between a suspected outlier and the closest value to it (gap), and the range, i.e. the difference between the maximum and minimum value. If the Q value is greater than corresponding value in Q-table, the suspected outlier is classified as an outlier.

$$Q = \frac{\text{gap}}{\text{range}}. \tag{4}$$

3.2. Receptor density algorithm

The RDA was developed through the extraction of features of the generalised T-cell receptor, and mapped onto kernel density estimation [25], a technique from statistical machine learning. The details of the development of the RDA [25] and related biology [26] are beyond the scope of this paper but interested readers are directed to relevant papers.

The RDA works as follows. The spectrum of input value is divided into an s discretised location and a *receptor* $\mathbf{x}_s$ is placed at each of these locations. A receptor has a length $\ell = \frac{1}{n \times h(\sqrt{2\pi})}$, a position $r_p \in [0, \ell]$, and a negative feedback barrier $\beta \in (0, \ell)$. At each time 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.

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, $r_n = 0$.

$$r_p(\mathbf{x}) = \sum_{i=1}^{n} \frac{1}{nh} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \qquad r_n(\mathbf{x}) = \begin{cases} r_p(\mathbf{x}) - \beta, & \text{if } r_p(\mathbf{x}) \geq \beta \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

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

$$r_p^t(\mathbf{x}) = b \times r_p^{t-1}(\mathbf{x}) + gb \times K\left(\frac{\mathbf{x} - \mathbf{v}}{h}\right) - \alpha \times r_n^{t-1}(\mathbf{x}) \tag{6}$$

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

$$c(\mathbf{v}) = \begin{cases} 1, & \text{if } r_p^t(\mathbf{x}) \geq \ell \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

The specific value for various parameters of the RDA is application-specific and also depending on specific performance metrics of interest. For readers interested on optimising the RDA parameters, refer to [27]. In this paper, the number of receptors is $= 20$, $\beta = 0.01$, $b = 0.1$, $gb = 1.1$, $\alpha = 1$, and the kernel is Gaussian kernel is defined as

$$K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{(\mathbf{x}-\mathbf{x}_i)^2}{2h^2}}.$$

4. Application of statistical classifiers for adaptive error detection

As mentioned previously in Section 1, data-driven error detection approaches infer the presence of a fault through the analysis on the operational data. In our foraging SRS, data regarding the `obj`, `eng`, and `dist` of each robot are collected in each control cycle. The values of these variables differ from one control cycle to another depending on factors such as the quantity of objects in the arena, and whether the robot is fault-free. Fig. 4 is a snapshot of the `obj` on a fault-free robot operates in an environment with time-varying amount of objects (a change occurred at control cycle 10). Given such data, one way to determine whether an error has occurred with statistical techniques is to use a sliding time window on the data to calculate some descriptive values (depending on the statistical technique) of the sample and compare these values against those from the previous time window. If the calculated values exceed defined thresholds, an error is reported. For example, assuming a Q-test at 95% confidence is used with a sliding time window of size 5 on data in Fig. 4. At control cycle 10, the calculated Q-value $= \frac{7-3}{8-3} = 0.800$ and corresponding Q-value from the Q-table is 0.710. Since $Q_{\text{calculate}}(0.800) > Q_{\text{table}}(0.710)$, an error is reported. Note that we analyse each variable separately and an error is detected if it was detected in any of the three variables.

However, since the 'significant' change in `obj` at control cycle 10 is an artefact of a change in the operational environment that affects all robots in the system, then classifying the data instance as an error is a false positive. In this case, detecting errors with data from a single robot is insufficient to differentiate between changes in data due to faults and those of dynamic environments. An alternative approach is to also include data from other robots within a local neighbourhood for cross-referencing and validation,

```
... 5 6 7 8 9 10 11 12 13 14 ... control cycle
... 8 7 8 8 8  3  0  0  0  0 ... obj
```

Fig. 4. An example of the `obj` on a robot at control cycle 5 to 14.

cc	R1				
.	.				
5	8	7	8		
6	7	8	8	7	8
7	8	7			
8	8	7	5	5	
9	8	8	6		
10	3	4	4	3	2
11	0	1	0	1	
12	0	1	1		
13	0	2	1	0	1
14	0	0			
.	.				

Fig. 5. R1's own data together with data from other robots within a communication range of R1. The label cc and R1 are placed for clarity and not in the input files.

this is the approach we advocate. With this approach, false detection, i.e. false positives and false negatives, can be reduced because a change in the environment is likely to affect all robots within the local proximity. Through cross-referencing, a robot can validate whether a similar change in behaviour is also experienced by surrounding robots. If it is, then it is likely to be an artefact of a change in the environment and can be safety ignored. With this, the detection is considered to be adaptive to environmental changes because the environmental effects are not miss-classified as a manifestation of a fault.

To obtain data from other robots, some form of communication is required. For the purpose of this work, we make use of a wireless communication network. During each control cycle, robots exchange data with other robots (neighbour) within a communication range of 2 m radius. Since the robots are mobile, there is a probability that a robot might not interact with another robot during a control cycle. Fig. 5 is a snapshot of aggregated `obj` for R1 showing the number of robots that interact with R1. The data is arranged in such a way that the first datum is from R1. Notice that the amount of data from other robots (neighbourhood) differs from one control cycle to another and the sources of data were not specified.

Data-driven error detection with neighbourhood data is the approach we have adopted during our experiments. An important assumption is that the detection should determine whether a robot itself is faulty (self-detection) and not whether the other robots are faulty: for this reason the identity of the sources of data is not important. We propose that such an approach should be more flexible as no additional storage is required to keep track of data sources, there is no centralised point of detection, and in principle, the approach is scalable.

4.1. Accessing error-detection ability

The error-detection ability of statistical classifiers was evaluated based on the true positive rate TPR, false positive rate FPR, and the time taken to detect the failure (referred to here as the `Latency`).

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{8}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}. \tag{9}$$

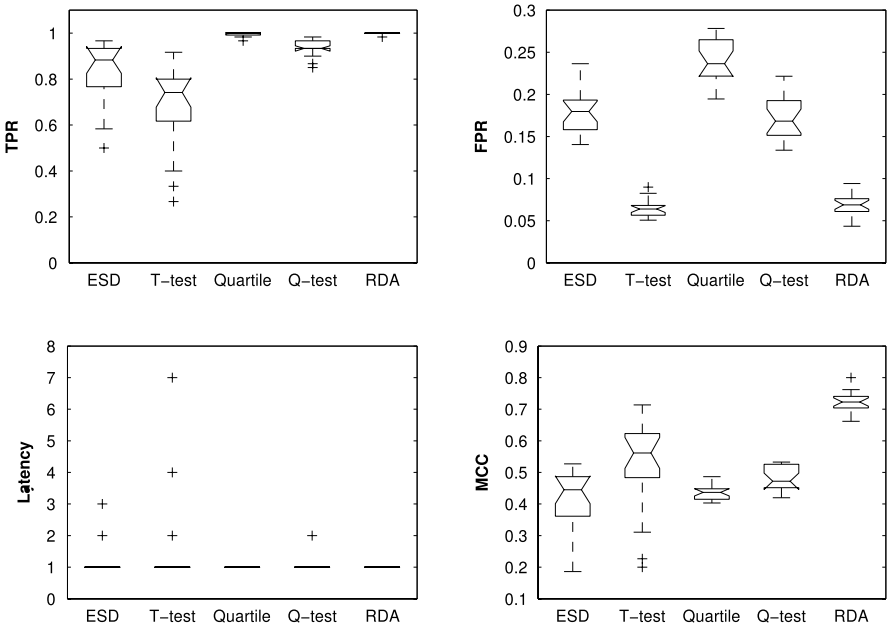


Fig. 6. Boxplots of the MCC, TPR, and FRP in detecting P_{CP} in CST.

TP = true positive, FN = false negative, FP = false positive, and TN = true negative.

$$\texttt{Latency} = \text{fault injection time} - \text{fault detection time}. \tag{10}$$

Given TPRs and FPRs of two techniques, it is often difficult to judge which one is superior if one technique has a higher TPR whilst the other has a lower FPR. In this case, the Matthews correlation coefficient (MCC) [28] score is useful and was included in the results. Although there is no single value to best describe the matrix of TP, FN, FP, and TN but MCC is regarded as one of the best such measures that include all of those metrics.

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}}. \tag{11}$$

5. Experiments and results analysis

In [4], a partial failure (P_{PT}) to the wheels was discovered to be the most critical and it significantly affects the ability of the swarm to carry out its task. Therefore, statistical classifiers defined in Section 3 were tested to detect these errors in dynamic environments. In addition, for completeness, we also tested two additional failure modes to the wheels: complete failure (P_{CP}), and gradual failure (P_{GR}). (Note that it is assumed that other components including the communication are working properly.) With P_{CP}, the left wheel on a faulty robot was set to left turn 10° causing the robot to move in a circle and unable to move to continue foraging. With P_{PT}, the robot wheels move less efficiently by reducing the speed of the wheels causing the robot to move more slowly but otherwise still able to search and collect objects. In the P_{GR} case, the wheels on a faulty robot move with a gradually reducing speed. These failure modes were independently simulated for the SRS in CST, V_{OPR}, and V_{ODS}. In the experiments, the fault was injected at control cycle 20 and persists until the end of simulation (permanent); the changes in V_{OPR} and V_{ODS} occur at control cycles [20, 40], [60, 80]; and each scenario was repeated 20 times. In theory, if the classifiers were not able to adapt to the environmental changes, then the FPR will be greater than $\frac{40}{80} = 0.50$ (i.e. environmental changes last for 40 control cycles of a simulation of 80 control cycles). Similarly, the TPR will be lower than 0.50.

5.1. Experiment A

This experiment investigates the ability of the statistical classifiers in detecting P_{CP}. This failure mode is the most severe among the three failure modes investigated in this paper and it should manifest itself very clearly in the data. Thus, it was expected that, in CST, the statistical classifiers should detect P_{CP} error easily, i.e. with a high TPR and a low Latency.

Results for P_{CP} in CST are summarised in Fig. 6, V_{OPR} in Fig. A.14, and V_{ODS} in Fig. A.15. In the boxplots, the centre line in each box is the median, the upper edge is the third quartile, and the lower edge is the first quartile. The whiskers extend to cover data points within 1.5 times IQR and outliers are plotted individually.

The boxplots showed that almost all statistical classifiers were able to detect P_{CP} with a median TPR greater than 0.80, FPR less than 0.25, and a Latency of 1 control cycle (immediate detection). One exception is the T-test in which the median TPR was about 0.74 in CST, 0.60 in V_{OPR}, and 0.65 in V_{ODS}. However, the T-test has the lowest FPR.

MCC score is useful as a measure (if necessary) to decide which classifier is superior when one classifier is better with a higher TPR whilst another is better with a lower FPR. For example in CST (Fig. 6), the Quartile and Q-test has a significantly higher TPR than with the T-test. However, on the results of the FPR, T-test a significantly lower FPR compared to the Quartile and Q-test. In this case, the MCC score is useful and it was determined that the T-test was superior.

In the case of V_{OPR}, the TPR results were lower and the FPR were higher compared to CST. The reason for this is the availability of objects in the arena changes and this directly affects the data. When the quantity of objects is minimal and approaching zero, many fault-free robots will not be able to collect any objects. Similarly, a faulty robot will not be collecting any object and thus the TPR is expected be lower compared with results in CST. This is because while the environment is changing, all robots will have nearly exact data values and thus the fault might not be reflected on the data.

Results for P_{CP} in V_{OPR} and V_{ODS} give evidence that our implementation of statistical classifiers are adaptive to changes in the environment. This is shown with a TPR significantly greater than 0.50 and a FPR significantly lower than 0.50 in those environments.

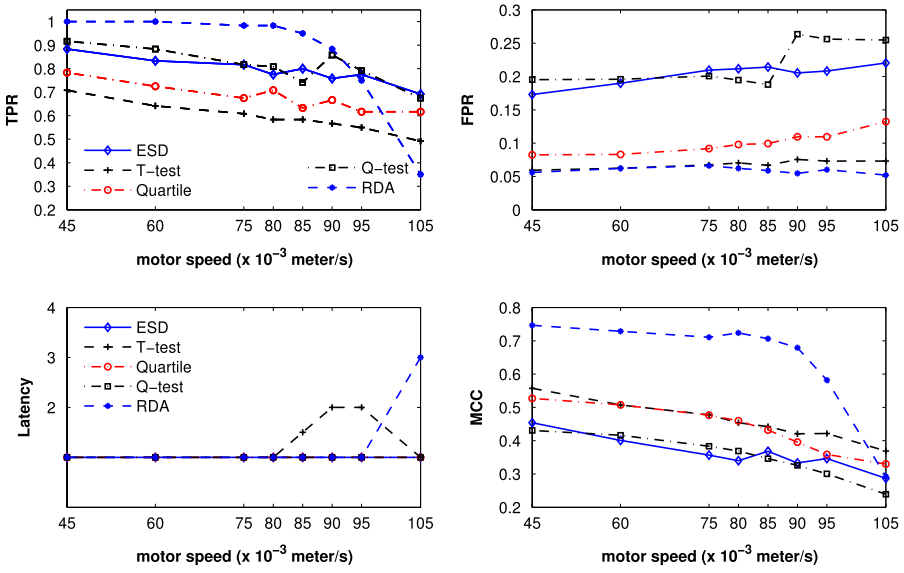


Fig. 7. The median TPR, FPR, Latency and MCC in detecting P_{PT} in CST.

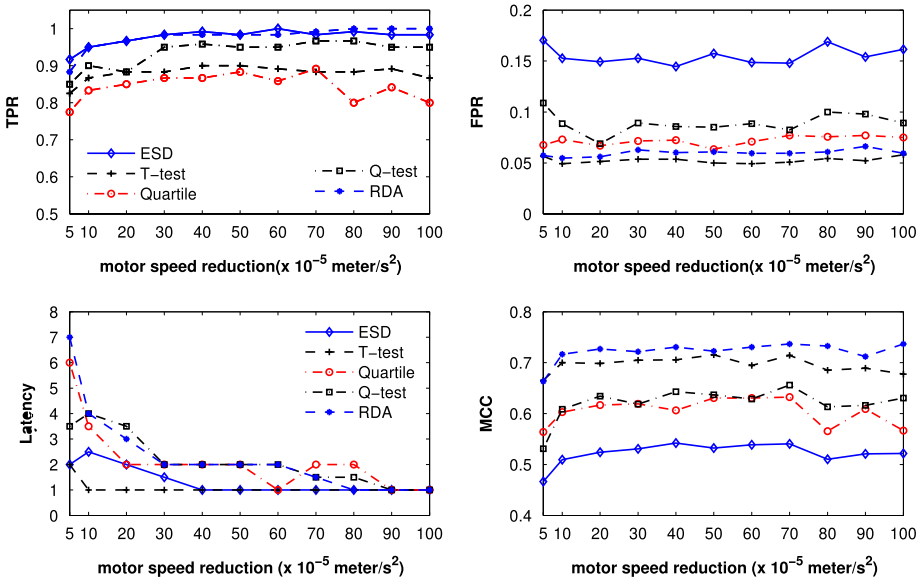


Fig. 8. The median TPR, FPR, Latency, and MCC in detecting P_{GR} in CST.

Among the statistical classifiers investigated, *T*-test and the RDA stand out and outperformed the others with the RDA being the one with the highest MCC score. We think because of the random movements of each robot, the data distribution is not strictly normal all the time. Therefore, non-parametric techniques work better than parametric ones. In addition, the RDA works on the density (frequency) rather than exact data values as with other classifiers. Thus, it is unaffected by the underlying data distribution.

5.2. Experiment B

This experiment investigates the detection of P_{PT} which is less severe when compared with P_{CP}, and thus harder to detect because the P_{PT} errors are less obvious compared with P_{CP} errors. The severity of P_{PT} directly influences the data values of the operational data; the more severe is the P_{PT}, i.e. slower wheels movement, the more obvious its effects as manifested on the data. Therefore, we test P_{PT} with failures ranging from severe ($P_{PT} = 45 \times 10^{-3}$ m/s) to subtle ($P_{PT} = 105 \times 10^{-3}$ m/s) to establish the performance (i.e. TPR, FPR, Latency) of the classifiers over a range of P_{PT} severity.

Results for P_{PT} in CST are summarised in Fig. 7, V_{OPR} in Fig. A.16, and V_{ODS} in Fig. A.17. Each graph shows the median value for a performance metric of the classifiers over a range of P_{PT} severity. The labels on the *x*-axis are the P_{PT} tested, and the lines connecting points are drawn for clarity.

At $P_{PT} = 45 \times 10^{-3}$ m/s, the TPR decreases whilst the FPR increases as the fault become more subtle. For instance in CST, whilst the TPR for Quartile decreased from 0.80 at $P_{PT} = 45 \times 10^{-3}$ m/s to 0.60 at $P_{PT} = 105 \times 10^{-3}$ m/s, the FPR increased from 0.08 to 0.13. This trend of decreasing TPR is seen for all classifiers. However, the FPR of *T*-test and the RDA seems to be not much affected by the severity of P_{PT} in which the median FPR is consistently the same throughout, i.e. 0.05 (in CST, V_{ODS} and V_{ODS}) for the *T*-test, 0.05 (in CST, V_{ODS}) and 0.10 (in V_{OPR}) for the RDA.

We suspect that the relatively unchanged FPR of the *T*-test is due to the subtle failure, the difference between the μ and ν in Eq. (2) is too small and insignificant to be classified as an error, and

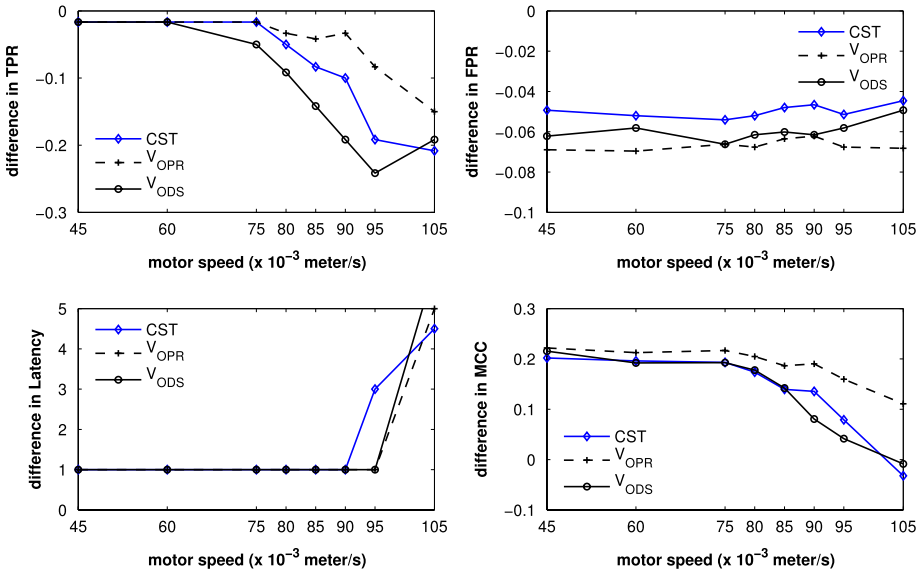


Fig. 9. The difference in performance between the original RDA with RDA-DW2, e.g. TPR(RDA-DW2)-TPR(RDA), in detecting P_{PT}. A positive value denotes an increment, a negative value denotes a reduction, and zero means no difference.

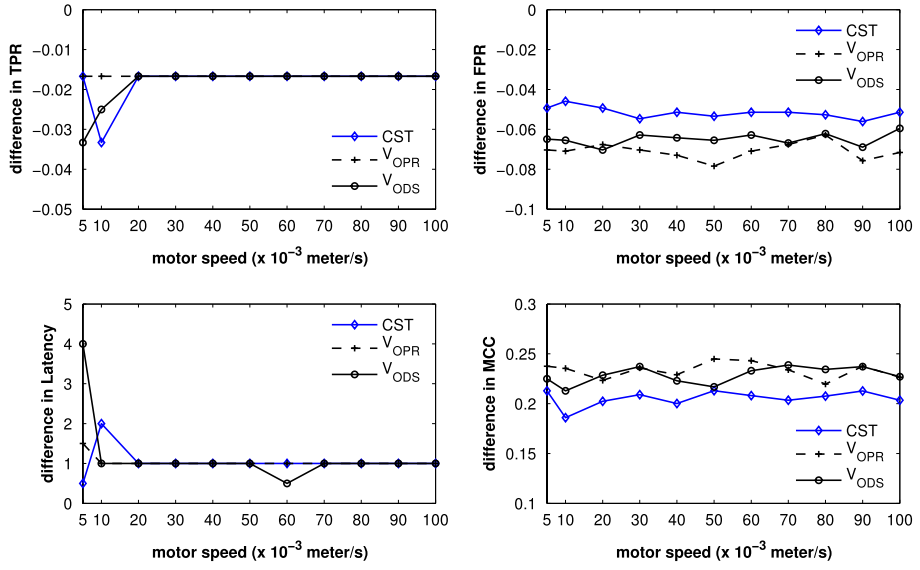


Fig. 10. The difference in performance between the original RDA with RDA-DW2, e.g. TPR(RDA-DW2)-TPR(RDA), in detecting P_{GR}. A positive value denotes an increment, a negative value denotes a reduction, and zero means no difference.

Table 2
The frequency of communication (broadcast) by each robot using different communication strategies in detecting $P_{PT} = 45 \times 10^{-3}$ m/s.

Env.	Strategy	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
CST	Original	80	80	80	80	80	80	80	80	80	80
	S1	60	0.5	0	0	0	0	1	0	0	1
	S2	65	19	18.5	19	20	18.5	19	19.5	19.5	19
V_{OPR}	S1	60	3	3	3	2.5	3	2.5	3	2	2
	S2	63.5	25	21.5	24.5	23	23.5	24	23.5	23.5	23.5
V_{ODS}	S1	60	2	2.5	2	3	3	2.5	2	3	2
	S2	65.5	22	22.5	21	19	21.5	18	22	20	20

thus a low value for both both TPR and FPR. As for the RDA, the aforementioned reason (i.e. detection based on density instead of data values) applies.

At $P_{PT} = 90 \times 10^{-3}$ m/s in CST, especially on the RDA's TPR and Q-test's FPR, a large decrease in the TPR and a large increase in the FPR can be seen. We suspect this is a critical point where the severity of P_{PT} is too subtle on the operational data for some classifiers. There is also evidence of this observation in V_{OPR} (Fig. A.16) and V_{ODS} (Fig. A.17).

Results for P_{PT} in V_{OPR} and V_{ODS} show that the implementations are adaptive to changes in the environment with a TPR significantly greater than 0.50 and FPR significantly lower than 0.50. One

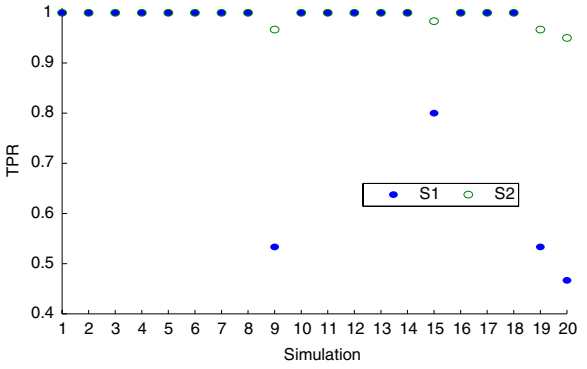


Fig. 11. The TPR in detecting $P_{PT} = 45 \times 10^{-3}$ m/s in CST for the 20 runs. At simulation 9, 15, 19, and 20, S1 missed some errors but with S2 was still able to detect them due to additional external detections subject to c and K.

exception is the T-test in which the TPR was below 0.50 for P_{PT} greater than 90×10^{-3} m/s in V_{OPR}.

Overall, the RDA achieved highest MCC score compared with other classifiers, whilst the T-test remained superior compared with other classical classifiers.

5.3. Experiment C

The severity of P_{GR} has an obvious impact on the `Latency`, i.e. a subtle fault will take longer before it has a significant impact on the data. Thus, the fault can only be detected at a later stage. In this experiment, we test P_{GR} from subtle (5×10^{-5} m/s^2) to severe (100×10^{-5} m/s^2), where a higher value of P_{GR} signifies a more severe fault.

The results for P_{GR} are summarised in Fig. 8 for CST, Fig. A.18 for V_{OPR}, and Fig. A.19 for V_{ODS}. Each graph shows the median value for a performance metric of the classifiers over a range of P_{GR} severity with labels on the x-axis for the P_{GR} tested. The lines connecting points are drawn for clarity.

As the severity of P_{GR} increases, e.g. from P_{GR} 5×10^{-5} m/s^2, it became easier to detect as evident from the increase in TPR and can be detected faster with a decrease in `Latency`. The trends were exhibited in all scenarios (CST, V_{OPR}, V_{ODS}) for all classifiers. However, the FPR remained relatively unchanged throughout.

Results for the `Latency` show four trends: $P_{GR} < 30 \times 10^{-5}$ m/s^2 (T1), 30×10^{-5} m/s^2 $\leq P_{GR} \leq 50 \times 10^{-5}$ m/s^2 (T2), 50×10^{-5} m/s^2 $\leq P_{GR} < 80 \times 10^{-5}$ m/s^2 (T3), and $P_{GR} \geq 80 \times 10^{-5}$ m/s^2 (T4). The median `Latency` for T2 and T4 is constant, whereas it is decreasing as the fault becomes more severe in T1 and T3. If we consider T2, T3, and T4 to be an acceptable `Latency`, then $P_{GR} = 30 \times 10^{-5}$ m/s^2 is the critical point at which

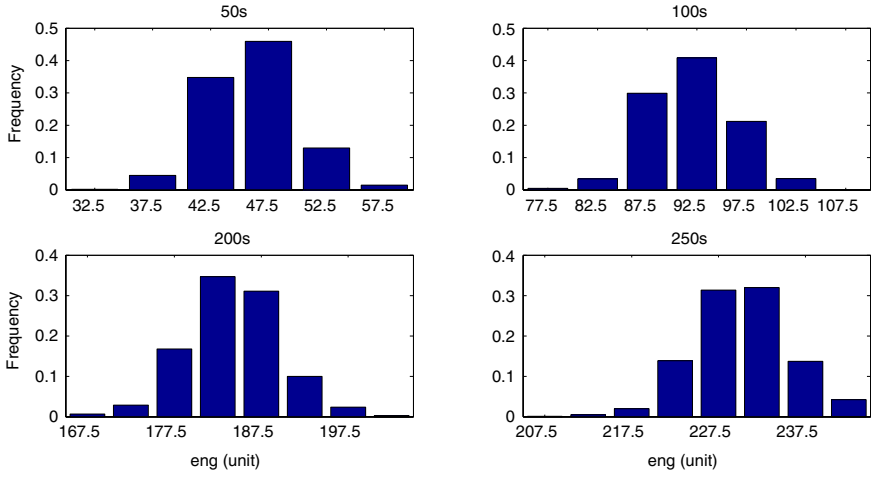


Fig. A.12. Histogram showing the distribution of the `eng` at system level over the lengths of a control cycle.

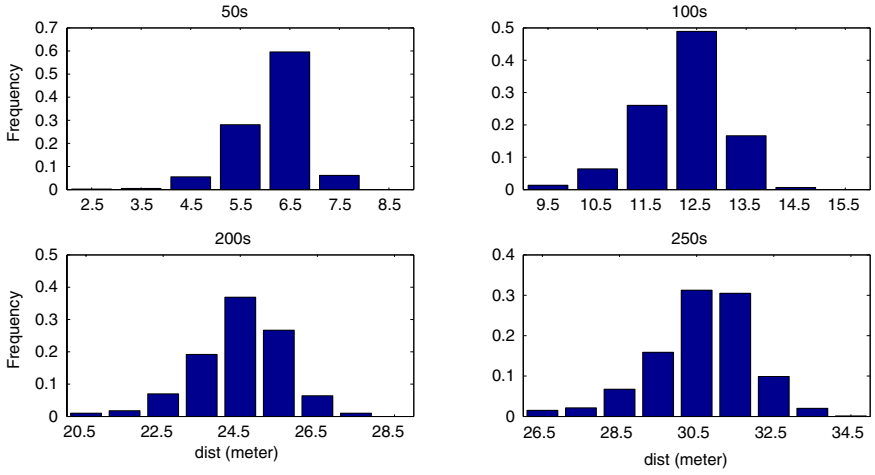


Fig. A.13. Histogram showing the distribution of the `dist` at system level over the lengths of a control cycle.

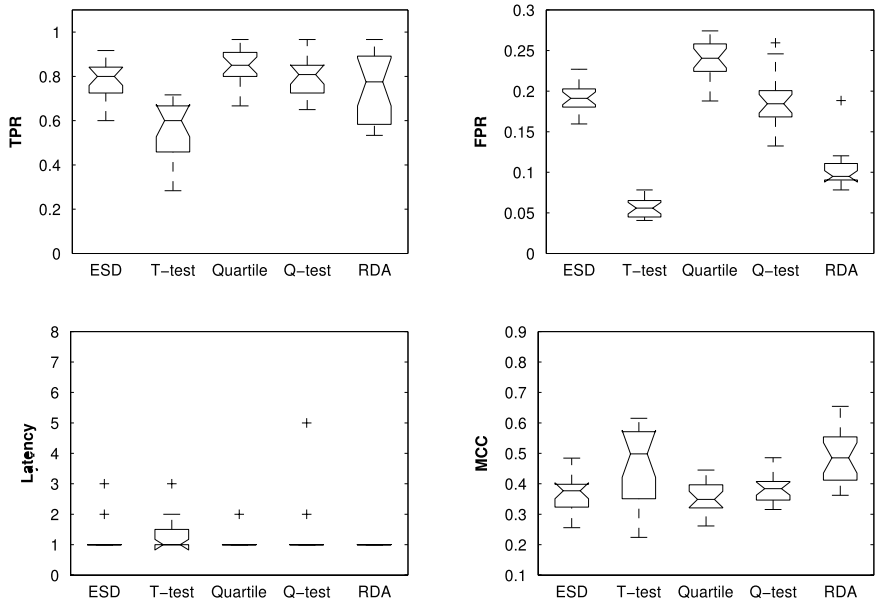


Fig. A.14. Boxplots of the MCC, TPR, FRP in detecting P_{CP} in V_{OPR}.

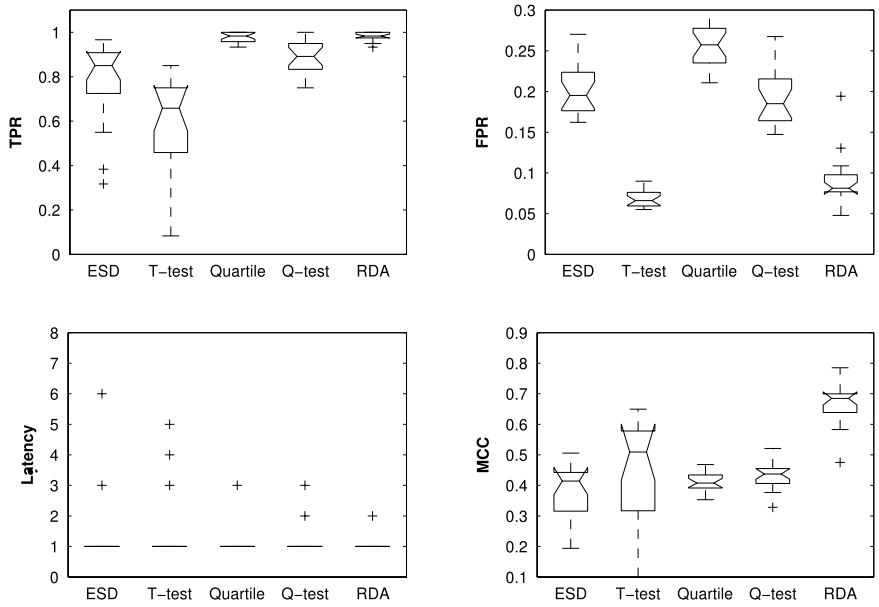


Fig. A.15. Boxplots of the MCC, TPR, FRP in detecting P_{CP} in V_{ODS}.

Table A.3
The median TPR, FPR, Latency, and MCC in detecting $P_{PT} = 45 \times 10^{-3}$ m/s.

Environment	Strategy	TPR	FPR	Latency	MCC
CST	Original	1.0000	0.0561	1.0	0.7463
	S1	1.0000	0.0027	1.0	0.9739
	S2	1.0000	0.0338	1.0	0.8259
V_{OPR}	Original	1.0000	0.0926	1.0	0.6377
	S1	1.0000	0.0155	1.0	0.8920
	S2	1.0000	0.0588	1.0	0.7247
V_{ODS}	Original	1.0000	0.0716	1.0	0.7169
	S1	1.0000	0.0128	1.0	0.9195
	S2	1.0000	0.0345	1.0	0.8484

a drastic increase in the Latency occurs. This also means that faults at $P_{GR} < 30 \times 10^{-5}$ m/s^2 are too subtle for the classifiers.

Results for P_{GR} in V_{OPR} and V_{ODS} are the same as in P_{CP} and P_{PT} in that the implementations are adaptive to changes in the environment with a TPR significantly greater than 0.50 and FPR significantly lower than 0.50.

Again, the RDA remained the superior classifier compared to other investigated classifiers in detecting P_{GR} with the highest MCC score.

6. Enhancements for improved performance

Having evidence to support that implemented statistical classifiers were able to detect faults to the wheels even in the presence of dynamic environmental changes, we wonder if the performance especially the FPR can be improved. Results for the TPR with P_{CP}, P_{PT}, and P_{GR} showed the RDA has the highest TPR. However, the FPR was consistently lower than *T*-test. Therefore, we investigate whether the FPR can be reduced with a larger detection window size (default is one). Also, assuming the exchanges of data between robots are through wireless communication, we seek

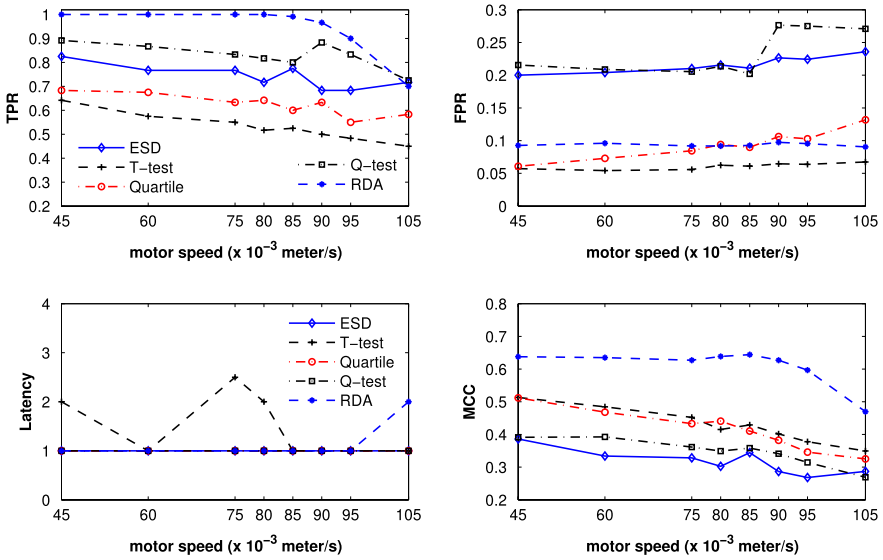


Fig. A.16. The median TPR, FPR, Latency, and MCC in detecting P_{PT} in V_{OPR}.

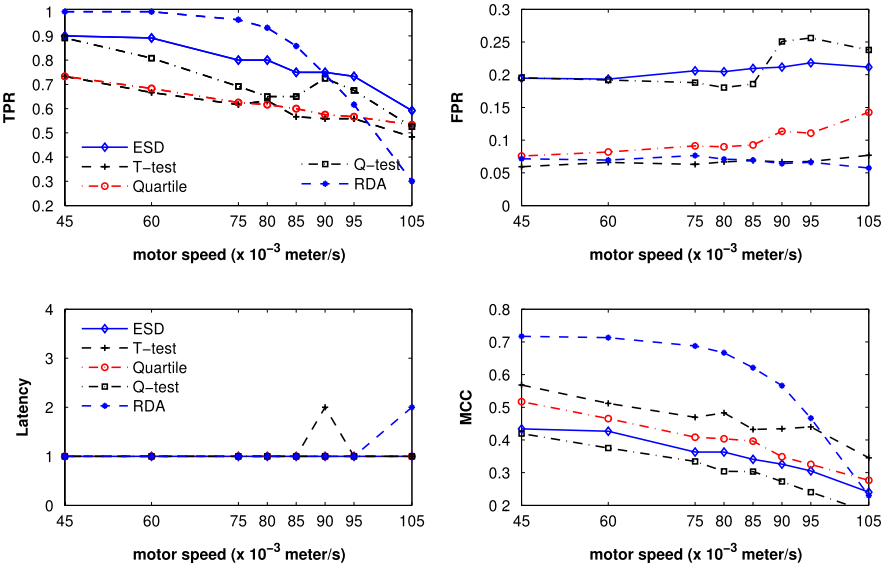


Fig. A.17. The median TPR, FPR, Latency, and MCC in detecting P_{PT} in V_{ODS}.

ways to reduce the communication overhead whilst maintaining the performance.

6.1. Reducing FPR by increasing the size of the detection window

In [29], it has been demonstrated that by increasing the size of the detection window (DW), sporadic or spikes in the data can be filtered and thus a lower FPR can be obtained. This assumes that faults last longer than the size of the DW. A DW of size one means that an error is flagged if it is detected at current (one) control cycle. If the size of the DW is two, an error is only flagged if it is detected for two consecutive control cycles (i.e. previous and current control cycle). This measure is appropriate with the experiments in this paper because the faults persist after being injected to a robot. However, there is a tradeoff between the Latency and the TPR due to the size of the DW. A larger DW will result in a longer Latency. An increase in the DW is also likely to result in reduction in the TPR as an error is only reported if it was detected consecutively within that DW. This means that one misclassified error (at a control cycle) will result in two false negatives.

We analysed the same data used in Sections 5.2 and 5.3 using the RDA with a DW of size two (RDA-DW2) and the results are shown in Figs. 9 and 10. The graphs shows the difference in performance between the original RDA and RDA-DW2. A positive value means an increase in the performance metric, a zero means no difference, and a negative value means a decrease in the performance metric.

It can be seen that by increasing the DW from one to two, a lower TPR and FPR is obtained. Even with a decrease in TPR, a better overall performance as measure with MCC score is achieved as evident from the MCC score. As for the Latency, as expected, an increase from one control cycle to greater than or equal to two control cycles was observed.

6.2. Reducing communication overhead

For our implementations, we assume the communication between robots is through a wireless medium by broadcasting (note

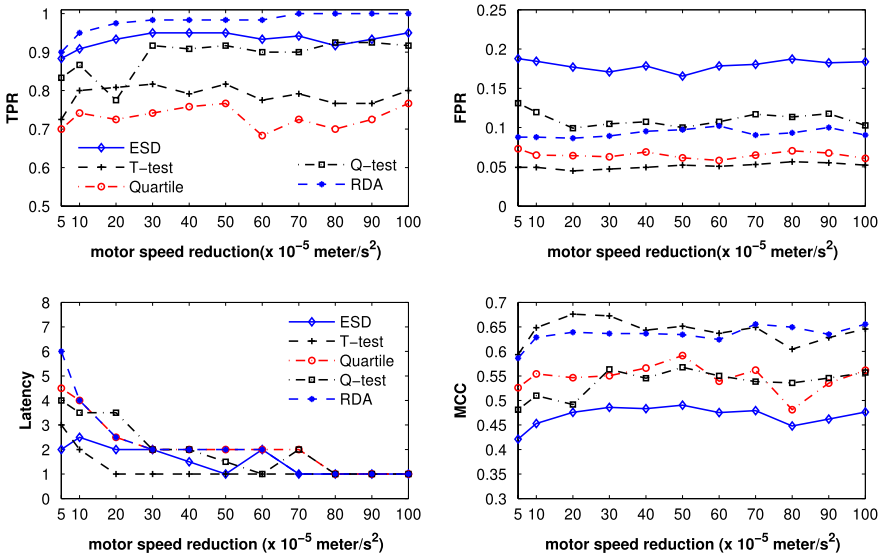


Fig. A.18. The median TPR, FPR, Latency, and MCC in detecting P_{GR} in V_{OPR}.

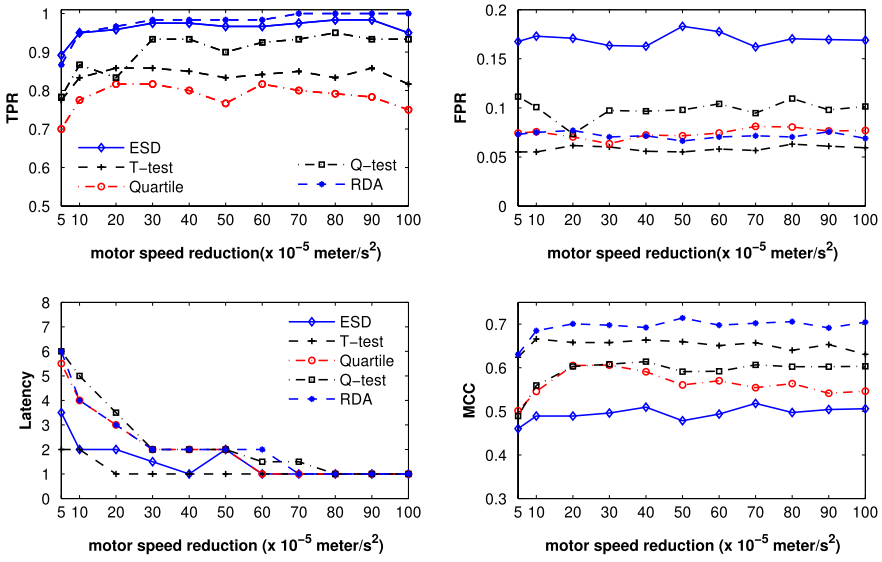


Fig. A.19. The median TPR, FPR, Latency, and MCC in detecting P_{GR} in V_{ODS}.

that possible problems with communication such as contentions, localisations, or transceiver are out of our research scope). However, wireless communications consume resources heavily [30]. Thus, we propose two communication strategies to work with the neighbourhood scheme. This does not apply if the exchange of data is through other mediums such as physical or stigmergic (indirect) interactions.

6.2.1. Strategy 1—optimistic communication strategy

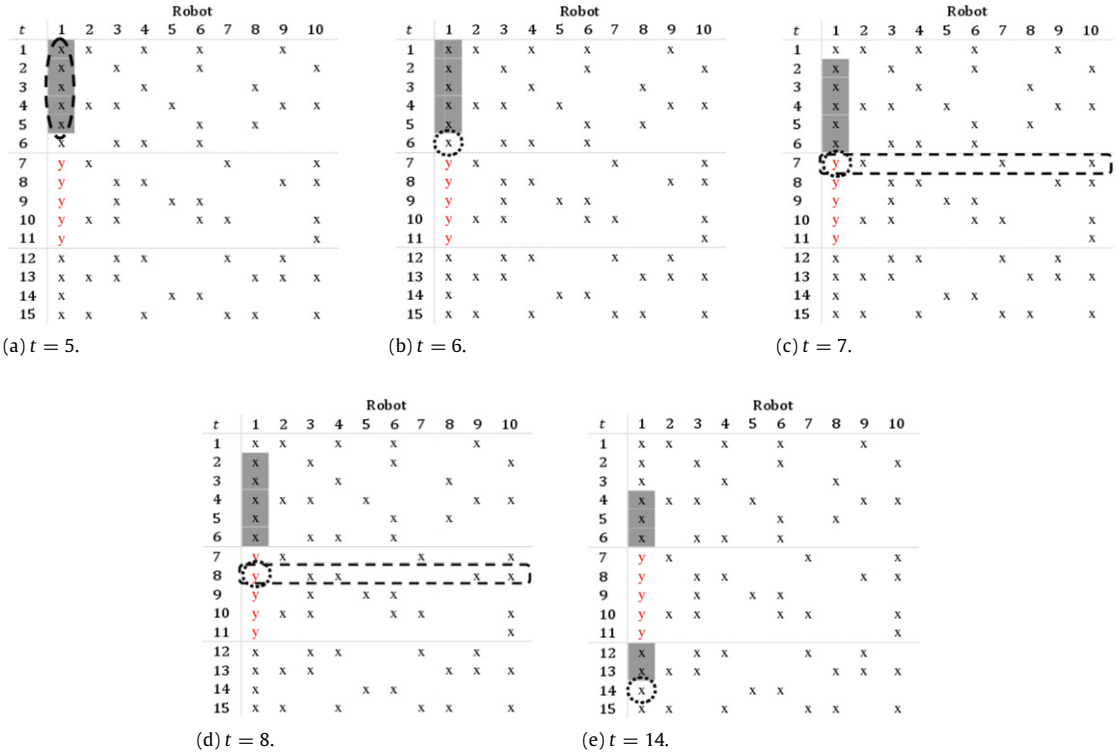
The first strategy is called Optimistic Communication Strategy in which external detection is only activated if an error is detected internally. For simplicity, we refer to it as S1. We differentiate between internal and external errors and refer to the former as err_{int} and err_{ext} for the later. Internal detection only deal with an individual robot's own data whilst external detection with a data stream from all robots within a robot's communication range.

Algorithm 1 is the pseudocode for S1. It starts with the initialisation of internal buffer $\mathcal{W}$ with a robot's own data from the first m control cycles (this is effectively using a time window as described in the first paragraph in Section 4). The size of the buffer

Input: current data instance v, neighbourhood data $\mathcal{DN}$, detection algorithm $\mathcal{A}$
Output: result of detection
initialise internal buffer $\mathcal{W}$;
foreach *control cycle t* **do**
  execute $\mathcal{A}(v, \mathcal{W})$;
  if err_{int} **then**
    execute $\mathcal{A}(v, \mathcal{DN})$;
    if err_{ext} **then**
      report err_{ext};
    else
      update($\mathcal{W}$);
    end
  else
    update($\mathcal{W}$);
  end
end

Algorithm 1: Pseudocode for Optimistic Communication Strategy

can vary; it is set to 5 in this research. After initialisation, at each control cycle t, the current data instance v is evaluated. If err_{int} is



(a) $t = 5$. (b) $t = 6$. (c) $t = 7$.

(d) $t = 8$. (e) $t = 14$.

Fig. A.20. Strategy S1: Optimistic Communication Strategy. Fault is introduced to robot R1 at $t = 7$ to $t = 11$. Data in $\mathcal{W}$ is shaded gray; dotted circle is current data instance; dotted rectangle is the neighbourhood data; x is normal; and y is anomalous. (a) Initialisation. Fill internal buffer $\mathcal{W}$ with data from $t = 1$ to $t = 5$. (b) Internal detection. Evaluate whether current data instance in circle is anomalous. If no err_{int} detected, update $\mathcal{W}$ with current data. (c) err_{int} is detected, get neighbourhood data. Detected err_{ext}, data will not be added to $\mathcal{W}$. (d) err_{int} and err_{ext} are detected. Same as (c). (e) Anomalous data are not added to $\mathcal{W}$. In this example, the number of inter-communication is only 5 whilst it is 15 with previous implementation.

detected, further analysis is carried out with neighbourhood data $\mathcal{DN}$. An error is considered to be detected if it is flagged by both the internal and external detection. If not, v replaces the oldest entry in $\mathcal{W}$. To illustrate how this strategy works, an example of error detection in robot R1 is shown in Fig. A.20.

6.2.2. Strategy 2—pessimistic communication strategy

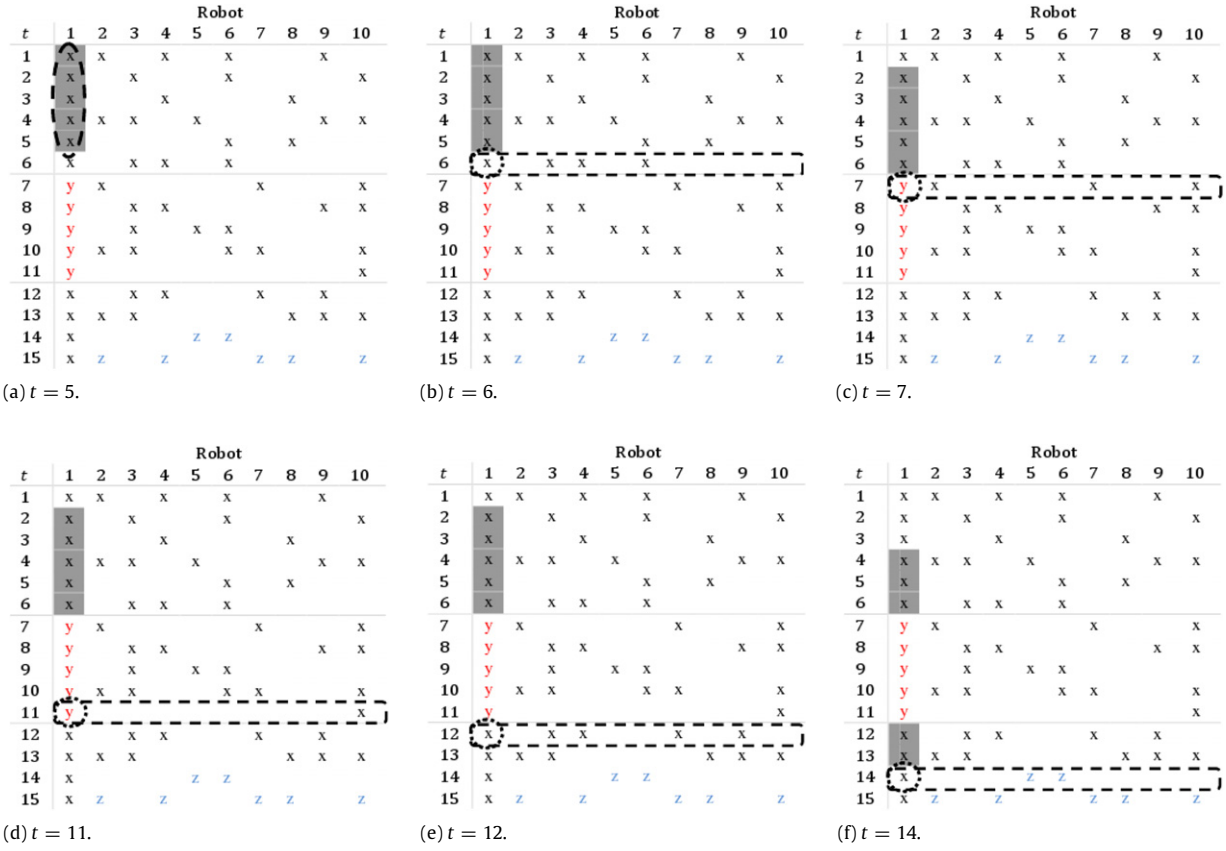
The second strategy is called Pessimistic Communication Strategy in which the frequency of external detection is based on the results from both internal and external detection. This strategy is similar to S1 but with the addition of a parameter c to control the frequency of communication. We refer to this strategy as S2.

Algorithm 2 is the pseudocode for S2. Similar to S1, it starts with initialisation of internal buffer $\mathcal{W}$. In addition, the external detection counter c is set to 1. This counter determines the next external detection, that is after the cth control cycle. For example, $c = 1$ means an external detection is performed at the first control cycle after the current cycle (which is the next control cycle), whereas $c = 4$ means an external detection at the 4th control cycle after the current cycle. The maximum value for c is controlled by the parameter K. An internal detection is still at every control cycle. If err_{int} is detected, subsequent external detection with neighbourhood data is conducted. If both err_{int} and err_{ext} are detected, an error is reported and c is reset to 1. Even if there is no err_{int}, external detection is still carried out subject to c. At this stage, if there is no err_{ext} the value of c is incremented by 1. The increment stops when c reaches K. Thus, the frequency of external detection changes according to both results from internal and external detection. An example of using S2 to detect errors in robot R1 is shown in Fig. A.21.

By examining the two strategies proposed, we can see that the number of communications is now proportional to the duration of faults and not the length of a simulation. This may significantly

Input: current data instance v, neighbourhood data $\mathcal{DN}$, detection algorithm $\mathcal{A}$, external detection parameter K
Output: result of detection
initialise sliding window $\mathcal{W}$;
counter $c \leftarrow 1$, $tc \leftarrow 0$;
foreach *control cycle t* **do**
  execute $\mathcal{A}(v, \mathcal{W})$;
  if err_{int} **then**
    execute $\mathcal{A}(v, \mathcal{DN})$;
    if err_{ext} **then**
      report err_{ext};
      $c \leftarrow 1$;
      $tc = t + c$;
    else
      update($\mathcal{W}$);
    end
  else
    if t == tc **then**
      execute $\mathcal{A}(v, \mathcal{DN})$;
      if err_{ext} **then**
        report err_{ext};
        $c \leftarrow 1$;
        $tc = t + c$;
      else
        if $c < K$ **then**
          c++;
          $tc = t + c$;
        end
      end
    end
  end
end

Algorithm 2: Pseudocode for Pessimistic Communication Strategy



(a) $t = 5$. (b) $t = 6$. (c) $t = 7$.

(d) $t = 11$. (e) $t = 12$. (f) $t = 14$.

Fig. A.21. Strategy S2: Pessimistic Communication Strategy. Fault is introduced to robot 1 at $t = 7$ to $t = 11$ and $t = 14$ to $t = 15$. Data in $\mathcal{W}$ is shaded gray; dotted circle is current data instance; dotted rectangle is the neighbourhood data; x is normal from R1's perspective; y is anomalous, and z is new normal. (a) Initialisation. Fill $\mathcal{W}$ with data from $t = 1$ to $t = 5$. Assign $c = 1$. (b) Internal detection and subsequent external detection. No err_{int} or err_{ext} detected, update $\mathcal{W}$ with current data. Increment c by 1, $c = c + 1$ (c) err_{int} is detected, communicate and evaluate data from neighbours to confirm. err_{ext} detected, $\mathcal{W}$ not updated and reset $c = 1$. (d) Both err_{int} and err_{ext} were detected, $\mathcal{W}$ not updated and reset $c = 1$. (e) No err_{int}, $c = c + 1 = 2$. (f) No err_{int} but due to c, external detection is conducted. err_{ext} was detected, reset $c = 1$. As for the number of inter-communication, it is 9 in this example, at $t = 6, 7, 8, 9, 10, 11, 12, 14, 15$.

reduce the communication overhead and thus power usage. Table 2 shows the number of communications (broadcasts) in the system in detecting $P_{PT} = 45 \times 10^{-3}$ m/s with the RDA. The robot R1 is faulty whilst the rest are fault-free. The number of broadcasts for R1 was greater than the number of faults (i.e. 60). For fault-free robots, the number of broadcasts was significantly lower, less than 5 with S1 and less than 25 with S2.

An obvious question is which strategy is better? The simple answer is it depends. S1 has the advantage of having the least communication overhead, subject to err_{int}. However, a misclassification is likely to be accumulated and affects the overall TPR as demonstrated in Fig. 11. In the figure, the TPRs over the 20 runs were plotted. It can be seen that at simulation number 9, 19, and 20, the TPRs are significantly different from other runs. This is the result of false negatives. On the other hand with S2, such false negatives were not accumulated as further cross-referencing were carried out, subject to the parameter c and K (refer Algorithm 2). Therefore, in some cases, there is a tradeoff between the communication overhead and the TPR. However, overall, both strategies do not result in a reduction of the error-detection ability as shown in Table A.3.

7. Conclusions and future work

Our work acknowledges the importance of fault tolerance in swarm robotics together with the complexities and challenges involved in achieving it through explicit detection. We have demonstrated the implementation of statistical classifiers for adaptive data-driven error detection and the results suggest that using statistical classifiers not only viable but practical in detecting different failure modes to the wheels under dynamic environments. We have also demonstrated that the RDA is a superior technique compared to classical statistical classifiers in our experiments. In addition, we have also explored a way to reduce the false positive rate by increasing the size of the detection window, and reducing the communication overhead for lighter resource usage with two communication strategies.

We acknowledge that our current work is based on simulation and results for the same experiments with physical robots might be different. However, the adoption of data-driven error detection, adaptive detection with neighbourhood scheme, and proposed communication strategies are still relevant and useful when dealing with uncertain environments. Thus, our future work involves the implementation of the statistical classifiers on Epuck[4] robots. In addition, we will also be looking into (simultaneous) failures on multiple robots.

Acknowledgments

The authors would like to thank Nick Owens and James Hilder for the help on RDA and thanks also to Mark Read on sensitivity analysis. This work is funded by Ministry of Higher Education of Malaysia KPT(BS) 761210136085 and Universiti Malaysia Sabah.

[4] www.e-puck.org.

Appendix. Referenced tables and figures

See Figs. A.12–A.21 and Table A.3.

References

[1] E. Şahin, Swarm robotics: From sources of inspiration to domains of application, in: E. Şahin, W. Spears (Eds.), Proc. 1st Int. Workshop Swarm Robotics 2004, Springer, 2005, pp. 10–20.
[2] L. Bayindir, E. Şahin, A review of studies in swarm robotics, Turkish Journal of Electrical Engineering and Computer Sciences 15 (2) (2007) 115–147.
[3] E. Şahin, S. Girgin, L. Bayindir, A. Turgut, Swarm robotics, in: C. Blum, D. Merkle (Eds.), Swarm Intelligence: Introduction and Applications, in: Natural Computing Series, Springer, 2008, pp. 87–100.
[4] A.F.T. Winfield, J. Nembrini, Safety in numbers: Fault tolerance in robot swarms, International Journal of Modelling, Identification and Control 1 (1) (2006) 30–37.
[5] J. Timmis, A. Tyrrell, M. Mokhtar, A. Ismail, N. Owens, R. Bi, An artificial immune system for robot organisms, in: P. Levi, S. Kernbach (Eds.), Symbiotic Multi-Robot Organisms: Reliability, Adaptability and Evolution, Springer, 2010, pp. 268–288.
[6] A. Avižienis, Design of fault-tolerant computers, in: Proceedings of the 1967 Fall Joint Computer Conference, 1967, pp. 733–743.
[7] A.L. Christensen, Fault detection in autonomous robots, Ph.D. Dissertation.
[8] R. Oates, J. Greensmith, U. Aickelin, J.M. Garibaldi, G. Kendall, The application of a dendritic cell algorithm to a robotic classifier, in: Proc. 6th Int. Conf. Artificial Immune Systems, Springer, 2007, pp. 204–215.
[9] R. Canham, A.H. Jackson, A. Tyrrell, Robot error detection using an artificial immune system, in: Proc. 2003 NASA/DoD Conf. on Evolvable Hardware, IEEE, 2003, pp. 91–100.
[10] M. Mokhtar, J. Timmis, A. Tyrrell, R. Bi, A modified dendritic cell algorithm for on-line error detection in robotic system, in: Proc. Congress on Evol. Comput. 2009, IEEE Press, 2009, pp. 2055–2062.
[11] L. Festinger, A theory of social comparison processes, Human Relations 7 (2) (1954) 117–140.
[12] R. Bi, J. Timmis, A. Tyrrell, The diagnostic dendritic cell algorithm for robotic systems, in: Proc. World Congress on Comput. Intell. 2010, IEEE Press, 2010, pp. 4280–4289.
[13] A. Winfield, Foraging robots, in: R. Meyers (Ed.), Encyclopedia of Complexity and System Science, Springer, 2009, pp. 3682–3700. doi:10.1007/978-0-387-30440-3_217.
[14] R. Brooks, A robust layered control system for a mobile robot, IEEE Journal of Robotics and Automation 2 (1) (1986) 14–23.
[15] W. Liu, Design and modelling of adaptive foraging in swarm robotic systems, Ph.D. Thesis, University of the West of England, 2008.
[16] B. Gerkey, R.T. Vaughan, A. Howard, The player/stage project: tools for multi-robot and distributed sensor systems, in: Proc. 11th Int. Conf. Advanced Robotics, 2003, pp. 317–323.
[17] R.D. Gibbons, Statistical Methods for Groundwater Monitoring, John Wiley & Sons, Inc., 1994.
[18] N. Owens, A. Greensted, J. Timmis, A. Tyrell, *T* cell receptor signalling inspired kernel density estimation and anomaly detection, in: P. Andrews, J. Timmis, N.D. Owens, U. Aickelin, E. Hart, A. Tyrell (Eds.), Proc. 8th Int. Conf. Artificial Immune Systems, Springer, 2009, pp. 122–135.
[19] N.D. Owens, From biology to algorithms, Ph.D. Thesis, University of York, 2010.
[20] F.E. Grubbs, Procedures for detecting outlying observations in samples, Technometrics 11 (1) (1969) 1–21.
[21] V.J. Hodge, J. Austin, A survey of outlier detection methodologies, Artificial Intelligence Review 22 (2004) 85–126.
[22] W. Trochim, J.P. Donnelly, The Research Methods Knowledge Base, Atomic Dog Publishing, 2007.
[23] J.W. Tukey, Exploratory Data Analysis, Addison-Wesley, 1977.
[24] NIST/SEMATECH, e-handbook of statistical methods, 2010. URL: http://www.itl.nist.gov/div898/handbook/ [Online; accessed 16.11.10].
[25] N. Owens, J. Timmis, A. Greensted, A. Tyrell, Modelling the tunability of early *T* cell signalling events, in: P. Bentley, D. Lee, S. Jung (Eds.), Proc. 7th Int. Conf. Artificial Immune Systems, Springer, 2008, pp. 12–23.
[26] N.D. Owens, J. Timmis, A. Greensted, A. Tyrrell, Elucidation of *T* cell signalling models, Journal of Theoretical Biology 262 (3) (2010) 452–470.
[27] J.A. Hilder, N.D.L. Owens, Peter J. Hickey, S.N. Cairns, D.P.A. Kilgour, J. Timmis, A.M. Tyrrell, Parameter optimisation in the receptor density algorithm, in: Proc. 11th Int. Conf. Artificial Immune Systems, Springer, 2011, pp. 226–239.
[28] B. Matthews, Comparison of the predicted and observed secondary structure of t4 phage lysozyme, Biochimica et Biophysica Acta 405 (1975) 442–451.
[29] A.L. Christensen, R. O'Grady, M. Birattari, M. Dorigo, Automatic synthesis of fault detection modules for mobile robots, in: Proc. 2nd NASA/ESA Conf. Adaptive Hardware and Syst., 2007, pp. 693–700.
[30] L. Doherty, B.A. Warneke, B.E. Boser, K. Pister, Energy and performance considerations for smart dust, International Journal of Parallel and Distributed Systems and Networks 4 (3) (2001) 121–133.

HuiKeng Lau received his B.IT (Hons) from Universiti Malaysia Sarawak, Malaysia in 2001 and M.Sc. from Universiti Teknologi Malaysia, Malaysia in 2006. He is currently a Ph.D. student at the Computer Science Department at the University of York, UK and his research interests include anomaly detection, swarm robotics, artificial immune systems, wireless sensor networks and artificial intelligence.

Iain Bate is a lecturer in Real-Time Systems at the Computer Science Department, University of York. His research interests include scheduling and timing analysis, design and analysis of safety-critical systems, and engineering of complex systems of systems including sensornets. He is the Editor-in-Chief of the Journal of Systems Architecture and a frequent member of programme committees for distinguished international conferences.

Paul Cairns received his BA in Mathematics from the University of Oxford in 1991 and his D.Phil in General Topology also from Oxford in 1995. He is currently a senior lecturer in HCI at the Computer Science Department, University of York. His research interests are in Human Computer Interaction generally but, with a background in mathematics, also in statistical methods for understanding user behaviour and mathematical knowledge management. Recently, he developed an interest in understanding the positive experience of using interactive systems, in particular, understanding what it means to be immersed in video games.

Jon Timmis received his B.Sc. (Hons) in Computer Science from the University of Wales, Aberystwyth in 1997, and his Ph.D. in Computer Science in 2000. He is currently a Professor of Natural Computation and hold a joint appointment between the Department of Computer Science and the Department of Electronics, University of York. His research interests includes artificial immune systems, bioinspired computing, biological modelling, fault tolerance and dynamic learning.