Collective Self-detection Scheme for Adaptive Error Detection in a Foraging Swarm of Robots

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

[1] Department of Computer Science, University of York
[2] Department of Electronics, University of York, Heslington, YO10 5DD, UK
[3] School of Engineering and IT, Universiti Malaysia Sabah, 88999 Kota Kinabalu, Sabah, Malaysia
{hklau,iain.bate,jtimmis}@cs.york.ac.uk

Abstract. In this paper we present a collective detection scheme using receptor density algorithm to self-detect certain types of failure in swarm robotic systems. Key to any fault-tolerant system, is its ability to be robust to failure and have appropriate mechanisms to cope with a variety of such failures. In this work we present an error detection scheme based on T-cell signalling in which robots in a swarm collaborate by exchanging information with respect to performance on a given task, and self-detect errors within an individual. While this study is focused on deployment in a swarm robotic context, it is possible that our approach could possibly be generalized to a wider variety of multi-agent systems.

Keywords: swarm robotics, error detection, receptor density algorithm, collective detection scheme, self-detection.

1 Introduction

Swarm robotic systems (SRS) refer to systems with a large number of simple and physically homogeneous robots interacting with each other and the environment to achieve certain tasks [3]. In order to allow a swarm of robots to perform its task over extended periods of time, the swarm needs to be tolerant to failures that can occur within the swarm. Distributed autonomous systems, such as SRS, are susceptible to failure, and as recently shown by [11] the assumption that SRS are immune to such issues is not necessarily the case. In this paper, we focus on producing a fault-tolerant swarm of robots, but rather than having a central point of control for the identification of errors within the swarm, the swarm itself is responsible for the detection. Therefore the swarm collectively self-monitors and identifies errors within the swarm, which in principle would allow for a greater degree of fault tolerance.

Implicit redundancy and explicit error detection-and-recovery are two ways to address fault tolerance in swarm robotics. With redundancy, uncompleted tasks by a failed agent is taken over by a redundant agent in the system. This approach has typically been preferable as it is straightforward to implement [11]. It works well when a minimal number of *healthy* robots are still available to complete a given task and also that failures are independent, i.e., faulty robots do not have undesirable effects on the overall swarm. However, these conditions do not always hold. Winfield and Nembrini [11] showed that a few failing robots can significantly affect the overall swarm and should (if possible) be repaired. Initial work into such a self-repairing approach has been outlined in [10], but assumes that an effective error detection system is in place. To activate any recovery measure, we need to first detect the error and identify the faults. This is the second approach to achieve a fault-tolerant system. These two approaches do not have to be used in isolation but rather are used together to complement each other for an improved overall performance.

With respect to error detection in swarm robotics, limited work is available especially when dealing with dynamically changing environmental conditions. There are many challenges in developing an error detection system in swarm robotics. First we require an accurate detection of errors, second we require an adaptive and low resource solution due to potential limitations of the hardware platform. Third, a quick response time coupled with robustness in detection are desirable, it is no use to detect an error in a time frame that makes it impossible to do any form of recovery or response.

In this work, we make use of an immune-inspired solution that has many of the properties we require for a detection system: it is lightweight in terms of computational overhead, it is adaptive and superior in performance to more traditional statistical approaches (such as quartile- and T-test based algorithms). We employ a T-cell signaling inspired algorithm called the Receptor Density Algorithm (RDA) [7] as part of a collective self-detection scheme and in this paper, we analyze the approach with respect to accuracy, adaptivity, responsiveness,and robustness in detection.

The rest of the paper is structured as follows. Section 2 briefly describes error detection in swarm robotics and the use of RDA as an error detection method. Section 3 provides details on the experimental set up. The results and discussions are presented in Section 4 and conclusions in Section 5.

2 Background and Related Work

2.1 Error Detection in Swarm Robotics

SRS are subject to anomalies due to reasons such as faulty hardware components, design errors, or deliberate sabotage [11]. To ensure dependability, these faults need to be dealt with to avoid more faults and errors. The error detection-and-recovery involves a 3-stage process: error detection, fault diagnosis, and recovery. Error detection examines the system's behaviour for errors. If an error has been detected, fault diagnosis is activated to identify the faults followed by corresponding recovery measures.

In swarm robotics, the work on error detection is limited. However, in robotics in general, neural networks and its variations have been used to detect faults in the joints of the robotic manipulator [9], wheels of a Robuter [8], and *treels* (combined tracks and wheels) [1]. These techniques work very well with the system behaviour being non-dynamic, i.e., not affected by the changes in the operational environment. Otherwise, re-training is required. This is undesirable as prior knowledge of changes might not be available. With SRS normally deployed in a dynamic environment, it is crucial for the detection to be adaptive. This is the focus of our work.

Mokhtar et al. [5] implemented a dendritic cell inspired error detection in a resource limited micro-controller as part of an integrated homeostatic system in SYMBRION project[1]. Their focus is on single individual detection utilizing only individual's own sensor data, i.e., standalone detection. The drawback with standalone detection is that a change in the behaviour of a robot could be caused either by faults or the external effects, e.g., change in the environment. Our work with collective detection scheme takes advantage of local interactions among individuals to exchange information for self-detection of errors. By cross-referencing one robot's behaviour with others, a more accurate detection can be achieved (Fig. 1). In Fig. 1, robot R1 exchanges its data with other robots within its communication range namely robot R2, R3 and R5. Note that the neighbourhood at different time instance varies as the robots move around in the environment.

2.2 Receptor Density Algorithm

Robots in SRS are typically simple with limited communication ability, processing and memory. For example, the e-puck robot is equipped with 64 MHz CPU with 16 MIPS peak processing power, 4kB RAM and 144kB flash memory [6]. Therefore, the error detection mechanism has to be lightweight and statistical methods such as the RDA offer a potential solution.

Owens *et al.* introduced the RDA in [7] an algorithm developed through the study of T Cell receptors' signaling mechanisms in the immune system. By extracting features of these receptors, a mapping was made onto kernel density estimation, a technique from statistical machine learning. Assuming the RDA is on R1 in Fig 1, it works as follows (illustrated in Fig. 2):

Step 1: Training

1. Calculate total stimulation $S(x)$ on each receptor x by input x_i from nb robots in a communication range of a robot (Fig. 2(a)). For R1 at T7, nb = 3, n = 4, and considering only variable a as input, $x_1 = {}^{R2}a_7$, $x_2 = {}^{R3}a_7$, and $x_3 = {}^{R5}a_7$.

$$S(x) = \sum_{i=1}^{nb} \frac{1}{n \times h} K_s(\frac{x - x_i}{h}) \tag{1}$$

where $K_s(x)$ is the kernel, h is the kernel width, and n is total number of robots in a robot's communication range including itself.

[1] SYMBRION - Symbiotic Evolutionary Robot Organisms project (`http://www.symbrion.eu`)

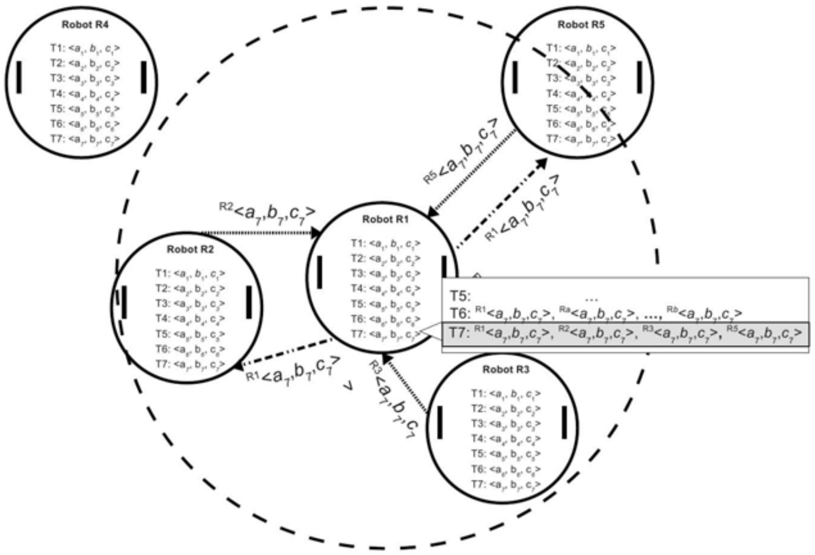


Fig. 1. Collective self-detection with exchanges of data between robots. The communication range of robot R1 is indicated by a dotted circle, and periodically R1 communicates (exchange data) with other robots within this range as indicated by dotted arrows. Rx <a$_t$, b$_t$, c$_t$ > is the data vector for robot Rx at time t. At current time T7, R1 exchanges data with robot R2, R3, and R5 and thus has input data from four robots (including itself) as highlighted in the diagram.

2. Calculate negative feedback $neg(x)$ for each receptor x (Fig. 2(b)).

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

where β is the base negative barrier.

Step 2: Testing

1. Set the receptor position $r_p(x)$=0 for all receptors.
2. Set receptor length l to the maximum height of the stimulation kernel K_s scaled by n and h, $l = \frac{1}{n \times h(\sqrt{2\pi})}$.
3. Calculate updated receptor position $r_p^*(x)$ with input v ($^{R1}a_7$) from the robot (Fig. 2(c)).

$$r_p^*(x) = b \times S(x) + gb \times K_s(\frac{x - v}{h}) - \alpha \times neg(x) \tag{3}$$

where b is receptor position's decay rate, gb is current input stimulation rate,

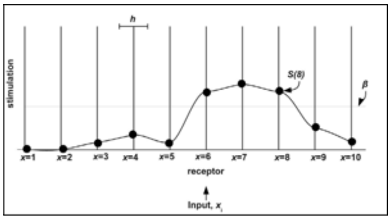
α is negative feedback's stimulation rate.

4. Classify v:

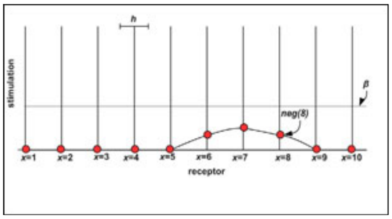
$$v = \begin{cases} Normal, & \text{if } r_p^*(x) < l \\ Anomaly, & otherwise \end{cases} \quad (4)$$

Step 3: Repeat 1 and 2 for every control cycle

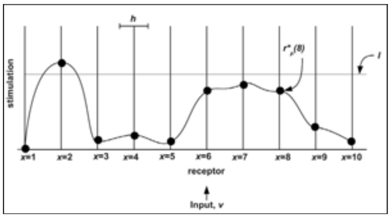
An receptor x is a point in the kernel density estimate. We normalized these points to fall within an interval of expected values for each variable of interest. For instance, if we choose to use 20 receptors to represent the kernel density estimate and the minimum and maximum value for input data is 0 and 100. By equally spacing the receptors to fall between 0 and 100, we have receptors from 0 to 100 evenly spaced by $\frac{100-0}{20-1}$. The variable x_i is input data from other robots that are within a communication range from a robot running the detection algorithm. Base negative feedback β, b, gb, and α is a constant to control the level of stimulation and suppression on each receptor.



(a) calculate stimulation from input



(b) calculate negative feedback



(c) calculate updated receptor position

Fig. 2. An illustration on using the RDA for error detection in a foraging SRS. (a) Calculate the stimulation level of each receptor from input data of neighbouring robots of a communication range. (b) If the stimulation level of a receptor is higher than the base negative barrier β, negative feedback is generated. (c) The stimulation level (receptor position) is updated when input from current robot is added. If any resulting receptor position is higher than a maximum stimulation level (receptor length) l, an error is detected, as seen at receptor x=2.

3 Experimental Setting

3.1 Simulation Setting

The foraging SRS is simulated with Player/Stage[2] [4]. Ten robots are placed within a 10-metre x 10-metre bounded octagonal shaped arena with a circular base at the centre. The task of the robots is to continuously search for, collect, and deposit objects at the base, this is a typical foraging task used in SRS. New objects are added to the arena at a rate referred to as the object placement rate (`OPR`) with `OPR`=0.10 in a non-dynamic environment.

A simulation begins with 100 initial objects placed randomly in the arena with a maximum of 200 objects at any time instance to avoid overcrowding. At time 5000s, a fault is injected to one robot and the fault persists until the end of simulation. Each simulation lasts for 20000s and this is further split into a smaller interval called a control cycle of 250s. Thus, there are 20000/250 = 80 control cycles in each simulation. A control cycle of a smaller length is possible and it is very much dependent on the specific task and input data. We chose 250s to reduce the amount of data to process and to smooth irregularities that might otherwise exist in one of the input data, i.e., number of objects collected. At each control cycle, robots exchange their behavioural data with others in a communication range. The data consists of the number of objects collected (V1), energy used (V2), and distance travelled (V3). We recorded the data[3] as comma separated variable (CSV) files and analyzed them off-line so that the same data can be used for further analysis, fair comparisons with other methods and parameters tuning. Each scenario is repeated 20 runs.

For the RDA, the initial setting was obtained from manual inspections: number of receptor = 20, β=0.01, b=0.1, gb=1.1, α=1, the interval for V1=[0,8], V2=[0,300], V3=[0,40], and the kernel is Gaussian kernel with kernel width h_{v1}=1, h_{v2}=12, and h_{v3}=3. We discuss the tuning process of these parameters in section 4.2.

3.2 Modes of Failure

Component faults on a robot can be due to wear-and-tear, power loss or damaged circuitry connections. These faults can occur either instantaneously or gradually, and can be either complete failure or partial failure. We simulate three failure modes on the robot wheels: complete failure ($\mathtt{P_{CP}}$), partial failure ($\mathtt{P_{PT}}$), and gradual failure ($\mathtt{P_{GR}}$). $\mathtt{P_{CP}}$ and $\mathtt{P_{PT}}$ occur instantaneously while $\mathtt{P_{GR}}$ is gradual.

With $\mathtt{P_{CP}}$, robot wheels stop responding completely by moving in circles and are unable to move to target objects. With $\mathtt{P_{PT}}$, the wheels suffer a sudden failure and the speed is reduced instantaneously to x meter/s from normal speed of 0.15 meter/s, while with $\mathtt{P_{GR}}$ the robot moves with gradually reducing speed by y meter/s per second. These faults are highly possible due to power loss

[2] Player 2.1 and a modified version of stage 2.1.0 from `http://www.brl.uwe.ac.uk/projects/swarm/index.html`

[3] Data online at `http://sites.google.com/site/researchmaterialshkl/data`

or damaged circuitry connections. Unless specified otherwise, $\mathtt{P}_{\mathtt{PT}}$ is simulated with an instantaneous motor speed reduction to 0.045 meter/s while $\mathtt{P}_{\mathtt{GR}}$ with a gradual speed reduction of 100 x 10 $^{-5}$ meter/s^2. These values are appropriate as they range from the easier to detect complete failure to more difficult gradual failure that are still feasible to detect.

3.3 Dynamic Environment

We simulate scenarios in which the concentration (or the availability) of target objects change with time. Three conditions for our SRS: non-dynamic (CST), varying object placement rate ($\mathtt{V}_{\mathtt{OPR}}$), and varying object distribution ($\mathtt{V}_{\mathtt{ODS}}$). In CST, the OPR is fixed at 0.1 with homogeneous object distribution. In $\mathtt{V}_{\mathtt{OPR}}$, the OPR in the arena changes between 0.1 and 0.025, and in $\mathtt{V}_{\mathtt{ODS}}$ the spatial distribution of objects in the arena is biased between top right and bottom left regions in the arena. Dynamic scenarios are simulated with a 2-cycle configuration in which dynamic changes occur at control cycle 20-40 and 60-80.

3.4 Performance Metrics

The performance of detection is evaluated based on the false positive rate (FP), the responsiveness (Latency), and true positive rate (TP). The Latency is time elapsed from the moment of fault injection until the moment of detection. For our problem, we are not only interested whether an error is detected but also how fast it can be detected. This is to prevent faulty robots being a cause of further disruptions to the rest of the swarm. Note that the Latency is also dependent on the control cycle length, a smaller control cycle may results in faster respond.

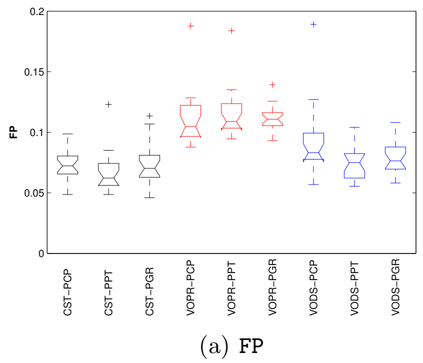
Given,
$N(pos)$ = Number of positives,
$N(tpos)$ = Number of true positives,
$N(neg)$ = Number of negatives,
$N(fpos)$ = Number of false positives,
T_{pd} = Fault detection time (in control cycle),
T_{ft} = Fault injection time (in control cycle),

Then, $\mathtt{FP} = \frac{N(fpos)}{N(neg)}$, and $\mathtt{Latency} = T_{pd} - T_{ft}$, $\mathtt{TP} = \frac{N(tpos)}{N(pos)}$.

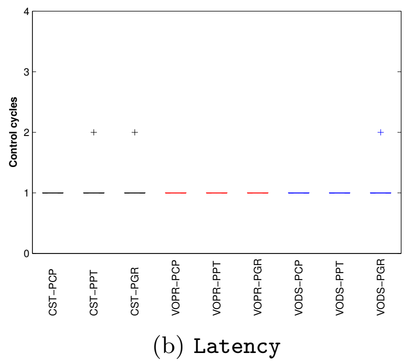
4 Results and Discussion

4.1 Performance

Boxplots of the FP from the 20 evaluation runs are shown in Fig. 3(a). In each box, the centre line 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 interquartile range. Outliers are plotted individually. In the figure, the results are plotted according to the operational environment with the first 3



(a) FP



(b) Latency

Fig. 3. Boxplots of the FP and Latency for collective self-detection of errors with the RDA

boxplots for CST followed by V_{OPR}, and finally V_{ODS}. The median FP is approximately 0.1 in all scenarios. Two important observations on the results. Firstly, the collective scheme with RDA produces a low FP. Secondly, the FP is consistent across all scenarios including those in dynamic environments. This shows that the proposed method is adaptive to environmental changes. Otherwise, the FP would be much higher ($\geq$0.5) in both V_{OPR} and V_{ODS}. We also note that in every trial, the error was detected (TP=1).

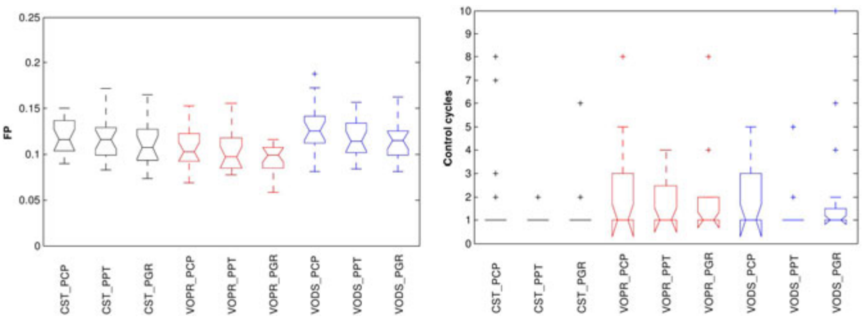
The results for Latency are shown in Fig. 3(b). The median response time is 1 control cycle for all scenarios. Since an evaluation is at every control cycle, this response is immediate. Similarly, the consistency in the Latency for all scenarios signify the ability to adapt accordingly.

We compared the results of proposed method using the RDA with two other detection algorithms based on quartiles and T-test. Results for quartile-based algorithm in Fig. 4 and T-test in Fig. 5. The RDA outperforms the other methods with a lower FP and a lower and more consistent Latency. These results are very encouraging. They indicate that through collective detection scheme with the RDA, an error within a robot can be self-detected within a short time even under dynamic environments. These results provide motivation to investigate other aspects such as fine-tuning the system (section 4.2) and the robustness of detection (section 4.3).

4.2 Parameter Tuning

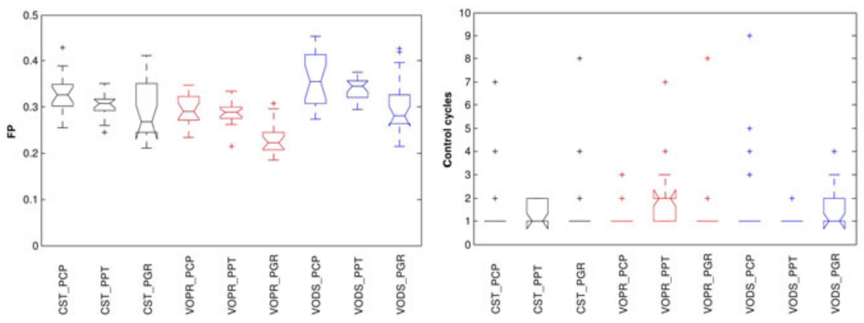
We adopted the hill-climbing method in tuning the RDA's parameters. A chosen starting value is gradually increased or decreased by a small constant amount to observe its effect on the performance metrics. One parameter is tuned at a time, and a found optimal value is then used for subsequent tuning of other parameters.

Fig. 6 shows results for the tuning of V1's kernel width (h_{v1}) with a starting value of 0.0 and an increment of 0.2. The results of FP and Latency are plotted



(a) `FP` (b) `Latency`

Fig. 4. Results of (a) `FP` and (b) `Latency` with quartile-based detection



(a) `FP` (b) `Latency`

Fig. 5. Results of the (a) `FP` and (b) `Latency` with T-test based detection

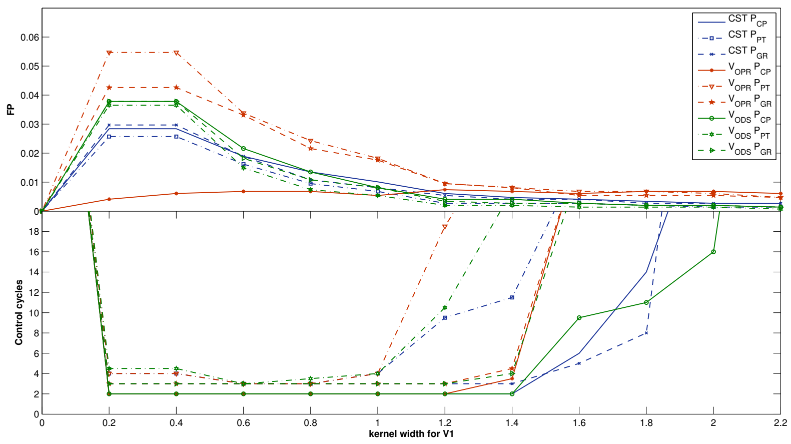


Fig. 6. Graphs of the median `FP` and `Latency` with different values of h_{v1}

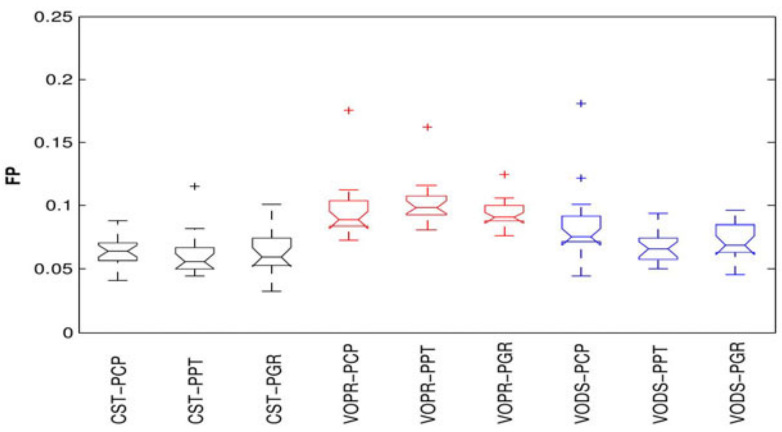


Fig. 7. Boxplots of the `FP` in detecting injected faults with the tuned RDA

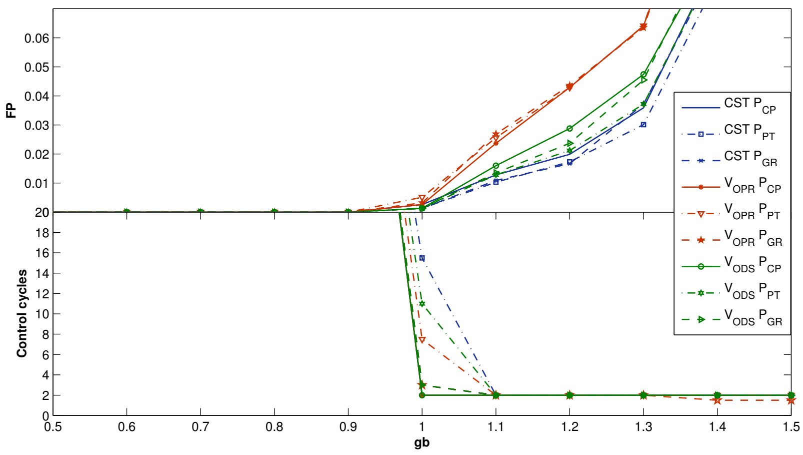
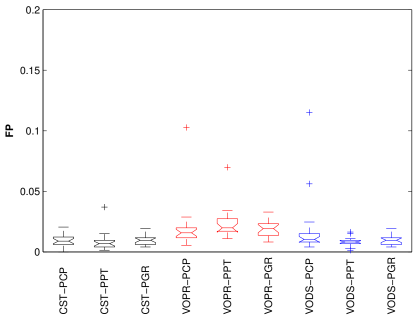


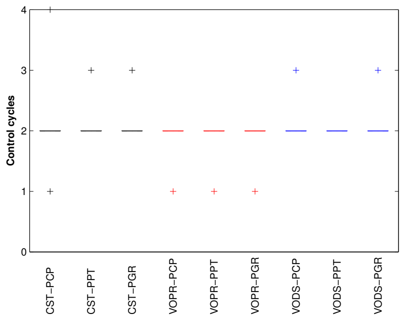
Fig. 8. Graphs for the median `FP` and `Latency` with different values of gb

side-by-side to show the effect of different h_{v1} on the performance. On the top figure, a sharp drop of `FP` is seen at h_{v1}=0.4 until h_{v1}=1.2 where the decrease became more subtle. On the bottom figure, a small drop in `Latency` is seen at h_{v1}=0.4 until h_{v1}=0.6 where it starts to rise. At h_{v1}=1.0, the latency rises sharply with $\texttt{P}_{\texttt{PT}}$ errors and eventually followed by $\texttt{P}_{\texttt{CP}}$ and $\texttt{P}_{\texttt{GR}}$ at h_{v1}=1.4. Here, a suitable value for h_{v1} can be selected depending on the preference of the performance metrics. For example, if a requirement of `FP`$\leq$0.04 and `Latency`$\leq$5 is given, h_{v1} within the range of 0.6 and 1.0 can be used.

By analyzing the results for each RDA parameter through hill-climbing, we have found a set of optimal value for our problem with h_{v1}=1.0, h_{v2}=12, h_{v3}=2.5, b=0.02, gb=1.1, α=1.7. `FP` results in Fig. 7 with these values showed that a lower `FP` is obtained in all scenarios compared to initial results in Fig. 3(a), significantly



(a) `FP`



(b) `Latency`

Fig. 9. Boxplots of the `FP` and `Latency` detecting injected faults with a size 2 detection window

better in $\mathtt{V}_{\mathtt{OPR}}$. The results for the median `Latency` and `TP`, on the other hand, are exactly the same (as in Fig. 3(b)) and are thus omitted here. These results show that through parameter tuning, the performance can be improved.

From the same tuning exercise, we found some interesting observations. For example, the parameter b and α has no obvious effect on the latency of detection, i.e., irrespective of what value of b and α, the `Latency` remained constant at 2. For these parameters, the selection for the optimal value is based solely on the `FP` results.

For parameter gb, its influence on `Latency` appears to be one-sided (Fig.8). From the figure, the `Latency` changes drastically as the value of gb approaching 1.1. Beyond that point it remained constant and unchanged irrespective of the value of gb. This is an interesting observation but can be explained as the role of gb is to control the amount of stimulation from test data point. Higher stimulation pushes the receptor position $r_p^*(x)$ beyond receptor length l and thus an earlier detection. However, note that an earlier detection does not mean a positive detection.

In order to further reduce the `FP`, we implemented a similar mechanism as in [2] by increasing the size of detection window (DW) from 1 to 2. This means that a positive detection requires a detection for two consecutive control cycles. We call this RDA-DW2. The results (Fig. 9(a)) showed that a significantly lower `FP` is produced. A bigger DW helps in reducing false alarms because a sudden and drastic change in behaviour that last only one control cycle will be ignored. On the contrary, by increasing the size of DW also meant that a longer response time is involved. The `Latency` is directly related to the size of DW, Latency $\geq$ DW. With DW=2, the median `Latency` also increased to 2 control cycles (Fig. 9(b)).

4.3 On Robustness

We view the robustness of an error detection method from 2 perspectives: scalability in implementation and scalability in detection. Our implementation of

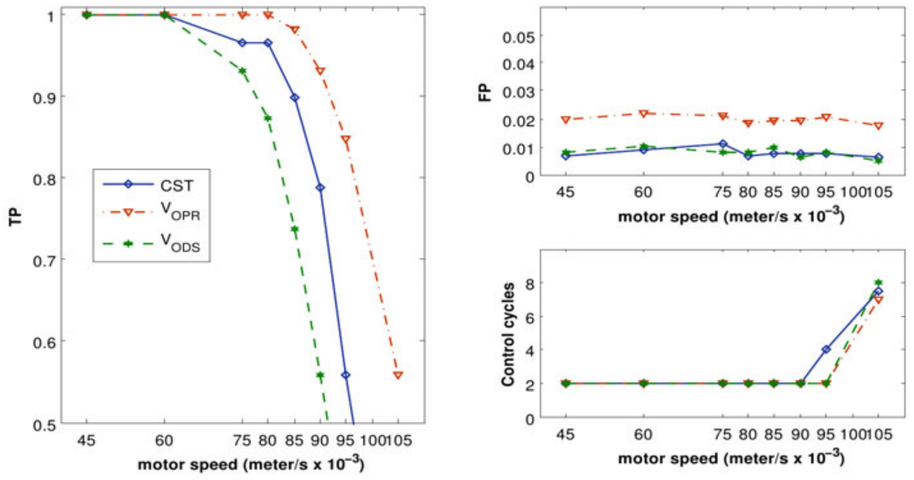


Fig. 10. The graphs of the median `TP`, `FP` and `Latency` in detecting various magnitudes of P_{PT}

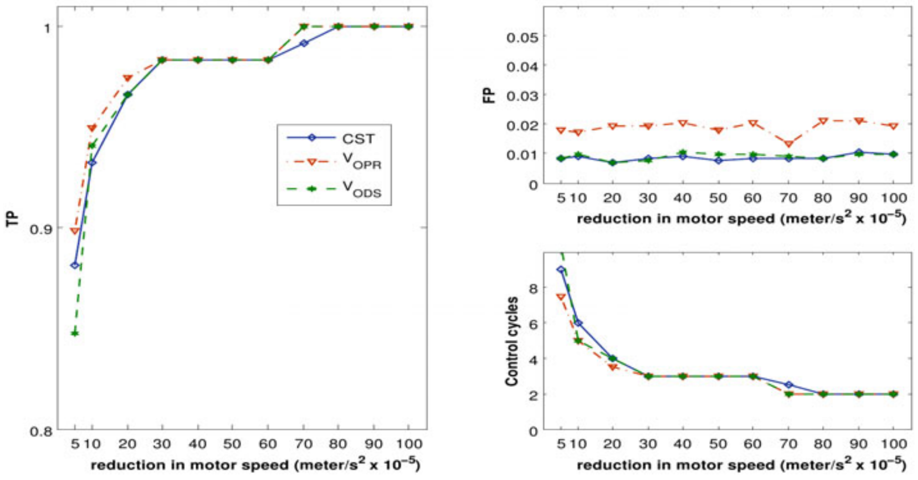


Fig. 11. The graphs of the median `TP`, `FP` and `Latency` in detecting various magnitudes of P_{GR}

the RDA-DW2 is on every robot and thus it is distributed and should scale to a larger swarm. To be aware of the magnitudes of faults detectable with our implementation, we conducted further experiments by injecting $\mathtt{P_{PT}}$ and $\mathtt{P_{GR}}$ with a range of magnitudes. If such a range can be established, we can then be aware of the detection capability of our implementation for informed decision for recovery actions.

We tested $\mathtt{P_{PT}}$ with different magnitudes: 0.105, 0.095, 0.090, 0.085, 0.080, 0.075, 0.060 and 0.045 meter/s. With $\mathtt{P_{PT}}$, a bigger value signifies a more subtle fault. Results for the median `FP` and median `Latency` are shown in Fig. 10. From the figure, the median `Latency` is 2 for $\mathtt{P_{PT}} \leq 0.090$ meter/s and starts to increase sharply with $P_{PT} > 0.090$ meter/s. On the other hand, the median `FP` does not change much. From the results, we know that the critical point of an increase in the `Latency` is at $\mathtt{P_{PT}}$=0.090 meter/s. This mean that for $P_{PT} > 0.090$ meter/s, the changes in behaviour of a faulty robot is too subtle, for this algorithm, to indicate the presence of an error. If so desired, re-tuning of parameters can be carried out to increase the sensitivity of detection. However, an increase in detection sensitivity also increases the false alarm rate.

For $\mathtt{P_{GR}}$, we tested $\mathtt{P_{GR}}$ with magnitudes from 5 x 10^{-5} to 100 x 10^{-5} meter/s^2. The interpretation of the magnitudes for $\mathtt{P_{GR}}$ is the direct opposite with $\mathtt{P_{PT}}$. With $\mathtt{P_{GR}}$, a smaller value signifies a more subtle fault. Results (Fig. 11) shows four trends in the latency of detection: $\mathtt{P_{GR}} <$ 30 x 10 $^{-5}$ meter/s^2 (T1), 30 x 10 $^{-5}$ meter/s^2 $\leq \mathtt{P_{GR}} \leq$ 50 x 10 $^{-5}$ meter/s^2 (T2) , 50 x 10 $^{-5}$ meter/s^2 $\leq \mathtt{P_{GR}}$ $<$ 80 x 10 $^{-5}$ meter/s^2 (T3), and $\mathtt{P_{GR}} \geq$ 80 x 10 $^{-5}$ meter/s^2 (T4). The median latency for T2 and T4 is constant with the `Latency`=3 in T2 and `Latency`=2 in T4. An increase is seen in the median `Latency` at T1 and T3. The increase in T3 is gradual and much smaller compared to T1. If we consider T2, T3, and T4 to be an acceptable `Latency`, then the critical point at which to expect a drastic increase in the `Latency` is at $\mathtt{P_{GR}}$=30 x 10 $^{-5}$ meter/s^2. This mean that with $P_{GR} <$ 30 x 10 $^{-5}$ meter/s^2, the fault is too subtle and only apparent after a while.

With these results, we have established the range of magnitudes of $\mathtt{P_{PT}}$ and $\mathtt{P_{GR}}$ that can be effectively detected by collective RDA-DW2. These findings provide us evidence that the proposed method with RDA is robust in detection.

5 Conclusions

In this paper, we presented an error detection approach with collective self-detection scheme of errors using an immune-inspired algorithm, the RDA in the context of a foraging SRS. Results show that our approach is able to produce an accurate detection with a low rate false alarms and adaptive to dynamic environments. To date, no approach (to our knowledge) has been proposed for this specific problem in the context of SRS. We have also shown the tuning of the RDA's parameters for optimal performance. To assess the limitations of our approach, we investigated the detection capability and established the critical points that is useful for subsequent recovery measures.

References

1. Christensen, A.L., O'Grady, R., Birattari, M., Dorigo, M.: Automatic Synthesis of Fault Detection Modules for Mobile Robots. In: Proc. 2nd NASA/ESA Conf. Adaptive Hardware and Systems, pp. 693–700. IEEE Computer Society Press, Los Alamitos (2007)
2. Christensen, A.L., O'Grady, R., Birattari, M., Dorigo, M.: Exogenous fault detection in a collective robotic task. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS (LNAI), vol. 4648, pp. 555–564. Springer, Heidelberg (2007)
3. Şahin, E.: Swarm Robotics: From Sources of Inspiration to Domains of Application. In: Şahin, E., Spears, W.M. (eds.) Swarm Robotics 2004. LNCS, vol. 3342, pp. 10–20. Springer, Heidelberg (2005)
4. Gerkey, B., Vaughan, R.T., Howard, A.: The player/stage project: Tools for multi-robot and distributed sensor systems. In: Proc. 11th International Conf. Advanced Robotics, pp. 317–323 (2003)
5. Mokhtar, M., Timmis, J., Tyrrell, A., Bi, R.: In: Proc. Congress on Evolutionary Computation, pp. 2055–2062. IEEE Press, New York (2009)
6. Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D., Martinoli, A.: The e-puck, a Robot Designed for Education in Engineering. In: Proc. 9th Conf. Autonomous Robot Systems and Competitions, pp. 59–65 (2009)
7. Owens, N.D.L., Greensted, A., Timmis, J., Tyrrell, A.: T cell receptor signalling inspired kernel density estimation and anomaly detection. In: Andrews, P.S., Timmis, J., Owens, N.D.L., Aickelin, U., Hart, E., Hone, A., Tyrrell, A.M. (eds.) ICARIS 2009. LNCS, vol. 5666, pp. 122–135. Springer, Heidelberg (2009)
8. Skoundrianos, E.N., Tzafestas, S.G.: Finding fault-fault diagnosis on the wheels of a mobile robot using local model neural networks. IEEE Robotics and Automation Magazine, 83–90 (2004)
9. Terra, M.H., Tinos, R.: Fault Detection and Isolation in Robotic Manipulators via Neural Networks A Comparison Among Three Architectures for Residual Analysis. Journal of Robotic Systems 18(7), 357–374
10. Timmis, J., Tyrrell, A., Mokhtar, M., Ismail, A., Owens, N., Bi, R.: An artificial immune system for robot organisms. In: Levi, P., Kernback, S. (eds.) Symbiotic Multi-Robot Organisms: Reliability, Adaptability and Evolution, pp. 268–288. Springer, Heidelberg (2010)
11. Winfield, A.F.T., Nembrini, J.: Safety in Numbers Fault Tolerance in Robot Swarms. International Journal on Modelling Identification and Control 1(1), 30–37 (2006)