Short Papers

Fault Detection in a Swarm of Physical Robots Based on Behavioral Outlier Detection

Danesh Tarapore, Jon Timmis, and Anders Lyhne Christensen

***Abstract*—The ability to reliably detect faults is essential in many real-world tasks that robot swarms have the potential to perform. Most studies on fault detection in swarm robotics have been conducted exclusively in simulation, and they have focused on a single type of fault or a specific task. In a series of previous studies, we have developed a robust fault-detection approach in which robots in a swarm learn to distinguish between normal and faulty behaviors online. In this paper, we assess the performance of our fault-detection approach on a swarm of seven physical mobile robots. We experiment with three classic swarm robotics tasks and consider several types of faults in both sensors and actuators. Experimental results show that the robots are able to reliably detect the presence of hardware faults in one another even when the swarm behavior is changed during operation. This paper is thus an important step toward making robot swarms sufficiently reliable and dependable for real-world applications.**

***Index Terms*—Collective behavior, fault detection, multirobot systems, robot swarms.**

I. Introduction

As robots become increasingly prevalent and take on new classes of tasks, safety and reliability are key issues to address. Robot malfunctions have the potential to cause damage to humans, other robots, infrastructure, vegetation, or animals present in the environment. Unless robot control can be explicitly designed to tolerate common malfunctions [1], reliable detection of the presence of faults is essential in many real-world scenarios to minimize the impact and risk associated with faults that inevitably occur. It is, however, challenging to give autonomous robots the capacity to detect faults in themselves as faults often cannot be detected directly. Instead, the presence of faults must be inferred from erroneous behavior [2]. While an autonomous robot operating alone must rely on its own sensors to detect the presence of faults, collective robots have the potential to detect faults cooperatively. In particular, robots that are able to observe one another have the potential to exogenously detect erroneous behavior. A robot is thus not limited to its own sensors, but can benefit from the presence of other robots in the environment to detect its abnormal behavior. A number of fault-detection approaches have been proposed for exogenous fault detection in decentralized multirobot systems. In some of the most basic approaches, fault detection is based on periodic communication [3], [4], which means that only those faults that render a robot unable to communicate can be reliably detected. In other approaches, actual behavior is compared to one or more models of expected or normal behavior [5]–[7], robots compare sensory readings when they meet [8], or they compare their individual contributions to the task-solving effort [9].

If robots adapt their behavior during operation [10], it may, however, not be possible to characterize normal behavior prior to deployment. Also, approaches that rely on comparison of performance require that individual task performance is easily quantifiable and performance assessment can be made relatively frequently, which is not always the case. In a series of past studies [11]–[14], we have proposed and developed a fault-detection system in which the presence of faults is inferred based on a continually learnt classification model that distinguishes between normal and abnormal behaviors of individual robots. Each robot characterizes the behavior of neighboring robots over a certain time window and then compares behaviors to detect any abnormalities that could be the result of a fault.

Large-scale decentralized mutirobot systems, or *robot swarms*, have significant potential to take on real-world tasks [15], but they are still mainly confined to research environments. Timely and robust detection of faults is essential in many real-world scenarios, but studies on faults and fault detection in robot swarms have almost exclusively been conducted in simulation (exceptions include [4], [16]). Since faults are intimately linked to robot hardware, experimentation on physical robots is, however, an important step toward bringing swarm robotics systems to real-world applications.

The main contribution of this paper is that we demonstrate decentralized fault detection based on behavioral observation in a physical swarm robotics system. Observed robot behaviors are encoded as feature vectors, and abnormalities in robot behavior are detected as outliers with the corresponding robots being labeled as faulty by its neighboring robots in the swarm. For our experiments, we use a swarm of seven mobile robots that perform a number of common swarm behaviors, and we consider different types of faults, both in sensors and actuators. We show that our fault detection system is robust to changes in swarm behavior and faults are reliably detected in a timely manner.

Manuscript received January 25, 2019; accepted July 4, 2019. This paper was recommended for publication by Associate Editor S.-J. Chung and Editor P. Robuffo Giordano upon evaluation of the reviewers' comments. The work of D. Tarapore was supported by a Marie Curie Intra-European Fellowship (GiFteD-MrS/623620) and in part by the Engineering and Physical Sciences Research Council (EPSRC) New Investigator Award under Grant EP/R030073/1. The work of J. Timmis was supported by the EPSRC under Grant EP/K040820/1. *(Corresponding author: Danesh Tarapore.)*

D. Tarapore is with the School of Electronics and Computer Science, University of Southampton, SO17 1BJ Southampton, U.K., and also with York Robotics Laboratory and the Department of Electronic Engineering, University of York, YO10 5DD York, U.K. (e-mail: d.s.tarapore@soton.ac.uk).

J. Timmis is with York Robotics Laboratory and the Department of Electronic Engineering, University of York, YO10 5DD York, U.K., and also with the Faculty of Technology, University of Sunderland, SR1 3SD Sunderland, U.K. (e-mail: jon.timmis@york.ac.uk).

A. L. Christensen is with the Embodied Systems for Robotics and Learning, The Maersk Mc-Kinney Moller Institute, University of Southern Denmark, 5230 Odense, Denmark, and also with the Instituto Universitário de Lisboa (ISCTE-IUL), 1649-026 Lisbon, Portugal (e-mail: anderslyhne@gmail.com).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org provided by the authors.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2019.2929015

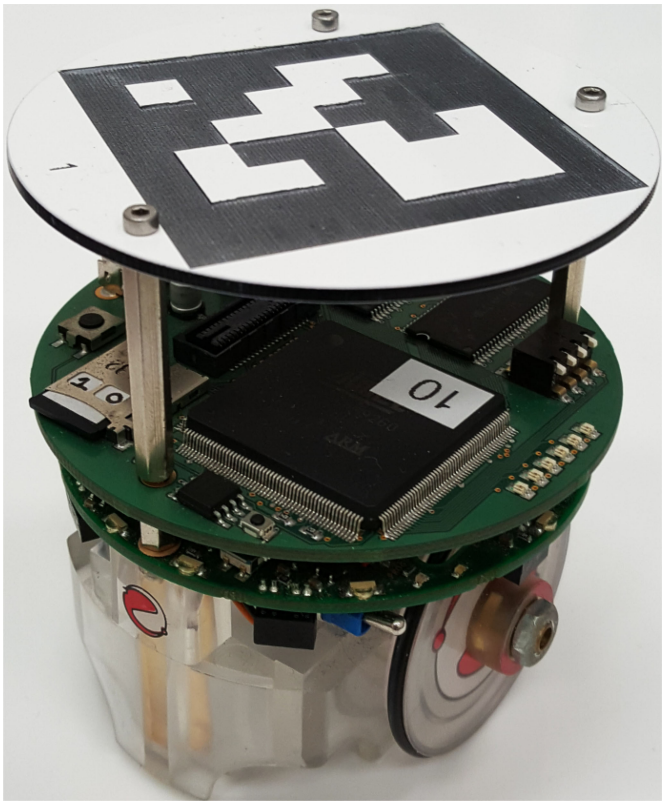

Fig. 1. E-puck robot of the swarm augmented with an ARM AT91SAM9260 extension board and with an ArUco fiducial marker tag.

II. Methodology

A. Behavior-Driven Fault Detection

Our process of exogenous fault detection in a robot swarm is comprised of the following phases, occurring in sequence. First, robots of the swarm sense their behavior over a window of time and characterize it as a binary feature vector, shared with neighboring robots in the swarm. Second, individual robots of the swarm execute an immunology-inspired outlier detection algorithm—the *crossregulation model* (CRM) [13]—to detect outlier feature vectors corresponding to abnormal behaviors, where abnormalities are consequent to faults in the neighboring robots. Second, a voting scheme is employed to consolidate an individual robot's decision on the identity of a faulty neighboring robot with decisions made by its other neighboring robots in the swarm.

The individual robots of the swarm execute the CRM to classify the behavioral feature vectors of their neighbors as inliers or outliers. The feature vectors corresponding to behaviors, which are exhibited by the majority of the neighboring robots, are treated as inliers. By contrast, feature vectors corresponding to behaviors exhibited by one or a few robots—abnormal behaviors—are flagged as outliers by the observing robot. In employing such an instance-based learning approach for outlier detection, the robot swarm is capable of establishing a dynamic signature of normality that allows fault detection to be resilient to changes in robot swarm behavior.

B. Experiment Hardware for Fault Detection

The hardware setup for our experiments comprises a swarm of e-puck robots (see Fig. 1). The e-puck is a two-wheeled differential drive open hardware mobile robot. The basic e-puck model is 7 cm in diameter and 5 cm in height, and comprises eight infrared (IR) transceivers positioned around the circular chassis of the robot to perceive obstacles in the robot's immediate environment. The e-puck's actuators are a pair of miniature stepper motors driving the two wheels at the base of the robot with speeds up to 13 cm/s.

The e-puck is also enhanced with a range-and-bearing sensor [17], which provides individual robots with sensor readings on the distance of neighboring robots (range up to 1 m) and the angular positions of neighboring robots (bearing between $[0^\circ, 360^\circ]$). However, the physical range-and-bearing sensor boards were not available for our experiments. Therefore, we employed a virtual range-and-bearing sensor for each physical robot of the swarm, using a realistic and noisy sensor model of Garattoni *et al.* [18] to emulate the hardware based on an overhead camera tracking system.

C. Behavioral Feature Vectors for Fault Detection

The observed robot swarm behaviors are characterized with respect to a number of predefined primitive swarm behavioral components (e.g., presence or absence of neighboring robots in close proximity based on readings from the range-and-bearing sensor, and the robot's motor reaction when neighboring robots are encountered). For behavior characterization as feature vectors, Boolean features associated with different behavioral components are set based on the presence (set to 1) and absence (set to 0) of the behavioral components in the observed swarm behavior. Binary features encoding the different behavioral components are then concatenated to form the behavioral feature vector.

Our design of a behavioral feature vector generalizable across various behavior tasks is the result of empirical data gathered from previous experiments in simulation [13]. We investigated the interplay between the choice of feature vector and fault detection performance with 15 different behavioral features that covered various characterizations of a robot's neighborhood, its absolute and relative speeds, and the nature of its interactions in the presence and absence of sensory stimuli. This paper revealed six Boolean features $\langle F_1, F_2 \ldots F_6 \rangle$ that provided a robust discrimination between 16 different normal-faulty behavior combinations, which are easy to sense and compute on the physical e-puck platform and are independent of the controller architecture used to execute the robot's behavior. The selected features encode for the robot's sensor inputs (features F_1 and F_2), motor outputs (features F_3 and F_4), and sensor–motor interactions (features F_5 and F_6). All computations for the feature vectors were executed on the ARM extension board on each of the e-puck robot platforms of the swarm.

1) Sensor Inputs: The first two features F_1 and F_2 encode the number of neighboring robots in proximity to the robot. We define a Heaviside step function $\mathrm{H}(x)$ whose value is 0 for a negative argument and 1 for a positive argument as

$$\mathrm{H}(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}. \tag{1}$$

The first feature F_{1_i} of robot r_i encodes the presence or absence of neighbors in close proximity of r_i. For feature $F_{1_i}(\tau)$ of r_i at time τ, we have

$$F_{1_i}(\tau) = \begin{cases} 1, & \text{if } \sum_{t=\tau-T_s}^{\tau} \dfrac{\mathrm{H}(|N_{c_i}(t)|)}{T_s} \geq 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where for robot r_i at time t, we define $N_{c_i}(t)$ as the set of neighboring robots in close proximity to r_i (up to 15 cm), and T_s is the number of control cycles of past observations. The feature F_{1_i} is set to 1

TABLE I
PARAMETERS OF THE BEHAVIORAL FEATURE VECTORS

Parameter	Description	Value
v_{max}	Maximum linear speed	10 cm/s
v	Desired linear speed	–
ω_{max}	Maximum angular speed	1.9 radians/s
ω	Desired angular speed	–
Δt	Length of control cycle	10 ms
N_c	Number of neighboring robots in close proximity of [0 cm, 15 cm)	–
N_f	Number of neighboring robots in far proximity of [15 cm, 30 cm]	–
T_s	Length of the time window for estimating number of neighbors and the distance traversed by observed robot	10 s
T_l	Length of the time window for estimating alterations in heading of observed robot	30 s

if robot r_i had at least one neighboring robot within [0 cm, 15 cm) for the majority of the past time window of length T_s (all feature vector parameters are listed in Table I).

The second feature $F2_i(\tau)$ for observed robot r_i encodes the presence or absence of neighboring robots in far proximity of r_i at time τ as

$$F2_i(\tau) = \begin{cases} 1, & \text{if } \sum_{t=\tau-T_s}^{\tau} \frac{\mathrm{H}(|N_{f_i}(t)|)}{T_s} \geq 0.5 \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where $N_{f_i}(t)$ is the set of neighboring robots further away from robot r_i (between [15 cm, 30 cm]). The feature $F2_i$ is set if r_i has at least one neighbor in far proximity for the majority of the past time window of length T_s.

2) Motor Outputs: The third and fourth features of r_i, $F3_i$ and $F4_i$, pertain to the motor actions performed by r_i during task execution.

The third feature $F3_i(\tau)$ at time interval τ encodes if r_i has moved in the past T_s time intervals

$$F3_i(\tau) = \begin{cases} 1, & \text{if } \mathrm{Dist}(r_i, \tau, T_s) > 0.15\ v_{\max} T_s \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

where $\mathrm{Dist}(r_i, \tau, T_s)$ at time τ is an estimate of the distance traversed by robot r_i over a time window of T_s (using a forward kinematics model of a two-wheeled nonholonomic differential-drive mobile robot), and $v_{\max}$ is the maximum linear speed of the robot. The feature $F3_i$ is set to 1 if the distance traversed by r_i exceeds 15% of the maximum traversable distance by the robot in the time window T_s. The 15% threshold is set to compensate for errors in e-puck odometry. The effect of odometric drift on the estimate of distance traversed by the robot is also reduced with the use of a short time window (T_s is set to 10 s in our experiments).

The fourth feature $F4_i(\tau)$ at time τ encodes the number of instances r_i has altered its heading in the past T_l time intervals. An alteration of the robot's heading is detected by a change in its angular acceleration, exceeding 15% of the maximum angular acceleration of the robot; the threshold was set to compensate for inaccurate estimates of changes in robot headings. For r_i, at time τ, an altered heading is detected as

$$M_i(\tau) = \begin{cases} 1, & \text{if } |\dot{\omega}_i(\tau)| \geq 0.15\ |\dot{\omega}_{\max}| \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where $\dot{\omega}_i(\tau)$ is the angular acceleration of robot r_i at time τ and $\dot{\omega}_{\max}$ is the maximum angular acceleration. Angular acceleration values are updated at each time interval and computed as the difference between angular velocity values at consecutive time intervals.

The feature $F4_i$ is set to 1 if the robot r_i has altered its heading at least once in the past T_l time intervals

$$F4_i(\tau) = \begin{cases} 1, & \text{if } \sum_{t=\tau-T_l}^{\tau} M_i(t) > 0 \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

3) Sensor–Motor Interactions: The fifth and sixth features of observed robot r_i, $F5_i$ and $F6_i$, account for the motor actions performed by r_i in the presence and absence of neighboring robots. For the feature $F5_i$, we define the function $Sm(t)$ as follows:

$$Sm(t) = \mathrm{H}(|N_{c_i}(t) + N_{f_i}(t)|) \wedge\ M_i(t) \quad (7)$$

where the Heaviside step function $\mathrm{H}(|N_{c_i}(t) + N_{f_i}(t)|)$ has value 1 if neighboring robots are present in proximity of the observed robot r_i at time t.

The feature $F5_i$, computed from (7), is set to 1 if r_i alters its heading at least once in the presence of neighboring robots in time window T_l

$$F5_i(\tau) = \begin{cases} 1, & \text{if } \exists\ t \in \{\tau - T_l \ldots \tau\} : Sm(t) \text{ is true} \\ 0, & \text{otherwise} \end{cases}. \quad (8)$$

The feature $F6_i$ encodes robot movement in the absence of neighboring robots. For this feature, we define function $Sn(t)$ as follows:

$$Sn(t) = \neg\, \mathrm{H}(|N_{c_i}(t) + N_{f_i}(t)|) \wedge\ M_i(t) \quad (9)$$

where $\neg\, \mathrm{H}(|N_{c_i}(t) + N_{f_i}(t)|)$ has value 1 if no neighboring robots are present in the vicinity of r_i at time t.

The feature $F6_i$, computed from (9), is set to 1 if r_i alters its heading at least once in the absence of neighboring robots over the time window T_l

$$F6_i(\tau) = \begin{cases} 1, & \text{if } \exists\ t \in \{\tau - T_l \ldots \tau\} : Sn(t) \text{ is true} \\ 0, & \text{otherwise} \end{cases}. \quad (10)$$

The behavioral feature vector $F(\tau)$ comprising $\langle F1(\tau), \ldots, F6(\tau)\rangle$ was computed for all time intervals τ in the duration of the experiment. Parameters for the behavioral feature vectors in Table I have been set according to the physical characteristics of the e-puck robot hardware. The length of the robot observation time windows were set to be long enough to allow for robust characterization of the robots' behavior while still being short enough to allow for timely fault detection.

D. Detection of Faulty Robots in the Swarm

In every time-interval of the experiment, each robot of the swarm shares its behavioral feature vector with neighboring robots in range and line-of-sight of its range-and-bearing sensors. Subsequently, an outlier detection algorithm—the CRM—is executed by each robot for the detection of outlier behavioral feature vectors exhibited by faulty robots in its neighborhood. The algorithmic implementation of the CRM and parameters for its execution are detailed in [14].

In order to trigger appropriate recovery actions for a faulty robot, the robots of the swarm have to first translate their individual decisions on their neighboring robots based on their own respective CRM instances,

into the swarm's decision on the normal/faulty state of the robots. A voting scheme is employed for this purpose: a robot is identified as faulty if its behavior is classified as an outlier by the simple majority of its neighbors; otherwise, the robot is treated as behaving normally by the rest of the swarm (for details see [13]).

III. Experimental Setup

For our experiments, we use a swarm of seven e-puck robots. The robots are situated in a 2.1×1.2 m^2 area surrounded by white walls. We evaluate our fault-detection approach in a series of experiments where the robots perform classic swarm robotics behaviors, namely aggregation, dispersion, and homing—behaviors widely used in previous studies, both in simulation and in physical robot laboratory experiments [19], [20]. The robots are placed at random locations and with random orientations in the beginning of each experiment. The onboard control programs follow the subsumption architecture [21], and they are written in C++ and compiled for the ARM extension board.

A. Faults

We inject simulated faults [2] in a robot by providing an erroneous input to its onboard control program in the case of sensor faults, and by ignoring or perturbing its output in the case of actuator faults. The behavioral control logic assumes that the robot on which it is executed is fully operational, and there are no mechanisms present to compensate for faulty sensors or actuators. The behavior exhibited by a faulty robot is thus the result of executing the control logic on the robotic platform with malfunction corresponding to the injected fault.

We consider the following sensor and actuator faults.

1) PMIN. The minimum possible value (0) is returned by the four front proximity sensors.
2) PMAX. The maximum possible value (1) is returned by the four front proximity sensors.
3) ROFS. An offset is applied to the readings from the robot's range-and-bearing board. The position estimate offset is sampled from a uniform distribution in the interval $[75\text{ cm}, 100\text{ cm}]$, and the orientation estimate offset is sampled from a uniform distribution in $[-180^\circ, 180^\circ]$.
4) LRACT. A fault in either the robot's left wheel or its right wheel actuator (selected at random at the start of the experiment) causing its speed to be set to 0 cm/s.
5) BACT. A fault in both the robot's left and right wheel actuator causing their speed to be set to 0 cm/s.

As discussed in [14], these fault types are representatives of five different scenarios involving permanent failure in the e-puck's sensors and actuators.

B. Swarm Behaviors

1) Dispersion: In dispersion, robots of the swarm combine obstacle avoidance and navigation for maximizing the sensing coverage of large areas in patrolling and monitoring missions [22]. In our implementation of dispersion, the robots, first, perform a simple random-walk behavior, moving straight in a randomly selected direction and selecting a new direction with probability 0.02 (per second), and, second, avoid obstacles within 5 cm of their position. The presence of obstacles is registered with the IR proximity sensors of the basic e-puck model.

2) Aggregation: In the aggregation behavior, the robots of the swarm are tasked to form a coherent and stable cluster. In our implementation, aggregating robots use their range-and-bearing sensors to move toward the center of mass of neighboring robots within range of 1 m while avoiding obstacles with the dispersion behavior.

3) Homing: In homing, the swarm is tasked to surround and cage a moving beacon, such as an intruder [22]. In our implementation of homing, one of the robots of the swarm serves as a beacon and performs dispersion behavior. The remaining robots of the swarm avoid obstacles while heading in the direction of the beacon when it is in range of their range-and-bearing sensors.

C. Experimental Protocol and Data Collection

At the beginning of each experiment, the server broadcasts a signal. On receiving the signal, the robots perform a random walk with obstacle avoidance behavior for a duration of 30 s, following which individual robots stop and send an acknowledgement signal to a server. Once the server has received acknowledgement signals from all the robots of the swarm, it broadcasts another signal and the robots start executing their tasks. Each experiment lasts for a duration of 300 s. During the experiment, we record sensor readings, control signals, operational states, and behavioral feature vectors of each e-puck in the swarm, used offline for the computation of behavioral outliers. In experiments requiring an injection of a fault, we inject the fault at the very beginning of the experiment.

IV. Results

In this section, we present the results of a series of experiments involving seven physical robots. We first analyze true-positive performance, that is, the robots' capacity to detect the presence of faults in one another. We then analyze false-positive performance, that is, the robots' capacity to avoid labeling fault-free robots as faulty. Finally, we show that robots are able to reliably tolerate changes in swarm behavior. Detailed results of our experiments are available online in Supplementary Material[1] (see Sections S2–S4).

A. True-Positive Performance

We conducted five replicates for each combination of task (aggregation, dispersion, and homing) and fault type (PMIN, PMAX, ROFS, LRACT, and BACT). The results indicating the true-positive performance, measured as the mean $\pm$ SD percentage of time a robot with a fault, was detected as behaving abnormally is summarized in Table II. The italicized values indicate the true-positive performance during the three consecutive 100-second intervals in the experiment. True-positive performance over higher resolution 30-second intervals is presented in Figs. S1 (aggregation), S3 (dispersion), and S4 (homing) in Section S2 of Supplementary Material.

The swarm's fault-detection performance often changes during an experiment and depends strongly on the type of fault (see Table II). Our fault-detection approach relies on online comparison of behavior, and thus only faults that result in a divergence from normal behavior can be detected. In the extreme case of the aggregation/PMIN experiments, in which the minimum value is returned by the four front proximity sensors, the faulty robot is never detected, because all robots—including the one with the fault–consistently aggregate into a single, coherent cluster. The PMIN fault thus does not provoke a divergence from normal fault-free behavior, and the fault is therefore not detected (see link to videos in Section S1 of Supplementary Material).

The results of the aggregation/LRACT experiments, in which either the left or the right wheel stops moving, show that the swarm's ability to detect the faulty robot decreases during the experiment (see Table II). The change in fault detection performance during an experiment is a result of the collective aggregation behavior. Initially, all robots try to

[1][Online]. Available: http://doi.org/10.5281/zenodo.3333824

TABLE II
TRUE-POSITIVE PERFORMANCE

	Aggregation			Dispersion			Homing		
PMIN	-			$\mathbf{70.1 \pm 3.1\%}$			$\mathbf{53.1 \pm 17.9\%}$		
				49.7%	*85.0%*	*75.6%*	*37.0%*	*63.6%*	*58.7%*
PMAX	$\mathbf{69.1 \pm 8.8\%}$			$\mathbf{85.7 \pm 10.3\%}$			$\mathbf{92.9 \pm 3.2\%}$		
	78.7%	*68.4%*	*60.3%*	*71.1%*	*89.5%*	*96.6%*	*79.7%*	*98.9%*	*100.0%*
ROFS	$\mathbf{50.5 \pm 3.7\%}$			-			$\mathbf{49.0 \pm 12.1\%}$		
	38.9%	*55.0%*	*54.0%*				*41.2%*	*56.2%*	*48.9%*
LRACT	$\mathbf{31.1 \pm 12.7\%}$			$\mathbf{83.5 \pm 5.4\%}$			$\mathbf{71.6 \pm 4.8\%}$		
	45.5%	*23.9%*	*23.9%*	*73.7%*	*86.6%*	*90.3%*	*58.6%*	*85.2%*	*70.9%*
BACT	$\mathbf{95.3 \pm 1.6\%}$			$\mathbf{94.7 \pm 1.6\%}$			$\mathbf{96.6 \pm 0.1\%}$		
	85.9%	*100.0%*	*100.0%*	*84.2%*	*99.7%*	*100.0%*	*89.8%*	*100.0%*	*100.0%*

Mean ± SD percentage of time for each type of faults is detected for the three different tasks. Percentages in italics are over intervals of 100 s. In the aggregation/PMIN experiment and in the dispersion/ROFS experiment, the injected fault had no impact on behavior or performance, and could therefore not be detected (see text for details).

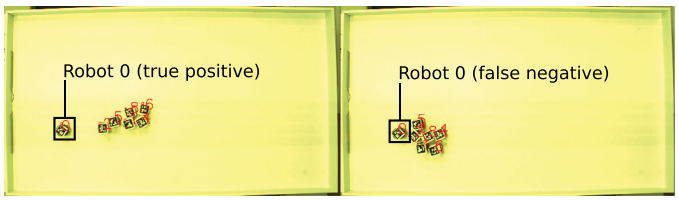

Fig. 2. Frames from one of the aggregation experiments in which a fault is injected in the right wheel actuator (RACT) of robot number 0. In the first frame ($t = 57$ s) the faulty robot is correctly detected by the rest of the swarm. In the second frame ($t = 1$ m 52 s), the fault is not detected as the robot has become a part of the aggregate and exhibits limited motion similar to the normally functioning robots of the tightly packed aggregate (for video, see Section S1 in Supplementary Material).

TABLE III
PROPORTION OF FALSE POSITIVES

	Aggregation	Dispersion	Homing
Unfiltered	0.06±0.05	0.09±0.06	0.09±0.07
Majority	0.02±0.04	0.05±0.05	0.05±0.06
Exclusive	0.00±0.00	0.00±0.02	0.01±0.03

Mean ± SD proportion of time for each type of classifications for the three different tasks.

move toward one another to form an aggregate, but since the robot with compromised locomotion is anchored by its unresponsive wheel, its behavior differs significantly from the other, fully operational robots. However, the fully operational robots eventually aggregate around the faulty robot, and its behavior therefore becomes indistinguishable from the other robots in the aggregate (see example in Fig. 2). In the experiments in which the injected fault affected both wheel actuators (BACT), the faulty robot is rendered completely immobile, which makes its behavior easily distinguishable from the other robots (see Table II).

In the aggregation experiments where a fault is affecting the range-and-bearing sensor (ROFS) and where the four front proximity sensors return the maximum possible reading (PMAX), the swarm's ability to correctly detect the faulty robot changes throughout the experiment with no clear trend. In those experiments, we observed that the fault would continually lead to the robot with the fault joining and leaving the aggregate.

In the dispersion and homing tasks, we observed significant differences in fault-detection performance with respect to the different faults (see Table II). While the robots were unable to detect the PMIN fault in the aggregation task, the fault was detected more than 50% of the time, on average, in both the dispersion task and in the homing task. The reason for the differences in fault-detection performances are again the interplay between the specific fault's impact on the individual robot's behavior, the task, and the collective behavior displayed by the other robots in the swarm. A fault in the range-and-bearing sensor can therefore be reliably detected in both the aggregation task and in the homing task, but not in the dispersion task, as the range-and-bearing sensor readings were not used by the dispersion behavior.

Similar to how the range-and-bearing sensor fault (ROFS) only periodically leads to abnormal behavior in the aggregation task (see example in Fig. S2 of Supplementary Material), several faults are only periodically detected in the dispersion and homing tasks. In the extreme case of the PMIN fault in the homing task (see Table II), the mean proportion of time the fault was detected is above 50%, but the minimum remains close to 0% after the first 30 s, whereas the maximum is close to 100% across the five replicates. In different replicates, the fault was detected at different times depending on the spatial configuration of the robots. In four of five replicates, the faulty robot was detected 100% of the time by the swarm in at least one 30 s time interval. In the fifth replicate, the faulty robot was detected over 80% of the time in one 30 s time interval. The fault was thus detected in all replicates but at different times.

B. False-Positive Results

We ran five experiments for each task in which no faults were injected to determine the swarm's capacity to correctly label fully operational robots as fault-free. The proportion of time a fully operational robot is wrongly labeled as faulty was on average relatively high at 0.08 across all tasks (for Mean ± SD see unfiltered classification in Table III), and reached over 0.19 in some cases (see Fig. S5 in Section S3 of Supplementary Material). A fully operational robot is wrongly labeled as faulty by the other robots when its behavior is dissimilar to the

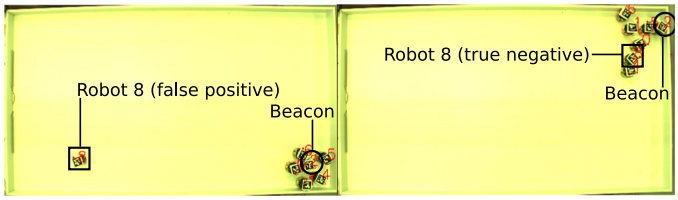

Fig. 3. Frames from one of the homing experiments in which all the robots are fully operational. In the first frame ($t = 24$ s), robot 8 is the only robot of the swarm that is unable to locate the beacon (encircled) and is therefore labeled as faulty by the rest of the swarm, whereas in the second frame ($t = 4$ m 23 s), robot 8 locates and homes toward the beacon, and is correctly labeled as being fully operational (for video, see Section S1 in Supplementary Material).

behavior of other robots. A robot's behavior can be distinct either as the result of a particular situation during the experiment or due to initial conditions. For instance, a robot may take longer than the rest of the swarm to localize the beacon in a homing task (see example experiment snapshots in Fig. 3).

The number of false positives can be significantly reduced when classification is done based on output of the CRM accumulated for a series of consecutive time steps, rather than based on the output in a single time step. We assessed two different approaches to classify the state of robots based on outputs across a time window: *majority* and *exclusive*. In majority-based classification, a robot is classified as faulty if the CRM has classified its state as abnormal for the majority of the time steps within the previous 10 s. In exclusive-based classification, on the other hand, a robot is only classified as faulty if the CRM has classified it as abnormal in *all* time steps in the previous 10 s. While basing classification on a series of consecutive outputs of the CRM significantly reduced the proportion of false-positive incidents (see Mean $\pm$ SD false-positive incidents in Table III), it leads to a high latency in fault detection [14].

C. Tolerance to Transitions in Swarm Behavior

Swarms of robots that operate in real-world scenarios must often display a number of different collective behaviors in order to complete a mission (see for instance [22]) or adapt their behavior on the fly (see for instance [23]). It is therefore important for a fault-detection approach to be able to tolerate changes in swarm behavior.

We established a series of experiments in which the swarm had to first perform one task and then gradually switch to perform another, in order to assess the capacity of our fault-detection approach to tolerate changes in swarm behavior. At the start of the task-switching experiments, the robots first performed the aggregation task. A probabilistic transition to the dispersion tasks was triggered halfway through the experiment, after which the robots would gradually switch from the aggregation behavior to the dispersion behavior. We tested two different mean task-switching times in separate and independent experiments, namely 5 s and 10 s. The times were selected to cover a range of transition rates whilst still allowing for all robots to reliably switch their behavior before the end of the experiment. Each task-switching experiment had a duration of 300 s and was replicated five times.

We recorded the feature vectors displayed by the robots at each control cycle during the task-switching experiments, and found that the feature vector distribution and the dominant feature vector changed after the behavior transition started. Despite the change in the distribution of feature vectors caused by the task switching, the proportions of false-positive incidents were not significantly affected (Mean $\pm$ SD for experiments with no change in behavior, 5 s behavior transition time, and 10 s behavior transition time were 0.08 ± 0.02, 0.08 ± 0.01, and 0.08 ± 0.0 respectively).

V. Discussion

Previously proposed fault-detection approaches rely on detailed descriptions of normal (no faults present) robot behavior [5]–[7], easily quantifiable task performance measures [9], and learnt models of normal behavior and faulty behavior [2], or are limited to a specific type of faults [3], [4], [8]. In our approach, on the other hand, the notion of normal and abnormal behavior is learnt online, based on a mutual interrobot observation. Robot behavior is characterized as a vector of six abstract binary features. The same binary features was used for all tasks, and we did not thus have to craft task-specific behavior characterizations to achieve reliable fault detection. Furthermore, as individual robots in the swarm detect faults in their neighboring robots, our fault-detection approach scales well to large-scale swarms (demonstrated in swarms of up to 100 simulated robots, see [13]).

The exclusive reliance on binary features to characterize behavior may, however, be inefficient. In this paper, for instance, two features (F_1 and F_2) were used to denote if a given robot had any neighbors within a range of [0 cm, 15 cm) and [15 cm, 30 cm], respectively. Those features and values were found to work well in general with respect to the robot platform and the traditional swarm robotics tasks considered. However, integer-valued or real-valued features that, for instance, denote the number of neighbors or the average distance to neighbors could be used instead. We speculate that such features could yield a better fault-detection performance as they enable more precise, yet still task-agnostic, characterization of behavior. Also, other methods, such as *local outlier factor* (LOF) or *k-nearest neighbors* (KNN), could be used for outlier detection in place of the CRM. We did, in fact, test both the LOF and the KNN on the data collected during our physical-robot experiments, and while their performance was found to be inferior to that of the CRM, they may perform well on nonbinary feature vectors.

Our behavior-driven approach to exogenous fault detection requires an accurate characterization of robot behavior—encoded as a behavioral feature vector—for the timely detection of the faulty robots in the swarm. For our experiments with the e-puck robot swarm, sensory data provided by the range-and-bearing sensors [17] were employed to characterize the robot's behavioral feature vector. While such reflected-signal-strength based IR sensor hardware may not function in sunlight, for outdoor robots employing our fault-detection system, odometric sensors, such as accelerometers, may be used to characterize the behavioral features pertaining to robot motion over short time scales. Additionally, GPS sensor data shared between robots of the swarm may be used to characterize if a robot has any neighboring robots in its vicinity, and the nature of their interactions, such as the frequency of heading angle, changes in the presence of neighboring robots in close proximity. Odometric and localization sensor hardware is commonly outfitted onboard most outdoor robot platforms, thus enabling our exogenous fault-detection system to be employed in outdoor mission scenarios.

In our experiments, fault detection was demonstrated on a robot swarm performing three tasks: aggregation, dispersion to maximize the sensing coverage of large areas, and homing toward a moving beacon that may represent an intruder. The selected tasks are simple and commonly used as benchmarks in the field of swarm robotics [19]. While our designed behavioral feature vector was found to yield a good fault-detection performance in the swarm robotics tasks considered in this paper, it may not be sufficient in other scenarios. For more complex swarms and tasks, for instance, it may be necessary to re-engineer the behavioral feature vectors to achieve reliable fault detection. Herein,

machine learning algorithms could enable the robots to learn relevant features for a robust discrimination between normal and faulty behavior conditions. Following the detection of the faulty robot, fault accommodation procedures may be executed to limit the impact of the faulty robot on the performance of the swarm. Accommodation procedures may comprise caging the faulty robot, diagnosing the nature of the fault [24], and consequently adapting the behavior of the faulty robot [25].

VI. Conclusion

Fault tolerance is an essential attribute for robots taking on real-world tasks. To ensure safe operation, robots must be able to detect abnormal behavior consequent to the presence of faults, such as those resulting from inevitable hardware failure. In this paper, we assessed the performance of our approach to detect abnormal behavior in a swarm of physical robots. We conducted extensive experiments, involving setups with three swarm robotics tasks, five types of faults ($3 \times 5 = 15$), one setup for each task where no fault was present (3), and two additional setups in which the swarm transitioned from one task to another (2). With five experimental trials conducted in each setup, a total of 100 physical-robot experiments were performed as part of this paper. Our results showed that decentralized fault detection based on interrobot behavioral observation and abnormal behavior detection—independent of the robot's controller architecture, and with no requirements on a measure of task performance—was feasible in a physical robot swarm. The capability of robots to distributedly detect a wide range of faults in one another is essential for swarm robotics systems to move out of tightly controlled laboratory environments and take on real-world tasks.

References

[1] H. Park and S. A. Hutchinson, "Fault-tolerant rendezvous of multirobot systems," *IEEE Trans. Robot.*, vol. 33, no. 3, pp. 565–582, Jun. 2017.

[2] A. L. Christensen, R. O'Grady, M. Birattari, and M. Dorigo, "Fault detection in autonomous robots based on fault injection and learning," *Auton. Robots*, vol. 24, no. 1, pp. 49–67, 2008.

[3] M. Ozkan, G. Kirlik, O. Parlaktuna, A. Yufka, and A. Yazici, "A multi-robot control architecture for fault-tolerant sensor-based coverage," *Int. J. Adv. Robot. Syst.*, vol. 7, no. 1, pp. 67–74, 2010.

[4] A. L. Christensen, R. O'Grady, and M. Dorigo, "From fireflies to fault-tolerant swarms of robots," *IEEE Trans. Evol. Comput.*, vol. 13, no. 4, pp. 754–766, Aug. 2009.

[5] M. J. Daigle, X. D. Koutsoukos, and G. Biswas, "Distributed diagnosis in formations of mobile robots," *IEEE Trans. Robot.*, vol. 23, no. 2, pp. 353–369, Apr. 2007.

[6] A. G. Mutambara, *Decentralized Estimation and Control for Multisensor Systems*. Boca Raton, FL, USA: CRC Press, 1998.

[7] A. G. Millard, J. Timmis, and A. F. T. Winfield, "Run-time detection of faults in autonomous mobile robots based on the comparison of simulated and real robot behaviour," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 3720–3725.

[8] R. A. Carrasco, F. Núñez, and A. Cipriano, "Fault detection and isolation in cooperative mobile robots using multilayer architecture and dynamic observers," *Robotica*, vol. 29, no. 4, pp. 555–562, 2011.

[9] H. Lau, I. Bate, P. Cairns, and J. Timmis, "Adaptive data-driven error detection in swarm robotics with statistical classifiers," *Robot. Auton. Syst.*, vol. 59, no. 12, pp. 1021–1035, 2011.

[10] H. Hagras, V. Callaghan, and M. Colley, "Learning and adaptation of an intelligent mobile robot navigator operating in unstructured environment based on a novel online fuzzy–genetic system," *Fuzzy Sets Syst.*, vol. 141, no. 1, pp. 107–160, 2004.

[11] D. Tarapore, A. L. Christensen, P. U. Lima, and J. Carneiro, "Abnormality detection in multiagent systems inspired by the adaptive immune system," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2013, pp. 23–30.

[12] D. Tarapore, A. L. Christensen, and J. Timmis, "Abnormality detection in robots exhibiting composite swarm behaviours," in *Proc. Eur. Conf. Artif. Life*, 2015, pp. 406–413.

[13] D. Tarapore, P. Lima, J. Carneiro, and A. Christensen, "To err is robotic, to tolerate immunological: Fault detection in multirobot systems," *Bioinspiration Biomimetics*, vol. 10, no. 1, 2015, Art. no. 016014.

[14] D. Tarapore, A. L. Christensen, and J. Timmis, "Generic, scalable and decentralized fault detection for robot swarms," *PLoS One*, vol. 12, no. 8, 2017, Art. no. e0182058.

[15] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," in *Proc. Int. Workshop Swarm Robot.*, 2004, pp. 10–20.

[16] A. F. T. Winfield and J. Nembrini, "Safety in numbers: Fault-tolerance in robot swarms," *Int. J. Model., Identification Control*, vol. 1, no. 1, pp. 30–37, 2006.

[17] Á. Gutiérrez, A. Campo, M. Dorigo, L. Magdalena, and F. Monasterio-Huelin, "An open localization and local communication embodied sensor," *Sensors*, vol. 8, no. 11, pp. 7545–7563, 2008.

[18] L. Garattoni, G. Francesca, A. Brutschy, C. Pinciroli, and M. Birattari, "Software infrastructure for E-puck (and TAM)," IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, Tech. Rep. TR/IRIDIA/2015-004, 2015.

[19] L. Bayındır, "A review of swarm robotics tasks," *Neurocomputing*, vol. 172, no. 8, pp. 292–321, 2016.

[20] M. Chamanbaz *et al.*, "Swarm-enabling technology for multi-robot systems," *Frontiers Robot. AI*, vol. 4, 2017, Art. no. 12.

[21] R. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Autom.*, vol. RA-2, no. 1, pp. 14–23, Mar. 1986.

[22] M. Duarte *et al.*, "Evolution of collective behaviors for a real swarm of aquatic surface robots," *PLoS One*, vol. 11, no. 3, 2016, Art. no. e0151834.

[23] T. H. Labella, M. Dorigo, and J.-L. Deneubourg, "Division of labor in a group of robots inspired by ants' foraging behavior," *ACM Trans. Auton. Adapt. Syst.*, vol. 1, no. 1, pp. 4–25, 2006.

[24] J. O'Keefe, D. S. Tarapore, A. G. Millard, and J. I. Timmis, "Adaptive online fault diagnosis in autonomous robot swarms," *Frontiers Robot. AI*, vol. 5, 2018, Art. no. 131.

[25] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.