

# Distributed Sensor Analysis for Fault Detection in Tightly-Coupled Multi-Robot Team Tasks

Xingyan Li and Lynne E. Parker

**Abstract**—This paper presents a distributed version of our previous work, called SAFDetection, which is a sensor analysis-based fault detection approach that is used to monitor tightly-coupled multi-robot team tasks. While the centralized version of SAFDetection was shown to be successful, a shortcoming of the approach is that it does not scale well to large team sizes. The distributed SAFDetection approach addresses this problem by adapting and distributing the approach across team members. Distributed SAFDetection has the same theoretic foundation as centralized SAFDetection, which maps selected robot sensor data to a robot state by using a clustering algorithm, and builds state transition diagrams to describe the normal behavior of the robot system. However, rather than processing multiple robots' sensor data centralized on a server, distributed SAFDetection performs feature selection and clustering on individual robots to build the normal behavior model of an individual robot and the entire robot team. Fault detection is also accomplished in a distributed manner. We have implemented this distributed approach on a physical robot team and in simulation. This paper presents the results of these experiments, showing that distributed SAFDetection is an efficient approach to detect both local and interactive faults in tightly-coupled multi-robot team tasks. Compared to the centralized version, this approach provides more scalability and reliability.

## I. INTRODUCTION

The demands for highly reliable robot systems are significantly growing due to the increasing needs for robotic applications in various challenging tasks. Therefore, it is necessary for robots to monitor their performance so that the abnormal behavior can be promptly detected. Here, we define a robot *fault* as a deviation from the expected behavior of the robot system, and *fault detection* as the process of automatically determining that a fault has occurred. In previous work, we have introduced an applicable centralized approach, called SAFDetection (which stands for Sensor Analysis based Fault Detection), to detect both local and interactive faults in tightly-coupled multi-robot team tasks [14], [15]. This approach provides a novel methodology for detecting robot faults in situations when motion models and models of multi-robot dynamic interactions are unavailable. To detect faults, SAFDetection maps selected robot sensor data to robot states using a clustering algorithm, and builds state transition diagrams to describe the normal behavior of the robot system.

The centralized SAFDetection approach treats the entire robot team as a single monolithic robot and performs all work centralized in a server. However, like many other centralized

systems, centralized SAFDetection faces the problems of a single point failure, a heavy computational load on the server, and difficulty in scaling to larger team sizes. To overcome these problems, we present a distributed SAFDetection approach. In this approach, rather than building the robot team's state directly from multiple robots' sensor data, the clustering algorithm is performed on individual robots to obtain the individual robot's states; the robot team's state is represented as a vector of individual robot states. Thus, the data dimension for clustering is limited to the number of features on a single robot and the curse of dimensionality is broken. In distributed SAFDetection, a peer-to-peer and client/server mixture model is used to distribute the computation work and failure risk. The creation of the individual robot state transition diagram, as well as local fault detection are accomplished locally by each of the robots in the team. The team robot state transition diagram is built on a server robot and is then distributed to all the client robots. The team faults are detected using a peer-to-peer mechanism, as each robot keeps a copy of the team robot state transition diagram. To detect team faults in distributed SAFDetection, each robot should know the latest state information of its teammates. A periodic and on-request mixed message transmission protocol is used to provide a balance between reliable information sharing and unnecessary message communication. With this mixed model, the original sensor data is stored locally and only robot states are transmitted among robots. Therefore, the network traffic is reduced. Compared to the centralized version, this approach provides more scalability and reliability.

The remainder of the paper is organized as follows. We present related work, including the centralized SAFDetection approach in Section II, followed by details of the distributed approach in Section III. In Section IV, we present results of using this approach in robot implementations of box pushing and robot following experiments. Conclusions are discussed in Section V.

## II. RELATED WORK

Fault detection for robots is a complex problem, for a number of reasons: the space of possible faults is very large; robot sensors, actuators, and environment models are uncertain; and there is limited computation time and power. Nevertheless, because of its importance, much prior work has been done in this area.

The most popular method for providing fault detection in robot systems is based on motion control modeling [8], [12], which compares the values estimated by the motion

X. Li and L. E. Parker are with the Distributed Intelligence Laboratory, Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN 37996-3450, {li, parker}@eecs.utk.edu

model and the current measurements, in order to detect a fault. Another widely used computer fault detection method is voting based on modular redundancy [9], [4], which is commonly used in highly reliable systems in which more than one module works redundantly to perform the same task given the same input data and the faulty module is voted out according to the module results. Analytical redundancy fault detection [13], [10] is a concept of comparing the histories of sensor outputs versus the actuator inputs to check failures. Particle filter techniques [19], [3], which have become popular recently for robot fault detection, estimate the robot and its environmental state from a sequence of noisy, partial sensor measurements. There are also many data driven fault detection methods, especially the data mining method in the robotics area. Commonly used techniques include artificial neural networks [17], [5], [18], Bayesian networks [6], [11], [16] and rule generation [20], [2].

Unlike other methods, our SAFDetection approach does not require knowledge of the internal control of the robot system or advance knowledge of the possible fault types. Additionally, no functionally redundant modules are required and no requirement is made for specifying the relationship between the measured variables. Instead, our approach treats the robot or robot team as a black box, learning the probabilistic robot state transition diagram from the histories of robot sensor data during normal operation based on a clustering algorithm. This diagram is then used together with on-line sensor data to detect faults in a real-time fashion. Furthermore, this approach has been applied to the domain of multi-robot systems, which has not been well-studied in the literature.

The structure of the previously built [14], [15] centralized SAFDetection approach is shown in Figure 1. This approach

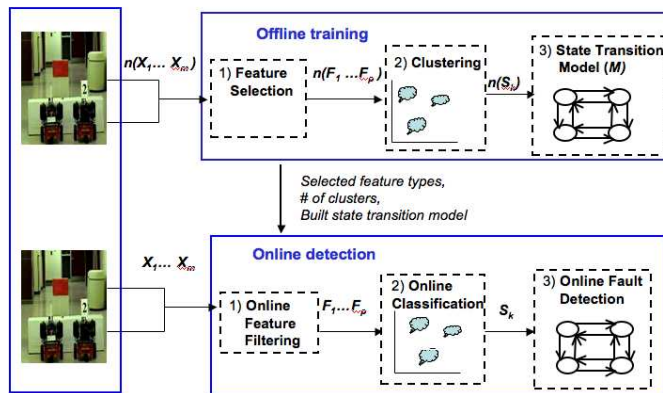


Fig. 1. The structure of the centralized SAFDetection approach.

is a training-classification based method. In the centralized training stage, data is preprocessed using Principal Components Analysis (PCA) to automatically select the features most relevant to the task. Then, the Fuzzy C-Means (FCM) clustering algorithm [1] is used to categorize the data into clusters that represent the relevant states of the system. (Alternative clustering techniques were also studied, with

FCM performing the best; see [15].) A probabilistic state transition diagram is learned from the data, which records not only the likelihood of transitions between states, but also the average time, and standard deviation of time, of duration in each state. This probabilistic state transition diagram represents the normal behavior of the robot team in performing their tightly-coupled task. At run time, online sensor data is compared to the centralized learned model, in order to detect three types of faults: *hard* faults, which reflect an abnormal robot state in which the sensor data does not match any previously seen before; *logic* faults, which reflect an abnormal state transition; and *coalition* faults, which reflect conflicts among team robots, in which the sensor model expectations of each robot differ. This approach was successfully demonstrated to allow robots performing a box pushing task to detect a variety of faults, such as a stuck robot, missing goal data, and communication failures.

### III. DISTRIBUTED SAFDETECTION

The centralized approach shown in Figure 1 uses a client/server model with message passing communication. However, it has two potential shortcomings: robustness and scalability. In this client/server model, the client robots send the original sensor data to the server robot and wait for the fault detection results. All the data is stored in a centralized location and all the computation work is performed on the server. It is well known that this client/server model lacks reliability, especially when the critical server is one robot in the team. Secondly, the curse of dimensionality in clustering restricts the scalability of the centralized approach. Although PCA has been used to reduce the dimension of the robot sensor data, the feature dimension for the entire robot team still becomes large as the robot team size increases, resulting in decreased clustering performance. In addition, traffic congestion on the network and limited computation capabilities on the server robot also become issues when more clients join the team. Thus, a distributed approach that overcomes these problems is needed that offers a more reliable and extensible mechanism to make SAFDetection scalable for applications with larger robot team sizes. The distributed SAFDetection approach presented in this paper shares the theoretic foundation with centralized SAFDetection but works in a distributed manner.

#### A. Training stage

Similar to centralized SAFDetection, the distributed approach is also a training-classification based method. In the training stage, PCA is performed on the history of each robot's local sensor data (i.e., training data) during normal operation to automatically select the essential features for individual robots. The selected feature data set is then mapped into a sequence of single robot states using the Fuzzy C-means clustering algorithm (Euclidean distance is used). After that, the state transition diagrams are built with the sequence of robot states. In centralized SAFDetection, the team robot is regarded as a single monolithic robot and only one state transition diagram is generated to describe

the normal behavior of the entire robot team. Distributed SAFDetection, on the other hand, learns two kinds of state transition diagrams in the training stage: the individual (local) one and the team (global) one (shown in Figure 2).

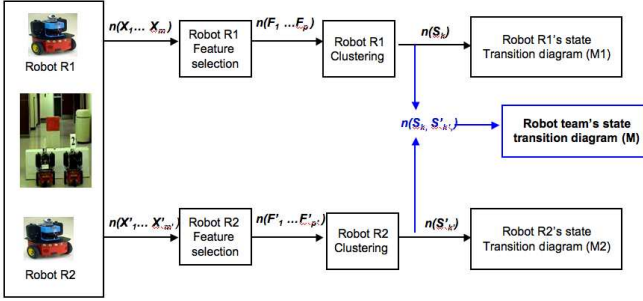


Fig. 2. The training stage in the distributed SAFDetection approach (for 2-robot team).

Compared to the centralized version (Figure 1), in distributed SAFDetection (shown in Figure 2), feature selection, clustering and the building of the local state transition diagram are all accomplished on the client side locally, while only the team state transition diagram is built on the server. More importantly, since the client robots perform clustering only with their own local sensor data, the data dimension is limited and the curse of dimensionality is broken.

The individual state transition diagram is built to describe the normal behavior of the single robot with its local sensor data. The steps to build the individual state transition diagram (Algorithm 1) are quite similar to the centralized approach, except that only local sensor data is used. The team state transition diagram is generated with a client/server model. The server (which can be either a robot or a PC) collects the individual states (achieved in Algorithm 1, step 3) from each client robot. The vector of those individual states is used to represent the team state; the team state transition diagram is generated with the sequence of team robot states in the same way. The steps for building the team state transition diagram are shown in Algorithm 2.

---

**Algorithm 1** Steps for building the individual state transition diagram in distributed SAFDetection.

---

- 1: For every robot in the team
  - 2: **for** each time step **do**
  - 3: Collect data from the robot local sensors during normal operation.
  - 4: **end for**
  - 5: Perform PCA on the history data and select features that cover sufficient (user defined) information.
  - 6: Perform clustering algorithm on selected features to create individual robot states.
  - 7: Build individual state transition diagram with the sequence of individual robot states.
- 

Algorithm 2 shows that the team robot state is a vector of size  $n$ , where  $n$  is the size of the robot team. Suppose that

---

**Algorithm 2** Steps for building the team state transition diagram in distributed SAFDetection

---

- 1: For the server robot in the team
  - 2: **for** each time step **do**
  - 3: Collect individual state index ( $S_{r_i}$ ) from all  $n$  client robots in the team.
  - 4: Create vector  $\langle S_{r_1}, S_{r_2}, \dots, S_{r_i}, \dots, S_{r_n} \rangle$  as the team robot state.
  - 5: **end for**
  - 6: Build team state transition diagram with the sequence of team robot states.
  - 7: Send the team state transition diagram to all  $n$  client robots.
  - 1: For the client robot in the team
  - 2: **for** each time step **do**
  - 3: Send individual state index ( $S_{r_i}$ ) to the server robot.
  - 4: **end for**
  - 5: Save the team state transition diagram created by the server robot.
- 

client robot  $r_i$  has number  $c_i$  individual states. The possible number of team robot states is  $c_1 \times c_2 \times \dots \times c_i \times \dots \times c_n$ , which is of exponential magnitude  $c_{max}^n$  ( $c_{max}$  is the maximum value of  $c_i$ ). However, since only robot team states that exist when the robot team is in the normal tightly-coupled behavior are created, the final number of robot team states is much less than the theoretical possible number in real applications. In addition, given the increasing memory that one robot can have nowadays, even a very large matrix can be stored easily.

One issue about using a combined vector to represent the team state is that it results in some transient states because of asynchronization among robots, which can cause false alarms during detection. For example, in one task, two robots in the same team are designed to move and stop at the same time; but in practice, these actions do not happen simultaneously. In centralized SAFDetection, all robot sensor data is collected to perform clustering; the small difference of speed is ignored given other correlated features are correct. However, with the distributed approach, the clustering is performed locally on the individual robot and the difference in speed is regarded as two states. When constructing the robot team state from the individual robot state, transient states such as  $\langle \text{move}, \text{stop} \rangle$  and  $\langle \text{stop}, \text{move} \rangle$  are generated. Thus, two rules are introduced to reduce this side-effect. First, when building the team state transition diagram, time durations in each team state (combined from individual states) are checked and those that have a very short time duration (e.g., 1 second or less) are removed. Second, in the detection stage, only faults that last for some minimum time will be reported as actual alarms.

### B. Detection stage

In the training stage, each robot on the team maintains its own individual transition diagram, as well as the team state transition diagram. Therefore, each robot on the team

knows the normal behavior pattern of itself, as well as of the entire team. In the detection stage, an individual robot uses the individual state transition diagram to detect local faults; it uses the team state transition diagram to detect interactive faults among robots.

1) *Information sharing and updating*: To detect interactive faults using the team state transition diagram, each robot keeps a copy of its teammates' latest communicated state to track the entire team's state in real time. Critical information sharing and updating are implemented with a peer-to-peer model via message passing. To reduce the asynchronization caused by message transition time delays, broadcasting is used without acknowledgment or other handshaking protocols. The message format used is <Update, sender id, receiver id, current state of sender>. To make sure that the information is shared and updated correctly and efficiently, two kinds of messages (with the same format) are transmitted among robots to update team information so that every team member can keep track of the state of the entire team. Each robot on the team broadcasts its current state to other teammates at a particular frequency, which can be defined by the user according to the network situation. This broadcast message is called a *periodic message*. When one robot detects a change of its own individual state, it broadcasts its new state to all other members on the same team. This broadcast message is called an *update message*.

This message passing strategy provides a solution to the need for information sharing and updating in distributed SAFDetection that balances reliability and efficiency. The periodic message reduces the message transmission loss risk; the update message reduces unnecessary communications among robots. Broadcasting without acknowledgment cuts down on the message transmission time between robots. Note that with this method of communication in the distributed approach, the original sensor data is stored locally and only robot states are transmitted among robots, compared to the need to transmit complete sensor data in the centralized approach. For example, in the box pushing task described later in this paper, the centralized implementation requires more than 400 real time sensor data readings to be collected and transmitted from the client robots to the server to detect real time faults. On the other hand, the distributed approach requires only simple cluster index values to be transmitted between the robots to update the entire team's real time state for fault detection. Therefore, the network traffic load is actually significantly reduced compared to the centralized approach.

However, with a very unreliable network, the message loss can greatly affect the performance of fault detection. Suppose the average dropped message rate of the network is  $p$ , the periodic message is sent with frequency  $f$ , and the message transmission time from robot A to its teammate robot B is  $t$ . Then, the average time between when robot A detects a state change and when robot B receives this information update

is  $T$ , as given by:

$$T = \sum_{n=1}^m (1-p) \times p^{n-1} \times [t + (n-1)/f] \quad (1)$$

where  $m$  counts how many times the update message is sent before robot B receives the update information. In theory,  $m$  can be arbitrarily large for any network whose average dropped message rate  $p$  is greater than 0, since there is always a non-zero probability that the message is dropped after  $m$  times. In real applications, we assume that messages are transmitted when the probability of successful transmission is  $\geq 99.99\%$ . In most cases, the message transmission time  $t$  for a particular network is fixed. If the network is not reliable and the dropped message rate  $p$  is high, then a higher periodic frequency  $f$  is needed to obtain an acceptable  $T$ . On the other hand, the user can choose a lower periodic frequency with a very stable and reliable network. (In our physical robot experiments, the network transmission time was thirty milliseconds, and the average dropped message rate was less than 1%. Thus, according to Equation 1 a periodic frequency of 0.2 Hz can be used to keep the message delay time  $T$  within 100 milliseconds. These are the values used in our experiments.)

2) *Fault detection*: Fault detection in distributed SAFDetection is quite similar to the centralized approach. The online sensor data is collected in a real time manner and filtered to select useful features according to the selected features in the learning stage. After that, the feature values are classified into a real time robot state according to the clustering information achieved during the learning stage. Then, this real time robot state is checked with the state transition diagram to detect faults. The difference from the centralized approach is that the data collection, feature filtering, classification, and local fault detection are completed on the local robot using its own local data. As previously described, distributed SAFDetection has two kinds of state transition diagrams: individual and team. These two diagrams are checked online to detect local fault alarms and team fault alarms in real time, as shown in Algorithm 3. These alarms will then be sent to the upper layer to detect the final faults.

#### IV. EXPERIMENTAL RESULTS

We have implemented both the centralized and the distributed SAFDetection approaches and tested them with several multi-robot cooperative tasks on physical robots and in simulation. In this section, a cooperative multi-robot box pushing task and a multi-robot following task are tested to demonstrate the feasibility of distributed SAFDetection, and to compare it to the centralized version. Simulation of the multi-robot following task is used to validate the robustness and scalability of the distributed approach. Performance is evaluated based on fault detection rate and false alarm rate. The physical experiments are implemented on Pioneer 3DX mobile robots in the hallway outside our laboratory (as shown in Figure 3), while the robot simulations are built using Stage [7]

---

**Algorithm 3** Steps for detecting fault alarms with distributed SAFDetection

---

```
1: Each robot does the following:
2: Initialize the local copy of teammates' state.
3: for each time step do
4:   if no update message received from teammate for time
      $T$  (from Equation 1) then
5:     Communication fault is detected.
6:   end if
7:   Collect real time data from the robot local sensors.
8:   Map the original sensor data to features that were
     selected in the learning stage.
9:   Classify the features into an individual robot state
     based on cluster centers achieved in the learning stage.
10:  Check the individual state transition diagram to detect
     local faults.
11:  if the individual state is different from the state in the
     last time step then
12:    Send update message to teammates.
13:    Check the team state transition diagram to detect
     team faults.
14:  else
15:    if it is the periodic time then
16:      Send periodic message to teammates.
17:    end if
18:  end if
19:  if receive message from teammate then
20:    Compare the newly arrived teammate's state to the
     local copy.
21:    if there is a change then
22:      Update the local copy.
23:    Check the team state transition diagram to detect
     team faults.
24:  end if
25: end if
26: end for
```

---

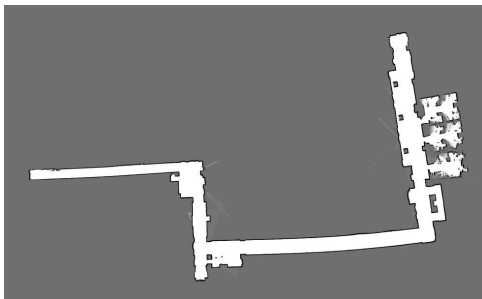


Fig. 3. Laser map of the experimental environment.

#### A. Cooperative multi-robot box pushing task

The cooperative multi-robot box pushing task requires two robots to cooperatively push a long box from a starting position to a goal position that is indicated by a red blob. During the pushing process (around 1-2 minutes), the robots can adjust their speed and pushing directions (i.e., alignment) to avoid losing the goal position; they communicate with

each other to coordinate their actions. To train SAFDetection, twenty (20) normal box pushing trials were performed to collect more than one thousand (1000) training data entries.

Using the centralized approach, PCA is performed on the following sensor features from the two-robot team:

- Both robots' laser range in 180 directions
- Both robots' sonar range in 16 directions
- Red blob's left, right, upper, bottom edge position in both camera images
- Both robots' speeds
- Both robots' turn rates
- Both robots' battery charge level

From the automated PCA analysis, ten (10) features are selected, as follows:

- Two principal components of laser and sonar data (1 for each robot)
- Four principal components of red blob data (2 for each robot)
- Three principal components of speed and turn-rate for robot team
- Battery charge

In the distributed approach, PCA is performed on each individual robot's available sensor features; six (6) features are automatically selected for each robot in the team, as follows:

- One principal component of laser and sonar data
- Two principal components of red blob data
- Robot speed
- Robot turn rate
- Battery charge

In the SAFDetection approach, Fuzzy C-means clustering is performed on the selected feature values to find the state of the robot system. The centralized approach generates five (5) team states for the entire robot team, while the distributed approach finds four (4) individual states for each robot on the team. With distributed SAFDetection, the robot team state is represented by a vector combination of the individual robot states, which, in this box pushing task, is  $\langle \text{left robot's state, right robot's state} \rangle$ . In theory, there are sixteen ( $16 = 4 \times 4$ ) possible combinations; however, only five (5) of these exist in the training data, which is consistent with the centralized approach's result.

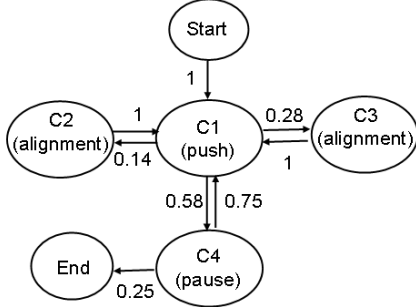
The learned individual state transition diagram of the left robot in the box pushing task is shown in Table I, along with the average state transition probabilities between states and the mean and standard deviation values of the time the robot remains in each state (i.e., before transiting to a different state). The learned team diagram is shown in Table II, along with the time durations in each state. This team behavior model of normal operation that is generated by distributed SAFDetection is consistent with the normal team behavior model generated by the centralized approach [14].

Both centralized and distributed SAFDetection detect faults efficiently but for different reasons. For example, in a scenario where the right robot gets stuck and the left robot keeps moving forward, the centralized SAFDetection

TABLE I

NORMAL BEHAVIOR MODEL FOR THE LEFT ROBOT IN THE BOX PUSHING TASK, BUILT BY DISTRIBUTED SAFDETECTION.

(a) State transition diagram.



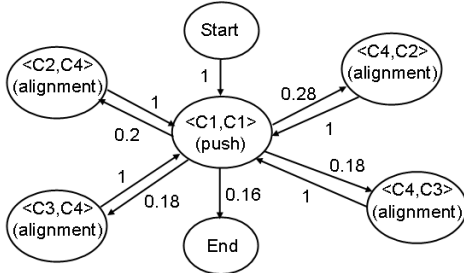
(b) Time duration in each state.

State	C1	C2	C3	C4
Mean time (s)	28.2	3.56	4.12	4.50
Standard derivation (s)	7.97	0.40	0.23	0.38

TABLE II

NORMAL BEHAVIOR MODEL FOR THE TEAM IN THE BOX PUSHING TASK, BUILT BY DISTRIBUTED SAFDETECTION.

(a) State transition diagram.



(b) Time duration in each state.

State	(C1,C1)	(C2,C4)	(C3,C4)	(C4,C2)	(C4,C3)
Mean (s)	28.8	2.56	4.42	4.53	3.91
Std (s)	7.92	0.72	0.23	0.68	0.79

approach detects the “stuck” fault because the server detects an unknown state. With FCM clustering, this means that the clustering result of the robot team’s sensor data does not have a high probability of being in any of the known clusters. In the distributed SAFDEtection approach, when the right robot gets stuck, it detects this state change (from “push” to “pause”) by clustering the local sensor data and comparing the current robot team state (i.e., state  $\langle \text{push}, \text{pause} \rangle$ , which is  $\langle C1, C4 \rangle$ ) with the team state transition diagram. It then checks the robot team’s normal behavior model; the “stuck” fault is detected since the current combined robot team’s state is unknown in the team state transition diagram. In addition, the right robot also broadcasts its state change to its teammate, causing the left robot to also detect the “stuck” fault by checking the team state transition diagram.

Another fault example, “blob missing”, was successfully detected by both centralized and distributed SAFDEtection.

In this scenario, the red blob that indicates the goal position is taken away during the pushing process. The centralized SAFDEtection detects the fault by observing an unknown team state by clustering the team robots’ sensor data. In distributed SAFDEtection, both the left and right robots detect the fault by clustering their local sensor data.

These examples show that both the centralized and distributed approach can detect faults efficiently. The centralized approach detects faults using the clustering result of the team robots’ sensor data. The distributed approach distributes the computational load and sometimes can detect faults with only local information (e.g., the blob missing case). Another advantage of the distributed approach is that faults can be detected by more than one robot in the team, instead of only the server. Table III compares the fault detection rates of centralized and distributed SAFDEtection with twenty (20) normal trials and twenty (20) abnormal trials of the box pushing task. These results show that distributed SAFDEtection has only a slightly higher false alarm rate, since transient states, as a result of the asynchronism between the robots, are detected as faults. In future work, further adapting the mechanism for handling asynchronism may lead to these approaches being nearly identical for this problem.

TABLE III

FAULT DETECTION RATES BY CENTRALIZED AND DISTRIBUTED SAFDETECTION FOR BOX PUSHING TASK.

	Centralized SAFDEtection	Distributed SAFDEtection
True positive	85%	80%
True negative	95%	95%
False positive	15%	20%
False negative	5%	5%

### B. Multi-robot following task

To examine issues of scalability to larger robot team sizes, we also applied both centralized and distributed SAFDEtection to the multi-robot following task, in which multiple (2-5) robot team members move from a starting position to a goal position, following each other by tracking the red blob adhered to the previous robot. We implemented this approach both on physical robots (involving 2 Pioneer robots) and in simulation in Stage (involving up to 5 robots).

With the two-robot following task, both the centralized and distributed approach can efficiently detect faults (such as a robot losing track of its leader robot, or a follower robot beginning to track spurious “non-robot” red-colored objects in the environment). Table IV shows the fault detection results for the robot following task by centralized and distributed SAFDEtection with ten (10) normal trials and ten (10) abnormal trials.

However, as the robot team size increases, the centralized SAFDEtection approach has difficulties. For instance, when the robot team size is five (5) (as shown in Figure 4), FCM clustering in the centralized approach is performed on the robot team’s twenty-three (23) essential features (selected by

TABLE IV  
FAULT DETECTION RATES BY CENTRALIZED AND DISTRIBUTED  
SAFDetection FOR 2-ROBOT FOLLOWING TASK.

	Centralized SAFDetection	Distributed SAFDetection
True positive	80%	70%
True negative	90%	90%
False positive	20%	30%
False negative	10%	10%

PCA) to determine the robot team's state. Different cluster numbers (from two (2) to twenty (20)) are tested to find the best results. However, the results show that clustering with two (2) clusters has the highest quality; this actually reflects the failure of clustering, since there are obviously more than two robot team states in this task. This failure is a result of the curse of dimensionality. Therefore, centralized SAFDetection is not appropriate for the five-robot following task. However, since the sensor data for individual robots is limited, distributed SAFDetection can be used in this task.

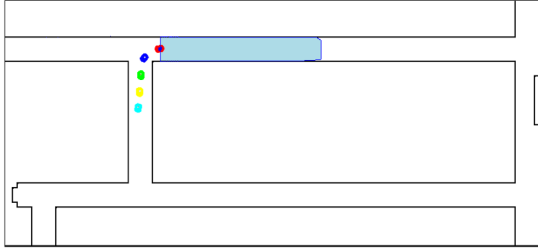


Fig. 4. Five-robot following task implemented with Stage.

With distributed SAFDetection, only six (6) essential features are selected for each individual robot in the team; FCM clustering maps the local sensor data of each robot to three (3) states by using six (6) essential features. As an example, the individual state transition diagram learned by the second robot in the five-robot following task is shown in Table V; the individual state transition diagrams of other robots in the team are similar.

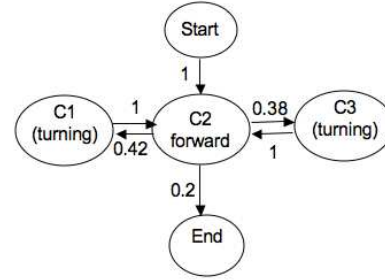
For the 5-robot team, since each individual robot has three states, there are 243 ( $3^5$ ) possible team robot states in total. However, only eleven (11) states actually exist in the training data. Table VI shows the robot team states resulting from the combination of individual robots and their meanings for this task and the state transition diagram of the entire robot team. The distributed SAFDetection approach is able to successfully use this model to detect faults in the simulated 5-robot following task, such as a robot getting stuck when turning a corner.

Table VII shows the number of features and states generated by the centralized and distributed SAFDetection approaches for different robot team sizes. As this data shows, the centralized approach is not scalable to team sizes greater than three (3), due to the increasing number of features.

An additional observation is that it is sometimes unneces-

TABLE V  
NORMAL BEHAVIOR MODEL FOR THE SECOND ROBOT IN THE  
FIVE-ROBOT FOLLOWING TASK.

(a) State transition diagram.

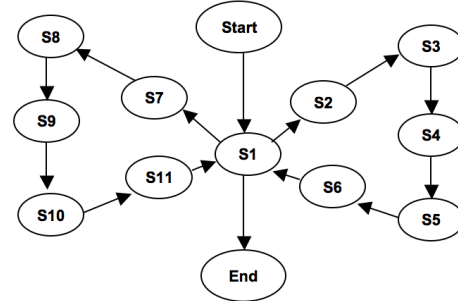


(b) Duration time in each state.

State	C1	C2	C3
Mean time (s)	4.91	20.1	5.47
Standard derivation (s)	0.81	9.22	0.34

TABLE VI  
NORMAL BEHAVIOR MODEL FOR THE TEAM IN THE FIVE-ROBOT  
FOLLOWING TASK, BUILT BY DISTRIBUTED SAFDetection..

(a) State transition diagram.



(b) Meaning of the states.

Team state	Combination	Meaning
S1	(C1,C1,C1,C1,C1)	The robot team moves forward
S2	(C2,C1,C1,C1,C1)	Robot $R_1$ is turning left
S3	(C1,C2,C1,C1,C1)	Robot $R_2$ is turning left
S4	(C1,C1,C2,C1,C1)	Robot $R_3$ is turning left
S5	(C1,C1,C1,C2,C1)	Robot $R_4$ is turning left
S6	(C1,C1,C1,C1,C2)	Robot $R_5$ is turning left
S7	(C3,C1,C1,C1,C1)	Robot $R_1$ is turning right
S8	(C1,C3,C1,C1,C1)	Robot $R_2$ is turning right
S9	(C1,C1,C3,C1,C1)	Robot $R_3$ is turning right
S10	(C1,C1,C1,C3,C1)	Robot $R_4$ is turning right
S11	(C1,C1,C1,C1,C3)	Robot $R_5$ is turning right

sary to build a robot team behavior model for the entire team; instead, these models can be built for certain subgroups of the team. For example, in the 5-robot following task, robot R1 is not directly related to robot R5; robot R5 can detect its own interactive faults by only checking the state of robot R4. Therefore, a subgroup state transition diagram can be used to replace the state transition diagram for the entire team.

TABLE VII  
PROBLEM SIZES FOR THE ROBOT FOLLOWING TASK, FOR VARYING  
NUMBERS OF ROBOTS.

Team size	# of features Centralized	# of features Distributed	# of team states Centralized	# of team states Distributed
2	8	5	5	5
3	13	5	6	7
4	18	5	2	9
5	23	5	2	11

### C. Analysis and future work

Our results show that both centralized and distributed SAFDetection are efficient for detecting faults when the robot team size is small ( $\leq 3$ ), although the distributed approach results in a slightly higher false alarm rate because of asynchronism between the robots. The centralized approach has been shown to be inappropriate for larger sized robot teams; however, the distributed approach continues to be appropriate as the team size grows. Note that even though larger team sizes can lead to a larger number of team states in the distributed version, this is not generally a problem for the approach; instead, the critical factor is the number of features, which remain relatively constant in the distributed approach. The choice of which SAFDetection approach to use is dependent on the user's requirements. In general, when the robot team size is small ( $\leq 3$ ), the communication bandwidth is sufficient, and the server of the robot team is reliable, the centralized approach is preferred. On the other hand, the distributed approach is much more appropriate if the robot team size is large, the communication bandwidth is limited, or there is no reliable server.

In future work, we plan to develop methods for incorporating prior knowledge of fault types into the model, and for reducing the sensitivity of the system to robot asynchrony. Additionally, we plan to apply this work to more complex multi-robot tasks, to fine-tune the approach to further improve its accuracy rate, to test robots with the same training task in different environments, and to test robots with reduced training data.

## V. CONCLUSION

In this paper, we have extended the SAFDetection approach from a centralized version to a distributed version, and illustrated how it can be implemented in a tightly-coupled multi-robot team task to successfully detect faults. The motivation for building a distributed system is to increase the scalability to larger team sizes by breaking the curse of dimensionality and by cutting down the network traffic. In addition, a peer-to-peer model is used to replace the client/server model in real time detection, which improves the reliability of the system. We presented simulation and experimental results to validate the approach, and to compare the centralized and distributed versions of SAFDetection. Experiments are performed in simulation and on physical robots in two applications: box pushing and following. The results show that the distributed approach offers a more

reliable and extensible mechanism to make SAFDetection scalable to applications of larger robot team sizes.

## REFERENCES

- [1] J. C. Bezdek, R. Ehrlich, and W. Full. FCM: The fuzzy c-means clustering algorithm. *Computers and Geosciences*, 10(2-3):191–203, 1984.
- [2] Swarup Bhunia and Kaushik Roy. Fault detection and diagnosis using wavelet based transient current analysis. In *Proceedings of the conference on Design, Automation and Test in Europe table of contents*, page 1118, 2002.
- [3] Z. Cai and Z. Duan. A multiple particle filters method for fault diagnosis of mobile robot dead-reckoning system. In *IEEE International Conference on Intelligent Robots and Systems*, pages 481–486, 2005.
- [4] W. Chen, R. Gong, and K. Dai. Two new space-time triple modular redundancy techniques for improving fault tolerance of computer systems. In *IEEE International Conference on Computer and Information Technology*, 2006.
- [5] Anders Lyhne Christensen, Rehan O'Grady, Mauro Birattari, and Marco Dorigo. Fault detection in autonomous robots based on fault injection and learning. *Auton. Robots*, 24(1):49–67, 2007.
- [6] Erick Delage, Honglak Lee, and Andrew Y. Ng. A dynamic Bayesian network model for autonomous 3D reconstruction from a single indoor image. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2418–2428, 2006.
- [7] B. Gerkey, R. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the International Conference on Advanced Robotics*, pages 317–323, 2003.
- [8] M. Hashimoto, H. Kawashima, and F. Oba. A multi-model based fault detection and diagnosis of internal sensor for mobile robot. In *IEEE International Conference on Robotics and Automation*, pages 3787–3792, 2003.
- [9] A. H. Jackson, R. Canham, and A. M. Tyrrell. Robot fault-tolerance using an embryonic array. In *NASA/DoD Conference on Evolvable Hardware*, pages 91–100, 2003.
- [10] B.P. Jeppesen and D. Cebon. Analytical redundancy techniques for fault detection in an active heavy vehicle suspension. *Vehicle System Dynamics*, 42:75–88, 2004.
- [11] B. Krishnamachari and S. Iyengar. Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. *IEEE Transactions on Computers*, 53(3):241–250, 2004.
- [12] I. S. Lee, J. T. Kim, and J. W. Lee. Model-based fault detection and isolation method using ART2 neural network. *International Journal of Intelligent Systems*, 18:1087–1100, 2003.
- [13] M. L. Leuschen, J. R. Cavallaro, and I. D. Walker. Robotic fault detection using nonlinear analytical redundancy. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 456–463, 2002.
- [14] Xingyan Li and Lynne E. Parker. Sensor analysis for fault detection in tightly-coupled multi-robot team tasks. In *IEEE International Conference on Robotics and Automation*, pages 3269–3276, 2007.
- [15] Xingyan Li and Lynne E. Parker. Design and performance improvements for fault detection in tightly-coupled multi-robot team tasks. In *IEEE SoutheastCon*, 2008.
- [16] J. P. Matsuura and T. Yoneyama. Learning Bayesian networks for fault detection. In *Proceedings of the IEEE Signal Processing Society Workshop*, pages 133–142, 2004.
- [17] M. H. Sadeghi, J. Raflee, and F. Arvani. A fault detection and identification system for gearboxes using neural networks. In *International Conference on Neural Networks and Brain*, pages 964–969, 2005.
- [18] E. N. Skoundrianos and S. G. Tzafestas. Modelling and fdi of dynamic discrete time systems using a mlp with a new sigmoidal activation function. *Intelligent and Robotic Systems*, 41, 2004.
- [19] V. Verma and Simmons. Scalable robot fault detection and identification. *Robotics and Autonomous Systems*, 54:184–191, 2006.
- [20] T. Yairi, Y. Kato, and K. Hori. Fault detection by mining association rules from house-keeping data. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2001.