Proc. of IEEE International Conference
on Intelligent Robots and Systems, Beijing, China, 2006.

Adaptive Causal Models for Fault Diagnosis and Recovery in Multi-Robot Teams

Lynne E. Parker and Balajee Kannan
Department of Computer Science, Distributed Intelligence Laboratory
The University of Tennessee, Knoxville, TN 37996–3450
Email: {parker, balajee}@cs.utk.edu

***Abstract*— This paper presents an adaptive causal model method (adaptive CMM) for fault diagnosis and recovery in complex multi-robot teams. We claim that a causal model approach is effective for anticipating and recovering from many types of robot team errors, presenting extensive experimental results to support this claim. To our knowledge, these results show the first, full implementation of a CMM on a large multi-robot team. However, because of the significant number of possible failure modes in a complex multi-robot application, and the difficulty in anticipating all possible failures in advance, our empirical results show that one cannot guarantee the generation of a complete *a priori* causal model that identifies and specifies all faults that may occur in the system. Instead, an adaptive method is needed to enable the robot team to use its experience to update and extend its causal model to enable the team, over time, to better recover from faults when they occur. We present our case-based learning approach, called LeaF (for Learning-based Fault diagnosis), that enables robot team members to adapt their causal models, thereby improving their ability to diagnose and recover from these faults over time.**

I. Introduction

Mobile robot teams are becoming increasingly useful for applications such as security, urban search and rescue, or autonomous warehousing. These applications require complex coordination among robots performing multiple tasks such as planning, mapping, localization, formation-keeping, and information sharing. For multi-robot systems to be commercially and practically viable, these systems must be efficient and robust. However, as these systems involve many subsystem behaviors, algorithms, and components, and typically operate in environments that are highly dynamic, they usually experience a high likelihood of some type of subsystem failure. Even with extensive system design and testing, breakdowns and faults are nevertheless commonplace, as reported in a recent study by Carlson and Murphy [4] cataloging how unmanned ground vehicles physically fail in the field. Faults can stem from a number of sources, including malfunctions, miscalibration, miscoordination, software errors, limited communications, power failures, and so forth.

Robustness is a key challenge in the face of uncertainty in complex, dynamic environments for intelligent agents [13]. The major drawback of most current methods is that the designer/programmer is responsible for accounting for all possible failures during the design stage. Certainly, the manual identification, classification and analysis of the possible faults that might be encountered by the multi-robot system provides a good starting point about how and when to apply fault diagnosis and recovery. However, despite extensive *a priori* analysis of the possible failure modes in a given application, physical experimentation nearly always reveals some previously unknown fault(s). The inherent explosion of state space complexity [3] for multi-robot teams operating in dynamic environments inhibits the ability of any designer to anticipate all possible failure modes in advance.

Relatively little research has addressed the issue of fault diagnosis and recovery in large-scale multi-robot teams. For a robot team, the faults that occur during the course of operation provide an opportunity for the system to learn about its environment, thereby adapting to it. Based on this key principle, we have designed a fault diagnostic architecture for multi-robot teams called LeaF (for Learning-based Fault diagnosis). LeaF uses a partial causal model for representing possible faults in the system. This pre-specified partial causal model provides the team with knowledge from the designer in advance of the application. When a new fault is encountered, the system uses a case-based reasoning (CBR) approach to attempt to extract a relevant recovery action from prior experience. The causal model is then adapted to include the new fault information, making it available for future diagnosis and recovery. The ability of the system to effectively learn from its own faults makes it a more robust and efficient team that is better suited for practical application.

In the remainder of this paper, we first present a brief review of related work in Section II. We then validate the idea of using a causal model in multi-robot team applications in Section III by presenting and analyzing extensive results from a physical robot implementation of a complex heterogeneous application [7]. This application was implemented using the causal model method (CMM) for a subset of the types of failures that the team was anticipated to experience. Based upon our extensive post-experimental analysis, we argue that it will nearly always be impossible to anticipate all the failure modes that such a system could experience. A much more robust solution would enable the robot team to adapt and extend its initial causal model on the basis of experience. In Section IV, we present our adaptive causal model approach, called LeaF, that enables a robot team to use case based learning to adapt and extend the causal model provided by the system designer. In Section V, we present some experiments and analyses that illustrate some of the benefits of the proposed approach. We offer concluding

remarks and comments on our ongoing work in Section VI.

II. Related Work

For large-scale swarm robotics, fault tolerance is a built-in feature of the system, due to the interchangeability of the robot team members. However, for large-scale heterogeneous robot teams performing complex tasks with many interdependencies, fault tolerance is not a built-in feature. An example of a complex heterogeneous multi-robot application we are addressing in this work is a "locate and protect" application [7], in which a heterogeneous mobile robot team must explore and map a large indoor environment, detect a valued object, deploy a sensor network, and use the network to track intruders within the building and protect the valued object. The capabilities of the robots dictate the particular combination of cooperative and single-robot algorithms and behaviors involved. In our research, this application scenario involves a complex combination of cooperative and single-robot behaviors, including laser-based localization, path planning, obstacle avoidance, vision-based autonomous teleoperation, simple vision-based following, and wireless ad hoc mobile communication. Because of the complexity of the behaviors and the operation of the application in a previously unknown environment, a wide variety of possible failure modes are possible.

According to a NASA survey conducted by Cavallaro and Walker [5], the reliability and fault-tolerance efforts in robotics are application-specific, with few existing standards and protocols for implementing system fault-tolerance. The experimental nature of typical mobile robotic equipment and its uncertain interactions with the environment make detailed modeling for multiple failures, such as the approach used in industrial robotics, difficult in teams of mobile robots. Because of this, most researchers have focused on single instance fault diagnosis systems for mobile robots rather than multiple instance failures.

A common approach to fault diagnosis makes use of knowledge-based techniques to reason about the state of the execution, detect anomalies, and to autonomously generate a recovery plan. Murphy and Hershberger have suggested a two-step approach: a strategy for classifying sensor failures, and a recovery strategy. The Sensor Fusion Effects Architecture (SFX-EH) [12] for handling sensing failures in autonomous mobile robots is based on this two-step methodology, using extensions to the generate-and-test method to classify failures based on a partial causal model of the sensor/environment/task interactions for the robot. Although work by Long, et al. [11], investigates extending the SFX-EH architecture from a single robot to a small team of distributed robots, the architecture was primarily designed for single robot sensing fault diagnosis.

Kaminka and Tambe present an approach for monitoring and diagnosis for multi-agent domains called SAM — Socially Attentive Monitoring [9]. SAM uses social psychology-based fault detection, in which an agent utilizes other agents as a source of information for detecting failures. Social diagnosis is performed by reasoning about failures using an explicit model of teamwork and uses model sharing to alleviate inefficiencies in model representation.

A similar approach is the Causal Model Method (CMM) [8], which is a model-based approach for fault diagnosis that predefines a decision graph for detecting and diagnosing problems that occur during system operation. The CMM was initially designed to address performance issues in situation-specific coordination strategies. In this method, the strategy used for agent coordination must be tailored to meet the specifics of the current environment and the coordination situations an agent will encounter. Our approach is built on the Causal Model Method (CMM); however, we make several modifications to the original CMM approach, to make it better suited to the multi-robot domain.

III. Validating the CMM in Large-Scale Multi-Robot Teams

To illustrate the utility of the causal model method for fault diagnosis and recovery in multi-robot teams, we present our extensive experimental studies using a pre-defined causal model for the "locate and protect" multi-robot application. First, we provide a brief background on the causal model method and our extensions to apply this work in the multi-robot domain. We then provide results from implementing this approach in physical robot experiments.

A. Causal Model Method

The original CMM as described by Horling and Lesser [6] was defined as a directed, acyclic graph, written left to right, used for organizing a set of diagnosis nodes. Each node in the graph corresponds to a particular diagnosis with the precision of reasoning increasing from left to right. As the nodes produce diagnosis, the causal model can be used to find more detailed levels of diagnosis to better identify the problem. The nodes on the extreme right, known as the leaf nodes, provide the most detailed level of diagnosis and are generally at the sensory level. Other nodes in the model are marked as triggerable, meaning that they periodically perform simple comparison checks to determine if a fault may have occurred. The system then checks for deviations from the expected value of the characteristics and if a deviation is detected, a flag is set for that error. This in turn triggers the diagnosis model to identify the exact source of the fault. This trigger-checking activity is a primary mechanism for initiating the diagnostic process.

Multiple faults are handled by incorporating a single fault node that uses the diagnosis of several other nodes for verification. The causal model allows easy addition and removal of nodes by the designer depending on the specifics of the diagnosis required for the system. The designer is also free to make the nodes as domain independent as possible. By making the design as generic as possible, the model can be applied to each agent or robot with little modification. On the other hand, the design and the precision of the model is also dependent on the individual capabilities, the computational power, sensors, etc., of each individual or type of robot in a multi-robot team.

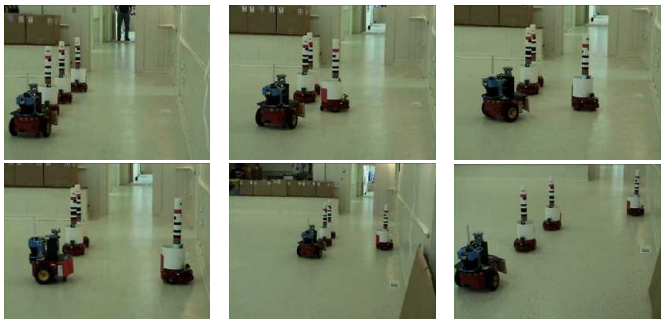

Fig. 1. Deployment of a sensor robot using assistive navigation: the lead robot first guides and then directs the sensor robot into position.

Therefore a system can have one or multiple causal models depending on the team composition.

To apply the CMM to the multi-robot domain, we made a number of modifications to the original model, including:

- Changing the representation to be an undirected graph, enabling the system to incorporate new information anywhere within the causal model;
- Making all nodes triggerable, meaning that every node actively monitors for a variation from known behavior knowledge;
- Representing behavior knowledge by symptoms associated with the fault, influence of the fault towards task completion, etc.,
- Incorporating the generate-and-test methodology from the SFX-EH architecture [12] to help trace faults; and
- Associating corrective actions with each node, consisting of a set of instructions that the robot has to successfully execute to recover from the encountered fault.

Using the CMM approach, we pre-defined a causal model for part of the "locate and protect" scenario, based upon our *a priori* experience in testing and debugging the complex heterogeneous robot behaviors of this application. In particular, the causal model was used for the deployment of the sensor network within the mapped building. Due to the limited capabilities of the sensor network robots, the deployment part of this application involved a complex set of cooperative behaviors implementing autonomous teleoperation [7], in which a leader robot led a chain of simple sensor net robots to their deployment positions. An example of this deployment process is shown in Figure 1.

Because the leader robots in this application were the most critical to the success of the mission, our causal model focused on recovery actions that enabled the leader robots to recover from anticipated failures. Figure 2 illustrates the implemented causal model for this part of the application. This causal model was implemented on the leader robots using a rule-based approach that defined the various possible failure modes represented in the causal model. Table I lists the rules used to recognize the failure states predefined and implemented for this application, along with the corresponding corrective actions. The comparison monitor used for fault detection was based on a case-based reasoning structure. A set of rules describing the fault characteristics were defined for the structure; in the event of a match, a flag was set, diagnosis performed, and a corrective action taken. For our implementation, the characteristic set describing the fault was restricted to a single defining element – the symptom representing a fault. The symptom was derived from the system exception that was raised when a fault occurred. By comparing the symptoms of the encountered fault with the existing rule-base, we can identify the fault.

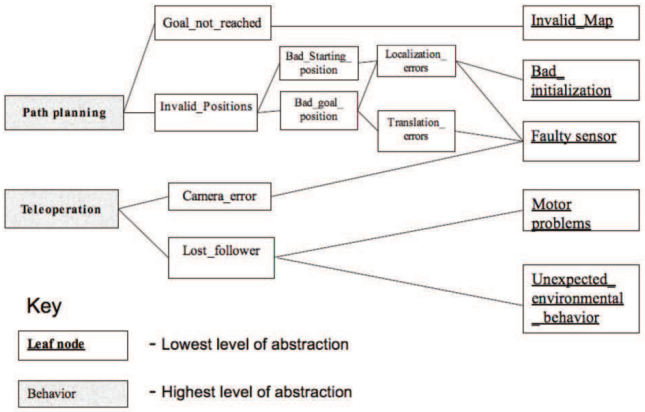

Fig. 2. Initial causal model for the deployment behaviors in the locate-and-protect scenario.

B. CMM Results in Physical Robot Application

This CMM was tested extensively on physical robots performing deployments of sensor network robots. The experiments consisted of repeated deployments of 1, 2, or 3 simple robots per team. Over the course of the experiment, various failures were encountered, some of which were expected and some that were totally unexpected. If a deployment failed on one experiment, the consequences of that failure were not corrected, except on rare occasions. Thus, the data reported here incorporates propagation of error from one experiment to the next. This is an important difference between single robot and multi-robot teams in terms of fault occurrence. In these experiments, a total of 61 simple robot deployments were attempted. The experimental data showed an overall deployment success rate of 60% - 90%, depending upon the environmental characteristics. In other words, for each attempt at deploying a simple robot, 60% - 90% of those robots successfully reached their planned deployment position. The (sometimes) low success rate for deployment was due to the large number of failure modes of the multi-robot system, which was composed of several non-trivial modules, including localization, path planning, navigation, helper following, visual marker detection, and inter-robot communication. Table II states the probability of success of each individual module in this implementation and the overall system probability, based upon the experimental results. Each component/module is designed to be highly reliable, yet due to the overall interaction between the components, the overall system probability is

TABLE I

RULE-BASE DESCRIBING THE SYMPTOM-FAULT RELATION AND THE ASSOCIATED CORRECTIVE ACTION

Behavior module	**Fault Node**	**Symptom**	**Generate-and-test strategy**	**Corrective action**
Path planning	Goal_not_reached	Time out in goto goal behavior	Perform path planner test	Reset path planner and re-attempt goto_goal behavior
	Invalid_positions	Invalid position information	Check starting and goal positions	Re-attempt path planning
	Bad_starting_position	Incorrect starting position	Check starting position information from initial config file	Reset starting position and re-attempt path_planning
	Bad_goal_position	Incorrect goal position	Check goal position from initial config file	Reset goal position and re-attempt path_planning
	Localization_errors	Inconsistent pose information	Perform laser test, re-calibrate laser	Check if simple robot is close enough to goal; if so, change simple robot state to sensor detection and inform human to replace laser
	Translation_errors	Inconsistent pose information	Request human user to check the translation algorithm for logical inconsistencies	Reset algorithm and re-attempt behavior
	Invalid_map	Map not found	Perform map test	Request human operator to replace map
	Bad_Initialization	Incorrect starting position	Test position values	Reset starting pose values
	Faulty_sensor	Bad sensor readings	Perform sensor test	Alert human to replace sensor
Teleoperation	Camera error	Image not found	Perform camera tests	If fault persists, leave simple robot(s) in wait state, send camera failure feedback to human operator and return home
	Lost_follower	Missing robot	Establish communications with simple robot and instruct it to perform motor tests	Check if simple robot is close enough to goal; if so, change simple robot state to sensor detection, and return home
	Motor_problems	Motor test failed	Perform simple robot status check	Change simple robot state to sensor detection and proceed

TABLE II

OVERALL SYSTEM SUCCESS RATE, AVERAGED OVER 45 TRIALS

Module	**Subsystem Success Rate**	**Experimental Success Rate**
Localization	0.83	
Path Planning	0.99 (est.)	
Navigation	0.95 (est.)	
Follow Helper	0.78 (est.)	
Marker Detection	0.98	
Communication	0.91	
Complete System	0.54 (est.)	0.67 (2-robot depl.) 0.48 (1-robot depl.) 0.59 (combined over all trials)
Helper Robot returning home		0.91 (over all trials)

only around 59%. In order to make each component highly reliable, each component must be able to identify, recognize and recover from errors.

Most importantly, however, was the success rate of the helper robots in making it back to the home base autonomously. Despite the limited number of failure states identified and implemented in this causal model, the success rate of the helper robots making it back home autonomously in these rigorous experiments was 91% (over 45 trials). Note that this success rate is higher than the predicted success rate for the multiplied probabilities of localization, path planning, and navigation (which would be 78%). The reason for the improved leader performance is primarily because the localization errors are transient; since we are using a Monte Carlo localization algorithm that can account for transient errors, the single leader robot is able to find its way back home even when occasional localization errors occur. However, when the leader robot is in the process of deploying simple robots, even these transient localization errors can cause problems, due to the resultant erratic motions of the leader robot. Thus, even though the leader robot does not often end up lost due to localization errors (i.e., it does better than the 83% reported in Table II), the following simple robots can get confused due to sharp, erratic turns, thus leading to team-level deployment errors due to lost simple robots. In any case, because the leader robots are able to make it back home, and because in this application the simple robots are plentiful, the overall application success was relatively high. We believe these experiments validate the utility of the causal model for practical multi-robot applications.

IV. LeaF: An Adaptive Causal Model Approach

The analysis of the post-experimental data highlighted certain unexpected faults that the system was not equipped to handle, such as certain types of communication failures and navigation challenges due to the shape of the floor boarding (causing some of the simple robots to sometimes get hung when moving close to certain walls). While such failures could be added manually to the causal model, this is potentially an infinitely iterative process due to the uncertain interaction dynamics of the environment and the robot team. Our experience convinces us that it is unlikely that the human designer will anticipate and incorporate every possible fault that the multi-robot system may encounter. Thus, the multi-robot team can achieve a much higher level of robustness if it is able to autonomously adapt its causal model based on experience.

Our LeaF adaptive causal model approach uses a case-based learning algorithm to adapt and categorize a new fault and add it to the causal model for future use. Figure 3 shows the LeaF architecture, which combines typical robot control processes with modules for adaptive fault diagnosis and recovery. In this architecture, the *Behavior Control Module* (BCM) contains application-dependent behaviors that enable the robot to accomplish its objectives within the given application. The *Fault-tolerance Module* (FtM) is responsible for fault diagnosis and recovery, and learning to adapt the causal model from experience. Due to space limitations, we focus here on the adaptive causal model approach incorporated in LeaF.

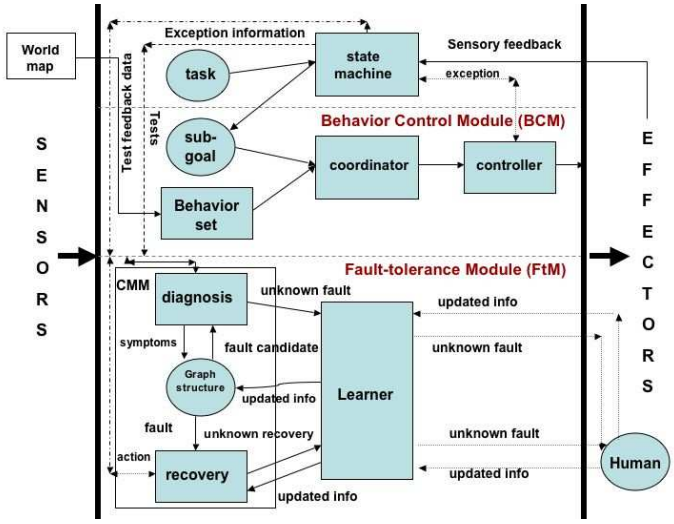

Fig. 3. Architectural details of LeaF.

A. Fault-tolerance Module (FtM)

The fault-tolerance module consists of two main blocks – CMM and the Learner. The CMM block contains the *Diagnosis* and *Recovery* sub-modules, and are based on an extended version of the SFX-EH architecture [12] to diagnose the failure. Using SFX-EH, the robot generates tests about the possible cause for failure, analyzes the resulting data and attempts to diagnose the failure by comparing with the nodes of the causal model. Since all nodes of the causal model have an action associated with them, when the cause for a fault is determined, the corresponding corrective action in the causal model is selected and executed.

Of course, when unknown faults occur, the predefined causal model is inadequate for fault diagnosis and recovery. Using case-based reasoning, a new fault can be diagnosed by identifying its similarity to one or several previously classified faults stored in the causal model and by adapting known solutions, rather than working out a new solution from scratch. In our approach, each capable robot maintains a copy of the causal model (perhaps in collaboration with its teammates) and the LeaF learner, and attempts to identify the new fault. In the event the LeaF learner cannot classify the fault, the robot communicates the relevant information to a human operator for assistance in updating the causal model.

The identification of similarities between faults is achieved using Armengol and Plaza's technique called Lazy Induction of Descriptions (LID) [1]. LID builds a symbolic similarity description, *similitude* [2], for the fault, finding the best match to one or more nodes in the causal model. Unlike other distance-based approaches for CBR, similitude measures the relevance between faults using relations among entities, rather than using arbitrary distance metrics. LID identifies relevance by only selecting the nodes with similar characteristics to that of the encountered fault. This approach provides a better reasoning for fault diagnosis and reduces overall system complexity and the time spent in fault diagnosis.

1) Overview of LID: We first describe the formal framework for LID, summarizing [1] and adapting it to our multi-robot domain. We then apply this technique to our multi-robot team application, giving an example of how LID enables us to create an adaptive causal model.

For our domain, we define the problem space, P, as the set of all possible fault descriptions that can be encountered by the system over the duration of the experiment, and the relationships between these faults. P includes the faults that are identified by the designer before experimentation as well as those that are encountered during testing and experimentation. Specifically, P=$<V, E>$, where the vertices $V = \{f_1, f_2, f_3, f_4, ..., f_m\}$ represent faults and the edges $E = \{\{f_1, f_2\}, \{f_1, f_3\}, \{f_2, f_3\}, ..., \{f_i, f_j\}\}$ represent relationships between the faults, according to the causal model. Typically, fault relationships indicate increasing (or decreasing) order of specificity in the fault description. Also, a path $\pi(f_i, f_j)$ is defined as the sequence of inter-related nodes going from node f_i to node f_j. In most applications, some faults occur more commonly than others. To prioritize faults, we associate a weight, called the probability of occurrence $\sigma(x, y)$, with every edge of the CMM, representing the probability of fault x being best explained by specific fault description y. The cumulative outgoing probability from any node sums to 1. All the edges in the CMM are either initialized with equal probabilities of occurrence or are initialized based on data from a training phase. These probabilities are continually updated based on experience at

run time. Finally, we define a path weight, $W(\pi(f_i, f_j))$, by multiplying the individual probability of occurrence for every edge involved in traversing from starting node f_i to final node f_j. Interpreting Table I and Figure 2 in this context, we can define the vertex set for our multi-robot test application as V = {*Goal_not_reached, Invalid_positions, Bad_starting_position, Bad_goal_position, Localization_errors, Translation_errors, Invalid_map, Bad_initialization, Faulty_sensor, Camera_error, Lost_follower, Motor_problems, Unexpected_Environmental-behavior*} .

The solution set, S, is the set of all corrective actions for the corresponding fault from the problem set. Specifically, $S = \{s_1, s_2, s_3, s_4, ..., s_m\}$, where s_j is the corrective action associated with fault f_j. For a system consisting of n robots, the sample case base, C, is defined as $C = \{C_1, C_2, C_3, ...C_n\}$, where $C_j = \{(P_j, S_j)\}$, and P_j is the problem space defined for robot R_j having a solution set of S_j. For the LeaF system, P_j represents all nodes and edges of the CMM for robot R_j and S_j constitutes the corrective action associated with the nodes in the CMM for that robot.

The collection of terms (ψ) define the characteristics of the fault, f_i, sorted in terms of their relevance. We define $\psi(f) = g[\psi_1, \psi_2, ...]$, where $\psi_1, \psi_2, ...$ are the fault characteristics and g is the application-specific sort function defined by the designer. To classify the encountered fault, f_i, the adapted LID builds a "similarity space", called D_f. The similarity space is formed by grouping all vertices from the causal model containing only the most relevant terms of the encountered fault, f_i (where "relevant" is determined via a user-defined distance metric). We define D_f for fault f_i as $D_f = \{f_k \mid f_k \in V; \psi(f_i) \cap \psi(f_k) \neq 0\}$.

For LeaF, we currently restrict the characteristic defining the problem to a single term that represents the fault symptom – i.e., $\psi = \{problemsymptom\}$. Hence, for LeaF, the similarity space is simply the set of all vertices in the causal model that have the same symptom as the encountered fault. To illustrate the concept of similitude, consider a scenario in which a helper robot encounters a fault, f_i, while teleoperating a simple robot. This fault has the symptom, ψ= {"inconsistent pose info"}. Applying the above criteria to the Table I, the generated similitude term, D_f, is given by {*Localization_errors*, *Translation_errors*}.

In order for the robots to diagnose and recover from a fault, it is not enough to identify a set of similar cases. The solution should be similar to the problem in aspects that are important to the current task. In order to identify the relevance of the elements in the similitude set, LID uses a top-down strategy that determines which of the characteristics present in the fault are the most discriminatory. This is then used to build the similarity space, D_f. The process of using only the most discriminatory characteristics eliminates unrelated cases and narrows down the similarity space to contain only the most relevant cases to the encountered fault. Unlike the original method, which builds a dynamic heuristic, the lack of prior knowledge of the distinct similarities between faults forces us to predetermine the most important characteristics for our causal model. As we mentioned above, currently for LeaF the characteristic of most significance is the symptom associated with each fault. (Future work involves analyzing the faults in the system to possibly design a more dynamic heuristic.)

The process of identifying relevant characteristics continues until the cases are pruned to just one case that provides the most specific reasoning for the encountered fault. Algorithmically, this is determined by checking for one of two termination criteria, as follows: (1) if there exists at least one leaf node (f_k) in the similarity space, $f_k \in D_f$, or (2) if adding further characteristics of f_i into D_f does not produce a leaf node f_k.

When the termination occurs due to the second criterion it means that system is unable to build a structural similarity between the fault f_i and any of the leaf nodes of the causal model. In this case, assuming problem f_i belongs to a subset of V_j, then $\forall i \in D_f, \exists$ no $i \in D_f$ such that i is a leaf node. Then, in order to prune the candidate cases down to just one most relevant case, we apply a diagnosis algorithm to D_f as sketched below.

Algorithm 1 Sketch of the Diagnosis algorithm for LID (adapted from [1])

1: For fault f_i, given D_f and V, do the following:
2: **while** $D_f \neq \emptyset$ **do**
3: Find edge (f_i,f_q) with highest $\sigma(f_i, f_q)$
4: Replace fault f_i with selected node f_q.
5: Build new D_f based on previously defined criteria for similarity space.
6: **if** $\exists$ a leaf node $f_k \in D_f$ **then**
7: $D_{final} = \{f_k\}$ and exit.
8: **else**
9: Calculate and store path weight to f_k and remove node f_q from D_f; $D_f = D_f - \{f_q\}$.
10: **end if**
11: **end while**
12: Retrieve f_q as the diagnosis, which is node with highest path weight; i.e., $D_{final} = \{f_q : max(W(\pi(f_q, f_k)))\}$.
13: **if** $(D_{final} = f_{root})$ **then**
14: Refer to human operator for classification. (i.e., the fault could not be classified.)
15: **end if**

The goal of the diagnosis algorithm is to identify the leaf node that best diagnoses the encountered problem, f_i. The first step in the algorithm involves selecting the edge with the highest probability of occurrence for the observed symptom. Once the edge is identified, the next step involves replacing the current problem, f_i, with the node, f_q of set D_f. Once the node f_q is selected, to ensure that the diagnosis is proceeding in the right direction, the generate-and-test strategy associated with f_q is executed (e.g., refer to Table I) to verify that the current vertex is descriptive of the actual problem being experienced. Assuming so, then we recursively reapply the LID strategy to the problem. (Note that this process of generate-and-test and the recursive application of LID is part of the complex Step 5 of Algorithm 1; due to space limitations, we have greatly sim-

plified this step in the algorithm sketch.) The recursive process is repeated until either a leaf node is eventually diagnosed or until all structurally similar nodes in the CMM are explored. If a leaf node is identified, it is selected as the final diagnosis and its associated recovery action is executed. In the event that the recovery action does not solve the encountered fault, the strategy resumes the diagnosis process by retracing a level of abstraction from the last known diagnosis and re-applying Step 5. If the system is unable to diagnose a leaf node, then the node from the similarity space having the next highest path weight is selected as the next diagnosis node. If the system is unable to build a similarity space for the encountered fault, the robot transfers control to a human for further analysis.

2) Updating the case base: When a fault is encountered in the system, two stages are involved in the recovery process. First, the system diagnoses the fault, and second, useful information is extracted from the fault for future use. Using the LID technique for identifying similitude enables us to consider only the most relevant cases to the encountered fault. This enables us to quickly and accurately narrow down the nodes responsible for the fault. To incorporate information on the current fault, we update our original case set based on new relevant information extracted from fault f_i. Let $\delta = D_{final}$. The updated vertex set is represented as $V' = \{f_1, f_2, f_3..., f_n\} \cup \delta$ and the edge set is updated to hold any new relation between the newly added node and existing nodes.

Once the appropriate action has been executed and the case base updated, the robot coordinates with other team members regarding the newly diagnosed fault. Information is exchanged as needed to ensure that all team members maintain current causal model information.

Compared to other traditional methods for case-based learning, LID is desirable for the multi-robot environment because of its ability to identify structural similarities between cases. For multi-robot fault tolerance, combining the causal model with LID provides us with a reliable and effective method for fault identification and classification. One of the disadvantages of using the causal model on its own is that every possible relation between faults needs to be explicitly defined. By adding LID to a base causal model, we can recombine, redirect and even add new nodes or faults to the causal model. The flexibility of the model means that we can classify previously unknown faults within the scope of the leaf nodes.

V. Experimentation and Analysis

To test the capability of the adapted LID algorithm, we apply it to the autonomous teleoperation task of the "locate and protect" multi-robot application described in Section III, using physical robot team members. Recall that the goal of the leader robot is to autonomously teleoperate the simple robots in an indoor environment in the presence of faults. The robots make use of an initial CMM (see Figure 2) that is provided to the system.

The first set of experiments is aimed at showing the diagnosis time advantage of the adaptive causal model approach of LeaF when compared to the non-adaptive CMM approach. In the experiments using LeaF, the probability values associated with fault likelihoods of occurrence were initialized to be equal, but were updated with experience; subsequent trials made use of the updated values. Table III compares the time taken for fault diagnosis for these two distinct systems, on physical robot teams. Note that these times represent the total time spent by the robot in fault diagnosis, from the time the exception was raised to the time the solution is identified, for the given experiment. The comparison is detailed based on system performance for two distinct faults: *faulty laser* and *camera failure due to environmental changes*. From Figure 2, we can see that *faulty laser* is a known type of fault (i.e., a special case of *faulty sensor*), whereas *camera failure due to environmental changes* has not been previously encountered. To experimentally create these errors, either the laser was covered (causing the *faulty laser* error) or the lights were turned off (causing the *camera failure due to environmental changes* error). From the results we see that the CMM system performs comparably with the LeaF system in the case of the known fault. However, we note that the time for the LeaF diagnosis decreases from trial to trial, due to the increasing probability of occurrence of this particular fault in the system. On the other hand, the disparity in system performances is clearly indicated in the event of a previously unknown fault. The CMM system is unable to provide any useful diagnosis even after a long time interval[3], whereas the LeaF system shows progressive improvement over time. Once LeaF identifies the new error, it adds a new edge in the CMM, thus reducing the time taken for subsequent searches.

TABLE III

Extent of learning exhibited by the system measured in terms of the time taken to diagnose faults

Fault name	**Trial number**	**Time (secs) for CMM**	**Time (secs) for LeaF**
faulty laser	1	44	45
	2	44	41
	3	45	40
	4	44	37
	5	45	35
camera failure due to environmental changes	1	360+	65
	2	360+	32
	3	360+	17

Table IV shows the adaptation of the probability values in the second fault case (i.e., camera failure due to environmental changes when the lights are turned off), along with the time taken to diagnose the fault. The data shows a rapid change in the probability values corresponding to the increased

[3]To save battery life, we halted the trials after 360 seconds of diagnosis.

TABLE IV

PROBABILITY ADAPTATION FOR ENVIRONMENTAL STATE CHANGE (LIGHT BEING SWITCHED OFF)

Fault name	**Trial number**	**No. of faults**	**Starting probability (%)**	**Ending probability (%)**	**Time taken to diagnose fault (secs)**
camera failure due to environmental changes	1	5	0	33	65
	2	5	33	47	24
	3	6	47	63	17
	4	4	63	76	17
	5	6	76	84	17
	6	2	63	88	17

likelihood of this particular failure occurring. While this may be desirable in the case of permanent environmental state changes, it also has the potentially negative effect of causing the team to forget the fault probability profile for the previous environmental state (i.e., when the lights are on). Our current work is extending LeaF to handle these distinct operational or environmental states when the fault profiles are different.

Another limitation of the current approach is its inability to solve for undiagnosed faults that cannot be mapped to any of the existing ones under the causal model, i.e., the encountered case lies outside the existing solution set. Currently, this situation is handled by transferring all available information to a human for further evaluation. Other autonomous methods are currently being investigated to overcome this limitation.

VI. CONCLUSIONS

In this paper, we have shown how the causal model approach can be implemented in a large-scale multi-robot team and successfully enable the team to diagnose and recover from pre-defined errors. We presented results from extensive physical robot implementations that illustrated the effectiveness of this approach. To the best of our knowledge, this is the first implementation of a casual model for fault diagnosis and recovery in a complex multi-robot application. We then argued that a static causal model approach is insufficient for guaranteeing robust solutions, since it is practically impossible to pre-define all possible error modes in a complex multi-robot application. We presented our approach for adapting causal models based on experience, called LeaF, which incorporates extensions of the causal model method of [6], the SFX-EH approach of [12] for diagnosing existing faults, and the LID work of [1] to create an approach that adapts causal models when new faults are encountered. We presented experimental data showing the benefit of this new approach for new robot fault conditions.

In other related work [10], we have developed application-independent metrics to measure fault-tolerance for multi-robot teams. The main focus is to capture the effect of intelligence, reasoning, and/or learning on the effective fault-tolerance of the system, rather than relying purely on measures of redundancy. Work is underway to further validate the LeaF approach in extensive experimentation both in simulation and on physical robots, using these new metrics we have developed.

REFERENCES

[1] E. Armengol and E. Plaza. Lazy induction of descriptions for relational case-based learning. In P. Flach L. De Raedt, editor, *Machine Learning: EMCL 2001*, Lecture Notes in Artificial Intelligence, pages 13–24. Springer-Verlag, 2001.

[2] E. Armengol and E. Plaza. Similarity assessment for relational CBR. In *Case-based reasoning research and development: ICCBR 2001*, Lecture notes in Artificial Intelligence, pages 44–58. Springer-Verlag, 2001.

[3] E. M. Atkins, E. H. Durfee, and K. G. Shin. Detecting and reacting to unplanned-for world states. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI)*, pages 571–576, 1997.

[4] J. Carlson and R. R. Murphy. How UGVs physically fail in the field. *IEEE Transactions on Robotics*, 21(3):423–437, June 2005.

[5] J. Cavallaro and I. Walker. A survey of NASA and military standards on fault tolerance and reliability applied to robotics. In *Proceedings of AIAA/NASA Conference on Intelligent Robots in Field, Factory, Service, and Space*, pages 282–286, March 1994.

[6] B. Horling, V. Lesser, R. Vincent, A. Bazzan, and P. Xuan. Diagnosis as an integral part of multi-agent adaptability. In *Proceedings of DARPA Information Survivability Conference and Exposition*, pages 211–219, 2000.

[7] A. Howard, L. E. Parker, and G. S. Sukhatme. Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment, and detection. *International Journal of Robotics Research*, 25:431–447, 2006.

[8] E. Hudlická and V. R. Lesser. Modeling and diagnosing problem-solving system behavior. *IEEE Transactions on Systems, Man, and Cybernetics*, 17:407–419, 1987.

[9] G. A. Kaminka and M. Tambe. What is wrong with us? Improving robustness through social diagnosis. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI)*, pages 97–104, 1998.

[10] B. Kannan and L. E. Parker. Fault-tolerance based metrics for evaluating system performance in multi-robot teams. In *Proceedings of Performance Metrics for Intelligent Systems Workshop*, 2006.

[11] M. Long, R. R. Murphy, and L. E. Parker. Distributed multi-agent diagnosis and recovery from sensor failures. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2506–2513, 2003.

[12] R. R. Murphy and D. Hershberger. Handling sensing failures in autonomous mobile robots. *The International Journal of Robotics Research*, 18:382–400, 1999.

[13] K. Toyoma and G. D. Hager. If at first you don't succeed ... In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI)*, pages 3–9, 1997.