Multi-Layered Model Based Diagnosis in Robots

Eliahu Khalastchi, Meir Kalech and Lior Rokach

Ben-Gurion University of the Negev, Be'er-Sheva, Israel, 84105, Israel
{eli.kh81,kalechm, liorrk}@gmail.com

ABSTRACT

The problem of fault diagnosis in the domains of robotics and autonomous systems are unique and interesting. A hidden internal fault affects not only other hardware components, as in any other machine, but also the different layers of abstraction and control in the sense-think-act cycle that the robot carries. We propose to implement a model-based diagnosis approach by utilizing the robot's control-architecture as a model to detect faults and to associate high-level abstractions with low-level hardware components. Furthermore, other approaches use behavioral models by representing the models with analytical equations, making it impractical to construct for very complex robots. We propose a structural model by representing only the dependencies in the model, which is significantly easier to construct. We demonstrate the feasibility of our approach by diagnosing high fidelity faults in a simulated Unmanned Aerial Vehicle (UAV) and a laboratory robot.

1 INTRODUCTION

Robots are susceptible to different types of faults, hardware wear and tear, false sensing, software bugs, environmental causes etc. Such failures can cause mission failure or even endanger the robot or its surrounding environment (for example a UAV crash). When a fault is detected it is important to know which internal components are involved (if any). This diagnostic information can be used for recovery or for decision-making purpose, for example, switching to undamaged redundant systems or re-planning.

The domain of robotics offers unique aspects to the problem of diagnosis. Robots are more than simply reactive machines. They sense the world using a different variety of external and internal sensors (e.g. camera, laser, sonar, infrared, accelerometer, etc.). They process these readings into abstract representations called beliefs (e.g. image processing to calculate the angle of the road related to the robot's position). Then a robot can apply a high level decision-making process (e.g. by using decision tree, automaton, planner, human operator) that according to the state of the beliefs the robot chooses a set of actions which will be referred as a task. The task's actions activate actuators (e.g. wheels, arms) which in turn, influence the world. These changes are sensed again making it the sense-think-act cycle (Brooks, 1986). Thus, when a task is executed, beliefs may be changed.

During task execution, faults might occur. These faults are expressed not only in the hardware components, but also in the sense-think-act control cycle. If a fault influences sensors then the beliefs might report incorrect readings and thereby the goal might be not achieved. If a fault influences an actuator then the correct process of the action is not guaranteed; thereby disrupting the execution of the task.

For example, consider the robot has a belief of its distance to a target, and during a task that aims to reach the target, we observe that the belief is unchanged. This observation can be explained by a fault to the wheels (actuators) or a fault to the wheels' power supplier (internal component that the actuators are dependent on). This observation can also be explained by a fault in the infrared sensors that are used to calculate the belief. Finally, the observation can be explained by a bug in the belief calculation.

In this paper, we model the different layers of the robot: the low level of the components, actuators and sensors and the high level of the beliefs, tasks and actions. In addition, we model the connections between the layers. Based on this model, we describe a fault detection and diagnosis techniques. As opposed to previous model-based approaches, no construction is needed for this model; it is always given to the robot's programmer. Previous model-based approaches in Robotics usually model only one layer of control (Isermann, 2005). We propose a multi-layer model which reflects the associations between the high level beliefs, tasks and actions to the low-level sensors and actuator hardware components. In addition, to deepen the diagnosis we suggest the use of a dependency model, which is very easy to construct compared to models that rely on analytical equations.

We demonstrate the feasibility of our approach by diagnosing high fidelity faults in a simulated Unmanned Aerial Vehicle (UAV) (FlightGear, website) and a laboratory robot (Robotican, website). We show that faults that are detected on high-level beliefs can implicate relevant low-level hardware components such as sensors and actuators.

The contributions of this paper are the follows: 1. Define a model-based approach for fault-detection and diagnosis. 2. Present a multi-layer model of the robot which enables to diagnose hardware faults by monitoring the beliefs of the robot 3. Present different types of faults related to the different layers of the robot. 4. Demonstrate several types of faults in simulated UAV and a laboratory robot.

2 RELATED WORK

Steinbauer *et al.* conducted a survey on the nature of faults of autonomous robot systems (Steinbauer, website). The survey participants are developers competing in different leagues of the Robocup competition (Robotcup, website). The reported faults were categorized as hardware, software, algorithmic and interaction related faults. We argue that these types of faults are the different locations in the sense-think-act cycle of the robot in which faults might occur. A fault in each location might have an impact on the other elements in the sense-think-act cycle. The survey concluded that hardware faults such as sensors, actuators and platform related faults have a high negative impact on mission success. In this paper we focus on diagnosing these hardware related faults.

MBD methods differ in the fault detection process, the model they use and the diagnosis process. Fault detection methods are usually analytical, data-driven or knowledge-based approaches (for examples refer to (Isermann, 2005; Steinbauer, 2005; Gspandl, 2011)). Analytical approaches use mathematical models to compare expected outputs with observed outputs and derive a residual that is used to determine whether or not a fault has occurred. However, expressing all the behavioral laws of every component in mathematical equations is a very hard task (Isermann, 2005; Varun, 2009).

Data-driven approaches are model-free statistical methods. These methods face the challenge of dimension reduction and a dependency in the existence of quality information that can be extracted from the data (Isermann, 2005; Varun, 2009). We propose to detect hardware related faults by monitoring the robot's beliefs, which is quality information already calculated, and simply compare them to their intended goals. Thus, not having to model each law mathematically nor having to know all faults a priori.

Steinbauer *et al.* (Steinbauer, 2010) recently suggested the diagnosis of robot's beliefs and emphasized its importance for coping with the real-world dynamic environment of the robot. Beliefs are diagnosed for their binary truthfulness. We monitor beliefs and how they differ from their intended goals.

Brandstötter *et al.* (Brandstötter, 2007) show a model-based diagnosis and reconfiguration framework which allows an autonomous robot to detect and compensate faults in a robot's drive. Their generic model should be implemented specifically to a given robot and take into account the dynamics and the kinematics of the drive. We argue that these types of faults will be expressed in the failure of beliefs to reach their intended goals and thus can be detected with an alternative implementation. Moreover, we show how to view the generic multilayered control architecture of the robot as a generic model that is able to associate faults detected on beliefs with low-level hardware components.

3 ROBOT'S CONTROL-ARCHITECTURE AS A MODEL FOR DIAGNOSIS

In this section, we describe the general multi layered architecture of behavior based (Arkin, 1998) control for a robot and the relation between low-level hardware components and high-level abstractions. We then describe how this architecture can be viewed as a model which can be used for fault detection and diagnosis.

3.1 General robot control-architecture

A robot control-architecture is usually built of layers (see Fig.1). The lowest layer is the robot's hardware made up of actuators and sensors. Other internal components such as power suppliers also exist but are not a part of the control architecture. The actuators and sensors provide an interface of electrical signals for software drivers to perform sensor reading and actuators activation. These drivers act as a hardware abstraction were each instance of a driver directly relates to a sensor or an actuator. The next layer of abstraction is the robot's API (Application Programming Interface) which contains a set of high-level commands. These commands provide a comfortable way to perform composite driver calls.

For example, a robot with a differential-drive would have a motor for its left wheel and another motor for its right wheel. These motors are part of the robot's actuators. A motor's driver would have two instances, left and right. A high-level command to move at a given speed and angle is defined in the robot's API and is implemented by setting different speed to each wheel. This high-level command invokes the drivers

which, in turn, invoke the actuators – the motors that cause the wheel spin and the robot to move.

The robot's movement affects the state of the robot and the state of the world. These states are the beliefs of the robot. The beliefs are constructed by the sensors readings of the environment.

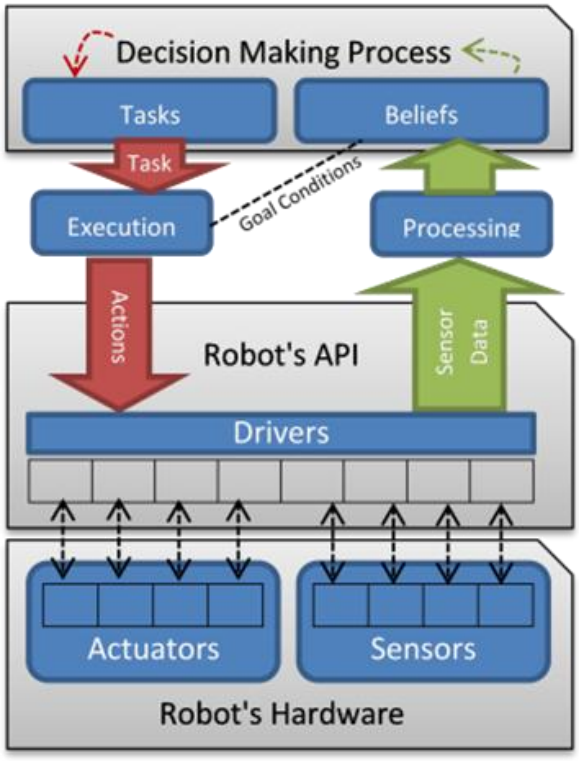

Figure 1: The Generic Control Architecture for Robots

The beliefs and a goal state i.e. a goal value per each belief, are the input of a decision-making process (the highest level). This process can be viewed as a planner, which issues tasks that supposed to achieve the goal state. A task is an algorithmic collection of actions from the robot's API which affect the robot's actuators. These actions are chosen by the planner to affect the world in such a way that the goal will be achieved. Each task is issued with a sub-goal state which is applied on beliefs. Upon achieving a task sub-goal state a new task is issued to fulfill the sub-goal. This process repeats until the robot's goal is achieved.

For example, assume a robot's goal is collecting 10 objects. The decision-making process maintains the belief of the number of objects the robot collected. While this number is less than 10, the decision-making process issues a task to locate an object which includes moving in a search pattern until an object is insight (task's sub-goal state). This belief is constructed by processing the readings of the camera sensor. Next, the decision-making process issues a task to approach the object. This task includes movement towards the object while keeping it insight until the belief of the distance to the object is 30cm (task's sub-goal state). Then, a task to pick up the object is issued etc.

Consider that during the task to approach the target object the distance belief is unchanged. If everything was healthy then it should have decreased as the task performs. Therefore, this event might express a fault somewhere in the sense-think-act cycle of the robot. Possible explanations can be relevant actuators - wheels that have not turned, relevant sensors – an infrared sensor that is stuck, software faults such as the movement action was not called, or the distance belief calculation has crashed.

We aim, in this paper, to diagnose faults in hardware components such as sensors and actuators. The beliefs that supposed to be influenced by the task and their goal state are provided by the control model. Therefore, fault detection with respect to the task's goal state can be made. The control model provides us with the knowledge of which sensors are used to construct the beliefs. Therefore, we can relate relevant sensors to the beliefs that are suspected of faulty behavior. The control model also provides us with the knowledge of which actions were called during the task and which actuators are invoked by each action. Therefore, we can point the relevant actuators as suspects as well.

3.2 The Control-Model

Let $S^t = \{s_1^t, \dots, s_n^t\}, s_i^t \in \mathbb{R}$ be the set containing the sensors values at time t. Let $B = \{b_1, \dots, b_p\}$ be the set of the robot's beliefs. A belief's value at time t is derived by a subset of the sensors.

Definition 1: [belief construction] *Given a set of sensors $S^{t'} \subseteq S^t$, the function $v_i^t(S^{t'}) = b_i^t$ calculates the value of the belief b_i at time t .We denote v_i^t as abbreviation and $V^t = \{v_1^t, \dots, v_p^t\}$ as the set of the beliefs' values at time t.*

To conclude the sensor set that influence a given belief, we define the following mapping:

Definition 2: [belief-sensors mapping] *Given belief b_i the function $\beta(b_i) = S'$ returns the set of sensors that are used to construct the belief b_i.*

Let $A = \{\alpha_1, \dots, \alpha_m\}$ be the set of actuators and let $ACT = \{a_1, \dots, a_k\}$ be the set of the robot's high-level actions. An action causes the activity of a subset of actuators.

Definition 3: [action-actuators mapping] *The function $\gamma(a_i) = A'$ returns the set of actuators $A' \subseteq A$ that are invoked by action a_i.*

A task is the main building block of the control model of the robot. A task dictates the activity of a set of actions based on the current beliefs to fulfill a goal belief.

Definition 4: [task] *The task* $T =< B_e, V_g, ACT', \phi: (V_e^t, ACT') \rightarrow V_g >$ *is a tuple, where:*
$B_e \subseteq B$ *is a set of beliefs,*
V_g *is a set of goal values of the beliefs in* B_e*,*
$ACT' \subseteq ACT$ *is a set of actions invoked by the task,*
$\phi: V_e^t, ACT' \rightarrow V_g$ *is a transition function which obtains the values of the current beliefs* B_e *and by invoking the actions in* ACT' *it transitions to the goal values* V_g *of the beliefs.*

For example, consider a UAV's takeoff task. This task includes accelerating on the runway, and after gaining enough speed, elevating the aircraft and maintaining attitude position until the desired altitude is achieved. $\mathrm{B_e} = \{\mathrm{speed, attitude, altitude}\}, \mathrm{V_g} = \{\mathrm{64kias, leveled, 1500feet}\}$,
$\mathrm{ACT'} = \{\mathrm{accelrate, elevate, maintain\ attitude}\}$. The takeoff task uses the actions in an algorithmic fashion:

1. accelerate
2. wait until speed > 64kias
3. elevate
4. while altitude < 1500feet
5. maintain attitude(leveled)

In addition, the control model provides the relevant actuators such as $\gamma(accelerate) = \{breaks, throttle\}$ or $\gamma(maintain\ attitude) = \{aileron, elevation, rudder, trim\}$, and relevant sensors such as $\beta(speed) = \{airspeed\ indicator\}$ or $\beta(altitude) = \{altimeter\}$.

The control model can be constructed automatically. Each sensor, actuator and action registers upon invoking, then all associations can be registered when calculating each belief or calling an action from a task.

3.3 Using the Control-Model for Diagnosis

The focus of this paper is about diagnosis, but first let us describe briefly our fault detection method. As explained in subsection 3.1, the decision-making process of the robot chooses an executing task to achieve a goal state. We expect the values of the associated beliefs with the current task to be affected by the actions taken through the task and gradually get closer to their goal condition values.

For example, as the robot moves towards a target object, the infrared distance sensor returns decreasing values which update the belief of object's distance to be closer and closer to the goal value of 30cm.

The values of the task's beliefs will get closer to their goal values only upon nominal behavior. Therefore, if the values of the task's beliefs B_e are not getting closer to their goal condition values V_g as the robot's operates then we report a fault. For this paper we used linear regression to measure the trend of the beliefs progress and compared it to the goal beliefs. However, there are two aspects which should be investigated: (1) the smoothing function of the beliefs progress and (2) the distance measures between the current beliefs and the goal beliefs. Since the focus of this paper is on diagnosis we leave them for future work.

We define $B_f \subseteq B_e$ as the set of beliefs that triggered the fault detection. When a fault is detected it could be the result of the following general cases: (1) hardware faults (2) software faults (3) environmental aspects. For example, consider the robot is trying to approach its target object. If the belief of the distance to the object is not getting smaller over time then it might be the result of hardware faults such as:

1. Faulty actuators – the motors gears are worn and therefore one or more wheels are not spinning correctly and the robot cannot get closer to the object.
2. Faulty internal component – a power supplier fails to deliver power to the motors of the wheels
3. Faulty sensors – the infrared distance sensor is stuck on its last reading, making the belief to appear unchanged even though the robot is approaching towards its target object.

It can also be the result of software faults. For example, the process that calculates the belief has crashed, therefore, the believed distance remains unchanged even though the robot is progressing and sensing correctly.

It can also be the result of environmental aspects. For example, the robot is stuck in front of a step. The wheels are spinning in place, and the distance is unchanged. However, nothing is wrong with the robot or its control software.

In this paper, we focus on hardware faults and how they are related to the control layer of the robot such as beliefs and tasks. The control-model provides us with the means to diagnose the relevant sensors and actuators. Since the fault is detected with respect to task T, we logically assume that this is due to a fault in hardware components that are related to the currently executing task T.

Definition 5: [suspected hardware component] *a suspected hardware component is either: a sensor, an actuator or an internal component which sensors or actuators are dependent on. The hardware component is suspected if its faultiness could explain the observed detected fault.*

The diagnosis procedure marks the relevant sensors and actuators as suspects. The suspected actuators are the set of the activated actuators by the actions that are called by the executing task: $\{\alpha_j | \alpha_j \in \gamma(a_i)\ \forall a_i \in ACT'\}$. We also mark the sensor set used to calculate a beliefs that caused the fault detection (i.e. did not get closer to its goal value): $\{s_j | s_j \in \beta(b_i)\ \forall b_i \in B_f\}$.

The control-model provides the means to reduce the diagnosis search space (i.e. all actuators and sensors) to the hardware components that are associated with the high level execution. However, this model does not cover the possibility that an internal component that is not a sensor or an actuator (e.g. a power supplier) has failed.

For that reason we also use a structural model that we describe in the following section. Once the list of suspects is provided the structural model is used to further diagnose whether additional internal hardware components should be suspected.

4 INTERNAL HARDWARE COMPONENTS DIAGNOSIS

In this section we describe the structural model and how it is used to diagnose faults in internal hardware components.

4.1 The structural-Model

In a robot's hardware structure there are hardware components that are neither actuators nor sensors, and thus not a part of the control-architecture. But these components play an important role in the healthy operation of the robot. For example, if a power supplier fails to deliver power to an actuator then the actuator cannot function (even though the actuator is healthy).

When we receive a list of suspected sensors and actuators that were extracted from the control-model we need to check whether their suspected faulty behavior is the result of their dependency to a faulty internal component. For that matter we use a structural model.

Let $C = \{c_1, \ldots, c_h\}$ be the set of hardware components that are not actuators or sensors $C \cap A = \emptyset, C \cap S = \emptyset$. The structural model defines the dependencies between the internal components and the sensors and actuators.

Definition 5 [structural model] *Given the set of sensors S and the set of actuators A and the set of internal components C, M_s is a set of tuples of the form $< c_i, D_i >$ where $D_i \subseteq A \cup S$. Given the healthy predicate $h(x)$ denotes the health of $x, \forall\, d_j \in D$: $\neg h(c_i) \rightarrow \neg h(d_j)$.*

If a component c_i is faulty then all of its dependent sensors and actuators ($s_j, \alpha_i \in D$) will report faulty data. However, if s_j or α_i are faulty it does not imply that c_i is faulty; s_j, α_i can be faulty themselves. We define mapping functions to conclude the actuators and sensors dependent on a given component, and the components associated with a given actuator or sensor:

Definition 6: [mapping functions] *Given a component $c_i \in C$ the function $\tau(c_i) = D_i$ returns the set of sensors and actuators that are dependent on component c_i. Given a sensor or actuator $d_i \in D$ t the function $\varphi(d_i) = C_i$ returns the set of the components that the sensor or actuator d_i is dependent on.*

Figure 2 illustrates our model. It presents a partial structural model of the cockpit panel of a Cessna 172p airplane. The dark rectangles represent the components and the bright rectangles represent sensors. For instance, to enable the speed indicator to return a correct reading both the pitot system and the static system need to be operating correctly. The altimeter is dependent only on the static system. The altimeter returns two data readings - altitude and pressure, each is considered as a one dimensional sensor that is dependent on the static system. The same is applied for the attitude indicator that returns the values of the Pitch, Roll and Yaw, which are all dependent on the vacuum system. The GPS is a redundant sensor that besides position values it also returns the speed and the altitude of the aircraft. Since the GPS is dependent only on the electrical system, it will still work in case of a static system failure.

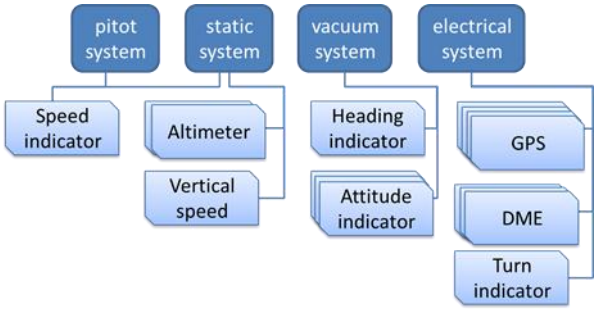

Figure 2: a partial structural model of a cessna172p aircraft

When sensor s_i or actuator α_j is suspected, we also return as a part of the diagnosis $\varphi(s_i), \varphi(\alpha_i)$ the set of internal components which the sensor or actuator is dependent on since a failure to these internal components can also explain the detected fault.

4.2 Isolating the Faults

When a fault is detected it is due to some beliefs that did not get closer to their goal value as the robot operates. The other beliefs that are monitored by the current executing task are considered healthy. Therefore, we can mark each of the sensors that are used to construct these healthy beliefs - healthy as well. These healthy sensors can clear internal components' suspicion. If an internal component c_i is suspected then if a healthy sensor s_j has a dependency on c_i then we can clear c_i of its suspicion: $\exists s_j \mid h(s_j) \wedge s_j \in \tau(c_i) \rightarrow h(c_i)$.

For example, consider a UAV is taking off the runway. The task contains the following actions: (1) set the throttle to full (2) wait until correct speed is achieved (3) elevate the aircraft (4) maintain aircraft roll, pitch and yaw angles and the aircraft's heading until desired altitude is achieved. Now, consider after some point in time during this task execution the belief of the aircraft altitude is unchanged – the aircraft does not get closer to its goal altitude.

Using the control-model we know that the actuators invoked by this task's actions are *{throttle, aileron, elevation, rudder}* and they are all reported as suspected. We also know that the sensors that are used to calculate the monitored beliefs are *{airspeed indicator, altimeter, attitude indicator, heading indicator}*. However, only the altimeter is reported as suspected since only the altitude belief is responsible for the fault detection. The rest of the sensors are healthy. The static system (in Figure 2) is a set of internal components that the altimeter is dependent on and therefore is suspected for a fault as well. However, since other sensors that are dependent on the static system (e.g. airspeed indicator) are healthy, we conclude that the static system is clear of suspicion.

5 EXPERIMENT SETUP AND RESULTS

To demonstrate the feasibility of our approach we use two domains. A high fidelity flight simulator – FlightGear (FlightGear, website) (see figure 3) and the Robotican1 laboratory robot (Robotican, website) (see figure 4). The simulator provides realistic physical behavior of aircrafts. FlightGear provides an interface for reading aircraft's sensors and activating different actuators that control the flight. Thus, aircrafts can be controlled by software as a robotic UAV. Moreover, FlightGear has built-in realistically simulated instrumental and system failures and offers dependency depth between internal components of an aircraft.

We used the FlightGear to simulate a UAV. The UAV enables the reading of 75 different sensors and the control of 32 different actuators. We implemented a set of high-level actions such as *accelerate* action which releases the breaks and push the throttle all the way, or *turn* action which invokes the aileron and rudder actuators to turn the UAV to a given heading in flight. We calculated abstract beliefs from raw sensors readings. For instance, the flown distance from the airport uses data samples from the GPS's readings.

We implemented a control model that enables the association of the different elements. We constructed a structural model that links the actuators, sensors and other internal hardware components according to their dependencies. Figure 2 illustrates a partial picture of the model we used.

Figure 3: FlightGear Simulator Screenshot

We demonstrate different diagnosed failures occurred during a take-off task. In order to show the advantages and disadvantages of our proposed approach we compare it to sensor-based fault detection and diagnosis approach (SFDD).

SFDD does not use the control model of the robot's beliefs and actions but relies on the existence of redundant sensors. Redundant sensors do not share component dependency but measure the same thing for backup purposes, e.g. altimeter and GPS indicated altitude. These sensors should react to the robot's behavior in the same way. However, if a fault has occurred to one of these sensors then the sensors should behave differently. This provides the knowledge whether a sensor is faulty or just reacts to the robot's actions.

We tested 20 takeoff flights; with each flight faults of different types were injected. These faults caused the belief of the UAV's altitude to be stuck or to drift downwards. Upon fault detection the diagnosis reported the relevant suspected sensors, actuators and internal components. In addition, we injected faults that are not relative to the task's beliefs in order to expose the disadvantages of the proposed approach.

Figure 4: Robotican 1

We would like to elaborate on the following study cases which show the advantages and disadvantages of the proposed approach.

Case 1: sensor failure (altimeter). The altitude belief is a function of the altimeter's readings. When failed, the altitude belief triggered a fault alarm. Since $altitude \in B_f$ and $altimeter \in \beta(altitude)$ the altimeter was included in the diagnosis report. The static system was suspected as well, but was cleared since the speed indicator which is also dependent on the static system is healthy: $speed\ indicator \in \beta(B_e \backslash B_f) \rightarrow h(speed\ indicator) \rightarrow h(static\ system)$.

Case 2: internal component failure (static system). When we failed the static system the belief of *speed* also triggered a fault alarm ($speed \in B_f$). This was caused by the fact that the speed belief is a function of the speed indicator's readings, and this sensor has failed (due to the static system failure). In this case the diagnosis report included the static system as a suspect. $\tau(static\ system) \cap \beta(B_e \backslash B_f) = \emptyset \rightarrow \neg h(static\ system)$.

In both above cases, the SFDD detected a fault related to the altimeter sensor since it had a different state than the redundant GPS indicated altitude sensor. The static system failure caused all the dependent sensors to be stuck or to drift down. Therefore, all the static system's dependent sensors were suspected, and thus the static system was reported by the SFDD.

Case 3: actuator failure (throttle, aileron). When we failed the actuators throttle or aileron, the UAV could not climb and started to drop; this triggered the alarm. The diagnosis report included these actuators since $throotle, aileron \in \gamma(ACT')$. The SFDD justifiably did not report anything since all the sensors reflected the UAV's behavior and redundant sensors had no contradicting states. This case presents the benefit of using a control model which can also implicate actuators.

Case 4: software faults. We tested faults like a software crash that stops the calculation of the altitude, and a wrong choice taken by the decision-making process that causes the UAV to stall and to drop altitude. These faults were detected since the altitude belief did not get closer to its goal value. However, as expected, the diagnosis report suspected the hardware components and not the software. The SFDD algorithm justifiably did not suspect the sensors since redundant sensors did not have contradicting states.

Besides having some false negatives that are described in the above study cases, the SFDD has some false positives as well. The SFDD false positive rate is 0.028. The proposed approach in this paper has no false positives since the fault detection is goal oriented and applied on beliefs. However, faults that are injected to sensors that are unrelated to the task's beliefs are undetectable by the proposed approach, but are detectable by the SFDD.

The second domain is the Robotican1 laboratory robot. Robotican1 has two wheels, 3 sonar range detectors in the front, and 3 infrared range detectors which are located right above the sonars, making the sonars and infrareds redundant systems to one another. Robotican1 also has 5 degrees of freedom arm. Each joint is held by two electrical engines. These engines provide a sensed reading of the voltage applied by their action.

To mimic some internal component depths we defined 4 abstract internal components: 1) sonar power supplier, 2) infrared power supplier, 3) arm power supplier 4) motors power supplier. We implemented high-level actions such as *move(speed,angle)* which invokes the motor actuators to spin the wheels in different speeds such that the robot moves at the given speed and angle. We calculated abstract beliefs from raw sensors readings. For instance, the distance to an obstacle is calculated over the filtered readings of the sonar sensors.

We evaluated our proposed approach on two tasks performed by Robotican1. The first task is to approach an object until the distance is within Robotican1's arm reach. The second task is to pick up the object. For each task we tested four scenarios: (1) no faults (2) actuator faults (i.e. wheels, arm joints) (3) sensor faults (i.e. infrared, arm joint feedback) (4) internal component faults (i.e. sensor power supplier).

The first task invoked a belief of the distance covered by the robot. The belief was calculated over filtered readings of the infrared sensor. An injected fault to the wheels or to the power supplier of the wheels' motors causes the robot to stop. An injected fault to the infrared sensor or to the infrared power supplier causes it to be drop to 0. These faults triggered the fault alarm as the distance belief did not get closer to its intended goal value.

The second task invoked a belief of the arm's grabber *3D* position. This belief was calculated by the feedback values that the arm's joints motors returned. An injected fault to a joint motor or to the arm power supplier causes the relevant joints to stop. An injected fault to the joint motor feedback causes it to be stuck on the same value. These faults triggered the fault alarm as the grabber's *3D* position belief did not get closer to its intended goal value.

In both tasks in every scenario the correct actuators and sensors were returned by the proposed approach due to the knowledge provided by the control model. However, power suppliers were always a part of the diagnosis repot even when the fault was not injected to an internal component.

The infrared power supplier was included in the diagnosis report of the first task scenarios since there are no other sensors, dependent on the infrared power supplier, that are associated with an invoked belief. Thus, there is no healthy sensor that can clear the power supplier of suspicion. Suspected actuators always implicate the internal components that these actuators are dependent on since the proposed approach has no policy on how to clear an actuator invoked by an executing task. Thus, the wheels power supplier was always included in the diagnosis reports of the first task scenarios, and the arm power supplier was always returned in the second task scenarios.

6 CONCLUSION and FUTURE WORK

In this paper we defined a model-based approach for fault-detection and diagnosis and presented a multi-layer model of the robot which enables to diagnose low-level hardware faults by monitoring the high-level beliefs of the robot. We present different types of faults related to the different layers of the robot and demonstrated these types of faults in simulated UAV and a laboratory robot and how they are detected and diagnosed by the proposed approach. We compared the proposed approach to the SFDD algorithm and discussed the advantages and disadvantages of each approach.

Preliminary tests of a combined approach define our future work. Initiating the SFDD when a fault is detected by the proposed approach might further isolate the diagnosis i.e. if the fault is not sensor related then it might be actuator related. We also wish to investigate a less than obvious approach for belief fault detection with respect to its intended goal value.

ACKNOWLEDGMENT

This research was supported in part by General Motors. Thanks to Gali Bodek and Yotam Shichel for their help.

REFERENCES

(Arkin, 1998) R. C. *Behavior Based Robotics.* New York, NY: Cambridge University Press.

(Brandstötter, 2007) M. Brandstötter, M. Hofbaur, G. Steinbauer, and F. Wotawa. *fault diagnosis and reconfiguration of robot drives*. IEEE International Conference on Intelligent Robots and Systems.

(Brooks, 1986) R. A. Brooks. *A robust layered control system for a mobile robot.* IEEE Journal of Robotics and Automation.

(Gspandl, 2011) S. Gspandl, I. Pill, R. Michael and G. Steinbauer. *Belief Management for High-Level Robot Programs* - Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence.

(Isermann, 2005) R. Isermann. *Model-based fault-detection and diagnosis—Status and applications.* Annual Reviews in Control, 29(1), 71–85.

(Steinbauer, 2005) G. Steinbauer and F. Wotawa. *Detecting and locating faults in the control software of autonomous mobile robots*. 19th International Joint Conference on Artificial Intelligence (IJCAI-05).

(Steinbauer, 2010) G. Steinbauer and F. Wotawa. *On the Way to Automated Belief Repair for Autonomous Robots.* International Workshop on Principles of Diagnosis (DX).

(Varun, 2009) C. Varun, B. Arindam and K. Vipin. *Anomaly detection: A survey*. The Association for Computing Machinery, Computing Surveys, 41(3):1–58.

(FlightGear, website) http://www.flightgear.org/

(Robotcup, website) Robotcup league http://www.robocup.org/

(Robotican, website) http://www.robotican.net/

(Steinbauer, website) G. Steinbauer. *a survey on the nature of faults of autonomous robot systems*. http://www.ist.tugraz.at/rfs/index.php/Main_Page