Available online at www.sciencedirect.com
SCIENCE DIRECT®
Robotics and Autonomous Systems 53 (2005) 73–88
Robotics and Autonomous Systems
www.elsevier.com/locate/robot

Execution monitoring in robotics: A survey

Ola Pettersson*

Center for Applied Autonomous Sensor Systems, Department of Technology, Örebro University, SE-70182 Örebro, Sweden

Received 20 September 2004; received in revised form 1 September 2005; accepted 13 September 2005

Abstract

Research on execution monitoring in its own is still not very common within the field of robotics and autonomous systems. It is more common that researchers interested in control architectures or execution planning include monitoring as a small part of their work when they realize that it is needed. On the other hand, execution monitoring has been a well studied topic within industrial control, although control theorists seldom use this term. Instead they refer to the problem of *fault detection and isolation* (FDI).

This survey will use the knowledge and terminology from industrial control in order to classify different execution monitoring approaches applied to robotics. The survey is particularly focused on autonomous mobile robotics.

Keywords: Robotics; Autonomous systems; Execution monitoring; Fault detection and isolation; Diagnosis

1. Introduction

The robot manipulator has been an essential tool for the development of the automated industry. In the near future these robots will be accompanied by intelligent mobile robots that will assist us in many different areas, including transportation, cleaning, mining, or agriculture. Such intelligent robots incorporate increased flexibility and need the ability to plan their actions and to execute them in a safe way. In order to operate in a changing and partially unpredictable environment, these robots also need the ability to detect when the execution does not proceed as planned, and to correctly identify the causes of the failure. An *execution monitoring system* is a system that allows the robot to detect and classify these failures.

In this survey we will use the terminology given from the field of industrial control in order to classify different execution monitoring approaches applied to robotics. This particular terminology is chosen since execution monitoring has been a well studied topic within industrial control for several years, and the work within this field is therefore relatively mature.

1.1. Motivation

Execution monitoring is needed in robotics in order to handle problems caused by uncertainties, both in the robot itself and in the environment. In [46, p. 7] the

* Fax: +46 19 303463.
E-mail address: ola.pettersson@aass.oru.se.

0921-8890/$ – see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2005.09.004

concept of uncertainty is classified for systems within the field of artificial intelligence. Based on the typology in this work, four main *sources of uncertainty* can be found in robotics:

- Missing information;
- unreliable resources;
- stochastic phenomena; and
- inherently vague concepts.

Since the world is not totally observable, the robot will occasionally suffer from *missing information*. An example of such an occasion is when the robot has no knowledge about what is behind a closed door. In some cases a used resource can be faulty. Examples of *unreliable resources* are a broken driving shaft, or a given map that is old and out of date. When measuring some physical quantity, there is always a certain variance in the value. An example of this so called *stochastic phenomena* is the sonar readings that include a stochastic noise component. *Inherently vague concepts* can occur when the world state or human knowledge are modeled. For example, concepts like "doors are most likely open," or "large red door" are inherently vague concepts.

Furthermore, the sources of uncertainty are present at several levels of abstraction within a robotic system. Many robots have not only the ability to move, but also the ability to avoid collision, localize itself, recognize objects, and reason about goals and actions. Therefore, many fundamentally different faults might occur including hardware faults to more abstract knowledge representation faults.

According to [67] the effects from uncertainty can be decreased by the use of three main ideas:

- eliminating uncertainty;
- reasoning about uncertainty; and
- tolerating uncertainty.

Some of the uncertainty can be eliminated by using better hardware, or engineering the environment. Examples of better hardware are high-precision mechanics and sophisticated sensors. The environment can be engineered by putting up artificial landmarks, or putting fixed tracks in the floor. When reasoning about uncertainty more complex models of uncertainty are used. In this case uncertainty can be modeled as a type of knowledge. For example, 50 sonar readings in the same direction indicating a free distance of about 8 m is a more certain knowledge compared to two sonar readings in the same direction, where one measurement indicate 4 m and the other 25 m.

The idea of eliminating uncertainty is very common in industrial robotics and results in good performance, but has a number of drawbacks. First, using high-precision mechanics or adding sophisticated sensors might significantly increase the cost of the robot. Second, engineering the environment decreases flexibility; for example, industrial robots that follow fixed tracks in the floor will not be flexible enough to work as multi-purpose service robots. Third, relying on an engineered environment might also decrease robustness of the robot. For example, the artificial landmarks could be hidden by dust or obstacles. Finally, not all sources of uncertainty can be eliminated by engineering the environment. For example, actions by humans cannot always be predicted.

Reasoning about uncertainty is necessary in the planning stage in order to weight different viable plans. Nevertheless, the reasoning do not necessarily increase robustness of the execution. In other words, no reasoning can obtain information that is missing. For example, the robot can only guess what is behind a closed door, but not really know before the door is opened.

Therefore, in order to act robustly in a partially unknown and dynamic world, the system must also tolerate uncertainty. In other words, the system must be prepared for failing execution. One way to do this is to introduce an execution monitor in the control architecture.

1.2. Definitions

There are many different definitions of execution monitoring in the literature. For example, within the field of artificial intelligence (AI), more specifically robot planning, we find the following definition [13, p. 178]:

In robot planning, the process of sensing the state to influence subsequent action is called execution monitoring.

Another more specific and informative definition of execution monitoring within the field of AI is given in [5, p. 26]:

Execution monitoring is an agent's process of identifying discrepancies between observations of the actual world and the predictions and expectations derived from its representation of the world, classifying such discrepancies, and recovering from them.

The first definition is rather general and perhaps not useful for this work. On the other hand, the second definition assumes that a model with prediction capacity is used. This comes from the notion of "predictions and expectations." With the use of this definition a system without a predictive model would not be performing execution monitoring. Within the field of industrial control we find a definition that does not make any commitment to prediction. The definition is given in [41, p. 710], and when slightly modified into our terminology it reads:

Definition 1 (Execution monitoring). *Execution monitoring* is a continuous real-time task of determining the conditions of a physical system, by recording information, recognizing and indicating anomalies in the behavior.

Compared to the first definition this one is more informative. At the same time, it is more general than the second one, since no assumptions about models are made. Note that this definition does not include the process of recovering from the detected faults. By referring to the definition of execution monitoring, we can simultaneously define a *fault* as an anomaly in the behavior of the monitored system.

According to Gertler [29, p. 3] a monitoring system could be divided into the following three functionalities:

- *Fault detection* that indicates that something is going wrong in the monitored system;
- *fault isolation* that makes a classification of what is going wrong; and
- *fault identification* that determines the magnitude of the fault.

While fault detection is an absolute must in any practical system, and isolation is almost equally important, fault identification may not always justify the extra effort it requires [29]. Therefore, many practical systems contain only the fault detection and isolation stages. In many cases the term *fault diagnosis* is used as a synonym to fault isolation.

As mentioned earlier, execution monitoring is a well studied topic within the field of industrial control, although control theorists seldom use this term. Instead they refer to the problem of fault detection and isolation (FDI). Many different monitoring methods have been validated in industrial applications since the end of the 1970s. Therefore, the amount of existing work within this area is large. Several text books have been written on the topic, see for example, [29] or [10]. Furthermore, the topic is also very well categorized within this field. Definitions to many terms used in FDI, together with a comprehensive survey on applied FDI systems are, for example, given in [41].

According to Chiang et al. [11] execution monitoring can be classified to one or more of three approaches, namely: *analytical*, *data-driven*, and *knowledge-based*. In the coming sections these three approaches will be introduced. An overview of existing execution monitoring systems applied to robotics is given. Particularly, mobile robotic systems are discussed. Each monitoring system is classified into one of the three approaches.

2. Analytical approaches

The analytical approach is model-based since mathematical models, often constructed from first principles, are used. By first principles we here mean basic models from physics, for example, fluid dynamics, mechanics or electricity. This approach relies on the concept of *analytical redundancy*. This means that two analytically generated quantities, obtained from different sets of variables are compared. The resulting difference, called *residual*, indicates the presence of a fault in the system. Fig. 1 illustrates the conceptual structure of such an analytical approach. As can be seen in this figure two main stages are performed, namely *residual generation* and *decision making*.

Residual generation: Residual generation is an algorithm that processes the measurable input $u(s)$ and output $y(s)$ of the system in order to generate the residual signal $r(s)$, where s denotes the system state given in the Laplace space. It uses the model describing the

relationship between those variables in exact mathematical terms, and any inconsistency in this relationship will indicate a fault in the system. The residual signal is given as:

$$r(s) = H_u u(s) + H_y y(s), \quad (1)$$

where H_u and H_y must be chosen such that:

$$\begin{cases} r(s) = 0 \text{ when no fault occurs, and} \\ r(s) \neq 0 \text{ when a fault occurs.} \end{cases} \quad (2)$$

Decision making: The residual is then examined for the likelihood of faults, and a decision rule is applied to determine if a fault has occurred. This decision process can be based on, for example, a simple threshold test, on the instantaneous values of moving averages of the residual, or it may involve methods from statistical decision theory. To obtain fault information the decision making must also include fault isolation. A single residual signal is sufficient to detect the presence of a fault but several residual signals are often required for fault isolation.

In the literature on analytical-based fault detection three different approaches to residual generation can be identified [59], namely: *parameter estimation*, *parity relations*, and *observers*. Parameter estimation together with observer-based approaches are the two most frequently applied methods for fault detection, especially for the detection of sensor and process faults. These two methods are used in nearly 70% of all applications [41]. Recently, it has been realized that there is a fundamental equivalence between observers and parity relations [29]. Actually the two techniques produce identical residuals if the generators have been designed for the same specification. A relationship between parameter estimation and parity relations has been found as well [28].

2.1. Parameter estimation

In this approach a reference model is created by first identifying the system's physical parameters in a fault-free situation. The physical parameters can be found using system identification theory methods [50]. Examples of such physical parameters are friction and mass velocity resistance. The parameters are then repeatedly re-identified online during monitoring. The residuals are calculated as the difference between the reference model parameters and the estimated model parameters.

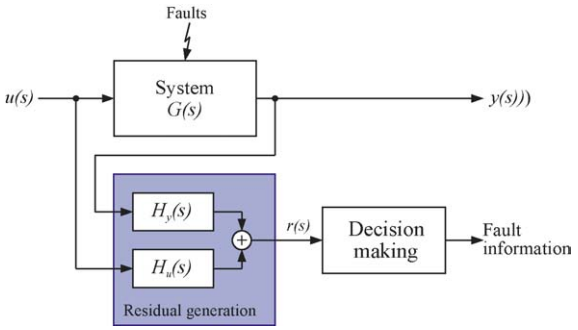

Fig. 1. A schema of an analytical approach to fault detection and isolation.

An overview of theory and applications of parameter estimation approaches is found in [38]. This approach is, for example, applied to fault detection in robot arms [39]. In [15] a complex kinematic model of a robot arm is used for fault detection. The residual is calculated from filtered torque signals. The main strength of this work is that no acceleration measures of the manipulator are required. In another, slightly more complex example, the parameters are calculated by a Hopfield neural network, and the decision making for fault detection and isolation is performed by a fuzzy expert system [36].

2.2. Parity relations

In this approach, mathematical equations referred to as *parity equations* are compared. The system's model parameters must be known a priori. The residual is generated by a consistency check of the reference model equation and the system equation generated from measurements. A consistency violation of the equations, an equation error, indicates that a fault is present in the system.

The origin of using parity relations for fault detection is found in aerospace related research, primarily sponsored by NASA in the 1970s. An early overview of this approach is found in [82]. In the textbook [29] a more recent survey is given. Yet another example of fault detection with parity relations is given in [85]. Here the parity equations are designed by the use of discrete time wavelet transforms.

2.3. Observers

Most execution monitoring systems reported in the robotics and AI literature falls into this category. As for parity relations, the system's model parameters must be known a priori. The underlying idea in observer-based approaches is to estimate the system outputs from available inputs and outputs of the system. The differences between the measured outputs and the estimated outputs are used as the vector of residuals. A very common special case of a state observer is the Kalman filter [26]. There is an advantage in using observer-based methods over parameter estimation. In the observer-based approach no special excitation is required, the method works in steady-state operation conditions as well. Therefore, the observer-based approach is appropriate if the faults are connected to problems in actuators, sensors, or the unmeasurable state variables.

The original idea of using observers, and perhaps also analytical monitoring in the first place, is given in [3]. An overview of observer-based fault detection methods is found in [24]. In [70] a fuzzy observer-based fault detection system is applied to a robot arm. More work on fuzzy observers is found in [4]. An other observer-based monitoring system applied to a robot arm, that also include fault isolation, is presented in [9]. Here the residual vector is used to distinguish between two types of faults, namely: actuator faults and sensor faults.

Already the first autonomous mobile robot Shakey had a system for fault detection, although it was involved at a high level of abstraction finding faulty plans. Shakey managed to accomplish tasks such as pushing boxes from one room to another in a physical, but highly engineered, environment. On-board Shakey PLANEX [19] was used for execution and monitoring of plans produced by the well known planner STRIPS [20]. The plan given from STRIPS was stored in a tabular form called a *triangle table*. A triangle table (see Fig. 2) is a triangular array where the rows correspond to the actions of the plan. The preconditions are given in the cells to the left of the action, and the expected outcome of each action is given in the cell below the action. Fault detection is realized by comparing the measured state to the expected outcome given from the triangle table. An advantage of the triangle table is that not all world states have to be modeled since the plan can be generalized. For example, the action gothru(d1, A, B) is a general model for crossing any door between two rooms. The main drawback of this method is the lack of robustness, since full observability of the robot's state is assumed.

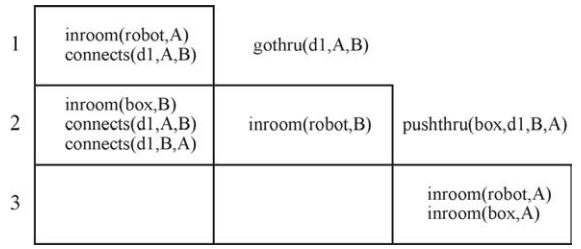

Fig. 2. An example of a triangle table.

A similar approach to PLANEX is presented in [18]. Faults are detected by comparing the current state with the model of the world and the plan. This approach also involves fault isolation: when a discrepancy arises between the expectations and the actual situation, a reasoning functionality tries to find suitable explanations and a recovery plan.

Kalman filters [26] are commonly used for sensor fusion. This is probably due to the fact that the Kalman filter is an optimal estimator in the minimum mean squared error sense when the sensor model predictions and system model are assumed to be corrupted with zero-mean Gaussian noise. Furthermore, Kalman filters can be applied in observer-based fault detection. In [69] Kalman filters are used to detect faults in a laser and gyroscope navigation system. In [53] a Kalman filter is used for fault detection and sensor improvement of a GPS navigation system tested on an outdoor mobile robot. Using one Kalman filter is often sufficient for sensor fusion or fault detection. If the faults also must be isolated, a bank of Kalman filters, that is, several Kalman filters in parallel, are often needed. In [66] three different sensor faults are isolated on a mobile robot using a bank of Kalman filters. Here each Kalman filter is designed to detect one of the following faults: a noisy gyro, a broken gyro (stuck to a fixed value), and a broken wheel encoder. In [30] the performance of this approach is increased by sending the residuals from the bank of Kalman filters to a neural network. Here the neural network performs the decision making. In the coming decade NASA is planning several missions involving a planetary rover moving across the surface of a planet and collecting samples. In [79] Kalman fil-

ters together with Markov decision processes perform fault detection and isolation on such an autonomous vehicle. Three types of hardware faults are isolated: a stalled motor, a broken gear, and a broken gear and a broken encoder.

The Procedural Reasoning System (PRS) [27] was developed as the execution and monitoring system on board the mobile robot Flakey. This system is inspired by the work on human reasoning called the Belief-Desire-Intention (BDI) model [7]. During execution a certain subplan, a so-called *intention*, is selected in order to achieve the given goal. Each intention has an expected outcome that is monitored by the system. When an intention fails, that is when the expected world-state differs from the real world-state, precompiled recovery routines can be called to handle the problem. PRS is therefore able to detect and isolate faults and pursue new goals dynamically. The LAAS architecture, see [56] and [2], is an architecture for autonomous agents. The fault isolation and recovery functionalities are based on PRS. When a fault is detected, that is, when the real outcome differs from the expected outcome, the execution is halted and a failure tree is traversed in order to isolate the fault. The failure trees are precompiled models of the known faults. The LAAS architecture has been applied in a variety of domains including the control of spacecraft systems [31], and the control of mobile robots [37]. More recent results in the improvement of the architecture is found in [49].

The Reactive Action Packages (RAP) system [21] was introduced as a plan execution and monitoring system for mobile robots and has much in common with PRS. In the RAP system a task is described by a RAP which is effectively a context sensitive program for carrying out the given goal. The RAP can also be thought of as describing a variety of plans for achieving the task in different situations. Similar to the intentions in PRS, each RAP has its own goal and expected outcome. Therefore, a RAP can, be programmed to handle modeled faults. The RAP system is the main component in the Animate agent architecture, see for example [22] and [23]. This architecture is implemented on the mobile robot Chip. Chip is able to find, identify, and track objects. It is also able to manipulate objects, for example, picking up trash and bring it to a trash can, using a robot arm.

Livingstone [81] is the name of the control architecture used in NASA's first New Millennium spacecraft. The autonomous spacecraft, named Deep Space One, was launched in 1998 to fly by asteroids and comets, photographing and sending back information to scientists on Earth. Livingstone performs state identification of the modeled system in order to detect faults and isolate them. In Livingstone 2 [47] the state identification is improved by the use of Markov decision processes [6]. The transition model between states in Livingstone 2 is restricted in order to reduce the state space when the problem space grow. In a set of simulations, Livingstone 2 is shown to perform better and to use less CPU time compared to Livingstone 1. Livingstone 2 is implemented, for example, on a free-flying mobile robot [16]. A free-flying mobile robot is a spheric science-fiction-like robot that can perform a number of tasks both inside and outside a space craft, such as remote sensing, or providing crew support.

The execution monitoring device on board the mobile robot Xavier [74] is programmed in TDL [73]. TDL is a language package for C++ that is used for planning, monitoring and dynamic execution. It is very similar to PRS and RAPS. The monitor can detect and isolate faults in the execution of navigation plans. A fault is defined as a significant difference between the observed state of the world and the expectation with respect to the nominal situation [17]. Therefore, this approach can also be categorized as observer-based monitoring. Yet another observer-based fault detection and isolation system evaluated on Xavier is Rogue [89]. An interesting feature in this system is its learning capabilities that are described in [90].

Rationale-based monitoring [77] is another planning and monitoring system. In Rationale-based monitoring, a set of states within the world-model are selected to be monitored. This approach dramatically decrease the state space to be observed. When a fault is detected, that is when a monitor identifies a potentially relevant change in the world-state, the current plan can be transformed in order to repair the problem. The Rationale-based monitor was initially implemented within the Prodigy planner [76], and tested in various simulated domains. In more recent work the Rationale-based monitor has been evaluated on a mobile robot [51].

A multi-agent approach to observer-based fault detection is presented in [8]. In this approach the different agents' states are combined into a joint state that is observed. Precompiled rules are used to recognize when

the current joint state indicates a fault. The approach has been evaluated in simulated battlefield scenarios.

3. Data-driven approaches

In contrast to analytical approaches, the data-driven approaches do not rely on mathematical models. Instead, the information used for fault detection and isolation is derived directly from input data. The decision making is often based on statistical methods. Modern industrial systems, whether an entire industrial plant or an autonomous robotic system are large-scale systems. With their heavy instrumentation, these systems produce an exceptionally large amount of data. The strength of data-driven approaches is their ability to transform the high-dimensional data into a lower dimension space; in which the important information is captured. By computing statistic measures, the monitoring system can be improved significantly in large-scale systems. The main drawback using this approach is that the performance is highly dependent on the amount and quality of the input data.

Execution monitoring based on data-driven approaches has, until now, been almost totally neglected in the AI literature. Nevertheless, these approaches have been adopted in some work on robotics within the field of industrial control, although analytical approaches are much more frequent.

The application of statistical theory to execution monitoring relies on the assumption that the characteristics of the data variations are relatively unchanged unless a fault occurs in the system. It implies that the properties of the data variations, such as the mean and variance, are repeatable for the same operating conditions, although the actual values of the data may not be very predictable.

The data-driven approach can be divided into two groups referring to the number of variables measured by the monitor. In *univariate statistical monitoring* only one variable is measured at a time, and in *multivariate statistical monitoring* several different variables are measured and combined.

3.1. Univariate statistical monitoring

As mentioned in Section 2, parameter estimation and observers are the most common residual generation approaches in the industry. In these model-based approaches the residual signal is often compared to a given limit value. If the measured variable exceeds the threshold limit a fault is indicated. This method is called *limit value checking* and is the most frequently used method for decision making in industry [40].

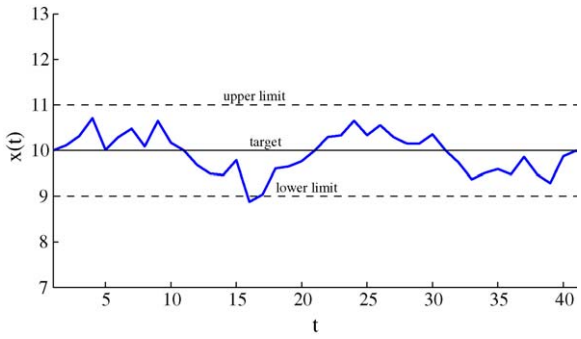

Fig. 3. An example of a Shewhart chart showing the measure of a certain observation variable $x(t)$. A fault is detected either when the measure exceeds the upper limit or goes below the lower limit.

However, limit value checking is not restricted to analytical monitoring. Instead a univariate statistical approach can be used to determine the thresholds for some *observation variables*. Observation variables are input data obtained directly from a sensor. In this approach a Shewhart chart [52] (see Fig. 3) is often used to visualize the limits when minimizing the number of false alarms and missed detections. Tight threshold limits for an observation variable result in a high false alarm rate, and a low missed detection rate, while limits that are too spread apart result in a low false alarm rate and a high missed detection rate. Given the threshold values, statistical hypothesis theory can be applied to predict the false alarm and missed detection rates based on the statistics of the data in a training set. The principles of statistical hypothesis theory can, for example, be found in [65].

An early example of fault detection using limit value checking on an autonomous robot is the work on the Thinking Cap [68]. The Thinking Cap is a robot control architecture including fault detection based on fuzzy logic. In this system a variable, that is a fuzzy composition of several variables, is compared to a given limit in order to detect faulty plans. When a fault is detected a recovery strategy, that is, replanning is called.

In [57] this work is extended by comparing the fuzzy variable to several limits in order to obtain the magnitude of the fault. Later work [58] has combined the execution monitoring system with ideas from the BDI model.

More recently, another univariate limit value checking approach is applied to a mobile robot [45]. Here the input variable is the motor current. The limit value is derived from empirical data when the mobile robot work under normal conditions. This work shows that the performance is not very reliable when only one variable is measured. For example, when the robot moves upwards in a slope the motor current increases even though there is no fault. To overcome this problem a second observation variable is introduced. When a gyro measuring the pitch direction of the robot is used to tune the motor current limit, the performance is increased significantly. In other words, a monitor based on multivariate statistics is more robust than one based on univariate statistics.

3.2. Multivariate statistical monitoring

For univariate statistical methods, the thresholds are determined for each observation variable individually without considering the information given from the other variables. In contrast, the multivariate statistical methods do consider the correlation between variables. This gives a much more powerful tool both for fault detection and fault isolation.

The quality and quantity of the training data have a large influence on the performance of data-driven statistical approaches. If sufficient data is not available to accurately estimate the mean vector and covariance matrix, this will result in suboptimal fault detection and isolation. One way to improve the performance is to reduce the number of input variables using dimensionality reduction methods. Once the dimensionality reduction is done, decision making is performed by applying discriminant analysis to the reduced-dimensional space.

Principal component analysis (PCA) is probably the oldest and best known method for multivariate analysis [33]. PCA is a linear dimensionality reduction technique, optimal in terms of capturing the variability of the data. Algebraically, principal components are linear combinations of a set of random variables. To distinguish between the transformed variables and the transformed observations, the transformed variables are called *principal components*, and the individual transformed observations are called *scores*.

Fault detection can be performed by monitoring each one of the scores using the univariate statistical approach discussed above. An advantage of using PCA in fault detection is the dimensionality reduction of the data. When the data is projected into the lower dimensional space using PCA, a smaller number of variables are checked compared to a system without the use of PCA. Also fault isolation may be performed using PCA. The simplest way is to construct a single *PCA model*, that is, the transformation matrix, and define regions in the lower-level dimensional space which classifies whether a particular fault has occurred. The classification can be made by statistical methods, for example, multiple linear regression (MLR). Unfortunately, this approach is unlikely to perform well when several faults can occur [88]. A better idea is to use several PCA models based on data collected from specific fault situations. In this case the data is transformed by all PCA models and then an arbiter chose the model describing the fault that most likely occurred. This method of creating several PCA models is often called *multi-model PCA* (mPCA).

Early work on fault detection applied to a planetary rover is presented in [25]. This is perhaps the first example of a data-driven execution monitor applied to an autonomous mobile robot. In this multivariate statistical approach the limit values for the different observation variables are given from an execution monitoring planner. This particular planner calculates the set of minimum and maximum limits by simulating the execution of a given plan. The simulation takes into account the uncertainty in the local terrain data derived from statistical measures.

Recent work on fault detection and isolation applied to a planetary rover is presented in [14]. Here the system is described as a discrete-time probabilistic hybrid automaton. This work is highly related to Kalman filters and the approach is similar to observer-based approaches. But instead of defining the different states in the automaton from first principles, the transition functions between states are given from statistical measures from several observed variables. Fault isolation is realized using *particle filters* [78]. A particle filter is a Markov chain Monte Carlo algorithm that approximates the belief state using a

set of samples (particles), and keeps the distribution updated as new observations are made over time. The faults addressed include: actuator faults, such as broken motors or gears; faults due to environmental interactions, such as a wheel stuck against a rock; and sensor faults, such as broken encoders.

Another multivariate statistical approach applied to a mobile robot is given in [75]. Here probabilistic models of both the sensors and the environment are used for sensor fault detection. This probabilistic sensor fusion approach is used for fault detection in the sonar sensors.

In the work on *model-free execution monitoring* [60] the use of mPCA together with MLR is applied to an autonomous mobile robot. The approach is called model-free since no predictive models of the system are used. As can be seen in Fig. 4 the activation levels of a set of behaviors act as input data to a standard pattern recognition system that performs both fault detection and isolation. Here a number of robust features are extracted from several activation level measures in a time interval. Both normal behavior and the different faults are learned on-line during execution. The idea is evaluated in a number of simulations when two faults are isolated: a closed door, and a corrupted world-model [61].

4. Knowledge-based approaches

Generally knowledge-based approaches are designed to simulate the problem-solving behavior of human experts. This category is not distinct from the former two approaches since both analytical and data-driven approaches apply to this class. Perhaps the most important strength using knowledge-based approaches is the opportunity to combine analytical and data-driven approaches in a hybrid monitoring system. The knowledge-based approaches can be divided into three categories: *causal analysis*, *expert systems*, and *artificial neural networks*.

4.1. Causal analysis

Causal analysis methods are based on causal modeling of fault-symptom relationships. These methods are primarily used for fault isolation.

One method for fault isolation by causal analysis is with the help of a *signed directed graph* (SDG) [11]. An SDG is a map showing the relationship between process variables. It reflects the behavior of the equipment involved as well as the general system topology. The nodes can represent process variables, sensors, system faults, or component faults. When an SDG is used for fault isolation, upper and lower threshold limits for each variable must first be defined. The limit values may be found using the methods discussed in Section 3. In the SDG a node takes the value of 0 when its measured variable is normal. When the variable is higher or lower than the given upper or lower limit the node takes the value of + or −, respectively. Assuming that a single fault affects only a single root node and that the fault does not change other causal relations in the SDG, the causal linkages will connect the fault origin to the observed symptoms of the fault.

The work on Overseer [44] differs from most previously referred work since it focuses on monitoring multi-agent systems, that is, several robots in parallel. In Overseer the agents are modeled as a signed directed graph. This graph is a map showing the relationship between belief and the different agents' statuses, and it also reflects the behavior of each agent as well as the full team hierarchy. Each node in the graph correspond to the known states of all the agents. According to [44] the state information can be obtained through either:

- team-member's communication (report-based monitoring); or
- observation (overhearing) of team-members.

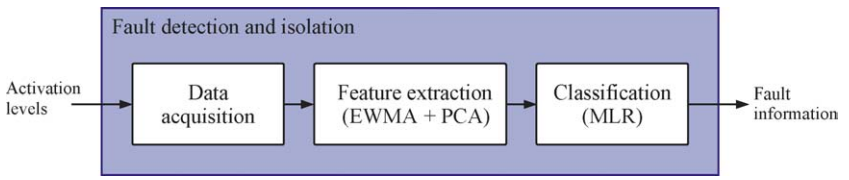

Fig. 4. Schema of the model-free execution monitoring system. Fault detection and isolation are realized by a pattern recognition system.

Since there are many problems involved in the communication between agents [43], their research is focused mainly on the observation of team-members. The key idea is to use various models of social relationships among agents, rather than goal-attentive models of the tasks. For example, a driver may not see a road-sign that tells it to turn, and therefore incorrectly continues straight ahead. But if the driver is driving in a convoy, where everybody shares the goal destination, the driver can infer the existence of the road-sign when everybody else turns. Overseer has been applied to a domain that involves automated pilot agents that participate in synthetic multi-agent battlefield simulation environments.

Other work that is focused on multi-agent monitoring is the work on Execution Assistants (EAs) [80]. This work has its focus on dynamic, data-rich domains where humans are ultimately responsible for team behavior. Therefore, the automated help should interactively support effective and timely decision making by the human. In contrary to Overseer, the EAs rely on state information that is communicated between agents. This communication is performed using peer-to-peer dissemination algorithms and forward fusion of sensor reports, in order to reduce the bandwidth requirements. Since peer-to-peer is fault tolerant, allowing any node to act as a server, the reliability of the communication is increased. The EAs use the BDI model and are implemented as multiple asynchronous PRS agents (discussed in Section 2.3). Each agent has beliefs about the state of the world, desires to be achieved, and intentions representing actions that the agent has adopted in order to achieve its desires. The EA approach has been applied to two different domains in simulation. One domain has hundreds of mobile, geographically distributed agents, a combination of humans, robots, and vehicles. The other domain has a handful of cooperating robots. Alerts were generated in a timely manner without inundating the user with too many alerts. Less than 10% of the alerts were unwanted, as judged by domain experts. Several faults were isolated, for example: an agent is out of position or late, an agent's actions were not effective, and an agent has adopted an action conflicting with other agents.

A heterogenous knowledge-based approach to execution monitoring is presented in [32]. In this approach, named Recovery, different types of knowledge are linked together with the use of partitioned semantic networks. Here the knowledge is categorized into five different types, namely: design knowledge, sensor knowledge, historical knowledge, mission knowledge, and fault knowledge. The key benefit in this approach is that a priori fault isolation knowledge, primarily gathered from human experts can be combined with available vehicle knowledge, for example, sensor knowledge. Recovery has been tested on a real autonomous underwater vehicle (AUV). In the experiments one type of actuator fault is isolated, namely a broken lift thruster.

4.2. Expert systems

Many fault isolation applications in the areas of engineering have made use of expert systems [42]. Expert systems are used to imitate the reasoning of human experts when isolating faults. The knowledge in the expert system is often formulated in terms of IF-THEN rules,

$$\text{IF } \textit{condition} \text{ THEN } \textit{conclusion}, \tag{3}$$

which can be found from first principles or a structural description of the system for isolating faults. Expert systems based on logic using terms that are either true or false are relatively sensitive to uncertainties. One way to overcome this problem is to use *fuzzy logic* [86]. Fuzzy logic provides an approximate, but still effective, means of describing complex ill-defined systems by using graded statements instead of strictly true or false.

An example of an expert system for fault detection and isolation in a flexible assembly work cell is given in [1]. A fuzzy expert system is applied on a robot manipulator in [84]. Some recent theoretical results on expert systems applied to robotics are presented in [12].

The monitoring system within the behavior-based Saphira architecture [48] is designed as an expert system implemented in linear temporal logic (LTL). In this approach a set of temporal fuzzy rules, operating on the activation levels of the behaviors, are used for fault detection and isolation. The general structure of the monitoring system, shown in Fig. 5, consists of:

- a monitoring knowledge base containing models of the expected and unexpected ideal behavior of the robot in terms of temporal constraints;
- a trace collector for tracking the activation levels of the behaviors;

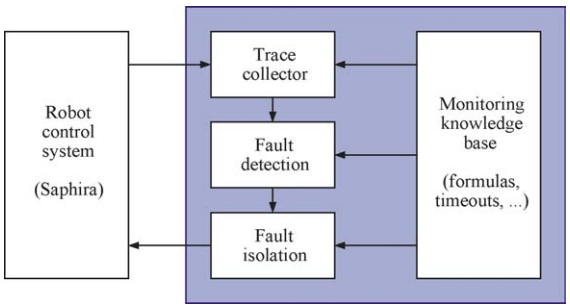

Fig. 5. Fault detection and isolation by a fuzzy expert system: the linear temporal logic (LTL) approach.

- a fault detection module for checking a temporal fuzzy logic formula over the current trace to determine whether or not it violates the formula; and
- a fault isolation module that evaluates a trace of the temporal logic verification process to determine the type of fault.

In contrary to many other expert systems the fault detection and isolation is not based on the current state solely. The use of LTL also allows notions of time and time intervals, and therefore a sequence of measures can be analysed. The activation levels of the behaviors are filtered to reduce noise using *ordered weighted average* (OWA) operators [83].

Work that is not directly focused on fault detection and isolation, but still related to execution monitoring is the work on the Dual Dynamics & Planning (DD&P) robot control architecture, see [35,72]. This work is related to execution monitoring since the DD&P architecture has an interesting technique for extracting information about the robot and its performance. This particular information, that is, symbolic ground facts, that are given to the world-model, are obtained by analyzing the activation levels of the behaviors. Four specific features are extracted from the activations levels over time [71]: rising edge, falling edge, high level, and low level. The key idea in extracting facts from activation levels is to consider patterns of qualitative activations of several behaviors that occur within the same interval of time. A set of rules, named *chronicle definitions*, are used to determine whether a certain ground fact holds or not within a given time interval. The method is evaluated on a mobile robot in simulation.

4.3. Artificial neural networks

Artificial neural networks (ANN) were motivated from the study of the human brain, which is made up of millions of interconnected neurons [33]. Similar to the data-driven approaches discussed earlier, ANNs are most useful when it is hard or even impossible to create mathematical models of the monitored system. The main drawback is that their performance is highly dependent on the amount and quality of the input data.

Several authors have applied ANNs to fault detection and isolation. In [54] a comparison between radial basis function networks and the classical parameter estimation approach is made. Their results show that the classical methods can compete if the assumptions for the structure are valid. However, if in practical applications the structure is not known the radial basis function network performs much better. In recent work presented in [87] it is shown that a combination of several neural networks can perform better than a single network.

In more recent work on the DD&P robot control architecture, discussed above, an ANN has been applied for the process of extracting facts from the activation levels [34]. In this work it is shown that the chronicle definitions can be learned. Here an *echo state network*, which is a particular type of recurrent neural network is used for learning. The method is demonstrated on data from a mobile robot simulator.

Also the work on model-free execution monitoring involves artificial neural networks. In [62] the use of ANNs has been evaluated in a hierarchical execution monitor. This monitor incorporates two artificial neural networks: one for fault detection and one for fault isolation. In Fig. 6 the hierarchical model-free execution monitor is shown. The approach is successfully tested in numerous experiments both on a real mobile robot and in simulation.

As mentioned, the main drawback using learning approaches for execution monitoring is that the training data must provide an adequate coverage of all the fault situations. Unfortunately, it is not always feasible to force the robot into all known fault situations, and some of these situations can be catastrophic for either the robot or its operating environment. An interesting idea to overcome this drawback is “learning from simulation” [63] which lets the execution monitor learn

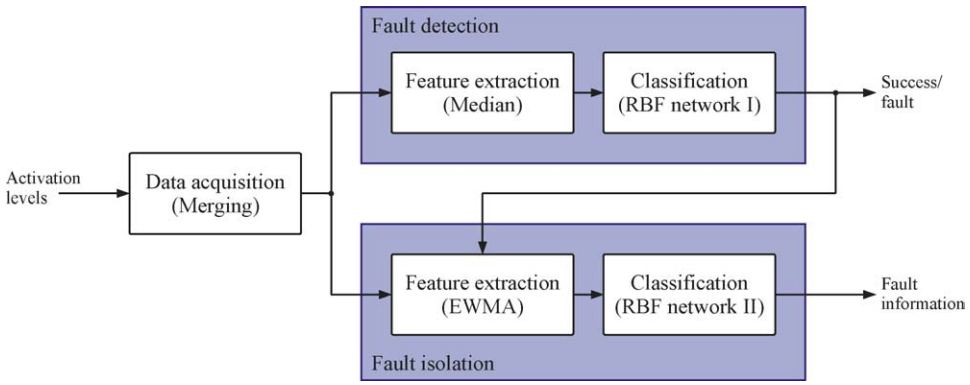

Fig. 6. Schema of the hierarchical model-free execution monitoring system. Fault detection and isolation are performed by two separated pattern recognition systems.

the fault classes safely in simulation, and then use the system to monitor a real robot.

Artificial neural networks can also be combined with other approaches. For example in [64] a multilayer perceptron feed forward network is used for fault detection and a fuzzy expert system is used for fault isolation. This monitor is evaluated on a simulated autonomous underwater vehicle (AUV).

5. Discussion

This work is an overview of existing execution monitoring systems applied to robotics, particularly mobile robotics. From the field of industrial control we have learned that execution monitoring can be categorized into one or several of three classes [11]: analytical, data-driven, and knowledge-based. This categorization has been used throughout the survey of the different monitoring systems.

Analytical approaches rely on the concept analytical redundancy. This means that two analytically generated quantities, obtained from different sets of variables are compared. These quantities are generated with the help of models, that are often basic models from physics, that describe the system's behavior. The analytical approaches are the most frequently used approaches for fault detection in robotics, including both robot manipulators and mobile robotics. Analytical monitoring can be divided into: parameter estimation, parity relations, and observers, where the latter is the most common approach for analytical monitoring. Analytical approaches are preferable when the monitored system is well understood and the amount of uncertainty is limited. Many parts within a robotic system are designed with the help of basic models and this knowledge might as well be used when designing the monitoring system. For example, the analytical approaches are well suited to detect faults in a wheel motor controller, such as a broken driving shaft, or a broken wheel encoder. The main weakness of this approach is the assumption that a precise mathematical model of the system is available. In many complex systems it is difficult, or even impossible to create such models. This concerns, for example, behavior-based controllers where the combination of different reactive behaviors may produce an overall behavior, sometimes called an *emergent behavior*, that is very difficult to model.

Data-driven approaches do not rely on analytical models. Instead, the information used for monitoring is derived directly from input data. The decision making is often based on statistical methods. The application of statistical theory to execution monitoring relies on the assumption that the characteristics of the data variations are relatively unchanged unless a fault occurs in the system. Data-driven monitoring can be divided into two groups referring to the number of variables measured by the monitor, that is, univariate and multivariate statistical monitoring. The main strengths of the data-driven approaches are their robustness to uncertainty, and their ability to transform high-dimensional data into a lower dimension space; in which the most important information is captured. Therefore, these approaches are well suited when several variables are

measured and their connection to the system behavior is not fully understood. An example of a fault that could be detected and isolated by the data-driven approach is a low battery charge level that causes increased noise in the sonar readings. The main drawback using this approach is that the performance is highly dependent on the amount and quality of the input data. If sufficient data is not available to accurately estimate the mean vector and covariance matrix, this will result in suboptimal fault detection and isolation.

The knowledge-based approaches are not distinct from the former two approaches since both analytical and data-driven approaches apply to this class. Knowledge-based monitoring can be divided into three categories: causal analysis, expert systems, and artificial neural networks. The main strength using the knowledge-based approaches is probably their ability to combine analytical and data-driven approaches in a hybrid monitoring system. In other words, these approaches are well suited to monitor complex systems, at several levels of abstraction, such as autonomous mobile robotics.

A common problem within the mobile robotics field is that many ideas rely heavily on a specific system. Therefore, it is hard to compare the performance of different approaches. Even though some researchers have raised the issue and claimed the need for common domains or benchmark systems, see for example [55], very few have prioritized this need and tried to create them. Nevertheless, if the same quantifiable criteria were used when evaluating the performance of a monitoring system, a comparison between different ideas would be easier. Gertler suggests that the following criteria could be measured when evaluating execution monitoring [29]: *reaction speed*, that is the ability to detect faults with reasonably small delays after their arrivals; *robustness*, that is the ability to operate in the presence of noise, disturbances and modeling errors, with few false alarms; and *isolation performance*, that is the ability to distinguish between faults.

To conclude, we claim that observer-based monitoring is by far the most common approach within robotics. When several sequential states are modeled, causal analysis, often realized by a directed graph or a state machine, is the most common approach. Only in the cases when it is very hard to develop an adequate model of the system, a data-driven approach is tested. The following example confirms our claim. In the work on monitoring in the New Millennium project by NASA, a model-based approach was developed. This observer-based monitor performed very well when applied to an autonomous space shuttle. The different states of a space shuttle are well known and the number of unexpected events limited. When the same monitor later on was evaluated on an autonomous rover, the performance was not overwhelming. Instead, a multivariate statistical approach was introduced to cope with all the uncertainties involved in the task of moving across the surface of a planet [78].

One explanation of why analytical monitoring is the most common approach could be the superior understanding of the underlying concepts. An analytical model created from first principles is easier to maintain, compared to black box approaches where the information is hidden. Note, that this applies only for people that have knowledge in the language that is used to describe the models. The robot user do not necessarily have knowledge in the formal language that is required for model compilation. Although most monitoring systems applied to robotics are based on analytical redundancy, the other approaches have some complementary advantages. Therefore, data-driven approaches together with knowledge-based approaches need to be further studied.

Acknowledgements

This work was partly funded by the Knowledge Foundation in Sweden. The author would also like to thank Alessandro Saffiotti and Lars Karlsson for support and numerous discussions.

References

[1] M.G. Abu-Hamdan, A.S. El-Gizawy, Computer aided monitoring system for flexible assembly operations, Comput. Ind. 34 (1997) 1–10.

[2] R. Alami, M. Herrb, B. Morisset, R. Chatila, F. Ingrand, P. Moutarlier, S. Fleury, M. Khatib, T. Simeon, Around the lab in 40 days, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), San Francisco, CA, USA, 2000, pp. 88–94.

[3] R.V. Beard, Failure Accommodation in Linear Systems Through Self Reorganization. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1971.

[4] P. Bergsten, Observers and Controllers for Takagi-Sugeno Fuzzy Systems. Ph.D. thesis, Örebro University, Department of Technology, Örebro, Sweden, 2001.
[5] M. Bjäreland. Model-Based Execution Monitoring. Ph.D. Thesis, Linköping University, Department of Computer and Information Science, Linköping, Sweden, 2001.
[6] C. Boutilier, T. Dean, S. Hanks, Decision-theoretic planning: structural assumptions and computational leverage, J. Artif. Intell. Res. 11 (1999) 1–94.
[7] M.E. Bratman, Intention, Plans, and Practical Reason, Harvard University Press, Cambridge, MA, USA, 1987.
[8] B. Browning, G.A. Kaminka, M.M. Veloso, Principled monitoring of distributed agents for detection of coordination failures, in: Proceedings of the International Conference on Distributed Autonomous Robotic Systems (DARS), Fukuoka, Japan, 2002.
[9] F. Caccavale, I.D. Walker, Observer-based fault detection for robot manipulators, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Albuquerque, NM, USA, 1997, pp. 2881–2887.
[10] J. Chen, R.J. Patton, Robust Model-Based Fault Diagnosis for Dynamic Systems, Kluwer Academic Publishers, Boston, MA, USA, 1999.
[11] L.H. Chiang, E.L. Russell, R.D. Braatz, Fault Detection and Diagnosis in Industrial Systems, Springer-Verlag, London, UK, 2001.
[12] M. de la Sen, J.J. Minambres, A.J. Garrido, A. Almansa, J.S. Soto, Basic theoretical results for expert systems: application to the supervision of adaptation transients in planar robots, Artif. Intell. 152 (2) (2004) 173–211.
[13] T. Dean, M.P. Wellman, Planning and Control, Morgan Kaufmann Publishers, San Mateo, CA, USA, 1991.
[14] R. Dearden, F. Hutter, R.G. Simmons, S. Thrun, V. Verma, T. Willeke, Real-time fault detection and situational awareness for rovers: report on the mars technology program task, in: Proceedings of the IEEE Aerospace Conference, Big Sky, MT, USA, 2004, pp. 826–840.
[15] W.E. Dixon, I.D. Walker, D.M. Dawson, J.P. Hartranft, Fault detection for robotic manipulators with parametric uncertainty: a prediction error based approach, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), San Francisco, CA, USA, 2000, pp. 3628–3634.
[16] G.A. Dorais, K. Nicewarner, Adjustably autonomous multiagent plan execution with an internal spacecraft free-flying robot prototype, in: Proceedings of the ICAPS Workshop on Plan Execution, Trento, Italy, 2003.
[17] J.L. Fernández, R.G. Simmons, Robust execution monitoring for navigation plans, in: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), Victoria, BC, Canada, 1998, pp. 551–557.
[18] M. Fichtner, A. Großmann, M. Thielscher, Intelligent execution monitoring in dynamic environments, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Acapulco, Mexico, 2003.
[19] R.E. Fikes, Monitored execution of robot plans produced by STRIPS, in: Proceedings of the IFIP Congress 71, Ljublijana, Yugoslavia, 1971, pp. 189–194.
[20] R.E. Fikes, N.J. Nilsson, STRIPS: a new approach to the application of theorem proving to problem solving, Artificial Intelligence 2 (3–4) (1971) 189–208.
[21] R.J. Firby, An investigation into reactive planning in complex domains, in: Proceedings of the National Conference on Artificial Intelligence (AAAI), Seattle, WA, USA, 1987, pp. 202–206.
[22] R.J. Firby, Task networks for controlling continuous processes, in: Proceedings of the International Conference on Artificial Intelligence Planning Systems (AIPS), Chicago, IL, USA, 1994, pp. 49–54.
[23] R.J. Firby, P.N. Prokopowicz, M.J. Swain, The animate agent architecture, in: R.P. Bonasso, D. Kortenkamp, R. Murphy (Eds.), Artificial Intelligence and Mobile Robots, MIT Press, 1998, pp. 243–275.
[24] P.M. Frank, Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy—a survey and some new results, Automatica 26 (3) (1990) 459–474.
[25] E. Gat, M.G. Slack, D.P. Miller, R.J. Firby, Path planning and execution monitoring for a planetary rover, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Cincinnati, OH, USA, 1990, pp. 20–25.
[26] A. Gelb, Applied Optimal Estimation, MIT Press, Cambridge, MA, USA, 1974.
[27] M.P. Georgeff, A.L. Lansky, Reactive reasoning and planning, in: Proceedings of the National Conference on Artificial Intelligence (AAAI), Seattle, WA, USA, 1987, pp. 677–682.
[28] J.J. Gertler, Diagnosing parameter faults: from parameter estimation to parity relations, in: Proceedings of the American Control Conference, Evanston, IL, USA, 1995, pp. 1615–1620.
[29] J.J. Gertler, Fault Detection and Diagnosis in Engineering Systems, Marcel Dekker, New York, NY, USA, 1998.
[30] P. Goel, G. Dedeoglu, S.I. Roumeliotis, G.S. Sukhatme, Fault detection and identification in a mobile robot using multiple model estimation and neural network, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), San Francisco, CA, USA, 2000, pp. 2302–2309.
[31] J. Gout, S. Fleury, H. Schindler, A new design approach of software architecture for an autonomous observation satellite, in: Proceedings of the Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space (SAIRAS'99), Noordwijk, The Netherlands, 1999.
[32] K. Hamilton, D. Lane, N. Taylor, K. Brown, Fault diagnosis on autonomous robotic vehicles with recovery: an integrated heterogeneous-knowledge approach, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Seoul, Korea, 2001, pp. 3232–3237.
[33] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice-Hall, Upper Saddle River, NJ, USA, 1999.
[34] J. Hertzberg, H. Jaeger, F. Schönherr, Learning to ground fact symbols in behavior-based robots, in: Proceedings of the European Conference on Artificial Intelligence, Amsterdam, Netherlands, 2002, pp. 708–712.
[35] J. Hertzberg, H. Jaeger, U. Zimmer, P. Morignot, A framework for plan execution in behavior-based robots, in: Pro-

ceedings of the IEEE International Symposium on Intelligent Control (ISIC), New York, NY, USA, 1998, pp. 8–13.
[36] B.N. Huallpa, E. Nóbrega, F.J. von Zuben, Fault detection in dynamic systems based on fuzzy diagnosis, in: Proceedings of the IEEE International Conference on Fuzzy Systems, New York, NY, USA, 1998, pp. 1482–1487.
[37] F. Ingrand, O. Despouys, Extending procedural reasoning toward robot actions planning, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Seoul, Korea, 2001, pp. 9–14.
[38] R. Isermann, Process fault diagnosis based on modeling and estimation methods—a survey, Automatica 20 (4) (1984) 387–404.
[39] R. Isermann, Estimation of physical parameters for dynamic processes with application to an industrial robot, in: Proceedings of the American Control Conference, Green Valley, AZ, USA, 1990, pp. 1396–1401.
[40] R. Isermann, Supervision, fault-detection and fault-diagnosis methods: an introduction, Control Eng. Pract. 5 (5) (1997) 639–652.
[41] R. Isermann, P. Ballé, Trends in the application of model-based fault detection and diagnosis of technical processes, Control Eng. Pract. 5 (5) (1997) 709–719.
[42] P. Jackson, Introduction to Expert Systems, Addison-Wesley, Reading, MA, USA, 1998.
[43] G.A. Kaminka, D.V. Pynadath, M. Tambe, Monitoring deployed agent teams, in: Proceedings of the International Conference on Autonomous Agents (Agent), Montreal, QC, Canada, 2001, pp. 308–315.
[44] G.A. Kaminka, D.V. Pynadath, M. Tambe, Monitoring teams by overhearing: a multi-agent plan-recognition approach, J. Artif. Intell. Res. 17 (2002) 83–135.
[45] K. Kawabata, T. Akamatsu, H. Asama, A study of self-diagnosis system of an autonomous mobile robot: expansion of state sensory system, in: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), Lausanne, Switzerland, 2002, pp. 1802–1807.
[46] P. Krause, D. Clark, Representing Uncertain Knowledge, Intellect Books, Oxford, UK, 1993.
[47] J. Kurien, P.P. Nayak, Back to the future for consistency-based trajectory tracking, in: Proceedings of the National Conference on Artificial Intelligence (AAAI), Austin, TX, USA, 2000, pp. 370–377.
[48] K.B. Lamine, F. Kabanza, History checking of temporal fuzzy logic formulas for monitoring behavior-based mobile robots, in: Proceedings of the IEEE International Conference on Tools with Artificial Intelligence, Los Alamitos, CA, USA, 2000, pp. 312–319.
[49] S. Lemai, F. Ingrand, Interleaving temporal planning and execution: IxTeT-eXeC, in: Proceedings of the ICAPS Workshop on Plan Execution, Trento, Italy, 2003.
[50] L. Ljung, System Identification: Theory for the User, Prentice-Hall, Englewood Cliffs, NJ, USA, 1987.
[51] C.E. McCarthy, M.E. Pollack, Towards focused plan monitoring: a technique and an application to mobile robots, Auton. Robot. 9 (1) (2000) 71–81.
[52] D.C. Montgomery, Introduction to Statistical Quality Control, John Wiley and Sons, New York, NY, USA, 1985.
[53] E.M. Nebot, H. Durrant-Whyte, S. Scheding, Frequency domain modeling of aided GPS for vehicle navigation systems, Robot. Auton. Syst. 25 (1–2) (1998) 73–82.
[54] O. Nelles, R. Isermann, Identification of nonlinear dynamic systems—classical methods versus radial basis function networks, in: Proceedings of the American Control Conference, Seattle, WA, USA, 1995, pp. 3786–3790.
[55] I. Noda, H. Matsubara, K. Hiraki, I. Frank, Soccer server: a tool for research on multiagent systems, Applied Artificial Intelligence 12 (2–3) (1998) 233–250.
[56] F.R. Noreils, R.G. Chatila, Plan execution monitoring and control architecture for mobile robots, IEEE Trans. Robot. Autom. 11 (2) (1995) 255–266.
[57] S. Parsons, O. Pettersson, A. Saffiotti, M. Wooldridge, Robots with the best of intentions. in: M. Wooldridge, M.M. Veloso (Eds.), Artificial Intelligence Today: Recent Trends and Developments, number 1600 in Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin, Germany, 1999, pp. 329–338.
[58] S. Parsons, O. Pettersson, A. Saffiotti, M. Wooldridge, Intention reconsideration in theory and practice, in: Proceedings of the European Conference on Artificial Intelligence, Berlin, Germany, 2000, pp. 378–382.
[59] R.J. Patton, C.J. Lopez-Toribio, F.J. Uppal, Artificial intelligence approaches to fault diagnosis for dynamic systems, Int. J. Appl. Math. Comput. Sci. 9 (3) (1999) 471–518.
[60] O. Pettersson, Model-Free Execution Monitoring in Behavior-Based Mobile Robotics. Ph.D. thesis, Örebro University, Department of Technology, Örebro, Sweden, 2004.
[61] O. Pettersson, L. Karlsson, A. Saffiotti, Steps towards model-free execution monitoring on mobile robots, in: Proceedings of the Swedish Workshop on Autonomous Robots (SWAR 02), Stockholm, Sweden, 2002, pp. 45–52.
[62] O. Pettersson, L. Karlsson, A. Saffiotti, Model-free execution monitoring in behavior-based mobile robotics, in: Proceedings of the International Conference on Advanced Robotics (ICAR), Coimbra, Portugal, 2003, pp. 864–869.
[63] O. Pettersson, L. Karlsson, A. Saffiotti, Model-free execution monitoring by learning from simulation, in: Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation (CIRA), Espoo, Finland, 2005.
[64] N. Ranganathan, M.I. Patel, R. Sathyamurthy, An intelligent system for failure detection and control in an autonomous underwater vehicle, IEEE Trans. Syst. Man Cybern. 31 (6) (2001) 762–767.
[65] S.M. Ross, Introduction to Probability and Statistics for Engineers and Scientists, Academic Press, San Diego, CA, USA, 2000.
[66] S.I. Roumeliotis, G.S. Sukhatme, G.A. Bekey, Sensor fault detection and identification in a mobile robot, in: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), Victoria, BC, Canada, 1998, pp. 1383–1387.
[67] A. Saffiotti, Handling uncertainty in control of autonomous robots. in: M. Wooldridge, M.M. Veloso (Eds.), Artificial Intelligence Today: Recent Trends and Developments, Lecture

Notes in Artificial Intelligence, Springer-Verlag, Berlin, Germany, 1999, pp. 381–408.
[68] A. Saffiotti, K. Konolige, E.H. Ruspini, A multivalued-logic approach to integrating planning and control, Artif. Intell. 76 (1–2) (1995) 481–526.
[69] S. Scheding, E. Nebot, H. Durrant-Whyte, The detection of faults in navigation systems: a frequency domain approach, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Leuven, Belgium, 1998, pp. 2217–2222.
[70] H. Schneider, P.M. Frank, Observer-based supervision and fault detection in robots using nonlinear fuzzy logic residual evaluation, IEEE Trans. Control Syst. Technol. 4 (3) (1996) 274–282.
[71] F. Schönherr, M. Cistelecan, J. Hertzberg, T. Christaller, Extracting situation facts from activation value histories in behavior-based robotics. in: F. Baader, G. Brewka, T. Eiter (Eds.), KI 2001: Proceedings of the Joint German/Austrian Conference on AI, number 2174 in Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin, Germany, 2001, pp. 305–319.
[72] F. Schönherr, J. Hertzberg, The DD&P robot control architecture: a preliminary report. in: M. Beetz, J. Hertzberg, M. Ghallab, M. Pollack (Eds.), Advances in Plan-Based Control of Robotic Agents, number 2466 in Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin, Germany, 2002, pp. 249–269
[73] R.G. Simmons, D. Apfelbaum, A task description language for robot control, in: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), Victoria, BC, Canada, 1998, pp. 1931–1937.
[74] R.G. Simmons, J. Fernandez, R. Goodwin, S. Koenig, J. O'Sullivan, Lessons learned from Xavier, IEEE Robot. Autom. Mag. 7 (2) (2000) 33–39.
[75] M. Soika, Sensor failure detection framework for autonomous mobile robots, in: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), Grenoble, France, 1997, pp. 1735–1740.
[76] M.M. Veloso, J. Carbonell, M.A. Pérez, D. Borrajo, E. Fink, J. Blythe, Integrating planning and learning: the PRODIGY architecture, J. Exp. Theor. Artif. Intell. 7 (1) (1995) 81–120.
[77] M.M. Veloso, M.E. Pollack, M.T. Cox, Rationale-based monitoring for planning in dynamic environments, in: Proceedings of the International Conference on Artificial Intelligence Planning Systems (AIPS), Pittsburgh, PA, USA, 1998, pp. 171–179.
[78] V. Verma, G. Gordon, R.G. Simmons, S. Thrun, Real-time fault diagnosis, IEEE Robot. Autom. Mag. 11 (2) (2004) 56–66.
[79] R. Washington, On-board real-time state and fault identification for rovers, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), San Francisco, CA, USA, 2000, pp. 1175–1181.
[80] D.E. Wilkins, T.J. Lee, P. Berry, Interactive execution monitoring of agent teams, J. Artif. Intell. Res. 18 (2003) 217–261.
[81] B.C. Williams, P.P. Nayak, A model-based approach to reactive self-configuring systems, in: Proceedings of the National Conference on Artificial Intelligence (AAAI), Cambridge, MA, USA, 1996, pp. 971–978.
[82] A.S. Willsky, A survey of design methods for failure detection in dynamic systems, Automatica 12 (6) (1976) 601–611.
[83] R.R. Yager, On ordered weighted averaging aggregation operators in multicriteria decisionmaking, IEEE Trans. Syst. Man Cybern. 18 (1) (1988) 183–190.
[84] B. Yan, T. Zhang, C. Xie, Fuzzy expert system for fault diagnosis of robotic, in: Proceedings of the World Congress on Intelligent Control and Automation, 2002, pp. 445–449.
[85] H. Ye, P. Zhang, S.X. Ding, G.Z. Wang, A time-frequency domain fault detection approach based on parity relation and wavelet transform, in: Proceedings of the IEEE International Conference on Decision and Control, Sydney, Australia, 2000, pp. 4156–4161.
[86] L.A. Zadeh, Fuzzy sets as a basis for a theory of possibility, Fuzzy Set. Syst. 1 (1) (1978) 3–28.
[87] J. Zhang, Improved on-line process fault diagnosis using stacked neural networks, in: Proceedings of the IEEE International Conference on Control Applications, Glasgow, UK, 2002, pp. 689–694.
[88] J. Zhang, M. Xu, S. Xu, Fault detection and classification through multivariate statistical techniques, in: Proceedings of the American Control Conference, Evanston, IL, USA, 1995, pp. 751–755.
[89] K. Zita Haigh, M.M. Veloso, Interleaving planning and robot execution for asynchronous user requests, in: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), Osaka, Japan, 1996, pp. 148–155.
[90] K. Zita Haigh, M.M. Veloso, Planning, execution and learning in a robotic agent, in: Proceedings of the International Conference on Artificial Intelligence Planning Systems (AIPS), Pittsburgh, PA, USA, 1998, pp. 120–127.

Ola Pettersson was born in Falkenberg, Sweden, in 1972. He received his MS degree in Electrical Engineering (specialization in photonics and image analysis) from the Halmstad University, Sweden, in 1995. In 2004 he received his PhD degree in Computer Science from the Örebro University, Sweden. His research interests include autonomous mobile robotics, execution monitoring, and pattern recognition.