2011 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 25-30, 2011. San Francisco, CA, USA

Online Data-Driven Fault Detection for Robotic Systems

Raphael Golombek, Sebastian Wrede, Marc Hanheide, and Martin Heckmann

***Abstract*— In this paper we demonstrate the online applicability of the fault detection and diagnosis approach which we previously developed and published in [1]. In our former work we showed that a purely data driven fault detection approach can be successfully built based on monitored inter-component communication data of a robotic system and used for a-posteriori fault detection. Here we propose an extension to this approach which is capable of online learning of the fault model as well as for online fault detection. We evaluate the application of our approach in the context of a RoboCup task executed by our service robot BIRON in corporation with an expert user.**

I. INTRODUCTION

Today's autonomous robotic systems are capable of performing a multitude of socially relevant tasks in barely unconstrained scenarios and close interaction with humans [2], [3], [4]. This increased performance leads to an enormous intrinsic complexity which is reflected in a vast number of sensors, actuators, dedicated software components and processes (e.g., object detector, speech recognizer, etc.), behaviours and interaction patterns. This development leads inevitably to a higher probability of faults in the system. Today, one of the most critical issues when developing robotic systems for close human-robot interaction is to guarantee safe robot behaviour at any time of operation.

A traditional way to improve dependability and reliability is to enhance the quality, reliability and robustness of individual hardware/software components and use redundancy to cope with faults. Yet, there exist at least two reasons why this concept does not guarantee a fault-free system performance. Firstly, it is simply not possible to consider all rationale which lead to a fault of a component in advance. Secondly, improving the aforementioned features of components does not tackle faults emerging from the interplay of fault-free components.Accordingly, state of the art robotic systems demand for monitoring and fault diagnosis in order to react upon faults and thus meet safety requirements.

In previous work we proposed a data-driven fault diagnosis method which learns a model of normal behaviour from temporal correlations in data communicated between components of a robotic system and utilizes the model to detect faulty situations [1]. This approach is based on the hypothesis that the communication and interaction between components contains structured information which can be exploited for fault detection. Compared with state of the art fault detection and diagnosis systems e.g. model-based techniques [5] our approach does not require to model normal and abnormal states beforehand thus freeing the developer from the burden of foreseeing each possible exceptional situation in order to be able to diagnose the system. Furthermore, our approach abstracts from system specific characteristics before training a fault model and is therefore applicable to a wide range of systems. We demonstrated the application of our approach on our service robot *BIRON* depicted in figure 1. We conducted experiments in the context of the RoboCup task *Follow Me* and demonstrated the fault detection performance of our approach for faults originating from external sources such as the environment itself or system failures such as resource starvation. However, the approach proposed in our previous work was exploited for off-line fault detection. Consequently, in this paper we propose an extension to this approach which enables the detection of faults in an online manner.

Fig. 1. BIRON performing at the Robocup@Home competition 2010.

Our contribution is organized as follows: Section II reviews related work on models and algorithms suitable for fault detection and diagnosis. Section III introduces our fault detection approach with a particular focus on the necessary extensions for online fault detection. Following up, Section IV describes experiments we conducted on

R. Golombek and S. Wrede are with the Research Institute for Cognition and Robotics, Bielefeld University, P.O. Box 100131, Bielefeld, Germany [rgolombe,swrede]@cor-lab.uni-bielefeld.de

M. Hanheide is with the School of Computer Science, University of Birmingham, UK m.hanheide@cs.bham.ac.uk

Martin Heckmann is with the Honda Research Institute Europe, Offenbach, Germany Martin.Heckmann@Honda-RI.de

978-1-61284-456-5/11/$26.00 ©2011 IEEE

BIRON while the robot performs a task defined in the RoboCup@Home competition, and discusses the results of the experiments. The evaluation is carried out based on various faults covering situations which are typical when working with robotic systems. Finally, Section V summarizes our contribution and provides an outlook on further challenges.

II. RELATED WORK

Faults are bound to occur in a complex system involving software, hardware and interaction with the environment. Thus, the capability to detect, diagnose and recover from faults is essential for a robot that displays dependable long-term behaviour.

Existing work on online fault detection and diagnosis (FDD) can be divided into two main approaches: *data-driven* and *model-based*. Model-based approaches follow either an analytical or knowledge-based method, they model each state beforehand and use this model to estimate the current system state (i.e., normal or a specific fault). Analytical approaches are used in the design of control systems where the models are constructed based on fundamental assumptions [6]. They are precise and mostly targeted at problems fairly close to the hardware as well as to the raw sensor data. However, up to now designing such a model is a tedious task and therefore mostly impractical for fault detection in complex cognitive systems. Attempts to combine these technique with data-driven diagnosis and thus improve design capabilities were made in [7]. In the knowledge-based domain, qualitative models of the system are used to detect and diagnose faults. One of the most successful approaches for fault identification and recovery has been consistency-based diagnosis [8]. This work has lead to the development of several model-based diagnosis systems, e.g. Livingstone [9], [5], Hyde [10], and Lydia [11]. A drawback of Livingstone, a widely deployed diagnosis engine [12], [13], [14], is that it does not support numeric representations of variables. A mean to overcome the limitation is to enhance the approach based on particle filters [15]. This yields a powerful technique for detecting faults in hybrid systems and has been already used for fault detection in Mars rovers and waiter robots [16]. Model-based approaches are powerful tools to detect and identify faults in the modelled system. However, they rely on expert knowledge to design the model (i.e., normal state boundaries and faulty configurations) and do not explicitly address unknown fault detection. In [17] model-based reasoning is used to detect faults in robot control software. Here, well understood reasoning techniques are used to build a precise model of the system, identify, and track faults with high accuracy. This approach also suffers from a time-intensive modelling phase.

In the data-driven approach, machine learning methods are used to reduce the dimensionality of vast amounts of log data generated by an appropriately instrumented system. Methods like Fischer discriminant analysis and canonical variate analysis have been used in several industrial scenarios [18] and proved to work on problems with relatively low number of discrete states with different dynamics. For systems with larger numbers of discrete states exist classification based approaches [19] assuming that it is possible to cluster the feature space in order to separate classes. Here, a pre-computed model is used resulting in a fast detection rate. Nearest Neighbour based approaches [20] assign data to a particular state of the system (normal or fault) based on a distance measure. Both techniques suffer from uncertainty and noise in the training data. Therefore, stochastic approaches based on different techniques like principal component analysis or partial least squares have been developed [18]. Further approaches based on Markov Chains [21] and Hidden Markov Models [22] have been proposed that capture structural information between data points ordered in time. Probabilistic approaches rest upon the concept that the system states can be represented with the help of probability distributions and the current system state can be determined by calculating the highest probability for a data point currently obtained from the system. Besides the inferred decision they provide a scoring of the states which can be interpreted as the confidence of the decision. However, a drawback of data-driven approaches is that they depends on the availability of training data in order to build models for normal states and faults.

Our recently developed fault detection and diagnosis approach belongs to the domain of probabilistic data-driven techniques. In contrast to aforementioned methods we do not explicitly assume the Markov Property in the time-series of data points gathered as training-data from the system in question. This makes the approach more flexible when monitoring a complex system which consist of several hidden processing chains. In the subsequent chapter we introduce our approach as proposed in [1]. Additionally, we explain necessary extensions in order to enable online application.

III. PROBABILISTIC FAULT DETECTION AND DIAGNOSIS MODEL

Our fault detection model is based on the following hypothesis: *We consider a robotic system as a set of functional components communicating with each other to fulfil a given task thereby generating temporal communication patterns. These pattern contain structured information which differ depending on the current state of the system (e.g., normal or faulty behaviour).*

We exploit this hypothesis and train a probabilistic model from the communication patterns representing normal behaviour of the system. While the robot is in field the model is populated with recorded communication time-series to determine whether the system is currently operating according the behaviour defined by the developer. The contribution of the present work in relation to the approach previously presented in [1] is the application of the approach in an online modus.

In the remainder of this section we explain i) how to build the fault detection model from a time-series recorded during normal behaviour, ii) how to asses whether a time-series belongs to this model and iii) how to perform fault detection in an online modus.

A. Building a Probabilistic Model

a) Encode Data: The components of a complex system communicate with each other thereby exchanging data from various domain spaces (e.g., numeric or symbolic). Thus, a record of component communication data in such a system is a time-series of heterogeneous data points gathered from different sources. In order to enable the application of our approach to varying system configurations and to different robotic systems in general we first build an abstract representation of this heterogeneous time-series. Therefore, we define a transformation function f which maps single data elements of the time-series to a common domain space:

$$f(d_j) : \mathcal{D}_j \mapsto \mathcal{E} := e \tag{1}$$

where d_j denotes an element of the sequence with the specific domain space $\mathcal{D}_j$ and $\mathcal{E}$ denotes the common domain space. We call $\mathcal{E}$ the event space and the representation e of the data entity d_j in $\mathcal{E}$ an event. It should be noted here that the algorithm always operates on the whole communication between the components of the system. In other words: there is no particular pre-selection of data before applying our method which simplifies the usage. We illustrate this idea by means of the transformation function used during the evaluation in section IV:

$$f(d_j) : \mathcal{D}_j \mapsto SOURCE \times SCOPE \times TYPE \tag{2}$$

This function is based on concepts of the event-driven architecture [23] exploited in our evaluation platform BIRON [2]. It maps each communicated data point d_j to a triple $e \in \mathcal{E}$ consisting of the information about the producing component of the data point, the scope where this data point can be perceived by other components and a generic type of the payload. The type takes one of the following values: *insert* (new data), *delete* (data becomes invalid) or *update* (content changes). This definition of f is very general and allows us to apply our approach directly to systems built upon other architectures at least sharing a data-driven or event-based robotics middle-ware concept, e.g. ROS [24], Yarp [25] or OROCOS [26]. However, more sophisticated mappings which incorporate further information (e.g., metrics on image or sound payload) are possible, too.

b) Training The Probabilistic Model: Let E_{train} be an encoded time-series of component communication data recorded during a specific system behaviour (e.g., normal system behaviour). Let $U_E = \text{Set}(E_{train})$ be the unique set of all events in E_{train}. For each tuple $(e_i, e_j) \in U_E \times U_E$ a distribution $P_{(i,j)} := P(t|e_i, e_j)$ is derived from E_{train}. $P_{(i,j)}$ describes the probability that the event e_i occurs at time stamp t_i and that the event e_j follows this event with a delay of t, respectively occurring at time stamp $t_j = t_i + t$. Additionally, we constrain e_i to be the last seen occurrence of this type of event as we want to model temporal correlations only between the current event and the last occurrence of a given event. The computation of $P_{(i,j)}$ works as follows: Each time an event e_j is discovered in the sequence of incoming time-series the timespan between the point in time t_i at which e_i has been seen last and the time stamp $t(e := e_j)$ of the just discovered event e_j is computed and $P_{i,j}$ is updated with this information. This is done for all other probability distributions $\{P_{(k,j)}|k \in U_E\}$, too. To model the distribution $P_{i,j}$ we use a Kernel Density Estimator (KDE) [27] parametrized with a Gaussian Kernel $K(u) = \frac{1}{2\pi} e^{-\frac{1}{2}u^2}$. Although the Kernel Density Estimator is not perfectly suited to model durations we benefit from its capabilities to cope with the (potentially) multi-modal nature of the data which is justified by the diverse and dynamic behaviour of the system. Finally, based on all distributions $P_{i,j}$, the model is defined as

$$M = \{P_{i,j}|(e_i, e_j) \in U_E \times U_E\}$$

i.e., the set of probabilities for all tuples in $U_E \times U_E$.

B. Online Fitness Estimate

While the system is in field we monitor the communication between the components and asses the fitness of the system. Thereby, a high fitness value computed for the currently monitored communication indicates a high probability that the system's behaviours fits the previously learned model. Assessing the system's fitness corresponds to calculating a *fitness value* for an event type e_j encoded from a communicated data d_j as explained earlier. We call the fitness value of e_j the *score* and denote it with s_j. Calculating s_j for an event e_j which occurred at the time stamp t comprises querying the model M to get all probabilities $p_{i,j} = P_{i,j}(\Delta t_i)$ and fusing them to a single score s_j. This again requires that the timestamps of the last occurrences of each event e_i are tracked to calculate the corresponding time spans Δt_i. The score value s_j is defined as:

$$s_j = \sum_{e_i \ \in \ E} w_{i,j} \cdot p_{i,j}. \tag{3}$$

Where E denotes the set of events tracked over time. The weight $w_{i,j}$ is defined as:

$$w_{i,j} = 1 - \frac{h_{i,j}}{H_j}. \tag{4}$$

$h_{i,j}$ denotes the entropy of the distribution $P_{i,j}$ and $H_j = \sum_{e_i}^{E} h_{i,j}$ is the sum of all entropy values of the distributions $P_{i,j}$ which have to be considered when calculating the score of the event e_j. The reason to weight each $P_{i,j}$ with its entropy value $h_{i,j}$ is that it provides valuable information about the correlation of the two corresponding events at low computational costs. A high entropy value indicates high uncertainty in the distribution and consequently low correlation.

C. Online Fault Detection

After online calculating the score value s_j for e_j based on the model M we decide whether the current system state during which e_j has been emitted is equal to the system state represented by the learned model M (i.e., whether the system behaves normal or not). This is done by a binary classifier:

$$\text{abnormal}(e_j) = \begin{cases} \text{True} & : s_j < s^* \\ \text{False} & : \text{else} \end{cases} \tag{5}$$

If s_j of e_j is higher then the threshold, then e_j (and hence the current system state) is declared as normal. Otherwise abnormality is assumed. The threshold s^* is calculated based on the following formula:

$$s^* = a \cdot s^*_{val} + b \cdot s^*_{val} \cdot \frac{S_{var}}{s^*_{var}} \tag{6}$$

where $a+b=1$. The idea behind this formula is that besides a constant threshold value s^*_{val} for the score, that has to be reached in order to classify the system as healthy, we assume that the variance S_{var} of consecutive scores $S = (s_{j,1}, \cdots, s_{j,n-1}, s_j)$ is lower for events belonging to the *normal* model M than for events which do not belong to M. If the variance of the history values is lower, we reduce the threshold value s^* which then can be then exceeded easier. Otherwise the s^* increases which makes it harder to surpass it. To realize this we add additional parameters s^*_{var}, $k = |S_{var}|$ and $a + b = 1$ to the classifier to incorporate the variance of a history of scores S into the threshold s^*. The weights a and b are used to balance the impact of the constant fraction of the threshold and the variance based one.

Next, we describe our evaluation. First, we explain the experimental set-up before discussing the results.

IV. EVALUATION

A. EXPERIMENTS

We conduct the experiments on BIRON [2] our human robot interaction platform. We evaluate the online capabilities in the context of the RoboCup task *Follow Me* executed by an expert in our laboratory. During this task the human partner uses commands to advice the robot to follow him, to stop, and either turn left or right. In return the robot is allowed to ask questions (e.g. do you want me to follow you?) if it is unsure which action to perform.

Each evaluation run consists of three phases, all performed online during the *Follow Me* task. In phase one data from three minutes of interaction is recorded and used to train the fault detection model. It should be noted here that no prior analysis of the inter-component communication of a *to be monitored*-system is necessary in order to generate the necessary mapping of communicated data $\mathcal{D}$ into the event space $\mathcal{E}$. Our algorithm is able to gather and encode the communicated data during the online-training phase.

Next, an additional time-series of 30 second length is recorded and used to optimize the parameter of the statistical classifier. Finally, we monitor the system during normal interaction and induce the *to be detected* fault. The monitoring part of an evaluation run is defined as follows: First we record 30 seconds of normal interaction. Afterwards, we induce the fault at a random point in time. By this means we always record a fixed time period of normal behaviour of the system which we use to calculate the false alarm rate. We monitor the system after inducing and detecting the fault for 30 seconds. In case that our fault detector does not report anything within 30 seconds after inducing the fault we stop the experiment. In this case we declare the run as failed.

To analyse the performance of our fault detection approach we measure i) the system's fault detection rate (FDR), ii) the tracking rate after fault detection (FTR), iii) the false alarm rate (FAR), iv) the seriousness of the false alarm i.e., the percentage of individual data false alarms in relation to all events recorded during normal behaviour(SFAR) and v) the delay between inducing the fault and it's reporting by the detector. Figure 2 visualizes the different measures along an exemplary time-series of scores for a recorded data.

First we induce a crash of the navigation component (NAV) which disables the robot's mobility. This simple form of a fault still occur sometimes and might lead to critical situations, especially in case of sophisticated components like the navigation. Next, we disturb the simultaneous localization and mapping component (SLAM) leading to errors in components relying on global planning. Furthermore, we introduce a dysfunction in the speech recognition sub-system (SR) which widely influences the human robot interaction. The nature of this dysfunction is the common case of inappropriate parameter settings not suited for the current environment. Finally, we simulate a resource starvation (RS) fault caused by a component which greedily occupies system resources having unforeseen impact on the system as a whole. Resource starvation faults may occur fairly often when integrating components developed by various research into a common robotic system.

B. RESULTS

So far we performed and evaluated 10 runs for each aforementioned fault. To increase the significance of these results we are continuously adding more runs and introduce novel fault patterns. Table I shows a summary of the aforementioned measures calculated for the performed runs. Our approach could successfully detect 90% of all induced faults and track about 84% of the detect faults until the end of the experiment. The mean response time to fault (delay) is 4.4 seconds. Table I states that the algorithm performs poorly concerning the False Alarm Rate (FAR) which is approximately 28%. Taking into account that the approach should be integrated into a closed loop of fault detection diagnosis and repair high false alarm rates lead to unnecessarily executed repair routines which might reduce the overall performance of the system. Having a look at the seriousness value of the false alarms (SFAR) one can see that only about $\%2$ of all data during normal behaviour is classified as faulty and lead to this high false alarm rate. Unfortunately, due to the individual data fault detection approach each data from this 2% leads to a false alarm. While

	FDR	FTR	FAR	SFAR	DELAY
NAV	100%	100%	25%	1.0%	2.1 Sec.
SLAM	80%	67%	25%	2.0%	8.2 Sec.
SR	100%	100%	30%	4.0%	5.5 Sec.
RS	80%	66%	30%	1.0%	1.7 Sec.

TABLE I

EXPERIMENT RESULTS FOR TEN RUNS OF EACH INDUCED FAULT.

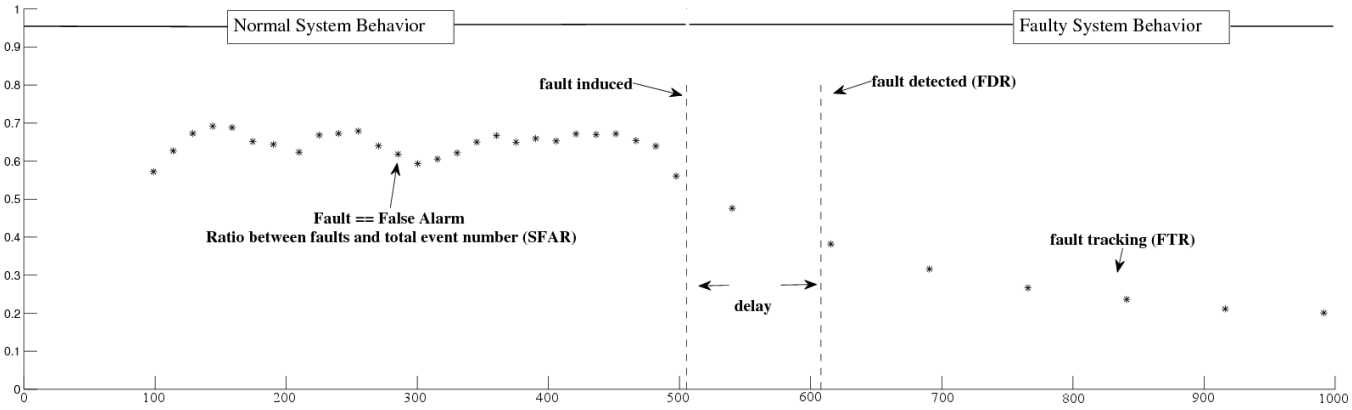

Fig. 2. Visualization of measures along an exemplary time-series. Each event classified as faulty during normal behaviour leads to a False Alarm. The seriousness of the false alarm is the ratio between fault classifications and all events during normal behaviour. The time between inducing and detecting the fault is the detection delay of the algorithm. Fault tracking takes place after the fault was induced.

this technique is sensitive to real faults and provides good response times to failures, it makes the approach vulnerable to false positives, especially for online fault detection where the recorded date intrinsically possesses a higher variance. To reduce the impact of the classification of single events we add another decision layer upon the present one thereby improving the false positive rate. First experiments with a moving average (MA) approach support this intuition. We applied a moving average with a window of approximately one second of recorded data to the individual data binary classification result (fault=1, no fault=0). By this means we obtained a smoothed representation of this classification results with values between 0.0 and 1.0 representing the percentage of events in the moving average range classified as faulty. Next we classified each event with a smoothed value higher than 0.6 as faulty which corresponds to the case that more than 60% of the events around this particular event has been classified as faulty in the individual data classification layer. This post-processing reduces the false alarm rate from 28% to $7,5\%$ while at the same time slightly increasing the response time to fault. Table II summarizes the outcomes.

	FDR	FTR	FAR	SFAR	DELAY
NAV	100%	100%	0%	0.0%	2.1 Sec.
SLAM	70%	97%	0%	0.0%	19.0 Sec.
SR	100%	71%	20%	1.0%	6.8 Sec.
RS	70%	67%	10%	0.2%	8.4 Sec.

TABLE II

EXPERIMENT RESULTS FOR TEN RUNS OF EACH INDUCED FAULT AFTER APPLYING A MOVING AVERAGE OF 1 SECOND TO THE INDIVIDUAL DATA CLASSIFICATION.

V. CONCLUSION AND OUTLOOK

In this paper we presented the online application of our fault detection approach for robotic systems. It is purely data-driven and exploits generic information extracted from the system's inter-component communication. The conducted experiments demonstrated that our approach is capable of detecting and tracking various induced faults online with high probability and acceptable delay. Initially, the algorithm suffers form false positives but introducing another decision-layer based on a moving average reduces the false positive rate. Although it would be most desirable, a comparison of our method with state-of-the-art fault detection approaches in the context of robotic systems turned out to be unfeasible due to the lack of a generally accepted benchmark.

Next steps will involve additional experiments to increase the significance of the results and evaluation of novel fault patterns. Furthermore, we will extend the detection model with sub-models representing different states of the system. This will involve different states of normal behaviour as well as states for already experienced faults. By this means we will on the one hand improve the fault detection and on the other hand enable fault diagnosis for experienced faults.

VI. ACKNOWLEDGMENTS

The research project "An Autonomic Computing Approach for Systemic Self-Regulation" is supported by the Honda Research Institute Europe.

REFERENCES

[1] R. Golombek, S. Wrede, M. Hanheide, and M. Heckmann, "Learning a probabilistic error detection model for robotic systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, October 2010.

[2] A. Haasch, S. Hohenner, S. Hüwel, M. Kleinehagenbrock, S. Lang, I. Toptsis, G. Fink, J. Fritsch, B. Wrede, and G. Sagerer, "Biron–the bielefeld robot companion," in *Proc. Int. Workshop on Advances in Service Robotics*, 2004.

[3] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent asimo: System overview and integration intelligent robots and system," in *Proc. Of IEEE/RSJ International. Conference on Intelligent Robots and Systems*, 2002.

[4] B. Graf, C. Parlitz, and M. Hägele, "Robotic home assistant care-o-bot® 3 product vision and innovation platform," *Proceedings of the 13th International Conference on Human-Computer Interaction. Part II: Novel Interaction Methods and Techniques*, p. 320, 2009.

[5] J. Kurien and P. P. Nayak, "Back to the future for consistency-based trajectory tracking," in *AAAI/IAAI*. AAAI Press / The MIT Press, 2000, pp. 370–377.

[6] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, *Diagnosis and Fault-Tolerant Control*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.
[7] J. Luo, M. Namburu, K. Pattipati, L. Qiao, and S. Chigusa, "Integrated model-based and data-driven diagnosis of automotive antilock braking systems," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 40, no. 2, pp. 321 –336, march 2010.
[8] J. de Kleer and B. Williams, "Diagnosing multiple faults," *Artificial Intelligence*, vol. 32, no. 1, pp. 97–130, 1987.
[9] B. C. Williams and P. P. Nayak, "A model-based approach to reactive self-configuring systems," in *Proceedings of AAAI-96*, 1996, pp. 971–978.
[10] S. Narasimhan and L. Brownston, "Hyde - a general framework for stochastic and hybrid model-based diagnosis," in *Proceedings of the 18th International Workshop on Principles of Diagnosis*, 2007.
[11] A. Feldman, "Approximation algorithms for model-based diagnosis," Ph.D. dissertation, Delft University of Technology, 2010.
[12] A. Bajwa and A. Sweet, "The Livingstone model of a main propulsion system," in *Aerospace Conference, 2003. Proceedings. 2003 IEEE*, vol. 2, march 2003, pp. 2_869–2_876.
[13] S. C. Hayden, A. J. Sweet, and S. Shulman, "Lessons learned in the Livingstone 2 on Earth Observing One flight experiment," in *Proc. AIAA 1st Intelligent Systems Tech. Conf., Am. Inst. Aeronautics and Astronautics*, 2004, pp. 1–15.
[14] J. Ernits, R. Dearden, M. Pebody, and J. Guggenheim, "Diagnosis of Autosub 6000 using automatically generated software models," in *Proc. of the Twenty First International Workshop on Principles of Diagnosis (DX 2010)*, Portland, OR, 2010, pp. 1–8.
[15] S. Narasimhan, R. Dearden, and E. Benazera, "Combining particle filters and consistency-based approaches for monitoring and diagnosis of stochastic hybrid systems," in *Fifteenth International Workshop on Principles of Diagnosis (DX 2004)*, Carcassonne, France, 2004, pp. 1–6.
[16] N. de Freitas, R. Dearden, F. Hutter, R. Morales-Menendez, J. Mutch, and D. Poole, "Diagnosis by a waiter and a Mars explorer," *Invited paper for Proceedings of the IEEE, special issue on sequential state estimation*, 2003.
[17] G. Steinbauer, M. Morth, and F. Wotawa, "Real-time diagnosis and repair of faults of robot control software," *Lecture Notes in Computer Science*, vol. 4020, p. 13, 2006.
[18] L. H. Chiang, E. L. Russell, and R. D. Braatz, *Fault Detection and Diagnosis in Industrial Systems*. Heidelberg: Springer, 2001.
[19] C. D. Stefano, C. Sansone, and M. Vento, "To reject or not to reject: that is the question-an answer in case of neural classifiers," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 30, no. 1, pp. 84–94, 2000.
[20] V. Chandola, E. Eilertson, L. Ertoz, G. Simon, and V. Kumar, *Data mining for cyber security*. Springer, 2006.
[21] N. Ye, "A markov chain model of temporal behavior for anomaly detection," in *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, 2000, pp. 171–174.
[22] M. Casar and J. A. R. Fonollosa, "Overcoming HMM time independence assumption using n-gram based modelling for continuous speech recognition," in *European Signal Processing Conference. EUSIPCO 2008*. EURASIP, August 2008, pp. 3–7.
[23] S. Wrede, "An information-driven architecture for cognitive systems research," Ph.D. dissertation, Technical Faculty – Bielefeld University, 2009.
[24] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *Proc. of the Int. Conf. on Robotics and Automation*, ser. Open-Source Software workshop, 2009.
[25] P. Fitzpatrick, G. Metta, and L. Natale, "Towards long-lived robot genes," *Robotics and Autonomous Systems*, vol. 56, no. 1, pp. 29–45, 2008.
[26] H. Bruyninckx, "Open robot control software," KU Leuven, Belgium, 2008, http://www.orocos.org.
[27] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *The Annals of Mathematical Statistics*, vol. 27, no. 3, pp. 832–837, 1956.