Adaptive Anomaly Detection for Autonomous Systems

Michael Hillebrand
Product Engineering
Fraunhofer IEM
Paderborn, Germany
Michael.Hillebrand@iem.fraunhofer.de

Pritha Gupta
Heinz Nixdorf Institute
Paderborn University
Paderborn, Germany
prithag@mail.upb.de

Florian Heiny
Heinz Nixdorf Institute
Paderborn University
Paderborn, Germany
florian.heiny@uni-paderborn.de

Nikhil Kumar Jha
Product Engineering
Fraunhofer IEM
Paderborn, Germany
Nikhil.Jha@iem.fraunhofer.de

Otthein Herzog
Intelligent Systems
Tongji University
Shanghai, China
o.herzog@uni-bremen.de

Roman Dumitrescu
Fraunhofer IEM
Product Engineering
Paderborn, Germany
Roman.Dumitrescu@iem.fraunhofer.de

Eyke Hüllermeier
Heinz Nixdorf Institute
Paderborn University
Paderborn, Germany
eyke@upb.de

***Abstract*—516.0pt252.0pt**
Smart sensor and communication technology combined with recent advances in online data analytics and machine learning are paving the way for autonomously acting robotic systems, which are learning from experience and evolving in the course of their operation. To fulfill their task and maintain their functionality, such systems must be able to recover from failures, which in turn presumes the ability to detect failures or potential issues in the system. In this regard, anomaly detection methods, which differentiate between normal and abnormal patterns in the data, play an important role. In this paper, we introduce an approach to adaptive anomaly detection, which is based on a new self-monitoring concept and suitable to cope with the evolving nature of the autonomous system and data. We validate our concept of adaptive anomaly detection for an autonomous mobile robotic system using virtual and physical testbeds.

***Index Terms*—Autonomous Systems, Self-Healing, Adaptive Systems, Anomaly Detection, Advanced Systems Engineering, Resilience Engineering**

I. Introduction

Modern robotic systems are driven by the amalgamation of increasingly powerful sensor and hardware technologies on the one side and advanced data analytics and machine learning methodology on the other side. With improved operational capabilities and an increasing level of autonomy of such systems, we are also facing new challenges. In particular, such systems need to be resilient against malfunctions and partial failures, and able to recover from an abnormal system state. One approach to guarantee this ability is the concept of self-healing, which aims to increase the availability and safety of the intended functionality. One major challenge in self-healing is the evolving nature of an autonomous system that adapts to its environment by learning from experience, which calls an adaptive anomaly detection in self-monitoring systems. In this regard, our contributions are as follows.

Project funded by Federal Ministry of Education and Research (BMBF)

1) *Self-Healing framework*: We present a self-healing framework for autonomous systems to increase availability and safety of the intended functionality.
2) *Adaptive anomaly detection*: A part of the framework is an adaptive anomaly detection component for self-monitoring, which can cope with the evolving nature of autonomous systems.
3) *Experimental validation of the concept*: We validate our concept on an autonomous mobile robotic system that is operating in indoor environments.

Following a general discussion of the challenges with respect to dependability in the next section, we introduce the concept of self-healing in autonomous systems in Section III. An important prerequisite of self-healing is the ability to detect anomalies in the system behavior. A brief account of related work in then given in Section IV. In Section V, we introduce our concept for adaptive anomaly detection and present the architecture as part of the self-healing framework. Finally, we validate our approach through an autonomous mobile robotic system in Section VI.

II. Autonomous Systems

In this section, we outline the concepts of autonomous system and reference structure. Further, we define the relevant notion of dependability. This leads to the concept of self-healing as a prerequisite of increasing the availability and safety of the intended system functionality.

A. Architecture

A system can be defined as autonomous if it is capable of independently achieving a predefined goal in accordance with its demands of the current situation without recourse either to human control or explicit programming [1]. Such systems use sensors to perceive the environment, proactively generate an appropriate plan of action according to the situation and execute the plan safely and reliably with actuators. The reference architecture focuses on sensors, self-regulatory and actuators as the three elements of autonomous systems. Self-regulation contains building blocks of detection and interpretation, planning and plan recognition, learning and reasoning, as well as communication and collaboration. Machine learning enables autonomous systems to generate new knowledge from data and expand their knowledge base. Such systems learn from experience and adapt their action plans accordingly [1], [2].

B. Resilience in Autonomous Systems

Recently, the term *Resilience* gains attention to address the evolving and changing nature of autonomous systems [1]: LAPRIE defines *Resilience* as the *"persistence of dependability when facing changes"* [3]. Changes may occur in nature (e.g. functional, environmental), prospect (e.g. unforeseen, foreseen) or timing (e.g. short-term, long-term). *Dependability* is defined as the *"delivery of service that can justifiably be trusted, thus avoidance of failures that are unacceptably frequent or severe"* [4]. A service failure is an *"event that occurs when the delivered service deviates from the correct service"*. This dependability is compromised by five *attributes:* availability, reliability, safety, integrity and maintainability [4]. LAPRIE ET AL. define the term availability as the readiness for correct service [4], whereas safety of the intended functionality is the absence of unreasonable risk due to hazards resulting from insufficiencies of the intended functionality [5].

Our self-healing framework aims to increase the availability and safety of the intended functionality of autonomous systems.

III. SELF-HEALING AUTONOMOUS SYSTEMS

The term self-healing was introduced around the year 2000 by IBM with the Autonomic Computing Initiative [6] and later shaped by the principles of *organic computing* and *artificial life*. GHOSH ET AL. define self-healing as follows: *"[...] a self-healing system should recover from abnormal (or unhealthy) state and return to the normative (healthy) state, and function as it was prior to disruption"* [7]. Self-healing implementations work by detecting malicious behavior and degradation of the autonomous system, diagnosing failure root cause, deriving a remedy, and recovering with a strategy while avoiding unpredictable side effects [8]. In order to integrate self-healing into the architecture of an autonomous system, we need a framework that consists of self-healing functions and functional building blocks. The self-healing framework consists of six stages:

1) *Monitor:* Continuous short-term self-monitoring of the performance of the autonomous system by processing and analyzing of sensor data streams.
2) *Anticipate:* To predict future states and trends. Anticipate changing environmental conditions or long-term system degradation.
3) *Diagnose:* Self-diagnosis based on the perceived state; probabilistic root cause analysis for abnormal state.
4) *Plan:* To plan and select an appropriate recovery strategy while avoid negative side effects.
5) *Respond:* Perform recovery actions on the autonomous system to return to a healthy state.
6) *Learn:* To learn the right lessons from experience and expand the knowledge base to improve each function.

A key aspect of self-healing systems is the detection of malicious behavior and the concept of health and performance monitoring in order to recover from an abnormal state. For that purpose, anomaly detection techniques are commonly used.

A. Anomaly Detection in Autonomous Systems

Anomaly Detection is part of *Monitoring* in self-healing systems. Autonomous systems comprise multiple sensors to observe various quantities inside and outside the system, such as temperatures, voltages, or other state variables. In the case of system degradation or failure, the nature of the underlying data generating process changes. These changes can be detected using anomaly detection techniques. An anomaly (or outlier) is usually defined as “an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism” [9]. Based on this assumption, anomaly detection has been used widely in quality control [10], fault detection [11], and system diagnosis applications [12], [13]. A more detailed overview of applications can be found in [14].

We structure our requirements on adaptive anomaly detection along input, detection, and output (see Figure 1) [15].

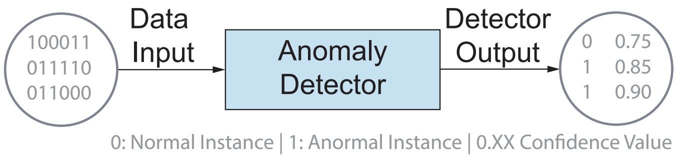

Fig. 1: Structuring requirements for anomaly detection

Input: The input for anomaly detection in autonomous evolving systems are recorded data or data streams of raw sensor values (e.g., temperature, speed, network traffic), i.e., a set of parallel data streams $\{X_1, \ldots, X_d\}$ produced by d sensors $S_1, \ldots, S_d$. More specifically, the i^{th} sensor gives rise to an open-ended sequence of time-stamped observations

$$X_i = \big\{(x_{i,1}, t_{i,1}), (x_{i,2}, t_{i,2}), \ldots\big\},$$

where $x_{i,j} \in \mathbb{R}$ is the j^{th} value observed for sensor S_i, and $t_{i,j} \in \mathbb{R}$ the time at which the value was recorded. In some cases, the data streams are synchronous $(t_{1,j} = t_{2,j} = \ldots = t_{d,j})$, but this cannot be assumed in general. The anomaly detector must preprocess those time series data, including windowing, filtering, and feature extraction.

Detection: CHANDOLA ET AL. [15] distinguish three types of anomalies:

1) *Point anomalies*: Single observations that are different from the rest of the data.
2) *Contextual anomalies*: Observations that are only anomalous in a specific context, but otherwise not.

3) *Collective anomalies*: Observations that are anomalous as a collection, but not necessarily each by itself.

It is important to note that contextual anomalies require some kind of context present in the data to occur, while collective anomalies can only exist if the observations are correlated among themselves, which is the case for temporal or spatial data, for example.

Anomaly detection is often treated as a machine learning problem. In the literature, a variety of algorithms are proposed. They can be classified into three categories based on the extent to which training labels are available [15]:

1) *Supervised case*: In the supervised case, the training set contains labeled normal and anomalous observations. Here, it is also possible to alternatively use algorithms for imbalanced binary classification.
2) *Semi-supervised case*: In this case, the training data only consists of normal observations. Since anomalous ones are not necessary, these algorithms are more versatile.
3) *Unsupervised case*: Here, no labels are provided at all, i.e., the data may or may not contain anomalies. This is the most difficult case to handle and at the same time the most flexible.

As autonomous systems adapt to new situations or operate in unknown environments, it is reasonable to assume that labeled training data does not exist in general. Hence, we focus on unsupervised methods for anomaly detection in our work.

In practice, the detection of anomalies is complicated due to various reasons. Firstly, the *presence of noise* makes the detection more difficult, as noise can happen to look similar to actual anomalies. Secondly, not all anomalous observations originate from the faulty behavior of the system; instead, some anomalies may originate from *unusual healthy events* like adaptable system behavior (e.g., due to reconfiguration mechanism). Thirdly, depending on the current *operating context* of the system, the same observation can either be normal or not. For example, the pH-values, heartbeat frequency, and pulse of a human body may vary during sports activities, sleeping, working or during resting phases. And finally, the definition of what is normal and what is not can evolve over time. This *concept drift* [16] might be due to altered subsystems, wear and tear of components, or a permanent change of context. All of these properties can be present in autonomous systems and will most likely lead to missed anomalies (false negatives) or erroneously detected anomalies (false positives).

Output: The output of an anomaly detector can be a discrete label categorizing the system as anomalous or not, or, more generally, a score expressing a degree of confidence in an anomaly. For example, a common goal is to learn a scoring function $f\colon \mathbb{T} \to \mathbb{R}$, where $\mathbb{T}$ is a set of time points and $f(t)$ an anomaly score reflecting the probability of a state being anomalous. To make a final decision about a possible alarm, the scores are often turned into discrete labels via thresholding (i.e., checking whether $f(t) \geq \nu$ or $f(t) < \nu$). To find an optimal compromise between false positives and false negatives, the threshold ν must be tuned very carefully.

For further diagnosis and self-monitoring, we also need the type and location of an anomaly as well as a confidence value and the timings (start and duration). The confidence value is an estimate that reflects the confidence of its prediction. Low confidence value means that the detector is uncertain whether it observed an anomaly. We then can define a threshold to process anomaly data in the self-monitoring pipeline.

IV. Related Work

In the literature, a wide spectrum of approaches has been proposed to detect faults or anomalies in technical systems. Traditional model-based methods, which rely on a model based on application-specific expert knowledge, are only able to detect faults that have been identified in the model at design time [17], [18]. They are not suited for detecting new and previously unknown faults in autonomous systems.

This motivates approaches that can learn normal and anomalous behavior automatically. The detection of anomalies has been researched in various areas of autonomous systems and robotics. In [19], an algorithm for the detection of anomalies in robots interacting with humans is proposed to increase safety. In [20], a mobile robot is presented that is able to detect visual anomalies on its way. In [21], the authors propose a fuzzy-rule-based learning approach of normal behavior to detect anomalies in a self-adaptive robot. In [13], a combination of statistical and logic-based detection techniques is evaluated on different cyber-physical systems. The authors of [22] use so-called spatio-temporal models for anomaly detection in robot behaviors. In [11], an anomaly detection technique is proposed and tested on data of different technical or autonomous systems, such as an unmanned aerial vehicle and an electric power system of a spacecraft.

None of these algorithms, however, provide functions to work in different system contexts. Solely the approach proposed in [11] can take a limited amount of temporal context in the form of past observations into account. However, the system does not consider any internal or external system context.

V. Adaptive Anomaly Detection in Self-Monitoring Systems

A. Self-Monitoring Concept

Based on our self-healing framework, we define the building blocks of our self-monitoring concept. Figure 2 shows the architecture with the data processing and the learning mechanism on an abstract level.

Data Collector: This agent collects the data from the sensors and other parts of the system. It also performs preprocessing, such as sliding windowing, filtering, and interpolation.

Health Monitor: The health monitor uses defined health signals processed by the data collector. It uses fuzzy logic and a so-called fuzzy pattern tree [23], [24] to predict the system health states (e.g., *normal, warning, critical, fatal*).

Anomaly Pipeline: The pipeline consists of preprocessing, feature extraction, and detection. It processes the data from the data collector and performs anomaly detection.

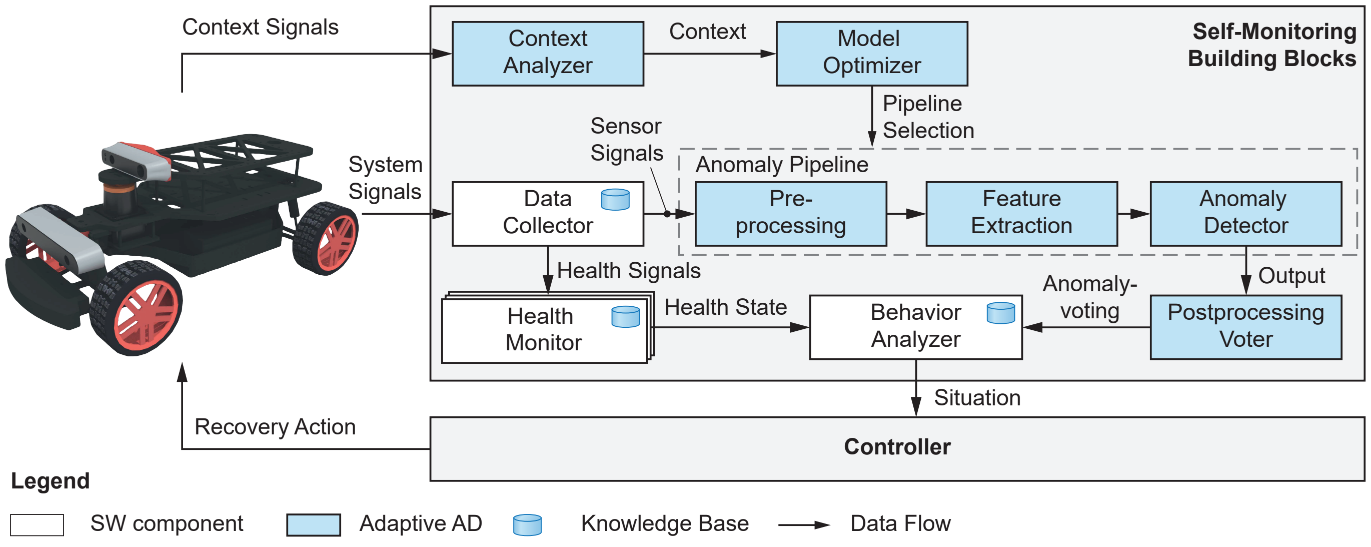

Fig. 2: Self-Monitoring Concept

Context Analyzer: This component determines the context of the system. We distinguish internal and external context, for instance operating mode and operating state such as driving, charging or environmental conditions.
Model Optimizer: The model optimizer selects an appropriate anomaly pipeline for the observed context by using a multi-objective decision logic (see Section V-B2).
Voter: The voter increases the overall reliability of the anomaly pipeline. The output of the anomaly pipeline is a score and a confidence value. The voter checks whether the scores are exceeding a given threshold and thus can be considered as anomalous for further processing.
Behavior Analyzer: The behavior analyzer associates anomaly detection with health states from the health monitor. It classifies the anomaly as *normal, warning, critical, fatal*.
Controller: The controller selects an appropriate recovery action on the system based on the observed situation. Recovery actions might be restart, reconfiguration, isolation, or fail-safe strategies. The recovery action depends on the level of criticality of an anomaly and health state.

B. Adaptive Anomaly Detection

Figure 2 highlights the components of the adaptive anomaly detection mechanism in self-monitoring systems. In the following section we describe those mechanisms in detail.

1) Context analysis: Contextual analysis is one of the major aspects while performing the task of anomaly detection. A contextual anomaly is a sample in the data that should be considered an anomaly only in a specific context. The features used to identify these sorts of anomalous behavior are broadly classified as contextual and behavioral features. While contextual features often refer to space and time information, behavioral features refer to operating modes of the autonomous system itself. The *Context Analyzer* uses a Rule-Based System to determine the current context of the system. While operating modes like *Teleop* and *Autonomous* are binary, other behavioral features are vague and imprecise: For example *vehicle speed* is a continuous value that is mapped to *high, medium or low*. For these cases we use a membership function. A membership function for a fuzzy set A on the context feature C is defined as $\mu_A : c \rightarrow [0, 1]$. Each element of C is mapped to a value between 0 and 1. It quantifies the degree of membership of a contextual feature in C to the fuzzy set A [25]. To tackle contextual anomalies, different baseline detectors are trained for the normal operation data for either context. The aforementioned technique helps in effectively and efficiently identifying anomalies in such an environment where the data is unlabeled and has uncertainty involved along with the presence of various contexts. In Section VI, we discuss how this technique is being used for performing inference on the evaluation data and identify anomalies.

2) Pipeline selection: Using the ground truth data and windowing techniques (discussed in Section VI-D), the quality of anomaly detectors are assessed based on their performance and computational complexity (running/inference time, resources required). Figure 3 shows an example of two anomaly detectors based on three quality dimensions. Based

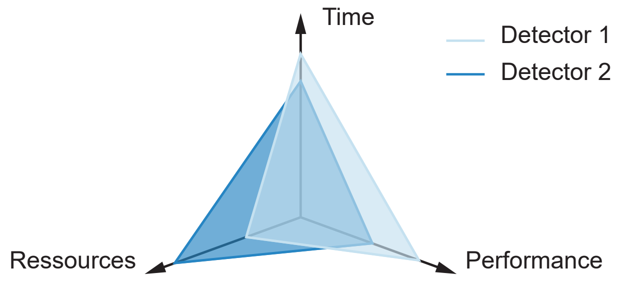

Fig. 3: Multi-objective decision space for anomaly detectors

on the current context, the *Model Optimizer* (see Section Section V-A) selects a detector model that performs best in terms of these quality metrics in a given context, which is a Pareto optimal solution of the multi-objective optimization problem (MOO) [26]. For simplifying this problem, we convert it to a single-objective optimization problem (SOO) using a linear scalarization technique, which guarantees Pareto-optimality of the solutions [27]. This function is expressed as a weighted sum of the evaluation quality of the detector d_j with respect to the different quality metrics:

$$p_j = p(d_j) = \boldsymbol{w}^\top \boldsymbol{m}_j = \sum_{i=1}^{k} w_i \cdot m_{j,i}\,,$$

where $\boldsymbol{w} = (w_1, w_2, \ldots, w_k)^\top$ is a weight vector satisfying $w_i \geq 0$ and $\sum_{i=1}^{k} w_i = 1$, with w_i representing the weight of the metric i. These weight factors could be predefined on the basis of domain knowledge or learned using techniques such as evolutionary algorithms [27]. Moreover, $m_{j,i}$ denotes the evaluation quality of the detector d_j with respect to the metric i. Finally, the detector with the maximum quality is selected by the Model Optimizer:

$$d = \underset{d_j \in D}{\arg\max}\, p(d_j)$$

3) Anomaly detection: *Describe the adaptive part* Learning (c, X_i) where c is a given context vector

4) Anomaly voting: The *Voter* uses a threshold mechanism ($f(t) \geq \nu$ or $f(t) < \nu$) to make a decision about the anomaly relevance (see Section III-A). Another approach is a redundant detection mechanism. In that redundancy concept we consider the relevance if $n : m$ detectors raise an alarm for any given observation $x_{i,j}$.

VI. Experimental Setup

In this section, we validate our approach with an autonomous mobile robot. Starting with brief description of the software and hardware platform, we describe our validation scenario and finally present the results.

A. Platform

Hardware: The mobile robotic system is based on the MIT Racecar platform. The system contains numerous on-board sensors and an NVIDIA Jetson TX2 [28] as an embedded AI computing device on a $1/10$th scale ackermann racecar [29]. The robot is connected with a host computer using WiFi onboard of the Jetson TX2. Onboard sensors are a laser rangefinder, a stereo camera, and an inertial measurement unit. The hardware platform is shown in Figure 4.

Software: The low level hardware abstraction layer is provided by NVIDIA Jetpack SDK and Linux Ubuntu as main operating system. The system uses the Robot Operating System (ROS) as middle-ware for robotic applications. The high level functions like sensor interfacing, localization and mapping, maneuvering as well as motion control are implemented on top of ROS. We are using InfluxDB time-series database for real-time data logging and processing.

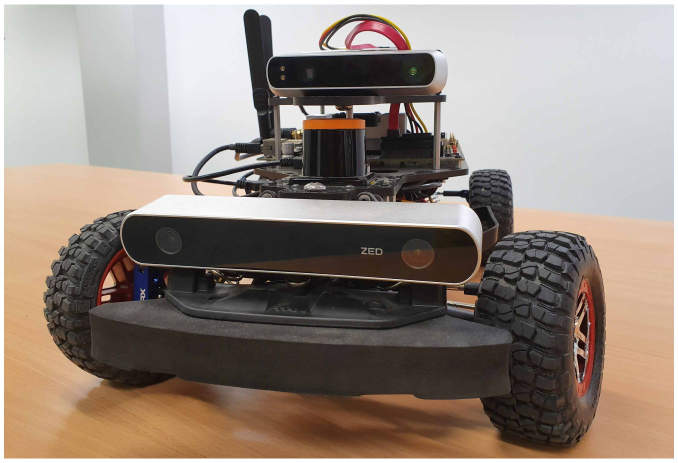

Fig. 4: Mobile Robot based on MIT Racecar Platform [29]

B. Scenario

The scenario under consideration is an autonomous mobile robot inside a warehouse facility, where the car is required to navigate from one location (station) to another. As discussed in the previous subsection, we have a hardware model as well as a simulation model implemented for this use case. In the simulation, not only the robot is modeled, but also the nearby surroundings are designed to mimic real-life behavior, Based on the context, the root operates in one of the two modes, teleop (*manual*) or autonomous. In manual mode, goals are explicitly known to the user and the car is manually maneuvered to reach from one station to another. For the autonomous mode, the robot is equipped with localization and planning algorithms, where the car simultaneous builds a map of the environment and localizes itself on the map. Then, it plans the most optimal routes to achieve arbitrary goals. The normal operation of the car can be disrupted by forcefully injecting some anomalies that resemble those encountered in real life, such as sudden force or torque due to a crash, wheel blockages, etc. These faults are injected using the scripts that transform the operation of certain components of the car or perform motion interference scenarios (hitting, get stuck, friction variation).

C. Methods for Anomaly Detection

For the experiment, we combined traditional unsupervised anomaly detection approaches with a SOO, referred to as Anomaly Detectors (ADs). Those approaches are briefly discussed below.

- ***Isolation Forest (IF)*** [30], [31]: Similar to the popular random forest algorithm, this is a tree ensemble method based on decision trees. It is one of the most efficient methods for high dimensional data. IF recursively selects a feature at random and a split value between the minimum and maximum values of that feature, thus creating a tree-like structure. The number of splittings required to isolate an observation is equivalent to the measure of its normality. An anomalous observation should have a smaller path from the root node, and thus will require a smaller number of splits.

- ***One-class Support Vector Machine (OCSVM)*** [32], [33]: Here, the margin principle of support vector machines is adapted from the discriminatory (binary classification) setting to the single-class case (data containing only normal instances). The goal is to find a decision boundary characterizing the data, in the sense of identifying an as small as as possible region of the data space (or actually a kernel-induced feature space) containing the data points. A new sample can then be classified as anomalous or not by checking if it is lying outside or inside the boundary. To optimize performance, the training requires fine-tuning the hyperparameters, which determine factors like contour smoothness, the proportion of outliers, etc.
- ***Angle-based outlier detection (ABOD)*** [34]: Most anomaly detection methods are based on evaluating Euclidean distances in full dimension, and thus are prohibitive for high dimensional data. This method rather considers variance between the difference vectors of a point to the neighboring data points, taking a high value as an indicator for abnormality.
- ***Local Outlier Factor (LOF)*** [35]: This algorithm is similar to a clustering approach and computes the local density deviation of a given observation with respect to its neighbors. Data points with density values significantly lower than their neighbors' values appear to be isolated and are considered as anomalous. Considering how isolated the observation is with respect to its surrounding neighborhood, this algorithm takes into account both the local as well as global properties of the dataset.
- ***Autoencoder (AE)*** [36]: AEs are neural networks that are trained to encode their input data to a compressed or expanded representation of itself and decode it again as the output. Being trained solely on normal observations, it will learn a mechanism/function able to reconstruct the normal observations, while the reconstruction error for anomalous observations is supposedly high. The magnitude of this reconstruction error can be used to identify anomalous behavior in the signals of the system.

D. Dataset

The information emanated from the sensors of the race-car is recorded in the InfluxDB database, which is later filtered and preprocessed to provide a fairly suitable dataset for our evaluation purpose. The data is recorded for the normal operation of the race-car and also for the cases where the point-wise anomalies are forcefully injected to set up some anomalous scenarios to evaluate. The time-series data is recorded for the following major components:

- *Inertial Measurement Unit (IMU)*: It is used to measure the linear acceleration, angular velocity, and the orientation of the vehicle.
- *Odometer*: It uses data from the motion sensors to estimate the change in position over time

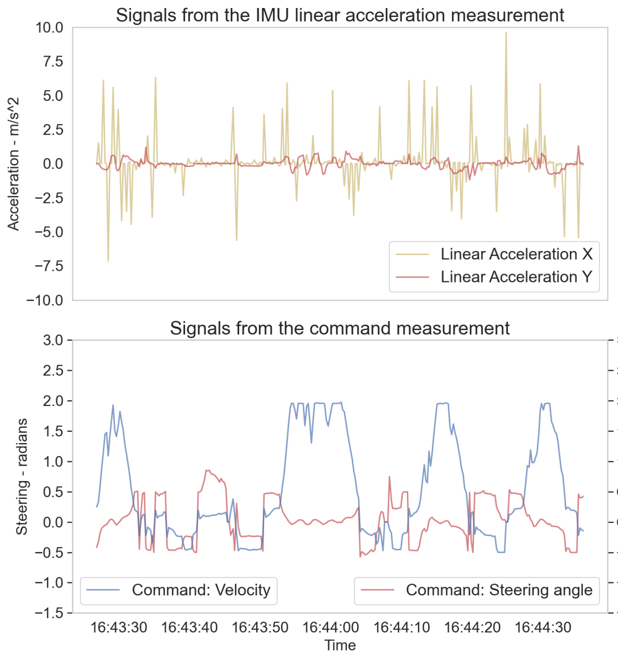

Fig. 5: Plotting the data for the IMU linear acceleration and the command signals

- *Command*: It records the steering angle and the velocity of the vehicle during its operation.

The data generated by each of the components is recorded around 10Hz, therefore the different components need not necessarily be synchronized with each other. To overcome this problem, the different datasets are combined into a single dataset, and a nearest-neighbor interpolation is performed on this dataset, while the instances with *NaN* values are dropped. The result is a dataset with regularly spaced timestamps [37].

For the purpose of evaluation, our dataset has 41 features. The normal operation (without recording) of the race-car is recorded for 56 minutes and consists of 367 424 samples. The evaluation dataset is recorded for 46 minutes and consists of 281 613 observations. Since the anomalies are injected manually using scripts, we have a rough idea of the labels of the observations. Although this ground truth is not reliable due to various factors like asynchronicity between various components, anomaly spread over an uncertain duration, etc., we still have a coarse knowledge of the timestamps where anomalies occurred. This information is utilized by using windowing techniques to evaluate the quantitative performance of the detectors using various metrics.

E. Metrics

For evaluating anomaly detectors, we use labels produced from the scenarios explained in Section VI-B. Anomalies are rare events and occur in less than 5 % of the time, so accuracy is not a suitable metric for performance evaluation [38]. Instead, the use of metrics like F_1-measure or Mathews

Correlation Coefficient (MCC) is suggested to provide better insight of performance of different ADs [38], [39].

To define the metrics formally, let y and $\hat{y}$ denote the ground-truth and predicted labels, respectively. The four quantities of the confusion matrix i. e., true positives, true negatives, false positives, and false negatives, shortly referred as $\widehat{\mathrm{tp}}, \widehat{\mathrm{tn}}, \widehat{\mathrm{fp}}$ and $\widehat{\mathrm{fn}}$, are defined as:

$$\widehat{\mathrm{tp}}(y,\hat{y}) = \sum_{i=1}^{n} [\![y_i = 1, \hat{y}_i = 1]\!],\ \widehat{\mathrm{tn}}(y,\hat{y}) = \sum_{i=1}^{n} [\![y_i = 0, \hat{y}_i = 0]\!],$$
$$\widehat{\mathrm{fp}}(y,\hat{y}) = \sum_{i=1}^{n} [\![y_i = 0, \hat{y}_i = 1]\!],\ \widehat{\mathrm{fn}}(y,\hat{y}) = \sum_{i=1}^{n} [\![y_i = 1, \hat{y}_i = 0]\!],$$

where n is the number of instances in the time-series anomaly detection dataset [40].

Recall and Precision: *Recall* is also known as the true positive rate or sensitivity value, which is the proportion of actual anomalies that are correctly identified by the AD [38], [41]. *Precision* is known as the positive predictive value, which denotes the proportion of predicted positive labels (anomalies) that are correctly identified [38], [41]:

$$\mathrm{d_{RE}}(y,\hat{y}) = \frac{\widehat{\mathrm{tp}}}{\widehat{\mathrm{tp}} + \widehat{\mathrm{fn}}}, \quad \mathrm{d_{PR}}(y,\hat{y}) = \frac{\widehat{\mathrm{tp}}}{\widehat{\mathrm{tp}} + \widehat{\mathrm{fp}}}$$

F_1-measure: The F_1-measure is defined as the harmonic mean of precision and recall. It can also be defined in terms of the quantities in the confusion matrix [38], [40]:

$$\mathrm{d}_{F_1}(y,\hat{y}) = \frac{2\cdot \mathrm{d_{PR}}(y,\hat{y})\cdot \mathrm{d_{RE}}(y,\hat{y})}{\mathrm{d_{PR}}(y,\hat{y}) + \mathrm{d_{RE}}(y,\hat{y})} = \frac{2\widehat{\mathrm{tp}}}{2\widehat{\mathrm{tp}} + \widehat{\mathrm{fn}} + \widehat{\mathrm{fp}}} \qquad (1)$$

MCC: MCC is a balanced measure that penalizes $\widehat{\mathrm{fp}}$ and $\widehat{\mathrm{fn}}$ equally. It can be used even if the classes are of very different size [38], [39]. The measure is exactly 0 in the case of a constant or randomly guessing detector. Formally, it is defined as follows:

$$\mathrm{d_{MCC}}(y,\hat{y}) = \frac{\widehat{\mathrm{tp}}\times\widehat{\mathrm{tn}} - \widehat{\mathrm{fp}}\times\widehat{\mathrm{fn}}}{\sqrt{(\widehat{\mathrm{tp}}+\widehat{\mathrm{fp}})(\widehat{\mathrm{tp}}+\widehat{\mathrm{fn}})(\widehat{\mathrm{tn}}+\widehat{\mathrm{fp}})(\widehat{\mathrm{tn}}+\widehat{\mathrm{fn}})}} \qquad (2)$$

F. Results

To evaluate the performance of the various detectors, we present some of the results in terms of the above metrics. The evaluation is done for the autonomous mode of operation, and the information regarding the dataset is provided in Section VI-D. The system used for experimentation has the following specification: 1.6GHz dual-core Intel Core i5, 4GB of RAM. There was no GPU present in either the training or the evaluation phase.

Inference time is the time taken by a trained model to process the complete evaluation dataset and produce an anomaly score for each data point. A two-layer network was used in the encoder and the decoder part of the AE and the model was trained for 20 epochs, while the other machine learning models except ABOD were trained with the standard default parameters provided in the *scikit-learn* package [42]. The ABOD approach was trained with standard parameters using the Python Outlier Detection (PyOD) [43]. The last metric shown is the memory required by the model to perform inference on the given evaluation data set.

Metrics	OCSVM	IF	ABOD	AE	LOF
	Results: Teleop Mode				
F_1-measure	0.39	0.33	**0.71**	0.39	0.22
Recall	0.25	0.50	0.83	0.25	**1.00**
Precision	**0.88**	0.25	0.63	**0.88**	0.13
MCC	0.46	0.35	**0.72**	0.46	0.35
Inference-time (Seconds)	**0.06**	0.63	88.04	18.2	0.07
Memory (MB)	0.99	**0.59**	32.55	7.80	4.31
	Results: Autonomous Mode				
F_1-measure	0.23	0.73	**0.88**	0.43	0.67
Recall	0.13	**1.00**	0.92	0.29	0.73
Precision	**0.93**	0.57	0.85	0.87	0.62
MCC	0.34	0.76	**0.88**	0.49	0.67
Inference-time (Seconds)	**0.07**	0.74	95.01	25.7	1.56
Memory (MB)	1.45	**0.89**	47.55	11.99	5.33

TABLE I: Evaluation of ADs with respect to different quality metrics. Best entry for each measure marked in bold.

VII. Conclusion

This paper presents a reconfigurable anomaly detection mechanism for self-monitoring in self-healing autonomous systems. Our experimental results shows the quality of different detectors in different context. The model optimizer can select an appropriate anomaly pipeline using a single-objective optimization technique. As for future work, one important challenge concerns the notion of context. In practice, this notion is often rather fuzzy, and the transition from one context to another gradual rather than abrupt. This needs to be addressed in our context analyzer. Besides, we plan to transfer the self-monitoring system to other industrial robotic applications. This requires an extension of the behavior analyzer and an integration of the controller mechanisms to perform recovery actions on the system. As a long-term goal, we seek to provide an overarching model-based design methodology for self-healing autonomous systems.

Acknowledgment

This work is part of the research project Validation of Artificial Immune Systems for Autonomous Systems (KI4AS). The project is granted under 03VP02551 by Federal Ministry of Education and Research (BMBF).

References

[1] "Fachforum Autonome Systeme : Chancen und Risiken für Wirtschaft, Wissenschaft und Gesellschaft : Abschlussbericht - Langversion; Redaktionsschluss: 08. März 2017," Berlin, Tech. Rep., 2017. [Online]. Available: http://publications.rwth-aachen.de/record/729403

[2] R. Dumitrescu, J. Gausemeier, P. Slusallek, S. Cieslik, G. Demme, T. Falkowski, H. Hoffmann, S. Kadner, F. Reinhart, T. Westermann *et al.*, "Studie" autonome systeme"," Studien zum deutschen Innovationssystem, Tech. Rep., 2018.

[3] J.-C. Laprie, "From dependability to resilience," *DSN, Anchorage, AK, USA*, vol. 8, 01 2008.

[4] J. C. Laprie, A. Avizienis, and H. Kopetz, *Dependability: Basic Concepts and Terminology*. Berlin, Heidelberg: Springer-Verlag, 1992.

[5] O. Kirovskii and V. Gorelov, "ISO 26262 and ISO PAS 21448 Road vehicles - Safety of the intended functionality," vol. 534, no. 1, p. 012019, 2019.

[6] A. Computing *et al.*, "An architectural blueprint for autonomic computing," 2006.

[7] "An architectural blueprint for autonomic computing," IBM, Tech. Rep., Jun. 2005.

[8] D. Ghosh, R. Sharman, H. Rao, and S. Upadhyaya, "Self-healing systems — survey and synthesis," *Decision Support Systems*, vol. 42, pp. 2164–2185, 01 2007.

[9] D. M. Hawkins, *Identification of outliers*. Springer, 1980, vol. 11.

[10] A. Konig, P. Windirsch, and M. Glesner, "Massively parallel vlsi-implementation of a dedicated neural network for anomaly detection in automated visual quality control," in *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, Sep. 1994, pp. 354–363.

[11] E. Khalastchi, M. Kalech, G. A. Kaminka, and R. Lin, "Online data-driven anomaly detection in autonomous robots," *Knowledge and Information Systems*, vol. 43, no. 3, pp. 657–688, Jun. 2015.

[12] R. A. Maxion, "Anomaly detection for diagnosis," in *[1990] Digest of Papers. Fault-Tolerant Computing: 20th International Symposium*, June 1990, pp. 20–27.

[13] N. Srivastava and J. Srivastava, "A hybrid-logic approach towards fault detection in complex cyber-physical systems," in *Proceedings of the Annual Conference of the Prognostics and Health Management Society, Portland, Oregon, USA*. Citeseer, 2010, pp. 1–11.

[14] C. C. Aggarwal, *Outlier Analysis*, 2nd ed. Cham: Springer International Publishing, 2017.

[15] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[16] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, vol. 23, no. 1, pp. 69–101, Apr 1996. [Online]. Available: https://doi.org/10.1007/BF00116900

[17] Y. Zhao, S. Oberthür, M. Kardos, and F.-J. Rammig, "Model-based runtime verification framework for self-optimizing systems," *Electronic Notes in Theoretical Computer Science*, vol. 144, no. 4, pp. 125–145, 2006.

[18] C. Priesterjahn, C. Sondermann-Wolke, M. Tichy, and C. Holscher, "Component-based hazard analysis for mechatronic systems," in *2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*. IEEE, 2011, pp. 80–87.

[19] R. Hornung, H. Urbanek, J. Klodmann, C. Osendorfer, and P. van der Smagt, "Model-free robot anomaly detection," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*. IEEE, 2014, pp. 3676–3683.

[20] P. Chakravarty, A. M. Zhang, R. Jarvis, and L. Kleeman, "Anomaly detection and tracking for a patrolling robot," in *Australasian Conference on Robotics and Automation (ACRA)*. Citeseer, 2007.

[21] B. Jakimovski and E. Maehle, "Artificial Immune System Based Robot Anomaly Detection Engine for Fault Tolerant Robots," in *Autonomic and Trusted Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 177–190.

[22] K. Häussermann, O. Zweigle, and P. Levi, "A Novel Framework for Anomaly Detection of Robot Behaviors." *Journal of Intelligent and Robotic Systems*, vol. 77, no. 2, pp. 361–375, 2015.

[23] R. Senge and E. Hüllermeier, "Pattern trees for regression and fuzzy systems modeling," in *2010 IEEE International Cnoference on Fuzzy Systems*. New Jersey: IEEE, 2010, pp. 1–7.

[24] R. Senge and E. Hüllermeier, "Fast fuzzy pattern tree learning for classification," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 6, pp. 2024–2033, 2015.

[25] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, no. 1, pp. 28–44, Jan 1973.

[26] M. Kaisa, *Nonlinear Multiobjective Optimization*, ser. International Series in Operations Research & Management Science. Boston, USA: Kluwer Academic Publishers, 1999, vol. 12.

[27] C.-L. Hwang and A. S. M. Masud, *Methods for Multiple Objective Decision Making*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1979, pp. 21–283. [Online]. Available: https://doi.org/10.1007/978-3-642-45511-7-3

[28] Nvidia, "Nvidia jetson," https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/, 2019.

[29] T. Henderson and A. Fishberg, "Mit racecar," https://github.com/mit-racecar, 2019.

[30] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 413–422.

[31] ——, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, p. 3, 2012.

[32] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in neural information processing systems*, 2000, pp. 582–588.

[33] J. Shawe-Taylor and B. Žličar, "Novelty detection with one-class support vector machines," in *Advances in Statistical Models for Data Analysis*. Springer, 2015, pp. 231–257.

[34] H.-P. Kriegel, A. Zimek *et al.*, "Angle-based outlier detection in high-dimensional data," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 444–452.

[35] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM sigmod record*, vol. 29, no. 2. ACM, 2000, pp. 93–104.

[36] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[37] M. Lepot, J.-B. Aubin, and F. Clemens, "Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment," *Water*, vol. 9, p. 796, 10 2017.

[38] D. M. Powers, "Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[39] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: an overview ," *Bioinformatics*, vol. 16, no. 5, pp. 412–424, 05 2000. [Online]. Available: https://doi.org/10.1093/bioinformatics/16.5.412

[40] O. Koyejo, N. Natarajan, P. Ravikumar, and I. S. Dhillon, "Consistent multilabel classification," in *NIPS*, 2015, pp. 3321–3329.

[41] D. M. Powers, "Recall & precision versus the bookmaker." International Conference on Cognitive Science, 2003.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[43] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection," *Journal of Machine Learning Research*, vol. 20, no. 96, pp. 1–7, 2019. [Online]. Available: http://jmlr.org/papers/v20/19-011.html