2014 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS 2014)
September 14-18, 2014, Chicago, IL, USA

Model-free robot anomaly detection

Rachel Hornung, Holger Urbanek, Julian Klodmann,
Christian Osendorfer, Patrick van der Smagt, *Member, IEEE*

***Abstract*— Safety is one of the key issues in the use of robots, especially when human–robot interaction is targeted. Although unforeseen environment situations, such as collisions or unexpected user interaction, can be handled with specially tailored control algorithms, hard- or software failures typically lead to situations where too large torques are controlled, which cause an emergency state: hitting an end stop, exceeding a torque, and so on—which often halts the robot when it is too late. No sufficiently fast and reliable methods exist which can early detect faults in the abundance of sensor and controller data. This is especially difficult since, in most cases, no anomaly data are available. In this paper we introduce a new robot anomaly detection system (RADS) which can cope with abundant data in which no or very little anomaly information is present.**

I. Introduction

Reliability and, depending on the task, high accuracy are critical for the successful application of a robot. Early detection of faults is crucial to ensure safety and high performance of the system. Faults can be soft- and hardware-related and include communication problems, wear-dependent deviation, and broken sensors. Especially in the early stages of a disturbance, these faults will not result in a failure or malfunction as defined by [13] since the control algorithms are sophisticated enough to handle changes in measurement and control values.

Besides increasing reliability of the system, utilizing redundant hardware can identify faults by comparing the outputs of two or more substituting systems. Unfortunately, a redundant setup increases costs, complexity, required space, and system weight [19].

Thus, robots often realize fault detection by setting thresholds on sensor data, which should not be exceeded [23]. Yet, simply applying thresholds disregards dependencies between different data and requires unacceptably large possible data ranges to avoid frequent false positives. Especially for robots, the acceptable data range strongly depends on the particular configuration.

Moreover, differences to concurrently running simulators give insight in aberrations. These model-based techniques vary from comparing the state estimates to the actual states and only accepting residuals below a predefined threshold [21], over imprecise models accepting measured values within a specified bandwith [20], to robot state estimating algorithms like particle filters [25]. Some of these approaches can only recognize faults with known failure patterns [9], which facilitates an immediate diagnosis of the fault, and others have a specific state for unmodeled faults [24].

However, all of these approaches have drawbacks. Simulations are merely approximations of the real system; The more detailed the model, the more cumbersome the generation and the more expensive the evaluation. Yet, the performance of model-based fault detection depends on model accuracy. Besides, using models implicitly defines the detectable faults, even if no specific fault pattern is required. Only those errors influencing modeled relations can be detected. E.g. if nothing but the relationship between desired Cartesian end effector position and joint position is modeled, a gear slippage forcing the controller to increase the impelling current will remain undetected until further damage is caused to the system.

Therefore, we focus on a data-driven approach. A major challenge is the impracticality to generate and record fault data. Such data are often caused by hardware failure, which can hardly be emulated (for instance, deliberate destruction does not reflect the effects of wear), but is also a combination of many different states, which cannot be exhaustively emulated. Similarly, the number of valid configurations is very large, and not all valid data combinations are seen during normal operation. Depending on the robot the data space can easily cover several hundreds of dimensions.

We introduce a semi-supervised Robot Anomaly Detection System (RADS) which incrementally learns valid data during normal operation, building up a compact representation of these states. After switching from training to application, aberrations of these are detected and correspondingly labeled. The method is validated with a KUKA/DLR Light-Weight Robot (LWR) III arm [3], but is applicable to other input-output systems.

The presented approach is aiming for the recognition of anomalies not yet prohibiting the proper operation of the robot, but potentially leading to increasing problems. In contrast to many fault identification and diagnostic systems, it does not model distinct fault states allowing for a clear diagnosis. However, it can detect faults not considered during modeling or occluded by model assumptions. Moreover, it is conceivable to employ RADS to recognize improper use of the robot in tele-operated systems, e.g. deviations from the common workflow. Thus, it should be considered an additional tool to ensure system reliability, but not as an ultimate replacement for current fault detection, isolation, and identification methods.

RH, HU, and JK are with the Institute of Robotics and Mechatronics, DLR / German Aerospace Center, {rachel.hornung, holger.urbanek, julian.klodmann}@dlr.de

CO and PvdS are with the Faculty for Informatics, Technische Universität München, {osendorf@in., smagt@}tum.de

PvdS is further with fortiss, An-Institut Technische Universität München

II. State of the Art

Few publications deal with the problem of anomaly detection in high-dimensional data while only positive samples are available. Using the following six requirements for novelty detection stated by [16] existing methods are assessed (see Tab. I):

1) robust trade-off: exclude novel samples, yet include known,
2) generalization: avoid false positives and negatives,
3) adaptability: capable to incorporate new information,
4) minimized complexity: applicable for online evaluation,
5) independence: handle varying dimensions and features,
6) parameter minimization: little input from the user.

Uniform data scaling further requested by [16] is part of preprocessing [26] rather than the novelty detection algorithm, and makes the application of many algorithms easier.

Statistical methods as aggregated by [5] evaluate whether a new data point belongs to the same distribution as the training data. Presuming the type of distribution is especially difficult for high-dimensional data, as well as the construction of hypothesis tests.

Density estimation methods such as Parzen windowing [4] do not scale with data dimensions. The same holds for variational Bayesian techniques [4]. Furthermore, the created clusters span over large, sparse areas, leading to overgeneralization.

Clustering algorithms such as k-means [15] require prior data space knowledge, e.g. the number of centers that make up the data model. More elaborate clustering algorithms, e.g. growing neural gas [17], can adapt the number of cluster centers but require many equally distributed data.

One-class Support Vector Machines [22] lead to an unmanageable number of support vectors, and compel a specified percentage of the training data to be considered as outliers. Extreme Learning Machines [12] use single-hidden layer feed-forward networks to approximate the data distribution, and are only applicable if counter examples are available.

Other algorithms inherently deal with time series effects, e.g. Peer Group Analysis [8]. However, these methods compare similarly behaving dimensions against each other, and classify sudden divergence as suspicious. In robot anomaly detection, many of the dimensions cannot be compared by common distance metrics as Euclidean distance.

TABLE I
Anomaly detection algorithms in the light of requirements

method	violated requirements
statistical methods	5, 6
density estimation	4
variational Bayesian techniques	1, 2, 4
clustering	1, 2, 6
one-class Support Vector Machines	1, 3
Extreme Learning Machines	5
Peer Group Analysis	5
genetic algorithms	3, 4
Replicator Neural Networks	1
Sliding window Mahalanobis threshold	1

[26] offers an extensive overview of outlier detection methods especially for high-dimensional data. However, either these methods require outliers to be present during training, or they evaluate test data with respect to all training points resulting in too slow evaluations, or the approaches use only selected subspaces of the data. E.g. [2] consider both high dimensionality and single-class discrimination by genetic algorithms, which project data onto several lower-dimensional subspaces. The idea is promising but very time consuming, and only those faults differing in the selected subspaces will be identified [26]. [10] use a multi-layer feed-forward network, called Replicator Neural Network, to compress and restore the data (similar to the method introduced in Sec. III-D). However, this technique cannot detect faults exhibiting a similar intra-data relationship as the training data.

In 2011 [14] have introduced another model-free approach to anomaly detection. They learn the distribution of measurements and control commands from the latest data history and use a similarity threshold on the Mahalanobis distance to evaluate whether new data points fit into that distribution. The algorithm has returned very good results in their evaluations. However, as this method is based on windows over the latest history, slinking appearance of errors, like wear, will not be detected. In the beginning these faults will not differ enough to be considered an anomaly, but the further the error proceeds, the more of it will be contained in the latest history and thus distort the training data. Although the training step in this algorithm is computationally efficient, it has to be repeated for each measurement, thus prohibiting too high update rates—in the paper the assessment is repeated with a frequency of at most 10 Hz.

None of the available approaches sufficiently considers all of the six requirements while being able to handle high-dimensional one-class data and preventing adjustment to faults (cf. Tab. I). The aim of this paper is to introduce a system that is capable of detecting anomalies in high-dimensional data while the robot is operating. First, a robust detection mechanism is developed. Subsequently, applicability to high-dimensional data is ensured. Each concept of RADS is evaluated according to the fulfillment of the above listed requirements and the detection capability to prove its satisfactory performance for anomaly detection.

III. Machine Learning-Based Anomaly Detection

We tackle the problem with a multi-stage approach. Our method starts off by calculating maps of valid and invalid states from the robot data; inter alia joint torques, motor torques, currents, temperatures in the joints, and the applied load. Either of the two maps can identify faults, but lack computational efficiency. Thus, the maps are used to generate two labeled classes. These classes are then separated using a Support Vector Machine (SVM) ensuring a quick decision. Whenever new information is available, e.g. because the

robot is equipped with a new tool or the robot is operating in a different area of the workspace, the system can be amended without requiring the former training data. Finally, dimensionality reduction ensures applicability to high-dimensional data.

Positive data comprises all areas of the data space accessed during regular use. Naturally, this space should be maximally sampled. In contrast, *negative data* looms in unknown areas of the data space. Any data point classified thus requires further investigation: Either the maps are incomplete and have to be amended or an invalid state is detected.

A. A Map of Positive Data

To test whether a data point is within the positive data it could be compared against all valid points. However, an infinite number of test points would be required to cover all possible measurement variations making training and evaluation infeasible.

Instead, we approximate the positive data space by summing K radial basis function kernels. The map assigns each point $\boldsymbol{x}$ in the data space a function value corresponding to the probability of belonging to the trained data. If this value is above a threshold α, $\boldsymbol{x}$ is considered known. Otherwise the data point is an outlier. The probability is evaluated using the χ^2 goodness-of-fit test [18].

Each kernel κ_k has a specific center $\boldsymbol{\mu}_k$ and variance Σ_k. The function value $\phi(\boldsymbol{x}, \kappa_k)$ of a data point $\boldsymbol{x}$ in a kernel κ_k only depends on the Mahalanobis distance,

$$m_k = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k)}, \tag{1}$$

to the respective kernel center and the dimensionality d of the data space:

$$\phi(\boldsymbol{x}, \kappa_k) = Q_{\chi^2}\left(m_k, d\right). \tag{2}$$

$Q_{\chi^2}\left(m_k, d\right)$ is the χ^2 goodness-of-fit test probability, describing the likelihood that a point with distance m_k to a kernel belongs to same distribution as the respective kernel in d-dimensional space.

Combining multiple kernels a multi-modal and multivariate distribution can be modeled:

$$\mathrm{rbf}(\boldsymbol{x}) = \sum_{i=1}^{K} \phi(\boldsymbol{x}, \kappa_k). \tag{3}$$

Although a single kernel might not surpass the threshold at a specific point, it is possible that the combination of several kernels still identifies this data space as known.

Training is performed sequentially. Once trained, the data can be discarded. For each training point $\mathrm{rbf}(\boldsymbol{x})$ is evaluated, i.e. $\boldsymbol{x}$ is tested against all kernels. If

$$\mathrm{rbf}(\boldsymbol{x}) > \alpha, \tag{4}$$

with α as threshold, $\boldsymbol{x}$ is situated in a known area, otherwise the data point is considered unknown.

If Eq. (4) rates $\boldsymbol{x}$ unknown, a new kernel κ_{new}, covering the area around $\boldsymbol{x}$, is added to the network. The initial center $\boldsymbol{\mu}_{\mathrm{new}}$ of the new kernel κ_{new} is placed at the position of $\boldsymbol{x}$. The covariance matrix has to be initialized in a way that the typical noise variation a, but not more, is covered. Accepting noise-like divergence establishes a d-dimensional ellipsoid of recognition around the initial data point. Any point within its border will be considered known. Assuming a normal distribution of measurement errors around the actual value, choosing a Gaussian kernel to include all neighbors is well suited. This does not imply that the underlying data distribution has to be Gaussian; indeed, the data can be arbitrarily distributed.

If $\boldsymbol{x}$ is in a known area, the center and the covariance of the corresponding kernel are adjusted:

$$\boldsymbol{\mu'}_i = \frac{\eta_i \boldsymbol{\mu}_i + \boldsymbol{x}}{\eta_i + 1}, \tag{5}$$

where $\boldsymbol{\mu}_i$ is the center of the kernel at the time of observation, $\boldsymbol{\mu'}_i$ the updated center, and η_i the number of points that have already contributed. The covariance has to be adjusted accordingly, ensuring that all points, covered before, will be covered after the transformation.

To test whether a data point is represented by the map during the application phase only Eq. (4) applies. If the probability threshold is surpassed, the point is accepted as inlier.

In case the map has to be extended, the same process as for initial training can be applied. The old map is used as starting point and, where necessary, amended c.q. extended.

B. A Map of Negative Data

A map of the unknown data space can also classify the data adequately. Instead of modeling a data distribution, owed to the fact of missing fault data, a geometric description of the negative space is desired. One way to generate such a map is Negative Selection (NS) [7]. Any space not covered by (positive) training data is filled with detectors reacting once a data point is within their area of influence. If a detector is activated during the application phase, an anomaly is detected.

A detector τ_i is modeled as a sphere with center $\boldsymbol{c}_i$ and radius r_i. The detector τ_i is *activated* whenever an observation is within the sphere, i.e. its distance to $\boldsymbol{c}_i$ is smaller than r_i.

Since there is no information on how the negative space is set up, a detector is generated by first drawing its center $\boldsymbol{c}_i$ uniformly from the sample space, determined by the minimum and maximum values of the *positive* data in each dimension. To avoid redundancy a new $\boldsymbol{c}_i$ is discarded if it activates any of the existing detectors. Otherwise, it has to be ensured that none of the positive data points is too close: $\boldsymbol{c}_i$ must have at least a distance of $r_{\min} + a$. The noise amplitude a ensures that data points differing less from the training data than the typical noise range will not activate the detector; $r_{\min}$ guarantees that τ_i has a minimum area of influence. Finally, the radius r_i of τ_i has to be determined. Since valid assumptions about the structure of the error space are intangible, a proportional divergence in all directions is

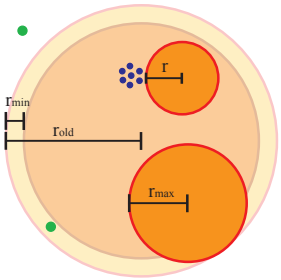


Fig. 1. NS Update Example: any of the small blue dots, representing new data, caused the large, pale detector in the back to turn invalid. Two exemplary replacements (bright orange) have been inserted. The radius of the smaller one is defined by the minimum distance to the training data, the larger is bound by size of the old detector. The two green potential detector centers have been discarded. One is outside of the bounding hull, one is too close to the hull. More replacement detectors have to be generated until the detector is sufficiently covered.

assumed and thus r_i is set to be the minimum distance to the training data D shielded by a:

$$r_i = \min_{\boldsymbol{x} \in D} \|\boldsymbol{c}_i - \boldsymbol{x}\| - a. \tag{6}$$

Accepting radii larger than the minimum radius reduces the number of required detectors to cover the invalid states. An established detector is not changed anymore. Detector generation is continued until the desired coverage c of the data space is reached, i.e. $1/(1-c)$ potential detectors have been discarded in a row. Once detector generation completed the training data can be discarded.

Since NS generally requires all data to be available during training, map updating cannot be conducted in the same way as initial training. Instead, the reaction of the detectors to each data point of a new batch of training data is evaluated.

If none of the detectors react, the point can be discarded without further processing. Otherwise, each *activated* detector is treated separately. If the active detector has only the minimum radius $r_{\min}$, no smaller detectors can be created to replace the old one, thus the detector is removed. Otherwise replacement detectors are generated. Since the available information on the space around the detector is insufficient given only the new batch of training data (the original training data has been discarded), a new detector may cover at most the same area as the old detector. Thus,

$$\|\boldsymbol{c}_{\text{new}} - \boldsymbol{c}_{\text{old}}\| < r_{\text{old}} - r_{\min} \tag{7}$$

with τ_{new} being a new, valid detector having the center $\boldsymbol{c}_{\text{new}}$, and the old detector τ_{old} with center $\boldsymbol{c}_{\text{old}}$ and radius r_{old}. $\boldsymbol{c}_{\text{new}}$ is drawn uniformly from within the sphere of τ_{old} (Fig. 1).

Similar to the previously described approach on the original data, a newly generated center is compared to the new training data and maintained if it is not too close to it. Because the new detector can not exceed the original one, the maximum radius $r_{\max}$ the new detector can have is the difference between the old radius and the distance of the two centers:

$$r_{\max} = r_{\text{old}} - \|\boldsymbol{c}_{\text{old}} - \boldsymbol{c}_{\text{new}}\|. \tag{8}$$

The new radius r_{new} will be the lower value of $r_{\max}$ and the minimum distance to the new training data r_i (Eq. (6)).

C. Fast Differentiation Between Two Classes

Both, the RBF-based approach and NS, provide stable error detection results. However, they are computationally intensive. While training may take a long time, the decision whether the actual data represents a fault has to be executed at a 1 kHz rate (for the LWR). In order to meet the required decision time frame, but still benefit from considering positive and negative data, we use the two maps as labled data to train an SVM, reverting to the standard libsvm [6] implementation.

We have chosen Gaussian kernels for the SVM and the parameters—C for misclassification punishment and σ^2 for kernel width—are determined by using a grid search together with cross validation.

D. Upgrade for High-Dimensional Data

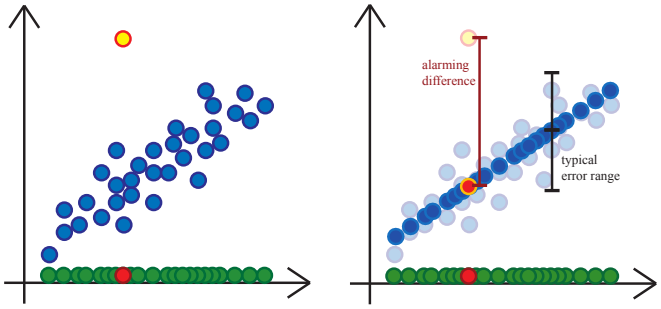
The previously defined RADS approach works well for low-dimensional observations. In order to evaluate complex behavior over time, the dimensionality increases to several hundreds of dimensions, leading to two major drawbacks.

On the one hand, calculations in the high-dimensional data space are expensive, prohibiting online evaluation of the data points. On the other hand, typical distance metrics like the Euclidean, respectively Mahalanobis distance have little meaning in high dimensions [1], [26].

1) Dimensionality Reduction: The simplest way to overcome the problem of relevance in high dimensionality is to use a distance metric that is less influenced by the number of dimensions than the Euclidean distance. [1] introduce fractional distances as a promising alternative. Switching the norm may improve the discrimination of data, but it does not solve the problem of computational complexity.

In contrast, dimensionality reduction can decrease the time required for testing, as d decreases. However, this introduces problems of its own. The only data available during training is valid data, therefore it is unknown how an error will appear. While projecting the known data onto a lower-dimensional manifold is straightforward, it is impossible to predict the projection behavior of errors. Projection might place a negative sample into the area of positive data, as depicted in Fig. 2a, making a decision about validity impossible.

In order to benefit from projection, back projection is essential. Back projection tries to move the data point back to its original position in the high-dimensional data space. If the point is an inlier, back projection will work fine and the point is returned to a position close to its initial location. However, an outlier projected onto the manifold will be reprojected to a part of the space related to inliers (cf. Fig. 2b). During training the system learns the regular back projection error. If a test error surpasses the acceptable value, the point is considered an outlier.



(a) Problems of dimensionality reduction: The correlated, blue 2D data points are projected onto a single dimension (green). However, the yellow outlier is also projected into the same area (red).

(b) Reprojection overcomes lack of projection: The green data is reprojected into the 2D space. A small error remains after reconstruction. The outlier is projected into the same area.

Fig. 2. Projection vs. Reprojection

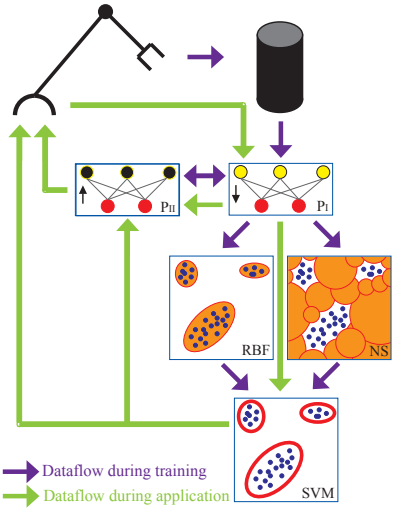


Fig. 3. Setup of HDRADS: First, projection P_I and reprojection P_{II} are computed. Using P_I the data is reduced and the basic RADS is applied. For evaluations the data is projected, evaluated with the SVM, and reprojected. If either the SVM or reprojection identify an anomaly, an alarm is issued.

2) Linear Projection: One type of dimensionality reduction is linear projection. The originally many dimensions are mapped onto a lower-dimensional subset of dimensions, e.g. through Principal Component Analysis (PCA) [4].

The transformation matrix P, expressing the principal components in the original data space, is determined from the training data. Reprojection is achieved using its pseudo-inverse $P^\dagger$. Since P is constant, $P^\dagger$ is calculated offline.

Untrained valid data should behave similarly to training data. Since projection is capable of projecting untrained, but comparably behaving data, it decreases the required amount of training setups and improves generalization.

Linear projection is well-suited to handle direct relations like the dependency between desired and actual current and can diminish constant values. However, PCA cannot exploit non-linear constraints, e.g. between joint angle and Cartesian position. Thus, a projection covering most of the variance will need more principal components than latent variables exist.

However, our initial results using autoencoders and stacked autoencoders [11] for non-linear projection do not exhibit any advantage within the data that was available. Thus, we confine ourselves to linear projection.

3) Combining RADS with Dimensionality Reduction: Although observing the reconstruction error from dimensionality reduction is auspicious, it is not sufficient. If an outlier is projected to a point outside of the projected training data, reprojection might locate it close to its original position. The reconstruction error would be in the acceptable range.

Combining dimensionality reduction with the previously introduced RADS in RADS for High-Dimensional Data (HDRADS) merges the advantages of both tools. Dimensionality reduction and reprojection decrease the time required for computation and identify outliers projected into the training data. The SVM discriminates the space of projected training data from the unknown projection space. Besides, projecting the data will reduce the number of required training samples due to improved generalization. It suffices to train on a few different configurations of the robot, rather than having to learn every possible setup, in addition reducing the need for retraining. Fig. 3 depicts the combined setup.

Data generated by the robot is recorded for training. First, a projection (P_I) and the corresponding backprojection (P_{II}) are calculated. Afterwards, the training data is reduced using P_I. The reduced data set is employed to create the positive (RBF) and negative (NS) maps, which are used to train the SVM. During the application, the data is projected with P_I, evaluated with the SVM, and reprojection with P_{II}. If either the SVM or reprojection classify the data as outlier an alarm is issued.

IV. Experiments

In the following, all five methods (positive map alone, negative map alone, RADS, reprojection alone, and HDRADS) are compared with regard to their anomaly detection performance. All tests have been conducted for an LWR [3].

A. Experimental Setup

Initial tests have been conducted running a simulation of the robot with 71 computed data dimensions. Several point-to-point trajectories have served as training data. Perturbing the simulation during test runs (adding constants and sine waves to single dimensions, or replacing data by constants) on the same trajectories has clarified, that unusual changes in a single dimension suffice to cause a RADS alarm while undisturbed data is recognized as valid—we can achieve true positive rates of 1, yet keeping the false positive rate at 0.

Although using the simulation facilitates the assessment according to a ground truth and the exact identification of false positive and negative rates, it is difficult to introduce realistic faults. Just to name a few of the necessary considerations: Which data dimensions are affected simultaneously? How is the correlation between these influences? What is the

a realistic fault behavior? Hence, the evaluations would forfeit validity and relevance, and we refrained from extending the simulation-based evaluation.

In order to assess the anomaly detection performance on authentic data, the data produced by a real robot during the motion along given trajectories has been recorded. To cover a wide range of the data space each of the robot joints has been moved from upper to lower joint limit and vice versa. The robot has remained at the limits for five seconds, before backtracing the trajectory.

Testing is performed against different recordings of the same trajectories. Generating real, yet predictable faults—to retrieve a ground truth—would require arbitrary damage to the hardware. However, that is without the bounds of possibility. Thus, we change hardware specific parameters: we increase and decrease in the maximum velocity, change the load applied to the endeffector, or switch the controller of the robot. Besides, in order to simulate acute and non-constant faults, collisions have deliberately been induced during an additional recording of the same trajectories by gently hitting the robot.

Without supplementary generated dimensions (*full state*), the data has 63 dimensions comprising control input and measurements. In order to observe the relationship between succeeding data points, the *enhanced state* set further contains first and second order derivatives of each dimension—except for quasi-constant dimensions like the currently applied load—and consists of 176 dimensions. As stated in [14] using the time derivative is reasonable, as when dealing with robots often the changes in states are of greater interest than the actual states. Each method is trained separately with *full* and *enhanced state* data.

B. Results of Basic RADS and its Components

Tab. II displays the percentage of data points that have been considered anomalous in the *full state* test data using the positive and negative map and the SVM, resp. RADS. The amount of resulting reactions for the *enhanced state* data are higher, but cover the same areas. For the validation set the same randomly selected 30% of the training data have been withheld from the algorithms during training. The results on the validation set show that the algorithm has a low rate of false positives if the trajectory is known, as none of the validation data points should be unknown. The table

TABLE II

ANOMALY ALARMS IN PERCENT OF THE NUMBER OF DATA POINTS. PARAMETERS USED FOR THE EVALUATION: POSITIVE MAP: $a = 0.2$, $\alpha = 0.05$; NEGATIVE MAP: $a = 0.2$, $r_{\min} = 0.2$, $c = 0.99$; SVM: $C = 1$, $\sigma^2 = 1/d$

Test Set	RBF	NS	SVM
validation	0	0.1	0
faster	19.8	15.1	0.001
slower	16.5	27.1	0.01
control	0.2	1.9	0
load	100	100	100
collision	3.2	1.9	1.3

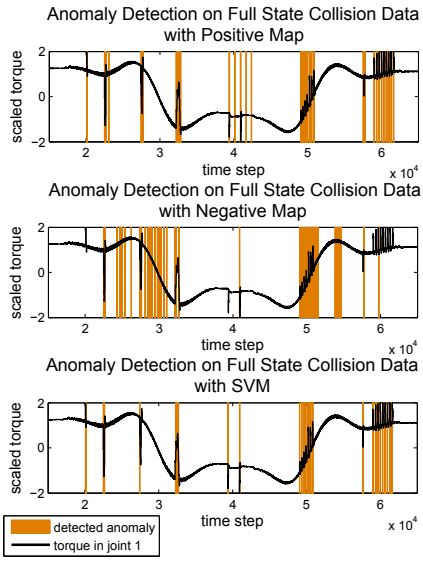


Fig. 4. Anomaly sensitivity of the different RADS components

further indicates, that an error has to be more pronounced to be detected by the SVM. The two preliminary stages assume that the majority of the movements are anomalous when a different velocity is used since currents, etc. vary from the trained data. They only accept static positions, which we expect to have the similar data values as in the training speed. The SVM, in contrast, fails to identify the comparatively marginal deviation. Whereas applying a different load to the endeffector influences the data dimensions more strongly and thus the entire sequence is considered anomalous. Of course, the load affects each data point in the test set, and thus anomalies on the entire sequence are expected.

Fig. 4 depicts a segment of the collision trajectory evaluated with the RBF-based approach, NS, and an SVM. The orange bars in the background indicate a detected anomaly. The black graph in front displays the torque in joint one. Spikes in the graph hint at an induced collision. Even if the collision is induced at a distal link, it will be apparent in proximal joints. Thus, all ascertainable collisions of this setup are visible in these graphs. The positive map and the SVM detect all collisions. Yet, the positive map shows a slightly higher amount of false negatives. NS lags behind the other two algorithms: while the robot is moving, the rate of false positives is much higher. Solely applying NS, a sufficient amount of valid training data would have to be higher. However, the RBF-based version is too computationally expensive, and the SVM cannot be trained without NS. Using the information from both preliminary stages, the SVM is able to differentiate between valid and anomalous states most accurately.

Using the *enhanced state* data the algorithm shows a marginal change of behavior. Due to the filter applied to smooth the data, the algorithm recognizes errors whenever they impact the filtered data. Using RADS with both data sets concurrently, the *enhanced state* data observation will react with a slight delay until a sufficient impact of the disturbance

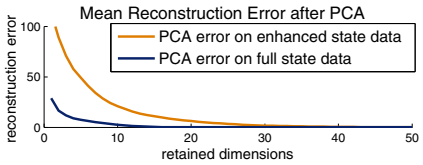


Fig. 5. PCA reconstruction error on training data vs. retained dimensions. The reconstruction errors between 50 and 63 retained dimensions for the *full state* data and between 50 and 176 for the *enhanced state* data are very close to zero, and are cropped for better visibility of the changes in lower dimensions.

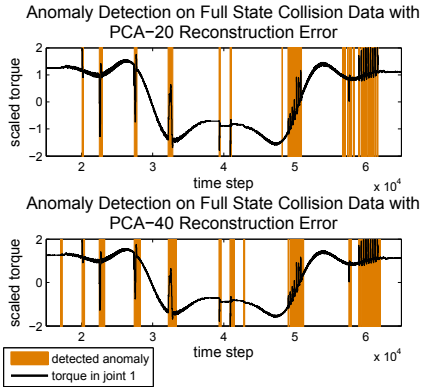


Fig. 6. Anomaly sensitivity based on the number of retained dimensions

is perceived. It will also notice a stop of the disturbance a little later than the *full state* observation because the filter needs to compensate the error impact.

C. Results using the Reprojection Error

As expected, the reconstruction error increases as fewer dimensions are retained during projection. Fig. 5 shows the reconstruction error resulting from PCA depending on the number of maintained dimensions.

The anomaly detection capabilities of (re-)projection alone are depicted in Fig. 6. The graph on top results from PCA with 20 retained dimensions, the one below from retaining 40 dimensions. The fewer dimensions are retained, the higher the typical error. Thus, to be identified as an error, the data points have to diverge more leading to smaller areas of detection. As in RADS, the *enhanced state* data have resulted in the same areas of detection—more pronounced due to filtering.

D. Results of HDRADS

For the evaluation of its generalization capability, HDRADS is trained with the same trajectory recorded at two different speeds, 0.2 m/s and 0.4 m/s. For the evaluation a third velocity, 0.6 m/s, is introduced. Neither reprojection nor RADS on the reduced data return any anomalies, although only two of all possible velocities have been used for training—the false positive rate is at 0.

The generalization of the algorithm improves, as it learns to generalize between different setups, in turn raising the alarm less often. Although fewer false positives are detected,

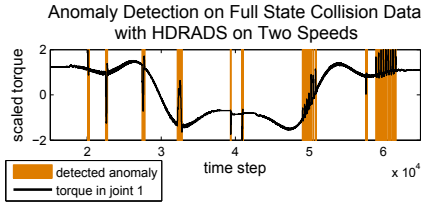


Fig. 7. Anomaly alarms from HDRADS during collision trajectory

overgeneralization is avoided. Fig. 7 displays the detected anomalies on a collision trajectory at a trained speed.

E. Evaluation of Experiments

Tab. III assesses HDRADS and its precursors with regard to the six requirements presented in Sec. II. The RBF-based approach identifies anomalies well while recognizing *positive data*, but is too computationally intensive in the test phase. NS detects too many anomalies or would require a higher amount of training data, but is faster at evaluating test points. Using an SVM, we can further accelerate the evaluation. However, taken by itself adapting the classifier is tedious and generalization is limited as we only have positive training data. Besides, for many training points, the decision time of SVMs increases with the number of support vectors, respectively training points. The drawback of depending on two distinct classes is overcome by training RADS with the positive and the negative model. This way we increase robustness and generalization while rendering model updates easy. Using only the two maps to train the SVM its complexity is forced to be at most that of the single maps.

The major problem—little generalization between different data parameters—can be overcome by including projection. Yet, projection and reprojection alone are not fail safe as anomalies that are not projected to the positive data space might not be identified as such.

Combining the entire setup the most accurate anomaly detection even in high-dimensional data is achieved, and the generalization compares to that of projection. Additionally, all parts of the algorithm can be improved with subsequent data using the available positive and negative maps, without requiring the initial training data. The dimensionality reduc-

TABLE III

FULFILMENT OF THE REQUIREMENTS (R) INTRODUCED IN SEC. II. 1=ROBUSTNESS AND TRADEOFF, 2=GENERALIZATION, 3=ADAPTABILITY, 4=COMPLEXITY, 5=INDEPENDENCE, 6=PARAMETER MINIMIZATION, ++=FULL COMPLIANCE, +=GOOD, 0=SATISFACTORY, −=UNSATISFACTORY, −−=DEFICIENT

R	RBF	NS	SVM	RADS	PCA	HDRADS
1	+	0	+	++	+	++
2	+	+	+	+	++	++
3	++	++	−	++	0	++
4	−−	−	+	++	++	++
5	+	+	−	+	+	++
6	+	+	+	0	+	0

tion incorporated in HDRADS further ensures fast training of the single maps. In case dimensionality reduction has to be improved, the reprojected *positive* and *negative* maps are sufficient to improve the mapping and reestimate the reconstruction error. Since HDRADS applies dimensionality reduction and could use both, positive and negative data for training, it is the most independent algorithm of those presented. A minor flaw of HDRADS is related to the least important requirement. Parameters for projection, the positive and negative map, and the SVM have to be identified.

Concluding, for low-dimensional data depending on few parameters, RADS performs well. It ensures sufficiently fast detection of unknown errors to run in parallel to the test robot. The higher the initial dimensionality, the more important the application of HDRADS to ensure satisfactory generalization in the increased parameter space. None of the basic algorithms can outperform RADS or its enhanced version.

V. Conclusion

By means of its redundant setup HDRADS is capable of detecting unknown anomalies reliably. While the system generalizes well enough to classify data points similar to the training data as known, anomalous divergence from the training data will cause an alarm. Supported by the projection and reprojection steps, the algorithm can handle high-dimensional inputs and can generalize between different inputs, thereby greatly reducing the number of required training samples.

The algorithm meets the requirements introduced by [16]. Yet, most testing has been performed on repeating trajectories, ensuring the applicability of RADS especially for industrial robots. Tracing predefined trajectories containing critical setups before putting the robot in service can ensure system operability if training all possible configurations is infeasible.

Moreover, instead of providing pre-specified derivatives to the algorithm, it might be helpful to consider other features calculated on the latest data. Alternatively, investigating the preformance when providing data of entire time windows rather than single points in time to the algorithm might show strong increases in performance. Using HDRADS on windowed data would automatically account for data smoothing and identify correlations between time steps that we are not aware of.

In the following, continuous evaluation of HDRADS in parallel to state-of-the-art fault detection algorithms has to demonstrate the performance on subtle and or slinking faults, while facilitating the assessment of the effective benefit. The projection mechanisms introduced in Ch. III-D are sufficient for the test cases. Yet, only a small set of all possible techniques has been considered. Further investigation could reveal more suitable methods.

References

[1] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the Surprising Behavior of Distance Metrics in High Dimensional Space," in *Proc. 8th Int. Conf. Database Theory*, 2001, pp. 420–434.

[2] C. C. Aggarwal and P. S. Yu, "Outlier detection for high dimensional data," in *Proc. 2001 ACM Int. Conf. Management of Data*, 2001, pp. 37–46.

[3] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger, "The DLR Lightweight Robot: Design and Control Concepts for Robots in Human Environments," *Industrial Robot: An Int. Jour.*, vol. 34, pp. 376–385, 2007.

[4] C. M. Bishop, *Pattern Recognition And Machine Learning*, 1st ed. Springer, 2006.

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys*, vol. 41, pp. 15:1–15:58, 2009.

[6] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Trans. Intelligent Systems and Technology*, vol. 2, pp. 1–27, 2011.

[7] D. Dasgupta and S. Forrest, "Novelty Detection in Time Series Data using Ideas from Immunology," in *Proc. 5th Int. Conf. Intelligent Systems*, 1995.

[8] Z. Ferdousi and A. Maeda, "Unsupervised Outlier Detection in Time Series Data," in *Proc. 22nd Int. Conf. Data Engineering Wksps.*, 2006, pp. 51–56.

[9] P. Goel, G. Dedeoglu, S. I. Roumeliotis, and G. S. Sukhatme, "Fault Detection and Identification in a Mobile Robot using Multiple Model Estimation and Neural Network." in *IEEE Int. Conf. Robotics and Automation*, 2000, pp. 2302–2309.

[10] S. Hawkins, H. He, G. Williams, and R. Baxter, "Outlier Detection Using Replicator Neural Networks," in *Data Warehousing and Knowledge Discovery*, ser. Lecture Notes in Computer Science, Y. Kambayashi, W. Winiwarter, and M. Arikawa, Eds. Springer, 2002, vol. 2454, pp. 170–180.

[11] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 212, pp. 504–507, 2006.

[12] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Exteme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, pp. 489–501, 2006.

[13] R. Isermann, *Fault-Diagnosis Applications—Model-Based Condition Monitoring: Actuators, Drives, Machinery, Plants, Sensors and Fault-tolerant Systems*. Springer, 2011.

[14] E. Khalastchi, M. Kalech, G. A. Kaminka, and R. Lin, "Online anomaly detection in unmanned vehicles," in *Proc. of 10th Int. Joint Conf. Autonomous Agents and Multi-Agent Systems*, 2011, pp. 115–122.

[15] S. P. Lloyd, "Least Sqares Quantization in PCM," *IEEE Trans. Information Theory*, vol. 28, pp. 129–137, 1982.

[16] M. Markou and S. Singh, "Novelty Detection: A Review - Part 1: Statistical Approaches," *Signal Processing*, vol. 83, pp. 2481–2497, 2003.

[17] T. Martinetz and K. Schulten, "A "Neural-Gas" Network Learns Topologies," *Artificial Neural Networks*, vol. I, pp. 397–402, 1991.

[18] R. L. Plackett, "Karl Pearson and the Chi-Squared Test," *Int. Statistical Review*, vol. 51, pp. 59–72, 1983.

[19] B. Rinner, "Detecting and Diagnosing Faults (Guest Editor's introduction)," *Telematik*, vol. 2, pp. 6–8, 2002.

[20] B. Rinner and U. Weiss, "Online Monitoring of Hybrid Systems using Imprecise Models," in *Proc. 5th IFAC Symp. Fault Detection, Supervision and Saftey of Technical Processes*, 2003, pp. 1811–1822.

[21] R. Schneider and P. M. Frank, "Fuzzy Logic Based Threshold Adaption for Fault Detection in Robots," in *Proc. 3rd IEEE Conf. Control Applications*, 1994, pp. 1127–1132.

[22] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the Support of a High-Dimensional Distribution," *Neural Computation*, vol. 13, pp. 1443–1471, 2001.

[23] J. Stoustrup, H. Niemann, and A. la Cour-Harbo, "Optimal Threshold Functions for Fault Detection and Isolation," in *Proc. American Control Conf. 2003*, vol. 2, 2003, pp. 1782–1787.

[24] M. Wang and R. Dearden, "Detecting and Learning Unknown Fault States in Hybrid Diagnosis," in *20th Int. Wksp. Principles of Diagnosis*, 2009, pp. 19–26.

[25] K. Zhou and L. Liu, "Unknown Fault Diagnosis for Nonlinear Hybrid Systems Using Strong State Tracking Particle Filter," in *Int. Conf. Intelligent System Design and Engineering Application*, 2010, pp. 850–853.

[26] A. Zimek, E. Schubert, and H.-P. Kriegel, "A Survey on Unsupervised Outlier Detection in High-Dimensional Numerical Data," *Statistical Analysis and Data Mining*, vol. 5, pp. 363–387, 2012.