Single-Class Discrimination for Robot Anomaly Detection

Masterarbeit

Rachel Hornung

DLR

Institut für Robotik und Mechatronik	BJ.: 2012
	IB.Nr.:

MASTERARBEIT

SINGLE-CLASS DISCRIMINATION FOR ROBOT ANOMALY DETECTION

Freigabe:	Der Bearbeiter:	Unterschriften
	Rachel Hornung	_____
	Betreuer:	
	Dipl. Inf. Holger Urbanek	_____
	Dipl. Ing. Julian KLodmann	_____
	Der Institutsdirektor	
	Dr. Ing. Alin Albu-Schäffer	_____

Dieser Bericht enthält 44 Seiten

Ort: Oberpfaffenhofen	Datum: 29.10.2012	Bearbeiter: Rachel Hornung	Zeichen:

Preface

The thesis at hand is split into two parts. The first part comprising Ch. 1–Ch. 5 introduces a new Robot Anomaly Detection System (RADS) including setup, functionality, and an extension for high dimensional data, and further evaluates the performance in a paper wise manner. App. A–App. I constitute the second part, and offer supplementary details on the concepts, the data, and additional tests.

I would like to express my gratitude to all those who helped me to complete this thesis; especially Holger Urbanek who proposed the application of a Radial Basis Function (RBF) network, Julian Klodmann who made validation against real robot data possible, Christian Osendorfer who contributed ideas especially for projection, and Patrick van der Smagt who pushed the thesis to its final state; all of whom helped a lot by proof reading and counseling with regard to latex and document layout. Finally, I thank Christian Kerl who constantly encouraged me, yet critically questioned my text.

Abstract

Early system error detection is essential for safe human-robot interaction. Detecting errors in the large amount of sensory and status data is, however, difficult, especially since the amount of fault data is very small if at all available; a problem for which no sufficiently stable methods exist. This thesis introduces a new Robot Anomaly Detection System (RADS) consisting of a multi-stage solution, that adapts to high dimensional input and detects errors without previous records of them. Application to real robot data proves its satisfying performance.

Zusammenfassung

Während Mensch-Roboter-Interaktionen ist es sehr wichtig Fehler und Schäden möglichst früh zu erkennnen um die Sicherheit aller Beteiligten zu gewährleisten. Dies wird hauptsächlich durch die Beobachtung von Sensor- und Zustandsdaten erreicht. Der Masse an Daten Herr zu werden ist schwierig, insbesondere da es wenige Beobachtungen von Fehlerzuständen gibt, sofern sie überhaupt vorhanden. Bisher existieren keine ausreichend stabilen Lösungen für dieses Problem. Im Rahmen dieser Arbeit wird ein neues Fehlererkennungssystem für Roboter (RADS) vorgestellt, das auch Fehler, zu denen keine Daten vorliegen, erkennt und den hochdimensionalen Eingaberaum verarbeiten kann. Durch die Anwendung auf reale Roboterdaten wird belegt, dass die Methode zielführend ist.

Table of Contents

List of Abbreviations

AE	Autoencoder
DMFA	Deep Mixture of Factor Analyzers
DLR	German Aerospace Center
DoF	Degrees of Freedom
ELM	Extreme Learning Machine
EM	Expectation Maximization
ES I	Evaluation Set I
ES II	Evaluation Set II
HDRADS	Robot Anomaly Detection System for High-Dimensional Data
ICA	Independent Component Analysis
iid	independent and identically distributed
LDA	Linear Discriminant Analysis
LWR	Light-Weight Robot
MCA	Minor Component Analysis
MFA	Mixture of Factor Analyzers
ML	Machine Learning
NS	Negative Selection
PCA	Principal Component Analysis
PGA	Peer Group Analysis
RADS	Robot Anomaly Detection System
RBF	Radial Basis Function
SVM	Support Vector Machine
XCA	Extreme Component Analysis

List of Symbols

a	noise amplitude
α	level of significance
C	SVM parameter for misclassification punishment
c	desired data space coverage
$\boldsymbol{c}_i$	center of detector i
χ^2	chi-squared distribution
$\chi^2(\alpha, n)$	the maximum distance a point can have from an RBF kernel in n dimensions, yet having at least a probability of α of belonging to the same distribution
D	data set
$d_{[i,j]}$	the data value of training point i in dimension j
$\delta_{j_{\max}}$	maximum justifiable distance to a training point
$\delta_{r_{\max}}$	maximum recognized distance from a kernel center
ϵ	mean of reprojection distances
ε	error threshold for reprojection distance
η	number of points that have activated a kernel κ
f	number of time steps for filtering
h	number of latent dimensions
i	iterations before re-evaluating whether $Q_{\chi^2}(m_k, n) < \alpha'/K$
$\boldsymbol{\iota}$	direction of translation between two data points, $\|\boldsymbol{\iota}\| = 1$
K	total number of RBF kernels
k	parameter for Machine Learning algorithms like k-means and k-fold cross-validation
κ_k	RBF kernel k
l_μ	Euclidean length of the translation between two kernels
λ	maximum distance to the cluster center in non-parametric k-means
L^p	L^p norm
LT	linear transformation matrix, e.g. retrieved from Principal Component Analysis (PCA)
$\text{LT}^\dagger$	pseudo inverse of LT
m_k	Mahalanobis distance from point $\boldsymbol{x}$ to kernel κ_k
$\boldsymbol{\mu}_k$	center of RBF kernel k
n	number of dimensions
$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \Sigma_k)$	Normal distribution over $\boldsymbol{\mu}_k$ and Σ_k
ν	numerator
p	percentage of a data set
$\phi(\boldsymbol{x}, \kappa_k)$	function value of $\boldsymbol{x}$ in RBF kernel k
$\psi(\boldsymbol{x})$	non-linear mapping of data point $\boldsymbol{x}$
$\boldsymbol{q}$	point at the border of recognition of a kernel
$Q_{\chi^2}(m_k, n)$	probability that a point with distance m_k to a distribution belongs to the same distribution if the data space has n dimensions
r_i	radius of NS detector τ_i
$r_{\min}$	minimum radius each NS detector has to have
$\text{rbf}(\boldsymbol{x})$	the response of the RBF network to data point $\boldsymbol{x}$
s	inverse of 1D variance, i.e. scale
σ	standard deviation
σ^2	1D variance
Σ_k	covariance of RBF kernel k
t	discrete point in time
$\tanh(\boldsymbol{x})$	hyperbolic tangent of $\boldsymbol{x}$: $\tanh(\boldsymbol{x}) = \sinh(\boldsymbol{x})/\cosh(\boldsymbol{x}) = (e^{2\boldsymbol{x}} - 1)/(e^{2\boldsymbol{x}} + 1)$. If $\boldsymbol{x}$ is a vector, the function is evaluated for each element of the vector
T	set of NS detectors
τ_i	NS detector i
w	number of sections per dimension for, e.g. Parzen windowing
$\boldsymbol{x}$	data point
$\boldsymbol{z}$	projected data point

1. Introduction to Anomaly Detection for Robots

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. Errors can be soft- and hardware-related and include communication problems, wear-dependent deviation, and broken sensors.

Typically, fault detection is realized by setting thresholds on sensory data, which may not be exceeded. Furthermore, differences to concurrently running simulators give insight in aberrations, i.e. [14, 19, 38, 44]. However, these approaches have drawbacks. Simply applying thresholds disregards dependencies between different data. Then, simulators are merely approximations of the real system, and errors which were not considered during modeling may be impossible to detect. Rather than relying on these coarse methods, this thesis introduces a model-free approach.

A major challenge is the impracticality to record fault data. Such data is often caused by hardware failure, which can hardly be emulated, but is also a combination of many different states, the combination of which cannot be spanned. Similarly, the number of valid configurations is very large, too, and not all valid data combinations are seen during normal operation.

The introduced Robot Anomaly Detection System (RADS) *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 the German Aerospace Center (DLR) Light-Weight Robot (LWR) [3] and the DLR MIRO [20], but can easily be applied to other input-output systems. Depending on the exact configuration and additional number of supplementary metrics, the data space can easily cover several hundred dimensions.

2. 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 Markou [35] existing methods are classified (see Tab. 2.1):

1. robustness and trade-off: excludes novel samples while including known states,
2. generalization: avoids false positives and negatives,
3. adaptability: capable to incorporate new information,
4. minimized computational complexity: applicable for online evaluation,
5. independence: handles varying dimensions and number of features,
6. parameter minimization: little input required from the user.

Uniform data scaling further requested by Markou is not contained in the novelty detection algorithm, but is part of preprocessing and makes the application of many algorithms easier (see App. B.1) [35].

Density estimation methods such as Parzen windowing [4] do not scale with data dimensions: Each dimension is split into w parts leading to a combinatorial explosion. The same holds for variational Bayesian techniques. Furthermore, the created clusters span over large, sparse areas, leading to over-generalization: Faulty data in these sparsely populated areas will be erroneously mapped to known models.

Clustering algorithms such as k-means [33] require prior data space knowledge, e.g. the number of centers that make up the data model. Also, the number of centers may grow too large. More elaborate clustering algorithms, e.g. growing neural gas [17, 37], can adapt the number of cluster centers but require a lot of equally distributed data.

Other algorithms inherently deal with time series effects, e.g. Peer Group Analysis (PGA) [15] applied in fraud detection. However, the 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.

One-class Support Vector Machine (SVM) [45] and Support Vector Domain Description [52] lead to an unmanageable number of support vectors, and force a specified percentage of the training data to be considered as outliers. Extreme Learning Machines (ELMs) [25], use single hidden layer feed-forward networks to approximate the data distribution, and are only applicable if counter examples are available.

The neural network predictor introduced by [51] classifies robustly, but can only be used if a good data model is available. Aggarwal et al. (2001) consider both high dimensionality and single-class discrimination by genetic algorithms, which project data onto several lower-dimensional subspaces [2]. The idea is promising but very time consuming.

None of the available approaches sufficiently considers all of the six requirements while being able to handle high-dimensional one-class data. The aim of this thesis is to introduce a largely deterministic RADS 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 the RADS is evaluated according to the fulfillment of the above listed requirements and its detection capability to prove its satisfactory performance for anomaly detection.

Table 2.1.: Robot anomaly detection algorithms in the light of requirements

method	violated requirements
density estimation	4
variational Bayes	1, 2, 4
clustering	1, 2, 6
PGA	5
one-class SVM	1, 3
ELM	5
genetic algorithms	3, 4

3. A Machine-Learning Based Anomaly Detection System

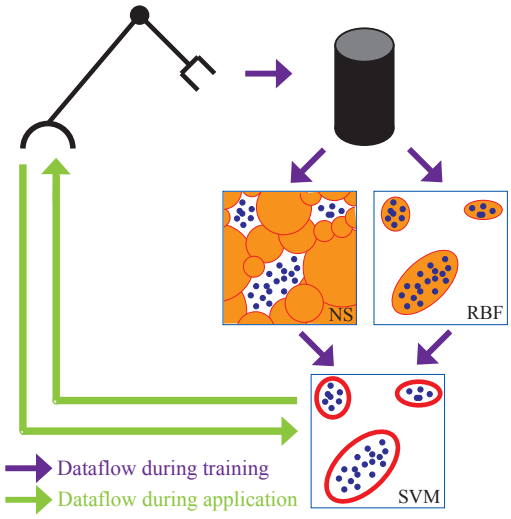


Figure 3.1.: Setup of the Robot Anomaly Detection System (RADS): Data generated by the robot is recorded for training. The data is used to create two models, one for valid data points (based on Radial Basis Functions (RBF)), one for invalid data points (based on Negative Selection (NS)). Once the models have been retrieved, they are used to train a Support Vector Machine (SVM). After training, only the SVM is necessary to decide whether any data point is erroneous.

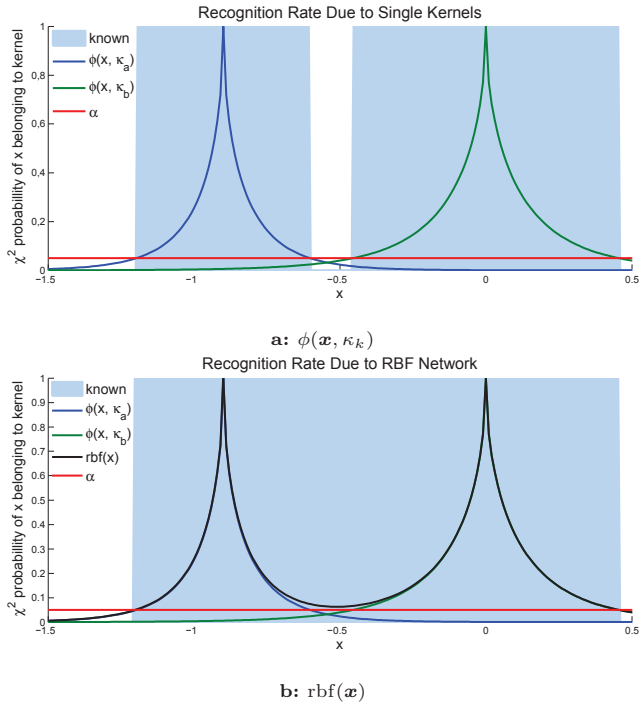
The problem is tackled by applying a multi-stage solution as depicted in Fig. 3.1. This method starts off by building models of valid and invalid states. These models are then used to generate two labeled classes that can be separated using a Support Vector Machine (SVM) ensuring a quick decision.

Positive data comprises all areas of the data space accessed during regular use. 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 model is incomplete and needs to be amended or an anomaly is detected.

3.1. A Model of Positive Data

The structure of the data is approximated by summing K Radial Basis Function (RBF) kernels, corresponding to multi-variate Gaussian functions. The model assigns each point $\boldsymbol{x}$ in the data space a function value according to the probability of belonging to the trained data. If the function value is above a predefined threshold α, $\boldsymbol{x}$ belongs to the model and is considered known. Otherwise the data point is an outlier. The probability is evaluated using the χ^2 goodness-of-fit test [41].

Each kernel κ_k has a specific center $\boldsymbol{\mu}_k$ and variance Σ_k. The function value $\phi(\boldsymbol{x}, \kappa_k)$ (see Eq. (3.2)) of a data point $\boldsymbol{x}$ in a kernel κ_k only depends on the Mahalanobis distance,



a: $\phi(\boldsymbol{x}, \kappa_k)$

b: $\mathrm{rbf}(\boldsymbol{x})$

Figure 3.2.: a: Two kernels constituting the network response. b: The black curve sums the two kernels and bridges the gap between the kernels. The kernels represent the χ^2 goodness-of-fit probability and thus are not Gaussian-shaped.

$$m_k = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k)}, \tag{3.1}$$

to the respective kernel center and the dimensionality n of the data space.

$$\begin{aligned} \phi(\boldsymbol{x}, \kappa_k) &= \phi(\boldsymbol{x}, \boldsymbol{\mu}_k, \Sigma_k) \\ &= Q_{\chi^2}(m_k, n) \end{aligned} \tag{3.2}$$

with $Q_{\chi^2}(m_k, n)$ being the χ^2 goodness-of-fit test probability that a point having a distance m_k to a kernel belongs to the same distribution as the respective kernel in an n-dimensional space.

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

$$\mathrm{rbf}(\boldsymbol{x}) = \sum_{i=1}^{K} \phi(\boldsymbol{x}, \kappa_k). \tag{3.3}$$

Although a single kernel might not surpass the threshold at a specific point in the data space, it is possible that the combination of several kernels still discovers this data space as known, as shown in Fig. 3.2.

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

$$\mathrm{rbf}(\boldsymbol{x}) > \alpha, \tag{3.4}$$

$\boldsymbol{x}$ is situated in a known area. Otherwise the data point is considered unknown[1]. If Eq. (3.4) rates $\boldsymbol{x}$ as unknown, a new kernel κ_{new}, covering the area around $\boldsymbol{x}$, is added to the network. Hereafter, $\boldsymbol{x}$ and its neighborhood will also be considered known.

The initial center $\boldsymbol{\mu}_{\text{new}}$ of 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, but not more, is covered. Accepting noise-like divergence establishes an n-dimensional sphere of recognition around the initial data point. Any point within this border will be considered known. The size of the sphere depends only on the initial covariance[2].

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

$$\boldsymbol{\mu}_{\text{new}} = \frac{\eta\boldsymbol{\mu}_{\text{old}} + \boldsymbol{x}}{\eta + 1} \tag{3.5}$$

where $\boldsymbol{\mu}_{\text{old}}$ is the center of the kernel at the time of observation, and η 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[3].

To test whether a data point is represented by the model during the application phase Eq. (3.4) applies. Only if the test point is closer to any kernel than the maximal distance or if the probability threshold is surpassed by the sum of several kernels, the point is accepted as an inlier.

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

3.2. A Negative Model of the Data

A model of the unknown data space can also classify the data adequately. One way to generate such a model is Negative Selection (NS) [11]. Any space not covered by (positive) training data is filled with detectors reacting once a new data point is within their area of influence. If a detector is activated during the application phase, a warning is issued.

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

A detector is generated by first drawing its center $\boldsymbol{c}_i$ uniformly from a sample space. The sample space is determined by the minimum and maximum values of the *positive data* in every dimension[4]. To avoid redundancy, $\boldsymbol{c}_i$ is discarded if it activates any of the existing detectors. If this is not the case, it has to be ensured that none of the positive data points are 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 of τ_i has to be determined. Since it is not possible to make any assumptions on the structure of the error, a proportional divergence in all directions is assumed and the radius r_i is set to be the minimum distance to the training data D shielded by a:

$$r_i = \min_{\boldsymbol{x} \in D} \|\boldsymbol{c}_i - \boldsymbol{x}\| - a. \tag{3.6}$$

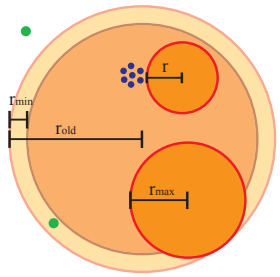


Figure 3.3.: NS Update Example: Any of the blue dots, representing new training points, caused the large, pale detector in the back to turn invalid. Two exemplary replacements have been inserted. The radius of the smaller one is defined by the minimum distance to the training points, the larger one is bound by the distance to the border of the old detector. The green potential detector centers have been discarded. One is outside of the bounding hull, one is too close to the hull. More detectors have to be generated until the detector is sufficiently covered.

In contrast to RBF kernels, an established detector is not changed anymore. Detector generation is continued until the desired coverage c of the data space, i.e. $1/(1-c)$ potential detectors have been discarded in a row, or the maximum number of accepted detectors is reached.

Since NS generally requires all data to be available during training, *model 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 any further processing. Otherwise, each *activated* detector is treated separately. If the radius of 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. In all other cases replacement detectors are generated. Since the available information on the space around the detector is insufficient given 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. This implies

$$\|\boldsymbol{c}_{\text{new}} - \boldsymbol{c}_{\text{old}}\| < r_{\text{old}} - r_{\min} \quad \forall \tau_{\text{new}} \in T \tag{3.7}$$

[1] Since no closed-form solution for the χ^2 goodness-of-fit test probability exists, calculating the probability of belonging to the network is the most expensive part of the evaluation. App. C.1.1 presents implementations to reduce the number of computations.
[2] Identifying suitable initial values can be done by grid search. App. C.1.2 presents a more data-driven approach to determine suitable parameters.
[3] App. C.1.3 provides detailed information on applicable covariance adjustment methods.
[4] App. D.3 states the restrictions on the sample space in more detail.

with $\tau_{\text{new}} \in T$ being a new, valid detector having the new 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. 3.3).

Similar to the previously described proceeding on the original data set 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 cannot 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:

$$r_{\max} = r_{\text{old}} - \|\boldsymbol{c}_{\text{old}} - \boldsymbol{c}_{\text{new}}\| \tag{3.8}$$

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. (3.6)).

3.3. Fast Differentiation Between Two Classes

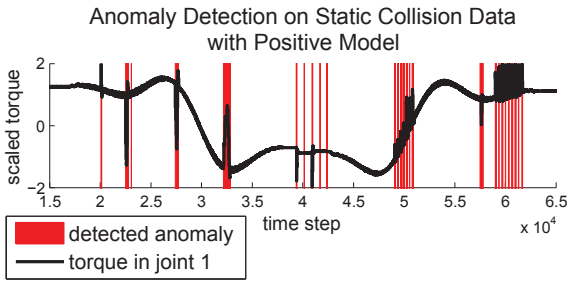
Both the RBF-based approach and NS provide stable error detection results. However, they are computationally intensive. While training may take a long time, the decision whether the actual data represents a faulty robot has to be executed in a 1 kHz rate (for the LWR). Reverting to a standard library for SVMs, libsvm [7], accelerates the decision.

For high-dimensional setups, it is sufficient to use the element centers (RBF and NS) as training data for the SVM since the data space in which the training data is located is sparse and few clusters represent the valid data. In lower-dimensional setups the space is much more populated and the boundary regions have to be defined more explicitly. Accordingly, each element needs to be sampled and those samples are then used for SVM training. Gaussian kernels have been chosen for the SVM and the parameters—C for misclassification punishment and σ^2 for kernel width—are determined by using grid search together with cross-validation[5].

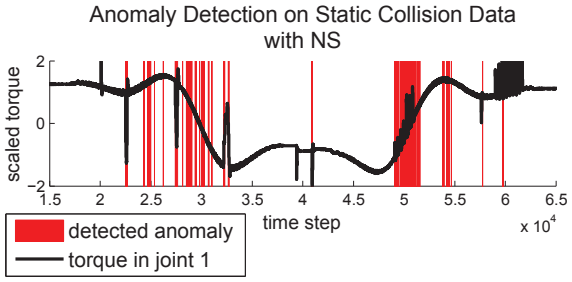
3.4. Evaluation of the Basic Approach

In the following, all three methods (positive model alone, negative model alone, and an SVM trained on valid and invalid data) are compared with regard to their anomaly detection performance. The tests have been executed on an LWR. Without additionally generated dimensions (*static*), the data has 63 dimensions comprising control input and measurements. The *dynamic* set further contains first and second derivatives of each dimension—except for constant dimensions—and consists of 176 dimensions[6].

Each of the methods has been trained with *static* and *dynamic* data on a pre-defined trajectory. Testing has been performed against the same trajectory with deliberate collisions, varying speeds, different controls, and applied loads.



a: Anomaly alarms from positive model during collision trajectory



b: Anomaly alarms from NS during collision trajectory

Figure 3.4.: RADS components sensitivity

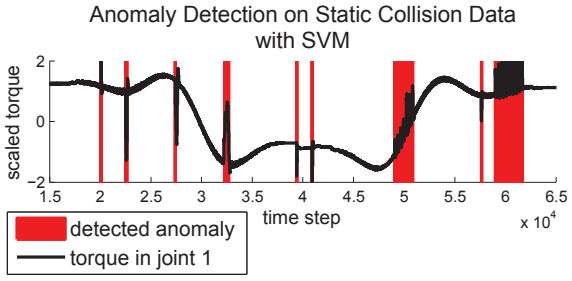


Figure 3.5.: Anomaly alarms from SVM during collision trajectory

Tab. 3.1 displays how frequently the different algorithms react to the static test data. The amount of resulting reactions for the dynamic test data are higher, but cover the same areas.

Fig. 3.4 depicts a segment of the collision trajectory evaluated with the RBF-based approach and NS, Fig. 3.5 displays the same for an SVM. The red bars in the background indicate a detected anomaly. The black graph in front displays the torque in one of the joints. Spikes in the graph hint at an induced collision. All apparent collisions are detected by the positive model

Table 3.1.: Anomaly alarms in percent of the number of data points: $a = 0.2$, $\alpha = 0.05$, $K = 13$; negative model: $a = 0.2$, $b = 0.2$, $c = 0.99$, $T = 3826$; SVM: $c = 1$, $\sigma^2 = \frac{1}{n}$, $SVs = 158$

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

[5] App. E offers additional information on the possibility to identify the parameters from the data.
[6] Additional information on the test set is provided in App. A.

and the SVM. NS lags behind the other two algorithms. 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 faulty data most accurately. Yet, as Tab. 3.1 shows, an error has to be more pronounced to be detected by the SVM. The two preliminary stages assume that the majority of the agitated sequences are faulty, and only recognize static positions since currents, etc. vary from the trained data. The SVM, in contrast, fails to identify the comparatively soft deviation.

In general, the *dynamic* data is more sensitive to outliers and does not only recognize errors the moment they are induced, but also slightly before and after, intensified due to the filtering for noise reduction[7,8].

[7] Results of MIRO tests are presented in App. H.1.
[8] App. G introduces methods to handle detected anomalies.

4. Upgrade for High-Dimensional Data

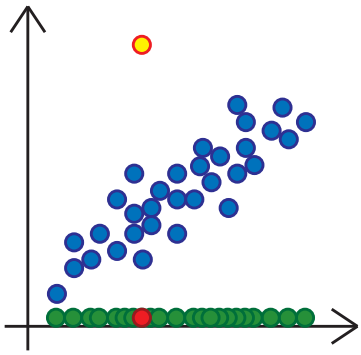


Figure 4.1.: Problems of dimensionality reduction: The correlated, blue, 2D data points are projected onto a single dimension (green). However, the yellow outlier is also projected into the same area (red).

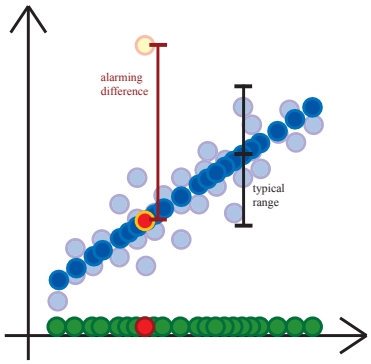


Figure 4.2.: Reprojection overcomes lack of projection: The green data is reprojected into the 2D space. A small error remains after reconstruction. The outlier is projected into the same area.

The previously defined RADS works well during *static* observation. In order to evaluate the dynamic behavior, 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 and time-consuming 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, 2].

4.1. 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. Aggarwal et al. (2001) introduce fractional distances as a promising alternative [1]. Nevertheless, switching the norm may improve the decision whether a point is good or bad, but it does not solve the problem of computational complexity.

In contrast, dimensionality reduction can decrease the time required for testing, as fewer calculations have to be performed on reduced data. Dimensionality reduction, however, introduces problems of its own. The only data available during training is valid data. Therefore, it is unknown how an error will present itself in the data. While it is fairly easy to project the known data onto a lower-dimensional manifold that represents the data, it is impossible to predict the projection behavior of errors. It may happen that the projection to fewer dimensions places a negative sample into the area of positive data, as shown in Fig. 4.1, making a decision about the 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. 4.2). During training the regular back projection error can be learned and if during testing the error surpasses the acceptable value, the point is considered an outlier[1].

4.1.1. Linear Projection

One kind 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) [40]. Other popular linear projection methods include Independent Component Analysis (ICA) and Linear Discriminant Analysis (LDA) [29]. LDA is inapplicable to this problem, as it searches for the vectors that discriminate different classes best [55]. ICA assumes that a maximum of one dimension is subject to Gaussian distribution, which according to App. A is much less accurate than the assumption of all dimensions being Gaussian as is the case for PCA [26].

The transformation matrix LT, expressing the principal components in the original data space, is determined from the training set. All data, including validation and test data, is right-multiplied to LT. Reprojection can be achieved by right-multiplying the pseudo-inverse $LT^{\dagger}$ of LT to the transformed data. Since LT is constant, $LT^{\dagger}$ is calculated offline.

Untrained, valid data should behave similarly to training data. Thus, linear projection should also be capable of projecting untrained data, in turn decreasing the required amount of training setups and improving generalization. Nevertheless, selection of training data

[1] Refer to App. F.1 for more details.

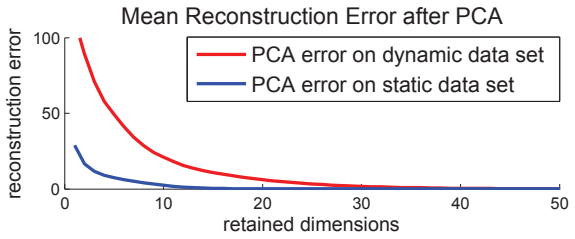
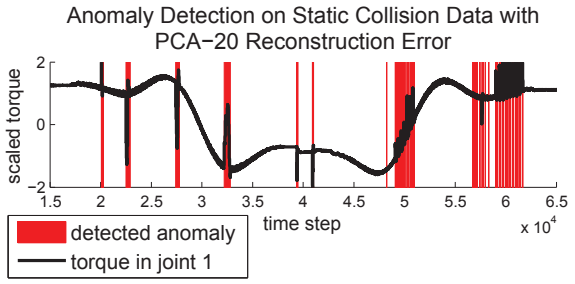
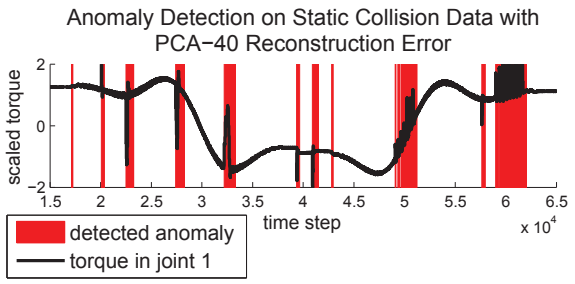


Figure 4.3.: PCA reconstruction error on training data vs. retained dimensions. The reconstruction errors between 50 and 63 retained dimensions for the *static* data and between 50 and 176 for the *dynamic* data are very close to zero, and are cropped for better visibility of the changes in lower dimensions.



a: Anomaly alarms from PCA to 20 dimensions during collision trajectory



b: Anomaly alarms from PCA to 40 dimensions during collision trajectory

Figure 4.4.: Projection sensitivity

has to be performed more carefully as retraining without all information is impossible for PCA.

Linear projections are 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 a wide range of the variance will need more principal components than latent variables exist. The initial results using autoencoders and stacked autoencoders [23] for non-linear projection do not exhibit any advantage within the data that was available[2]. Thus, the current implementation employs linear projection only.

4.1.2. Experimental Results

As expected, the reconstruction error increases as fewer dimensions from the initial data set are retained. Fig. 4.3 shows the Euclidean error resulting from PCA depending on the number of maintained dimensions.

[2] App. F.2.1 provides information on the examined algorithms.

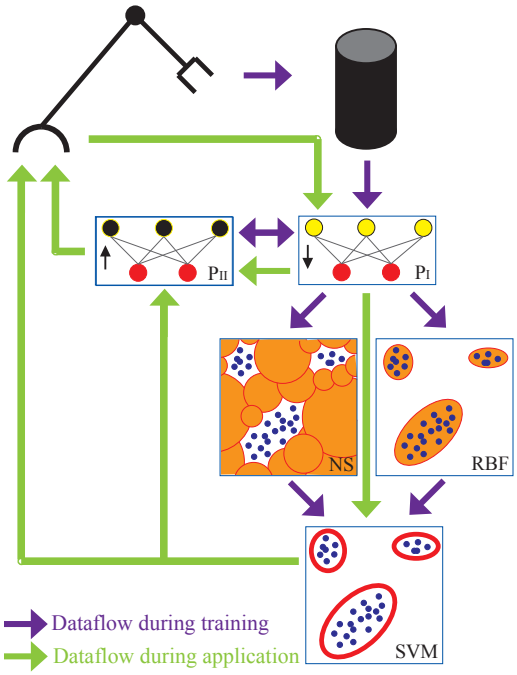


Figure 4.5.: Setup of the HDRADS: First, projection P_I and reprojection P_{II} are computed. Using P_I the data is reduced and the basic RADS is applied. For evaluations the data is projected, evaluated with the SVM, and reprojected. If either the SVM or reprojection identify an anomaly, an alarm is issued.

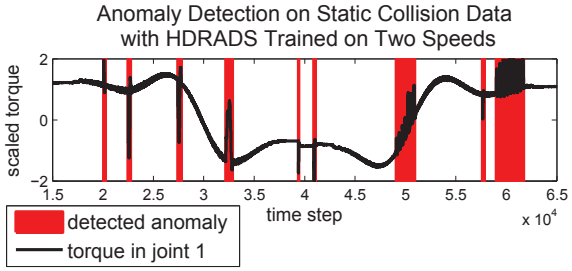


Figure 4.6.: Anomaly alarms from HDRADS during collision trajectory

The anomaly detection capabilities of (re-)projection are depicted in Fig. 4.4. 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 reconstruction error. Thus, to be identified as an error, the data points have to diverge more strongly leading to smaller areas of detection. As in the basic RADS, the dynamic data resulted in the same areas of detection—more pronounced due to the filtering before derivation.

4.2. Combining the RADS with Dimensionality Reduction

Although observing the reconstruction error from dimensionality reduction provides promising results, it is not sufficient. If an outlier is projected to a point outside of the projected training data, reprojection might locate it close to the original data point. Thus, the reconstruction error would be in the acceptable range.

4.2.1. Setup of the Algorithms

Combining dimensionality reduction with the RADS described in Ch. 3 merges the advantages of both tools. Dimensionality reduction and reprojection decrease the time required for computation and take care of outliers projected into the training data. The SVM discriminates the space of projected training data from the unknown projection space. Besides, projecting the data reduces the number of required training samples. 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. 4.5 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 and negative models, which are used to train the SVM. During the application, the data is projected with P_I, evaluated with the SVM, and if the data is considered valid, reprojection P_{II} is conducted to ensure that the data point is a true inlier. Outliers immediately cause an alarm.

4.2.2. Experimental Results

To evaluate the generalization capability of the Robot Anomaly Detection System for High-Dimensional Data (HDRADS), it is trained with the same trajectory recorded at two different speeds. For evaluation a third velocity is introduced. Neither reprojection nor RADS on the reduced data return any anomalies, although only a small subset of possible velocities is used for training. The generalization of the algorithm is improved, as it learns to generalize between different setups, raising the alarm less often. To demonstrate that overgeneralization is avoided, the anomaly detection on a collision trajectory at a trained speed is displayed in Fig. 4.6.

Finally, Tab. 4.1 assesses the HDRADS and its precursors with regard to the six requirements presented in Ch. 2. The RBF-based approach identifies anomalies well, but is too computationally intensive in the test phase. NS detects too many anomalies, but is faster at evaluating test points. Yet, only using the SVM sufficiently fast evaluation is possible. The drawback of depending on two distinct classes is overcome by training the RADS with the positive and the negative model. The RADS robustly identifies anomalies and recognizes *positive data*.

The major problem—little generalization between different data parameters—can be overcome by including a projection in the method. PCA and reprojection alone have a limited anomaly detection capability. Moreover, improving the projection algorithm at a later point in time is impossible. However, the generalization capability of the projection will make most of retraining superfluous. Besides, additional but rare information most likely will not influence the setup of the projections strongly, especially since all valid data should behave in a similar way.

Combining the entire setup the most accurate anomaly detection even in high-dimensional data is achieved, and the generalization compares to that of projection. Besides, at least the RADS models can be improved with subsequent data. The dimensionality reduction incorporated in the HDRADS also ensures fast training of the single models. A minor flaw of the HDRADS is related to the least important requirement. Parameters for projection, the positive and negative model, and the SVM have to be identified. Nevertheless, this thesis presents methods the reduce the search space (see App. C–App. F).

Concluding, if the data provided is low-dimensional and depends on few parameters, the basic RADS performs sufficiently well. It ensures fast detection of unknown errors. The higher the initial dimensionality, the more important gets the application of HDRADS to ensure satisfactory generalization in the increased parameter space. None of the basic algorithms can outperform the RADS or its enhanced version.

Table 4.1.: Fulfillment of the requirements (r) introduced in Ch. 2 by the basic approach. 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	++	+	−	+	−	+
4	−−	−	++	++	++	++
5	+	+	−	+	++	++
6	+	+	0	0	+	0

5. Evaluation of the Machine Learning-Based Robot Anomaly Detection System

By means of its redundant setup the 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, 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 [35]. Yet, most testing has been performed on steady trajectories, ensuring the applicability of the RADS especially for industrial robots. Tracing a predefined trajectory containing most critical setups before putting the robot in service can ensure that the system is operable, even if training all possible configurations is infeasible.

The projection mechanisms introduced in Ch. 4 show sufficient performance for the test cases. However, only a small set of all possible techniques has been considered (see App. F.2). Further comparing other projection methods could reveal more suitable techniques[1].

[1] Future areas of research are the subject of App. I.

Appendix

A. Robot Data

Figure A.1.: Picture of the LBR 4+ used for ES I [30]

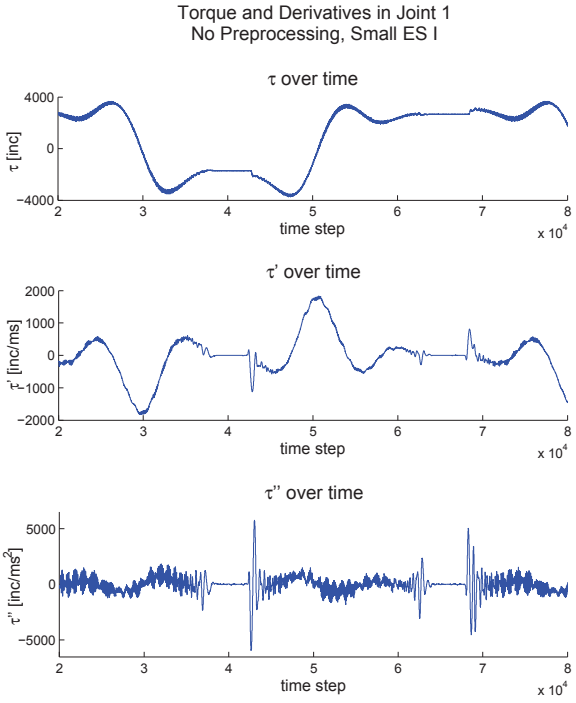


Figure A.2.: Plots of torque in joint 1 and its derivatives

A.1. Setup of Evaluation Data

In order to compare the performance of the different algorithms, data sets have been recorded and artificial errors induced.

A.1.1. Evaluation Set I

The main test data Evaluation Set I (ES I) are recorded using the LWR depicted in Fig. A.1 [3]. All seven joints are moved simultaneously from upper to lower joint limit and vice versa. After reaching a stop the robot remains in that position for five seconds, before back-tracing the trajectory. The desired maximum velocity

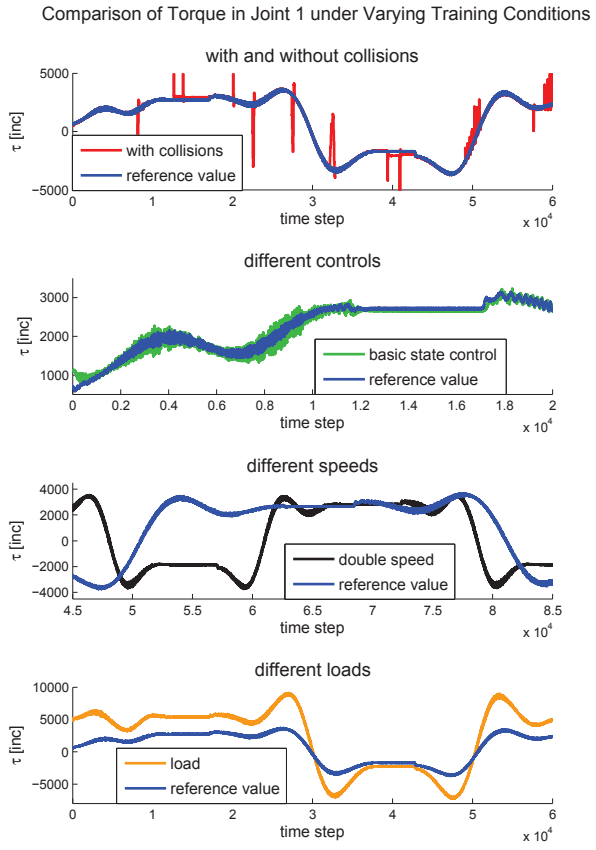


Figure A.3.: Comparison of torque with different parameter settings. For the reference torque the LWR uses adaptive gain control, does not carry any load, moves at a medium speed and operates free of collisions.

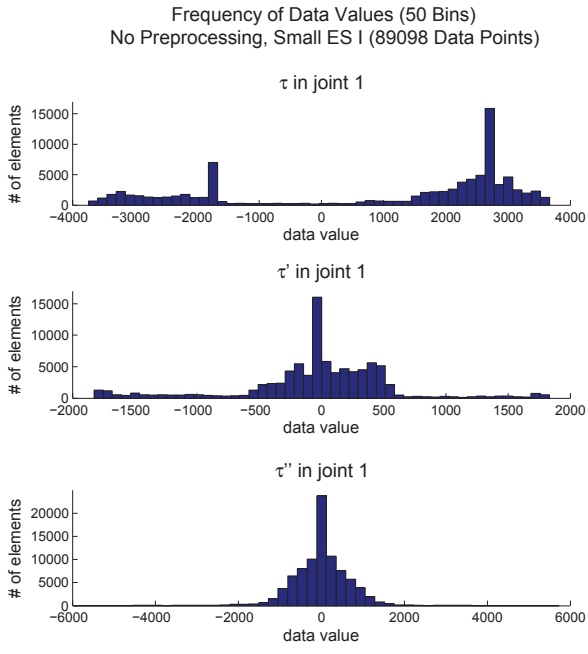


Figure A.4.: Histograms of small ES I without preprocessing

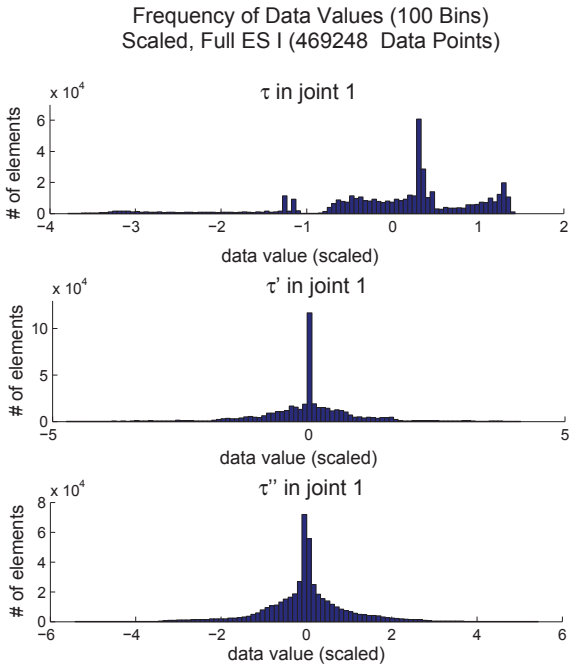


Figure A.5.: Histograms of full ES I after scaling

Figure A.6.: Picture of the DLR MIRO used for ES II [13]

is reached in the middle of two joint limits.

The recorded data includes a time stamp, the diagonal elements of the mass matrix with respect to the joint position, temperatures for each joint, link sided joint positions sensed by potentiometers, torques in the joints measured in potential increments, actual motor torques, desired motor torques, drive sided joint angles, and desired joint angles. Further, the attached load, the center of gravity of the load, the center of gravity of the robot base and the controller used[1] are included in the data. Throughout each trial these values remain constant.

Derived metrics are the first and second derivatives (see Fig. A.2). To ensure a smooth structure, a Savitzky-Golay filter with window length=1001 and a polynomial of order 4 is applied (see App. B.2) [43].

The *static* set—without derivatives—contains a total of 63 dimensions. A total of 176 dimensions are evaluated for the *dynamic* set. Instead of the time stamp, that will always be new, only the differences between the current and the preceeding time stamp is maintained, even if only *static* data are observed. Due to problems with the software interface for reading out temperatures the measurement values are unreliable. Whenever the robot is interrupted the measurements jump. Thus, temperatures are not considered for the (HD)RADS.

The small training set only contains training data of one maximum speed, one controller type and one load. The full training set contains training data of different speeds, different loads and different controller types. Fig. A.4 and Fig. A.5 display the distribution of the data within one generic dimension in the small, resp. full ES I. The main tests are conducted with data recorded under the same preconditions. However, occasional collisions—hitting the robot at different positions—are induced. Further, the recordings of different speeds and loads, etc. can be used as test cases (see Fig. A.3).

A.1.2. Evaluation Set II

Additionally, the algorithm is tested using data from the DLR MIRO displayed in Fig. A.6 [20] (ES II). The same method is used to generate trajectories.

The MIRO data includes desired current, actual current, desired torque, desired joint angles, actual joint angles, velocity, actual torque, a time stamp, and temperatures from the electronic units in the joints. Besides, the constant values of the applied load, the center of gravity of the load, as well as the center of gravity of the robot base are added. In contrast to the LWR generating data at a 1 kHz rate, the MIRO provides data at a 3 kHz rate, thus the amount of data available is greater and if plotted against each other, the two robots will display different time lines.

The *static* set initially included 65 dimensions. However, as with ES I, the temperature values are problematic and are removed from the data leading to a total of 61 dimensions. The *dynamic* set includes 196 dimensions. Again, different speeds and different applied loads have been tested against artificial collision data.

References to *validation* data imply that the training data is split into two parts, one—containing 70% of the total number of training points—is used for training, and the other—containing the remaining 30% of the training points—is used to test the approach. Correspondingly, the parameters for data generation are the same, but the algorithm is tested against previously unseen data. Validating the models with data generated under the same circumstances ensures a suitable representation of the data. If the model finds anomalies in the validation data, further evaluations can be omitted as the model is insufficient.

[1] adaptive gain reducing over-shooting or basic control

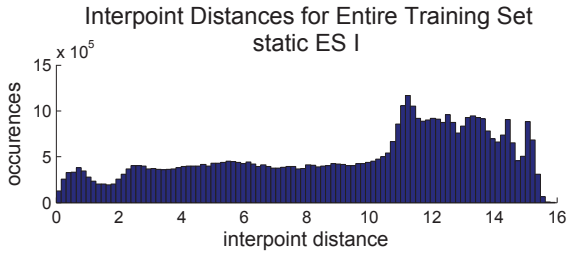


Figure A.7.: Histogram of interpoint distances on small, scaled, *static* ES I

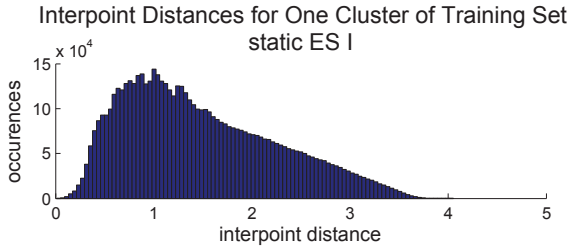


Figure A.8.: Histogram of interpoint distances in one cluster of small, scaled, *static* ES I

A.2. Initial Data Analysis

Additional analyses using simple statistical or Machine Learning (ML)-based techniques provide further insight into the structure of the data and can validate assumptions for the algorithms.

A.2.1. Identifying the Number of Clusters

Non-parametric k-means tries to identify the optimal number of data clusters k just by defining the maximum acceptable distance to a cluster center λ [31]. Usually, the lower λ, the higher the number of required clusters. Testing against different values of λ, the interval that spans the values between the largest distance of data to the *real* cluster center and the smallest distance between two clusters typically is the greatest interval in which the number of clusters does not change—except when the number of clusters is one or the number of clusters matches the number of training points.

The application of non-parametric k-means does not provide meaningful information on the number of clusters for either of the test sets. Besides the computational complexity limiting the number of λ values which can be evaluated, the number of clusters continuously increases while reducing λ without any plateaus. There are two explanations for this effect. If some of the clusters have a broader distribution than the distance between other clusters, the algorithm is doomed to fail because λ is assumed to be equal for all clusters. Further, if the clusters are not strictly separated but have fluent passages, non-parametric k-means cannot identify the corresponding λ value.

Accordingly, it is important to use approaches that can handle concave data, that are independent on the number of clusters, and that do not require an equivalent distribution of data within different clusters.

A.2.2. Lower-Dimensional Manifold Assumption

According to [32] if the interpoint distance distribution does not follow a Gaussian distribution, not all dimensions of the data are independent. Exploiting this assumption to validate that the data is located on a lower-dimensional manifold is difficult because no cluster affiliations are known. Taking the interpoint distances across the entire data set the Gaussian assumption clearly does not hold as Fig. A.7 depicts. However, this could also result from the fact that the data points do not belong to one single distribution.

Instead, using the basic RBF approach kernels, c.q. clusters, are generated. After identifying which cluster corresponds to which data points, i.e. which cluster is closest to each point, interpoint distances within the clusters are evaluated. As Fig. A.8 illustrates, even evaluating according to clusters does not result in a Gaussian distribution of interpoint distances.

This is owed to the fact, that many of the dimensions are physically correlated, e.g. current and torque, the influence of joint 7 on the values of joint 1, and a different center of gravity changes the torques in all joints. Besides, robots only have a limited number of Degrees of Freedom (DoF), seven for LWR and MIRO. Taking into account that further the load and the centers of gravity, as well as the desired trajectory have an influence on the values, and maybe a few influence factors that have not been identified, expecting a latent dimensionality for the *static* setup greater than 20 is already set high. Assuming the data to lie on a sub-dimensional manifold is reasonable.

B. Data Preparation

B.1. Methods for Normalization

Due to different measurement units, the spread of the data dimensions differs strongly. While joint angles only vary between $\pm\pi$, the potential increments measured to calculate the applied torque have their mean in the three-digit area and have a standard deviation in the same order.

Comparing such divergent data dimensions is infeasible. On the one side, many algorithms using matrix calculus—especially matrix inversion—suffer from the bad scaling, rendering reliable calculations technically impossible. On the other side, comparing the variance is intricate.

Several methods for normalization have been evaluated. For example, each element of the training data $d_{[i,j]}$ can be scaled to a hypercube with edges from [0..1]:

$$d'_{[i,j]} = \frac{d_{[i,j]} - \min\limits_i d_{[i,j]}}{\max\limits_i d_{[i,j]} - \min\limits_i d_{[i,j]}} \tag{B.1}$$

with j being the data dimension and i the respective observation. This creates a bounding box for the valid data, which is especially helpful for NS (see Sec. D.3). However, the variance stays inconsistent.

Another option is scaling the data with the standard deviation per observation often used in image processing:

$$d'_{[i,j]} = \frac{d_{[i,j]} - \overline{d_j}}{\sigma_j} \tag{B.2}$$

with $\overline{d_j}$ being the mean value over all dimensions in one observation and σ_j being the respective standard deviation. However, in contrast to pixels, the different sensors of a robot do not behave in a similar manner, thus variance inconsistency remains.

Alternatively, one can scale each dimension with the standard deviation over time σ_i instead of scaling over observations:

$$d_{[i,j]} = \frac{d_{[i,j]} - \overline{d_i}}{\sigma_i}. \tag{B.3}$$

This form of scaling assumes normal distribution of the data within each dimension and returns a standard normal distribution $\mathcal{N}(0,1)$. In case $\sigma_i \ll 1$, slight differences in the actual value compared to the mean can cause large differences—comparable to outliers—after scaling. To avoid these, it is recommended to set all values in that dimension to zero indicating that deviations from the mean are highly unlikely. Alternatively, only subtracting the mean in that dimension is conceivable. However, this amplifies the acceptance of deviations rather than restricting it.

Values normalized by the latter are much more alike and can be compared to each other. However, independent and identically distributed (iid) sampling of the data is impossible. Thus, the central limit theorem—the mathematical justification for an $\mathcal{N}(0,1)$ normalization—does not hold for the data sets. Nevertheless, the outcome of the RADS is most accurate and most reliable using the $\mathcal{N}(0,1)$ normalization per dimension removing the dependency on units.

No matter which scaling method is used, values used for scaling during the training phase, e.g. σ_i^2 and $\overline{d_i}$, have to be stored to ensure that validation and test data are scaled in the same manner. Using the same standard deviation on training and test data stresses outliers clearly.

If sliding window normalization was applied, i.e. the normalization was calculated over the most recent samples instead of all data available, the normalization would adapt to slow increases or decreases thereby covering slow appearance of errors.

B.2. Including Temporal Information

Often, the absolute sensor values may be in an acceptable range, while the velocity or acceleration get out of hand causing the robot to move too fast and abrupt. In contrast to [15] directly observing the behavior over time the described RADS generates models in the state space. Thus, only if additional information is available unusual effects over time can cause an alarm.

Calculating further metrics embodying temporal information and providing these to the algorithm as supplementary dimensions mitigates the restriction to static observations. Although evaluation is performed on vectors in the state space each vector contains information on the changes over time.

Encoding velocity and acceleration the first and second derivative with respect to time are important metrics. Calculating the difference between two following data points and dividing by the time step (cf. Eq. (B.4)) would be a sufficient approximation of the derivative if the sensor measurements were perfect.

$$\Delta t = 1, \tag{B.4a}$$

$$\Delta y(t) = y(t) - y(t + \Delta t), \tag{B.4b}$$

$$y'(t) = \frac{\Delta y(t)}{\Delta t}, \tag{B.4c}$$

with t being a discrete point in time and $y(t)$ the sensor value at that time.

Unfortunately, the measurements are noisy. As derivation increases the noise ratio, the second derivative could be replaced by white noise as it does not

encode any reliable information anymore. Accordingly, more sophisticated techniques have to be applied.

The Savitzky-Golay filter [43] is a reasonable method to calculate the derivatives. It tends to preserve features such as maxima and minima—important for the analysis—better than, for example, the moving average filter [47]. However, not choosing parameters carefully causes the filter to be less efficient at removing noise. Moreover, since the Savitzky-Golay filter fits a polynomial to approximate the real data values, it can be set up without a time delay. Although filter delays are not critical to this application, using an algorithm without delay is desirable to improve the performance.

Kalman filters designed to estimate the real signal from noisy measurements could be applied as well. However, they return most accurate results if the matrices are modeled according to the system [36]. Even though pseudo models can be used to filter the data, it requires prior knowledge of the setup objecting the purpose of model independence.

Other metrics, e.g. the oscillatory behavior of the system, might be interesting. Maybe a low frequency oscillation of the current is typical since the robot switches from acceleration to deceleration frequently. High-frequency oscillations, on the other side, could hint at an over-shooting control. One option to incorporate oscillations into the data is to evaluate simulated damped oscillators. For several frequency bands different oscillators are observed. The higher the energy in the damped oscillator, the stronger the oscillation at the respective frequency in the system [24].

B.3. Randomization

As stated in App. C.3, training with ordered data can be hazardous. Either the generated models overgeneralize to the time effects, or overfit, or, especially in neural networks, forget about early training stages and focus on the latest data.

Except for algorithms that use the entire data stack at once (e.g. NS) or that are based on sequence effects (e.g. PGA), randomization of the order can improve the performance. Since sequence effects are not considered by the algorithms applied and since the order does not matter for NS, the training data is shuffled after calculating the desired additional metrics.

C. Supplementary Information on the Positive Model

C.1. Implementation Details

The RBF network evaluation is time consuming. Even parallelization on a GPU does only reduce the problem to a certain degree. Further, some of the parameters are difficult to estimate. The following section introduces some enhancements to speed up the algorithm and to improve parameter search.

C.1.1. Evaluating Closeness of Data Points to Kernels

The calculation of the network response is computational intensive because there is no closed form solution for the probability according to the χ^2 goodness-of-fit test. To reduce calculations, the maximum distance according to the χ^2 distribution $\chi^2(\alpha, n)$—depending on the dimension n and the level of significance α—a data point can have from a single kernel without being significantly different can be computed in advance. If

$$m_k \leq \chi^2(\alpha, n) \tag{C.1}$$

with m_k being the distance between a point $\boldsymbol{x}$ and a kernel center $\boldsymbol{\mu}_k$, the probability threshold α is surpassed by that kernel. $\boldsymbol{x}$ is considered an inlier without having to compute the full network response $\mathrm{rbf}(\boldsymbol{x})$.

Moreover, the maximum distance that $\boldsymbol{x}$ can have to all K kernels can be calculated in advance. If $\boldsymbol{x}$ has a probability

$$Q_{\chi^2}(m_k, n) < \frac{\alpha}{K}, \quad \forall k \in [1 \dots K], \tag{C.2}$$

it is impossible that the network surpass the threshold in sum. I.e., if the minimum distance $\boldsymbol{x}$ has to any kernel is greater than $\chi^2\left(\frac{\alpha}{K}, n\right)$, $\boldsymbol{x}$ definitely is not represented by the current model.

If neither of the two criteria apply, it is least computationally expensive to compute $\mathrm{rbf}(\boldsymbol{x})$ iteratively. The probability of belonging to each kernel k, $Q_{\chi^2}(m_k, n)$, is added in order of ascending distance:

$$\phi(\boldsymbol{x}, \kappa_0 \dots \kappa_{k+1}) = \phi(\boldsymbol{x}, \kappa_0 \dots \kappa_k) + \phi(\boldsymbol{x}, \kappa_{k+1}). \tag{C.3}$$

Once α has been surpassed, the iterations can be stopped, as the likelihood of belonging to the model is high enough. Only if $\boldsymbol{x}$ is not represented by the model, all kernel probabilities have to be calculated.

Typically, the data points will be inliers. Thus, the computation time will be reduced significantly. Further, experiments showed that usually only few kernels have an impact on $\mathrm{rbf}(\boldsymbol{x})$. Therefore, every i steps, it is evaluated whether the spare $K - i\nu$ kernels with $\nu \in \left[1 \dots \left\lfloor \frac{K}{i} \right\rfloor\right]$ can surpass the remaining probability threshold $\alpha' = \alpha - \phi(\boldsymbol{x}, \kappa_0 \dots \kappa_{i\nu})$. Eq. (C.2) transforms to

$$Q_{\chi^2}(m_k, n) < \frac{\alpha'}{K} \quad \forall k \in [i\nu \dots K]. \tag{C.4}$$

C.1.2. Initializing the Covariance Matrix

For initializing the covariance Σ_{new}, all dimensions are assumed to be independent. Accordingly, the covariance is a diagonal matrix and its entries have to ensure, that points not deviating further from the new kernel center $\boldsymbol{\mu}_{\mathrm{new}}$ than the usual noise amplitude a are recognized as inliers (see Fig. C.1).

The noise amplitude a can be approximated from the measurements. Especially well suited are constant states as the only variance in the data should be caused by measurement noise. It may be different for each dimension. However, the differences are negligible for the test data.

The probability according to the χ^2 goodness-of-fit test takes into account that neighboring points have a larger distance value because deviations in more dimensions are possible. However, it also accepts larger deviations for each dimension (see App. C.6). Therefore, the scaling factor is adjusted to at most accept a deviation of the typical noise amplitude in all dimensions.

Assuming that all dimensions have the same noise ratio[1], the maximum justifiable distance $\delta_{j_{\max}}$ to the center is

$$\delta_{j_{\max}} = \sqrt{a^2 s n} \tag{C.5}$$

with n dimensions and a scaling factor s, also called precision.

The maximum distance that will be recognized $\delta_{r_{\max}} = \chi^2(\alpha, n)$. Equating $\delta_{r_{\max}}$ with $\delta_{j_{\max}}$ resolves to

$$s = \frac{\delta^2_{r_{\max}}}{a^2 n}. \tag{C.6}$$

Therefore, the initial Σ_{new} for κ_{new} is an $n \times n$, diagonal matrix with all entries on the main diagonal equal to $1/s$.

C.1.3. Covariance Adjustment

Fig. C.2 shows that the new covariance Σ_{new} is supposed to cover the initially data space, plus an increase of $\pm\boldsymbol{\iota}_\mu$, the translation from the old to the new center. l_μ is the length of the translation, $\boldsymbol{\iota}$ is the direction of $\boldsymbol{\iota}_\mu$ with Euclidean length 1.

[1] If this assumption is too restrictive for the given data the sum of the scaled squares of each noise amplitude should be used instead.

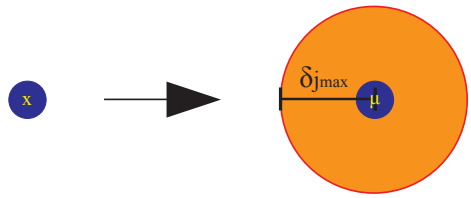


Figure C.1.: Starting from a single data point (blue) an area of recognition (orange) is established.

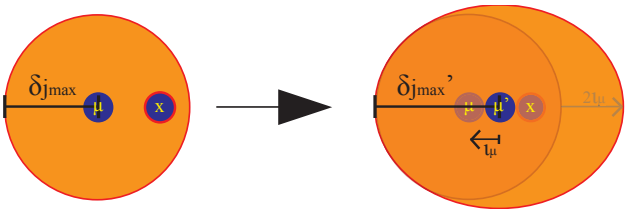


Figure C.2.: A kernel recognizes a new training point (red border). The center and the covariance have to be adjusted.

$$\boldsymbol{\iota}_\mu = \boldsymbol{\mu}_{\text{new}} - \boldsymbol{\mu}_{\text{old}}, \tag{C.7a}$$
$$l_\mu = \|\boldsymbol{\iota}_\mu\|, \tag{C.7b}$$
$$\boldsymbol{\iota} = \frac{\boldsymbol{\iota}_\mu}{l_\mu}. \tag{C.7c}$$

A computationally simple, but approximate method is chosen to scale the covariance which results in very good recognition properties during tests. The new covariance Σ_{new} has to be stretched to continuously cover the data space that has been covered by the initial covariance Σ_{old}, although being placed at a different center. Thus, the covariance has to be increased relative to the center translation $\boldsymbol{\iota}_\mu$.

Scaling the outer product of $\boldsymbol{\iota}_\mu$ with itself by $\delta^2_{r_{\max}}$ and adding it to Σ_{old} returns the new covariance:

$$\Sigma_\iota = \frac{\boldsymbol{\iota}_\mu \boldsymbol{\iota}_\mu^T}{\delta^2_{r_{\max}}}, \tag{C.8a}$$
$$\Sigma_{\text{new}} = \Sigma_{\text{old}} + \Sigma_\iota. \tag{C.8b}$$

C.2. Alternative Implementations

In addition to the presented implementation of the RBF network, alternatives have been considered. Two of them were intended to replace current code, but did not provide as good results; two of them are left out due to insufficient performance.

C.2.1. Different Scaling of the Covariance During Training

Instead of the scaling routine introduced in App. C.1.3 two further methods have been evaluated.

A comprehensible approach is to generate the new covariance matrix from the bounding hull of the recognized area. First, the length l_{old} that makes $\boldsymbol{\iota}$ point to the farthest, recognized data point $\boldsymbol{q}$ of the old kernel in the direction of $\boldsymbol{\iota}$ is identified:

$$\boldsymbol{q} = \boldsymbol{\mu}_{\text{old}} + l_{\text{old}}\boldsymbol{\iota}. \tag{C.9}$$

Scaling $\boldsymbol{\iota}$ by $l_\mu + l_{\text{old}}$ (l_μ is the distance between the old and new center) and adding the result to, resp. subtracting it from $\boldsymbol{\mu}_{\text{new}}$, offers two data points at the edge of recognition. Subsequently, the maximum recognized length of any orthogonal vector in the old kernel has to be identified. Adding the scaled, orthogonal vectors and their negatives to the new center $\boldsymbol{\mu}_{\text{new}}$ results in another $2(n-1)$ points, with n being the number of dimensions.

In total $2n$ points unambiguously determining the shape of the new bounding hull are available. Using [28] the smallest, enclosing, multi-dimensional ellipsoid can be constructed. The result is a precision matrix—the inverse of the new covariance matrix. To make sure that multiplying distance vectors with that matrix returns the correct values for the χ^2 goodness-of-fit test, it has to be scaled by $\delta^2_{r_{\max}}$.

However, this approach has two disadvantages. First and foremost, the algorithm to determine the enclosing ellipse is iterative, thereby further expanding the computational complexity of training. Second, since the border points have the same distances from $\boldsymbol{\mu}_{\text{new}}$ as their counterparts have from $\boldsymbol{\mu}_{\text{old}}$, the resulting covariance cuts off previously recognized space in the border region. To ensure that the points will be placed on the border of the ellipse, they have to be constructed from orthogonal vectors. Therefore, it is not possible to instead use former border points.

A further approach directly rescales the covariance matrix. $\boldsymbol{q}$ is the farthest, recognized point from κ_{old} as well as κ_{new}. Taking into account that $\boldsymbol{\mu}_{\text{new}}$ and $\boldsymbol{\mu}_{\text{old}}$ are translated along $\boldsymbol{\iota}$

$$\boldsymbol{q} = \boldsymbol{\mu}_{\text{new}} + l_{\text{new}}\boldsymbol{\iota} \tag{C.10}$$

with $l_{\text{new}} = l_{\text{old}} + l_\mu$. Since $\delta_{r_{\max}}$, Σ_{new}, and $\boldsymbol{\iota}$ are known, Eq. (C.11) can be solved for l_{old}.

$$\boldsymbol{\iota}_{\text{old}} = \boldsymbol{\mu}_{\text{old}} + l_{\text{old}}\boldsymbol{\iota} - \boldsymbol{\mu}_{\text{old}} \tag{C.11a}$$
$$\delta_{r_{\max}} = \sqrt{\boldsymbol{\iota}_{\text{old}}^T \Sigma_{\text{old}}^{-1} \boldsymbol{\iota}_{\text{old}}}, \tag{C.11b}$$
$$l_{\text{old}} = \frac{\delta_{r_{\max}}}{\sqrt{\boldsymbol{\iota}^T \Sigma_{\text{old}}^{-1} \boldsymbol{\iota}}}. \tag{C.11c}$$

The same applies for l_{new}.

Rearranging the above equations (see Eq. (C.12)) to display what happens to each element of the covariance matrix shows that each entry $[r, c]$ of Σ_{old}^{-1} is multiplied with the entries $[r]$ and $[c]$ of the scaled translation vector, c.q. Σ_{new}^{-1} for κ_{new}.

$$\delta_{r_{\max}}{}^2 = \sum_{r=1}^{R}\sum_{c=1}^{C} \iota_{\text{old}[r]}\iota_{\text{old}[c]}\Sigma^{-1}_{\text{old}[r,c]}. \tag{C.12a}$$
$$\delta_{r_{\max}}{}^2 = \sum_{r=1}^{R}\sum_{c=1}^{C} \iota_{\text{new}[r]}\iota_{\text{new}[c]}{\Sigma_{\text{new}}}^{-1}{}_{[r,c]}. \tag{C.12b}$$

Equating Eq. (C.12a) and Eq. (C.12b) only Σ_{new} is variable. Solving for each element of Σ_{new} separately

leads to

$$\Sigma_{\text{new}}{}^{-1}{}_{[r,c]} = \Sigma^{-1}_{\text{old}\,[r,c]} \frac{\iota_{\text{old}\,[r]}\iota_{\text{old}\,[c]}}{\iota_{\text{new}\,[r]}\iota_{\text{new}\,[c]}}. \quad \forall r, c \in [1..n] \tag{C.13}$$

In case any element of $\boldsymbol{\iota}$ equals zero, the corresponding elements of Σ_{new} have to be set to the respective value of Σ_{old}, since division by zero is prohibited but no change in that direction has taken place.

This form of scaling increases the covariance in the direction of $\pm\boldsymbol{\iota}$, but does not affect orthogonal directions. But, this scaling is only applicable if the underlying covariance matrix does not contain zero values in any of the directions that have to be scaled. Since no assumptions about correlations between the different dimensions are available in the first step, a more suitable approach has to be chosen.

C.2.2. Using the Normal Distribution

Instead of using the χ^2 goodness-of-fit test, the function value of the normal distribution $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \Sigma_k)$ according to Eq. (C.14) was supposed to surpass a threshold value.

$$\text{rbf}(\boldsymbol{x}) = \sum_{k=1}^{K} w_k \mathcal{N}(\boldsymbol{x}, \boldsymbol{\mu}_k, \Sigma_k), \tag{C.14a}$$

$$\mathcal{N}(\boldsymbol{x}, \boldsymbol{\mu}_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_k|}} e^{\frac{(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\boldsymbol{x}-\boldsymbol{\mu}_k)}{-2}}. \tag{C.14b}$$

Although requiring fewer computations than the worst case χ^2 calculations, this method has drawbacks. It is difficult to define a sensible recognition threshold, i.e. which function value indicates a significant difference? Further, changing the spread of the covariance matrix makes it very hard to maintain a similar recognition area, respectively recognition distance. The larger the covariance, the flatter the function. Therefore, setting a function value that has to be surpassed is delicate.

In low dimensional spaces, using the local integral of the normal curve would be reasonable. Given a specific normal distribution it describes the probability of a sample appearing in that area. However, the space is high-dimensional, and for each dimension of the domain D a further integral

$$\int \dots \int_D f(x_1, \dots, x_n) dx_1 \dots dx_n \tag{C.15}$$

has to be evaluated.

Since there is no analytic expression for the cumulative distribution function of a multivariate distribution, it is computationally impossible to evaluate the probability in a high-dimensional space.

C.2.3. Expanding and Shrinking Covariances

Initially, it was intended to shrink the covariance in the directions with little difference between the observation and the center, i.e. the dimensions where the difference vector $\boldsymbol{\iota}_\mu$ is close to zero. This would have facilitated having a random covariance in the beginning, that automatically adjusts to the data.

However, this results in allegedly large known areas in sparsely populated space and border regions as too little information is available for sufficient shrinking. Besides, reducing the covariance, i.e. shrinking the area of coverage, is at risk of uncovering areas populated by training data.

C.2.4. Merging of Similar Kernels

If two kernels having compatible covariances are close enough, they can be represented by one kernel, as depicted in Fig. C.3. Merging these kernels can reduce computational costs.

In order to maintain the strongly structured border area, a number of limiting factors have to be considered. Most important, the two kernels that will be united have to be close. Closeness can be defined by distance. Having used Mahalanobis distance and the χ^2 goodness-of-fit test throughout the approach to take different local variances into account, it is reasonable to proceed using this metric. If the χ^2 goodness-of-fit test probability that one kernel center belongs to another kernel is greater than $\alpha = 5\%$ they are not significantly different, i.e. they may be similar[2].

In order to ensure most accurate coverage of the initially recognized space, the kernels additionally have to have a similar shape. Fig. C.3a shows pairs of two-dimensional kernels that have shapes which can be represented in one kernel, Fig. C.3b depicts two-dimensional kernels, that are close, but too different in shape for being merged.

To reduce the amount of kernels during training merging kernels every k training points has been introduced. However, the calculations evaluating whether two kernels are compatible are costly and only few kernels are merged during the course of calculation. This results in an increase of computation time compared to the time required from full network construction. Besides time, merging increased inaccuracy. Therefore, it is refrained form using the add-on in the final implementation.

C.3. Effects of Sequential Training

Passing new training points in the order of generation bears the hazard of the kernel following the path of generation. Most valid data points will be neighbors to their predecessors. As Fig. C.5 depicts, the kernels are likely to increase their size in the direction of the next neighbor without taking into account that the points might not be representable by a single kernel.

To avoid this sequence effect the order of the training data is randomized. If, by chance, two neighbors appear sequentially no harm is done. Only consequent sequential passing has to be evaded.

[2] The choice of $\alpha = 5\%$ is arbitrary, but conforms to many statistical evaluations [16, 48].

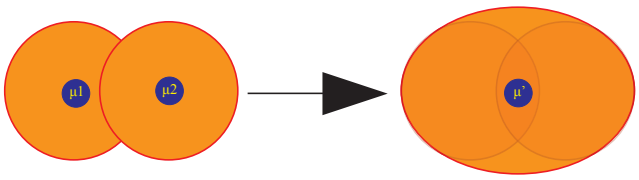
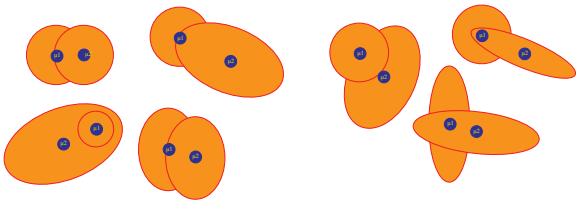


Figure C.3.: Two separate kernels can be represented by just one kernel if they are close enough and have compatible covariances.



a: close, compatible kernels

b: close kernels that should not be merged

Figure C.4.: Juxtaposition of kernels that can be merged and others that are not compatible

A further enhancement of the positive model could be to generate the model as described above. However, after a first round of training, each training data point is assigned to the closest kernel. Afterwards, for each kernel the most suitable center and covariance are calculated from the allocated training points, as in k-means. Using the initial model generation the appropriate number of kernels and their approximate positions are identified. Completely independent of sequence effects the readjustment optimizes the setup.

Already, the performance of the initial setup is highly satisfying. Therefore, the potential improvement has neither been implemented nor tested.

C.4. Handling Outliers

For improving the resistance towards outliers it is sensible to discard kernels that have never been activated during the entire training phase. Never being activated implies, that the only data point close to the particular kernel is the one that caused its creation. Having no neighbors despite of the high sampling rate is a likely indicator for being an outlier caused by transmission or sensing errors.

In case a discarded kernel did cover a valid data area it can be included during the next improvement phase. Even two or three activations could result from outliers. But, these outliers show some repeatability and might reoccur consistently due to specific constellations.

C.5. Number of Required Kernels

In order to achieve high accuracy, a small value for the initial covariances is required. Although being able to grow, a single kernel is unlikely to cover the whole data space. Only data points already activating the kernel will influence the spread of the covariance. The more points have already influenced the data point, the smaller is the change of center and spread.

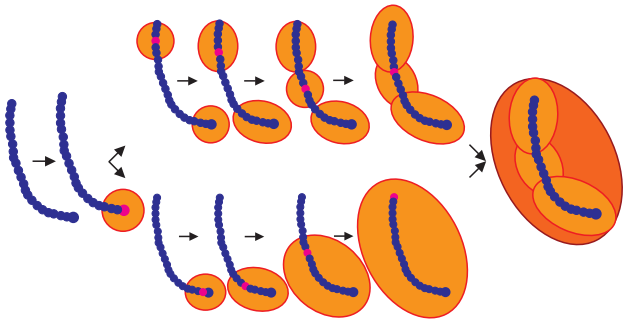


Figure C.5.: Since most new training points will be neighbors to their predecessor, passing them in the order of generation can lead to misshaped kernels. In the randomized data set (top row) several kernels are introduced. The bottom line depicts what would happen if all training points were passed in the order of generation. The last step compares the area of coverage from both sequences, clearly displaying that random training provides a more accurate model, yet being more complex.

From any position of the kernel center $\boldsymbol{\mu}_i$ and the corresponding covariance Σ_i, follows the maximum recognition distance to the center l_i. Accordingly, the point at the border of recognition in direction $\boldsymbol{\iota}$ is situated at $\boldsymbol{\mu}_i + l_i\boldsymbol{\iota}$. Assuming a new training point is located there, the resulting center $\boldsymbol{\mu}_{i+1}$ and the new maximum distance of recognition l_{i+1} are computed by

$$\boldsymbol{\mu}_{i+1} = \frac{i}{1+i}\boldsymbol{\mu}_i + \frac{1}{i+1}(\boldsymbol{\mu}_i + l_i\boldsymbol{\iota}) = \boldsymbol{\mu}_i + \frac{l_i\boldsymbol{\iota}}{i+1}, \tag{C.16a}$$

$$l_{i+1} = l_i + \|\boldsymbol{\mu}_{i+1} - \boldsymbol{\mu}_i\| = l_i + \frac{l_i}{i+1}. \tag{C.16b}$$

Further requiring that $\boldsymbol{\iota}$ remains the same for each iteration, the recursive formulation reveals that the increases are arithmetic sequences, with an initial center $\boldsymbol{\mu}_0$ and an initial recognition distance d_0:

$$d_i = \frac{i+2}{2}d_0, \tag{C.17a}$$

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_0 + \frac{1}{4}\frac{i(i+3)}{2}d_0\boldsymbol{\iota}. \tag{C.17b}$$

Both sequences diverge. Stating any limits with regard to the minimum number of required kernels is impossible, except if all points were neighbors, and the training points were received in a sequential order, one single kernel would suffice. However, as the data points are provided in random order, in the worst case, i.e. all space within the limits is known,

$$\prod_{j=1}^{n} \frac{\max_i \boldsymbol{x}_{[i,j]} - \min_i \boldsymbol{x}_{[i,j]}}{d_0} \tag{C.18}$$

kernels have to be evaluated, with j indicating the dimension, and i the training point.

C.6. χ^2-Probability Assumes Different Variations per Dimension in Higher-Dimensional Space

The χ^2 goodness-of-fit test probability is aware that the higher-dimensional the space the greater, the distance two points can have from each other. However, assuming a fixed level of significance α different deviations per dimension are accepted. Assuming that all dimensions have the same deviation, $\chi^2(\alpha, n)$ for n dimensions can be transformed to a maximum distance of acceptance

$$\delta_{\max} = \sqrt{\frac{\chi^2(\alpha, n)^2}{n}}. \tag{C.19}$$

Table C.1.: Recognition distances in a single dimension based on the χ^2 goodness-of-fit test probability

α	n	$\chi^2(\alpha, n)$	$d_{\max}$
0.05	1	3.8414	3.8414
	2	5.9914	4.2365
	3	7.8147	4.5118
	100	124.3421	12.4342
0.1	1	2.7055	2.7055
	2	4.6051	3.2561
	3	6.2517	3.6094
	100	118.4980	11.8498

Tab. C.1 shows the effect for two levels of significance in five different dimensions. Tab. C.2 on the contrary displays, that for real data the number of clusters remains much the same if the covariance is scaled as described in App. C.1.3. In order to prevent the data itself from occluding the results, instead of using data sets containing different information—e.g. with and without derivatives—the data set is extended with itself (ES I' = [ES I, ES I]).

Table C.2.: Clustering behavior in different dimensions after scaling the covariance

n	a	α	K	# of alarms
63	0.4	0.05	18	1713
126			18	1717
63	0.2	0.05	62	2940
126			62	2902
189			62	2902
252			62	2905
63	0.15	0.05	116	3713
126			118	3730
63	0.2	0.01	62	3290
126			62	3190
63	0.2	0.1	62	2690
126			62	2774

D. Supplementary Information on the Negative Model

D.1. Reducing the Number of Detectors

Although testing against the already existing detectors limits the number of overlaps, it is likely that subsequently generated detectors cover the entire activation area of previously defined detectors. They can have a larger distance to the closest known point and thus a larger radius.

Identification of detectors completely covered by other detectors is achieved by calculating the distance $\Delta\tau_i$ of all detectors τ_i to the potentially covering detector τ_c. If any $\Delta\tau_i$ plus the respective radius r_i is smaller than the radius r_c of τ_c, all points covered by τ_i are also covered by τ_c, among others. Thus, removing τ_i can be performed without any loss of information.

$$\Delta\tau_i + r_i < r_c \tag{D.1}$$

is checked after detector generation has been finished. Successively, each detector becomes τ_c and is tested against all other remaining detectors to ensure all potential overlaps are discovered. Reducing the number of superfluous detectors increases the performance of the system during testing.

D.2. Handling Outliers

Since NS is sensitive to any training point resistance towards outliers comparable to that of the RBF approach can only be achieved by a workaround.

Once an outlier has been identified by the positive model (see App. C.4) it should be removed from the training set handed to the NS algorithm, thus restricting the order of execution. In case the removed data point was valid it can be recovered in the updating phase.

D.3. Dependency of NS on the Ratio between Training Data Space and Sampling Area

For negative selection the sampled data space has to closely match the actual training data distribution. If the sampling area is much larger than the area containing data, a few large detectors will cover the largest portion of the space. A high coverage of the sampling space is guaranteed, yet outliers are represented poorly.

Therefore, it has to be made sure that the training data does not contain obvious outliers, e.g. after $\mathcal{N}(0,1)$-normalization typically constant values with few points of small absolute deviation will cause the bounding area to be extremely large, due to the division by the extremely small standard deviation. Outliers prohibit reliable detector generation. Thus, analogous effects have to be eliminated.

If large bounding areas are desired in order to detect highly unlikely data points with this approach the detector generation should be split into two parts. The first execution generates detectors close to the actual data, e.g. at a maximum distance of twice the minimum radius $r_{\min}$ as proposed by [27]. The second execution creates detectors around the old detector set. If the training data is retained for the second part, the same algorithm can be applied. Only the minimum and maximum detector positions in each dimensions would have to be increased as much as desired.

In case the bounding area has to be extremely large, it might be advisable to repeat the space enlargement several times to avoid too strong differences between the two subspaces. Of course, as this problem highlights, it is impossible to detect infinite divergence. Fortunately, this is not the task of the RADS. It is rather desired to identify small, but unusual divergence.

D.4. Enhancements to Further Improve NS

The detectors are generated at random. Any potential detector that is lying too close to the training data and thus considered lying inside of the known area is discarded entirely.

The efficiency of detector generation could be improved by reducing randomness. In robot motion and obstacle avoidance it is common to first generate random path points. Each point that coincides with an obstacle is gradually moved in orthogonal directions until a point in free space is created. This way, not only fewer potential points have to be generated, but also the area around obstacles is represented in more detail [6,8]. Transferring this idea to NS, each potential detector located in a known area could be moved a fixed distance in orthogonal directions for at most k iterations or until an unknown area is discovered. Thereby providing a better representation of the data borders.

Besides, sampling detectors in less known areas is sensible, as well. One way to achieve this is weighted sampling. Whenever a point is generated within a specific region the weight of that region is decreased relative to the other regions. Thus, in the next iteration the probability of sampling in the same region is reduced [6].

E. Adapting the SVM Parameters to the RADS

In order to handle the non-linear distribution of inliers and outliers, an RBF kernel is applied. Identifying the most appropriate parameters C for misclassification punishment and σ^2 for kernel width is impossible beforehand. Each setup is different and varying circumstances require adapting parameters. Usually using cross-validation combined with exhaustive grid search, or more elaborate search measures, can return the most suitable parameter set by comparing the accuracy resulting from different parameter sets [49].

For the given problem, this approach is insufficient. The negative model diverges from the valid data in every direction. Real errors influence specific dimensions and thus vary mainly in these dimensions. Using only the NS model to generate test data for cross validation the difference between valid and invalid samples is strong enough to ensure 100% accuracy for a large set of parameters.

However, testing against data with artifically induced collisions shows that the range of appropriate parameters is much smaller. Therefore, instead of k-fold validation against the model, a faulty trajectory, e.g. with artificially induced anomalies as for testing (see App. A), should be recorded and labeled (valid and invalid observations) either by hand, by NS, or the RBF-based approach. The trajectory then can be used to evaluate the performance of the SVM. The set of parameters returning the most accurate label prediction is used to perform the final training of the SVM. Using labels from either the RBF-based approach or NS the procedure can be automated and relieves the user.

F. Supplementary Information on Projection

F.1. Learning Reprojection Distances

In order to identify the reprojection error threshold ε, that may not be surpassed without being considered an outlier, the differences during training have to be evaluated. The PCA reconstruction error of valid data is a one-dimensional normal distribution scattering around its mean ϵ retrieved by averaging all training errors. After subtracting ϵ from all error values, the standard deviation σ can be calculated. Any value that is greater than ϵ and that has a probability $\alpha < 0.05$ of being from the same distribution is significantly different. Significant difference implies, that the corresponding point probably is a member of a different distribution. If that is the case, the point is considered an outlier. Using the χ^2 goodness-of-fit test

$$\varepsilon = \epsilon + \frac{1}{\varsigma}\chi^2(\alpha, 1) \tag{F.1}$$

can be evaluated.

The Euclidean distance—often applied in ML—is a poor discriminating factor in higher dimensions. Although the Euclidean error has been sufficient to identify errors in the test data, two alternatives shall be presented.

The typical distances in each dimension can be evaluated separately. As soon as the distance in just one dimension surpasses the typical error a reaction is triggered. However, just observing the single dimensions neglects dependencies between the dimensions. While a difference of one unit might be ok for any unit, a difference of one unit in all dimensions may not be acceptable.

Further, the fractional distance metrics introduced in [1] could be used. Instead of emphasizing outlier values they stress the typical distances. Thus, they are less dependent on noise and supposedly suffer less from the curse of dimensionality. However, before applying these metrics, it has to be evaluated whether they maintain their superior performance on real data rather than the artificial data used in [1]. Besides, all algorithms would have to be adapted to apply the new distance metric prohibiting the black-box integration of existing code.

F.2. Additionally Evaluated Projection Mechanisms

Besides PCA applied in the HDRADS different projection mechanisms including autoencoders (AE) and Mixture of Factor Analyzers (MFA) have been examined.

F.2.1. Projection with Autoencoders

One method for non-linear dimensionality reduction are AEs. They are based on neural networks and unlike most other algorithms inherently handle reprojection. An AE generates a lower-dimensional code of the data and optimizes the coding and decoding strategy to produce the closest match between original and reconstructed data [23, 42].

In order to facilitate non-linear dimensionality reduction, some non-linear function $\boldsymbol{z} = \psi(\boldsymbol{x})$, e.g. $\psi(\boldsymbol{x}) = \tanh(\boldsymbol{x})$ with $\tanh(\boldsymbol{x})$ being the hyperbolic tangent, has to be applied to the reduced input. Otherwise, the mapping of the AE will be linear and comparable to that of PCA.

AEs are well suited to identify non-linear dependencies if a non-linear mapping is applied. However, the non-linear mapping also prevents optimal linear reduction. Besides, the time required for training is much longer for AEs than for e.g. PCA since no closed form solution is available and stepwise improvement of the algorithm is required. One more drawback is the danger of getting stuck in a local minimum.

In order to use an AE for the HDRADS, the entire training data—except for validation data—is used to train the AE. Once a sufficiently small reprojection error—also on the validation data—is reached the training can be stopped. Then the entire data—including the validation data—is reduced with the AE but reprojection is omitted. The reduced data is used to train the positive and negative models and subsequently the SVM. Once the testing phase starts, the original data is processed by the AE and, additionally, the reduced data is evaluated using the SVM.

Beyond the basic setup of AEs there are several modifications, e.g. the following three that have been considered for the HDRADS.

F.2.1.1. Denoising AE

As the name suggest the denoising AE is supposed to remove the disturbances in a noisy signal. The AE correlates several dimensions to reconstruct the input. To ensure the identification of correlations, a specified percentage of each input is corrupted. For each sample the dimensions to corrupt are chosen at random. Typical corruption values vary between 30 and 50 %. The mapping is trained on corrupted data. However, the reconstruction error is evaluated against the original input. Thus, the net has to identify which of the dimensions encode similar information. If one of them is corrupted, the others have to bypass the missing information [53].

Different corruption methods are available:

- zeromasking: all corrupted values are set to zero;
- salt and pepper noise: the corrupted values are either set to the minimum or maximum value;
- Gaussian noise: a Gaussian noise is added to the corrupted values [53].

In the robots all sensor readings are naturally blurred by noise. Thus, additional Gaussian noise is not likely to improve the performance, especially since there is no ground truth for the actual value. Disturbed information would have to be compared against blurred data points. Salt and pepper noise have little difference in the effect compared to zeromasking. However, zeromasking is easier to implement. Thus, only the latter is taken into account for the HDRADS.

Using a denoising AE the bottleneck layer does not necessarily have to have fewer neurons than the in- and output layer. Learning of the identity is impossible because the input data will never be exactly the same as the output data. Hence, the neurons will encode a different representation of the data [53]. Yet, the aim of applying an AE in the RADS is to reduce dimensions inherently requiring a bottleneck.

F.2.1.2. Relational AE

Another extension to the AE are relational AEs considering the pairwise products of two input sets as input to construct the hidden layer. Relational AEs especially represent co-occurrences between two inputs [39].

Memisevic further combines the basic relational AE with a denoising AE to model the covariances of input values. Instead of applying two different inputs, the same input is corrupted independently [39].

F.2.1.3. Stacked AE

In contrast to linear projections consecutively applying two non-linear projections provides a further Degree of Freedom DoF. Thus, more suitable, lower-dimensional representations of the data can be identified.

Training a stacked, resp. deep, AE is most efficient in several steps. Instead of training all levels at once independently trained levels—subsequent levels have fewer dimensions than preceeding ones—are stacked on top of each other. After training the first layer, the entire training data is reduced to the lower dimensional representation and used to train the next layer. Once the desired depth is reached an additional fine tuning by e.g. conjugate gradient methods [22, 46] can improve the overall performance as interdependencies between the different levels are considered when processing the entire stack [23].

Also, any of the modifications are applicable for stacking. The better the AE identifies interdependencies, the better will be the code representation and the lower will be the reconstruction error. Therefore, algorithms with more sophisticated coding strategies are most likely to exhibit a superior performance after stacking.

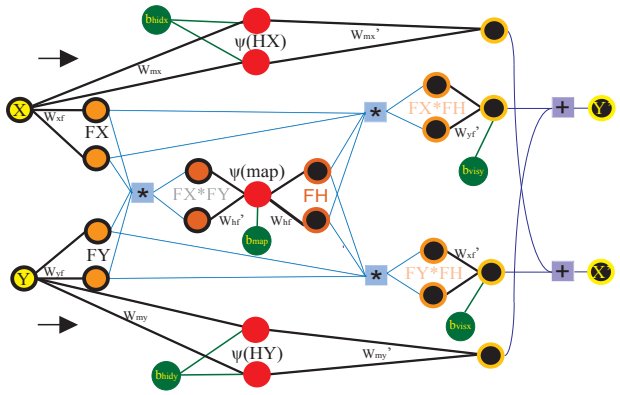


Figure F.1.: A relational AE with denoising on the inputs X and Y.

F.2.1.4. Stacking PCA and AE

Instead of stacking several AEs stacking different dimensionality reduction algorithms is tenable. Merging both PCA and an AE should in the first step take care of linear dependencies and in a second step recognize the non-linear dependencies, exploiting the positive aspects of both algorithms.

It is reasonable to first remove linear dependencies using the PCA to prevent the AE from wasting neurons on linear dependencies. Besides, the time required by PCA is comparatively short. After retrieving the PCA transformation matrix LT, it is multiplied with the training data. The reduced data is used to train the AE. Normalizing the reduced data can improve the performance of the AE.

As explained in App. F.1, the typical reconstruction error is learned and used to identify outliers that have been projected into the lower-dimensional manifold. Since both algorithms contribute to the reconstruction error the total error is observed.

F.2.1.5. Performance of Non-linear Projection in the RADS

Contrary to the initial assumption none of the examined non-linear projection mechanisms outperforms the basic PCA.

The reprojection error of the basic AE is larger than that of PCA if few dimensions are removed. In contrast, if only one third of the dimensions is retained, the AE performs better with respect to reprojection. Nevertheless, the training and evaluation time for the AE is longer than that of PCA. Further, the anomaly detection capabilities of PCA are equivalent to the AE if few dimensions are retained, while PCA outperforms the non-linear method with many retained dimensions. Since the major decision criterion for performance is not the magnitude of the reconstruction error, but the anomaly detection rate, PCA is considered better than the AE.

To improve the performance of the non-linear projection a denoising AE is employed. The denoising policy increases the reprojection error. The values oscillate around the minimum but do not converge even

if decreasing the learning rate. Nevertheless, it improves the anomaly detection rate. Yet, the major problem results from correlation. The denoising improves the mapping of the data. Instead of only using a very small proportion of the new data space, like the basic AE does, the denoising AE almost exploits the entire space. Although this extensive usage implies that a more reasonable mapping is identified, it also prohibits the sensible application of the RADS. If all data space is populated by *positive* data, there is no opportunity to detect *negative* data. Instead, the strategy would have to be changed to relying on projection only.

Despite being very useful for image processing [39], especially as the learning of input covariances is well suited for noisy data, the relational approach bears several problems. To ensure that all information is taken into account the hidden layer of the denoising AE and the two hidden layers of the relational AE have to be observed for anomaly detection, thus inherently retaining a large number of dimensions. Further inspecting [39] (see Fig. F.1) additionally to the product units also the input information is required for reconstruction. Accordingly, no true dimensionality reduction is achieved. While the filters constructed when applying the relational AE to images can easily be interpreted, the return values do not provide *readable* information about the robot data. The reconstruction error is very small. Yet, using part of the input data for reconstruction this is expectable.

Although the AE performance presumably improves if the dimensionality reduction is performed over several layers, stacking of AEs has not been tested. The performance of the simple AE is not satisfying and identifying the cause of failure gets more difficult the more AEs are involved. Instead stacking of PCA and an AE is evaluated. Using first linear and subsequently nonlinear projection reduces the overall reconstruction error if few dimensions are retained. Keeping the parameters for the AE constant and only changing the number of retained dimensions after PCA exposes that for an AE with a fixed number of parameters there is an optimal PCA reduction. If too many dimensions are retained, the neural net is overwhelmed by the input. The nodes available in the AE cannot sufficiently represent the data resulting in a high reconstruction error. If too few dimensions are kept, the increasing PCA error propagates through the stack and reduces the overall performance and especially increases the reconstruction error.

However, despite the reduced reprojection error with suitable combinations the problems of the basic AE remain. As the AE increases the computational effort but does not improve or even impairs the performance, it is refrained from applying any of the AE variations.

F.2.2. Mixture of Factor Analyzers

MFA, as well, is intended to identify the latent variables in the data set. Using Expectation Maximization (EM) [12], MFA identifies the k Gaussian components with h latent dimensions that explain the n-dimensional data best (typically $h \ll n$). Each of the k components has its own center, own diagonal covariance, and own factor loading matrix, i.e. uses a different subset of the n dimensions [18]. Calculating the likelihood of a specific data point $\boldsymbol{x}$ in the factor model should be sufficient to classify a data point $\boldsymbol{x}$ as *positive* or *negative*. Deep Mixture of Factor Analyzers (DMFA) introduced by [50] improves the performance of MFA by stacking several layers of MFA on top of each other such that each parent layer is a generalization of its children.

Nevertheless, MFA is inapplicable for the RADS. Already the observation of few selected dimensions shows, that the subtle delimitation of valid data from unknown space is impossible unless specifying a large number of clusters. This, in turn, results in high computational complexity. To identify the most suitable parameters k and h in a two layer DMFA for the *static* LWR data, a brute force grid search with cross-validation was intended. However, the evaluation has been aborted without useful results after 15 days of computation on an Intel Core i5 CPU.

G. Handling a Detected Anomaly

If the RADS identifies an anomaly three further questions have to be considered:

1. When does the robot react to an anomaly?
2. What is the reaction to an anomaly?
3. How to determine the cause of the anomaly?

G.1. Filtering Anomalies for Outlier Resistance

Hardware—like a robot—is not necessarily predictable. All sensors suffer from noise, and sometimes sensors return invalid data without being broken. On the other side, real errors most likely persist over a period of time.

No matter how the RADS reacts, frequent superfluous error recognitions might be justifiable, but will be irritating and can prohibit reasonable deployment of the robot. Instead, adding a filter to the system is beneficial. The filter keeps track of the last f evaluations. Only if more than p percent of these f evaluations appear to be irregular, the alarm is issued. The size of p and f depend on the desired sensitivity of the system and the acceptable delay.

Receiving new data at a 1 kHz rate $f = 1000$ and $p = 0.2$ results in a 0.2 second delay if the error is permanently visible. Reaction times of humans vary between 0.2 and 0.6 seconds depending on the type of stimuli, the desired reaction, and whether the human is trained for fast reactions [5, 9, 21]. Thus, stopping the robot by hand (see Sec. G.2) would take at least 0.4–0.8 seconds. Yet, this is tenable as the algorithm is not intended to detect sudden misbehavior causing immediate damage. Rather, it aims at identifying slow changes indicating that something is about to break or is already broken, but does not yet prevent the robot from completing its tasks.

G.2. Reactions of the Algorithm to an Anomaly

Once an anomaly is perceived, an action based on a predefined handling strategy has to be taken. Two methods are conceivable. Either the robot is stopped automatically or a warning is issued and the user has to decide what to do.

Both options have assets and drawbacks. If the robot is stopped automatically, a malfunctioning robot cannot cause further harm. Yet, immediate deactivation of the system might not be in the interest of the user, as moving the robot out of a danger area will be difficult.

Leaving the decision what to do next to the user causes a time delay and requires a responsible reaction from the user. Nevertheless, this is the method of choice for the current implementation. As stated in App. G.1, detecting immediately hazardous errors is unlikely. Therefore, the user can take the most suitable action to bring the task to an acceptable stop while ensuring safety of everyone affected.

Not every anomaly has to be caused by a real error in the system, but might be caused by an unusual configuration of the robot. An experienced operator could realize whether such a strange set up of the robot prevails. He could finish the current task before checking—thereby recording new training data—and, if applicable, retrain the robot.

G.3. Identifying the Cause of Error

Finally, the task of identifying the cause for the anomaly and fixing it comes into effect. Since the algorithm is intended to signal before the user notices the error by himself, identifying the cause is even more difficult than identifying the cause for total failure.

However, each type of error will cause changes in specific dimensions. Automatic classification of the error remains infeasible, because the system aims at unknown errors, i.e. errors which have not appeared before, or that cannot be modeled. But, identifying those dimensions with the strongest divergence from the closest known point can assist troubleshooting.

Instead of searching for the closest point in the training set, it is likewise sensible to use the closest radial basis kernel substituting the real data points. Neither NS nor the SVM can provide sufficient information since they do not provide information on the structure of the valid data space.

For calculating the closest kernel, it is reasonable to use the Mahalanobis distance as a larger value in the covariance hints at typically stronger variation in that dimension. Although the absolute difference might be higher, the relative difference can be less alerting. Alternatively, any L^p metric can be used.

Once the closest point is examined, the distances within the single dimensions have to be assessed. Either the absolute difference $|\boldsymbol{x} - \boldsymbol{\mu}|$ is evaluated for each dimension, or, to take advantage of Mahalanobis properties, Σ^{-1} is used in combination with the Hadamard product:

$$(\boldsymbol{x} - \boldsymbol{\mu}) \circ \left(\Sigma^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right). \tag{G.1}$$

Comparison can be conducted as with absolute values.

Prevalence of the error in one dimension only is unlikely. Instead of returning the single most distant dimension, the dimensions will be sorted according to their divergence.

As explained in App. G.1, the error will appear more than once. Accordingly, a ranking of the divergence in each dimension with regard to all the error values before the RADS reacts can further improve the expressiveness. In order to incorporate the order of dimensions for each outlier in the overall ranking, the dimensions can be weighted according to their ordinal number. The weights for each dimension are added. The resulting list is sorted according to the summed weights. The lowest value indicates the farthest dimension[1].

Even with the ranking, it will remain challenging to find the cause for divergence. However, if a certain joint is especially affected, most likely the dimensions related to that joint will differ, as well as a transmission error in the gear will most likely increase or decrease the current and will have little influence on the position of the joint. Thus, at this point model knowledge is required and will enable the useful application of the RADS.

[1] Results for the identification process are presented in App. H.2.

H. Additional Test Results

H.1. Results from ES II

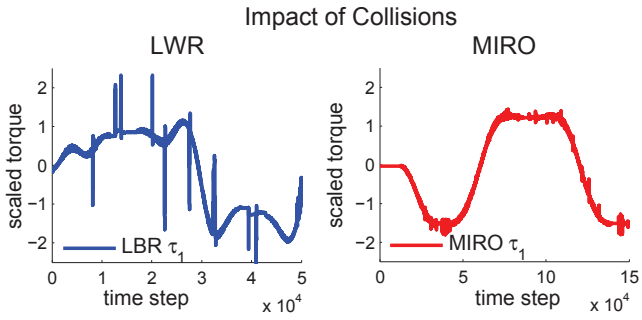


Figure H.1.: Torque in the first joints of the LWR and the MIRO with induced collisions

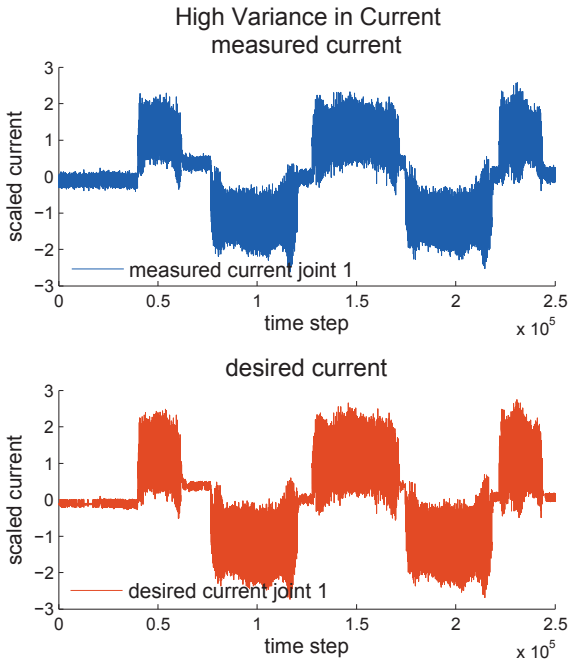


Figure H.2.: The measured and desired current in joint 1 of the MIRO

The results of (HD)RADS on the MIRO data are not as good as the LWR results. Nevertheless, they facilitate fairly reliable outlier detection. A major difference between the LWR and the MIRO is the lower stiffness in the medical robot. Lower stiffness makes the robot more susceptible to disturbances. Therefore, the collisions are induced with less force as Fig. H.1 displays[1].

Besides, the MIRO returns a different set of measurements. In the setup used for data generation the LWR does not provide any current information. The current changes quickly in order to compensate control errors as depicted in Fig. H.2.

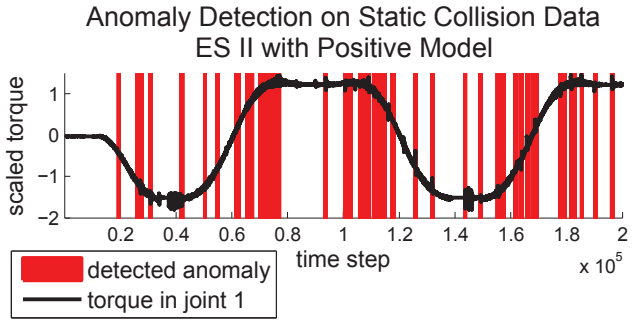


Figure H.3.: Anomaly alarms from positive model during collision trajectory of ES II

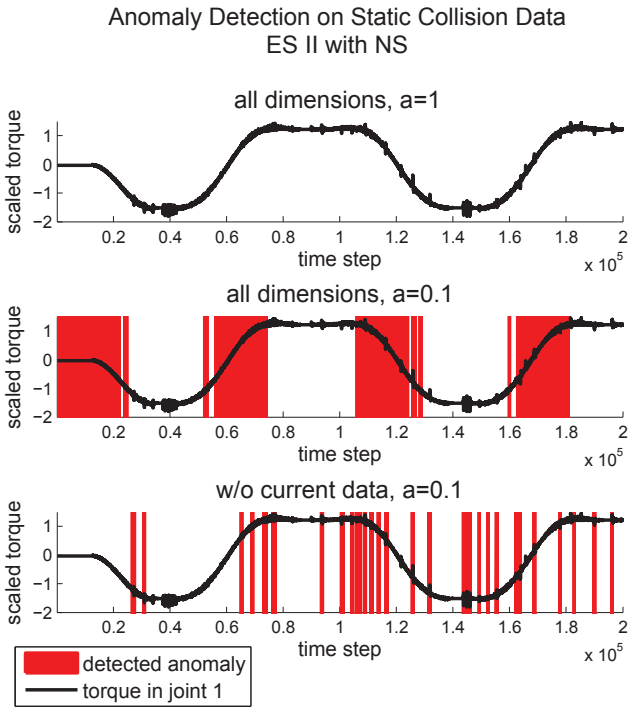


Figure H.4.: Anomaly alarms from negative model during collision trajectory of ES II on *static* data

The RBF-based approach can handle the aggravated conditions in ES II and returns satisfying detection results as depicted in Fig. H.3. In contrast, NS suffers from the variance of the current. Only using the *static* data of ES II choosing small values for the noise amplitude a results in alarms at any time of movement—collisions in the held positions are not detected due to the low coverage $c = 0.9$—or if a is large no alarms are raised. On the other side, if the current related values are excluded from the data set, the detection rate is satisfying (see Fig. H.4). Using the entire *dynamic* data the detection rate improves drastically. Still, removing the current data optimizes the results and facilitates

[1] Being able to detect collisions with lower impact further confirms the validity of the approach.

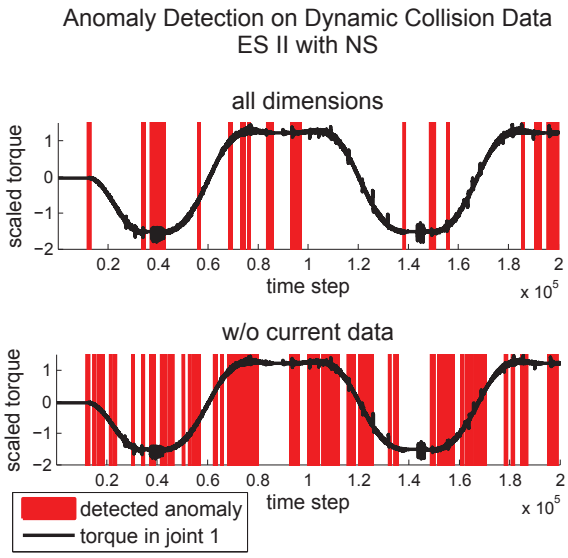


Figure H.5.: Anomaly alarms from negative model during collision trajectory of ES II on *dynamic* data

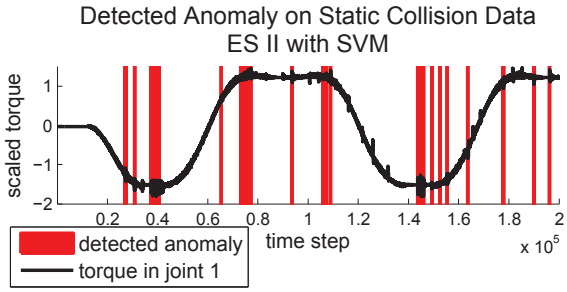


Figure H.6.: Anomaly alarms from SVM during collision trajectory of ES II on *static* data

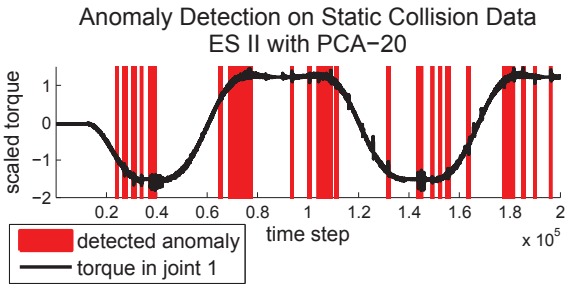


Figure H.7.: Anomaly alarms from PCA with 20 retained dimensions during collision trajectory of ES II on *static* data

detecting most anomalies. The results are displayed in Fig. H.5. As with the LWR filtering of the *dynamic* data causes the areas of detection to be broader than the actual collision.

For the evaluation of the MIRO only a short sequence of data is used. Extending the training period is likely to counterbalance the influence of the current data. Using more training data decreases the required security border in NS and more of the possible combinations of noisy values are captured.

For testing the SVM the NS results received without the current data are used to be able to make a statement about performance. Accordingly, the positive model has to be trained on the smaller data, as well. All collisions that have been missed using all dimensions are detected

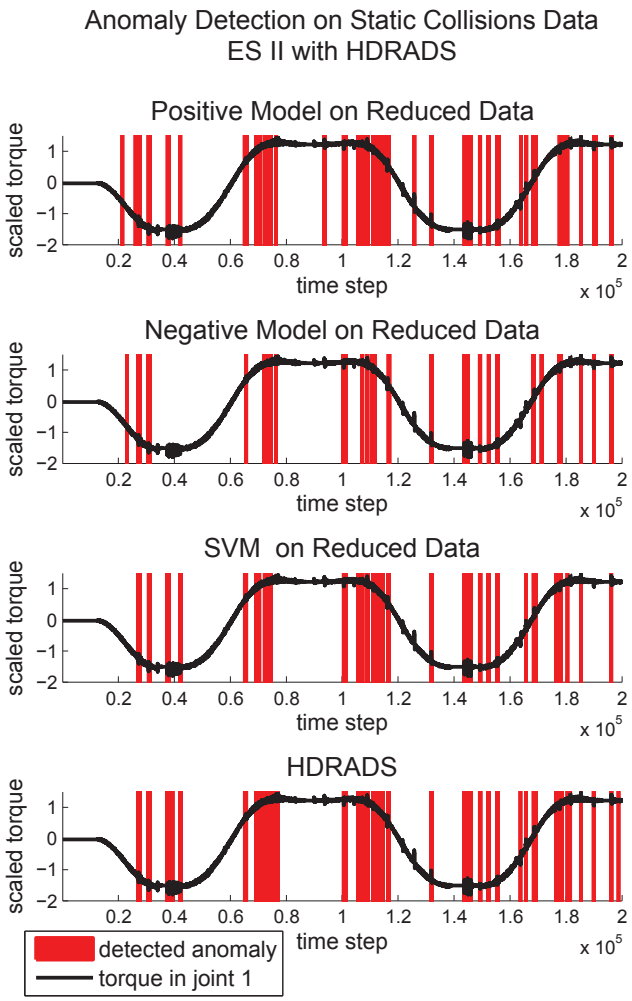


Figure H.8.: Anomaly alarms from HDRADS during collision trajectory of ES II on *static* data. The data is reduced to 20 dimensions by PCA

using the reduced data. As Fig. H.6 shows, there is room for improvement, but the RADS detects most of the collisions in *static* data. Applying the algorithm to *dynamic* data the results further improve because the starting values are more sensitive.

Projection performs comparable on LWR and MIRO data. As expected the reconstruction error increases the fewer dimensions are retained during projection. Using a sufficiently small number of retained dimensions ensures that the reprojection error is applicable to identify errors. For *static* ES II keeping 50 of the 60 dimensions is too much, hardly any of the collisions are perceived. Only keeping 20 dimensions provides good results, as depicted in Fig. H.7. For anomaly detection through the reprojection error on *dynamic* MIRO data it is nearly indifferent whether 20 or 50 dimensions are retained. Both detection rates are satisfying. The results are slightly better if all data—also current—is included.

Stacking the basic RADS on top, the system becomes more sensitive to the number of retained dimensions. The fewer dimensions are retained, the smaller is the worst case number of calculations. However, the fewer dimensions are retained, the larger is the variance across dimensions. Thus, a larger area of the space has to be

represented by the positive model. The higher number of kernels increases the required computations. Since the computational complexity for the Mahalanobis distance is $O(n^2)$, but the complexity is linear with respect to the number of kernels, a reduction of dimensions is desirable if the number of kernels does not grow exponentially. Moreover, the variance in the retained dimensions differs from the initial variance and has to be reassessed. For the MIRO data the noise amplitude a is tripled if reducing the *static* data to 20 dimensions. Taking this change into account the number of kernels grows much slower than the dimensionality decreases. Thus, the time required for computation decreases noticeably.

The performance of the HDRADS is illustrated in Fig. H.8. The joint approach outperforms any of the single methods. It is faster than using the basic RADS—even during training—it detects significantly more collisions than the reprojection error and identifies a few more collisions than the SVM on the projected data. However, some of the collisions are not recognized and some valid areas are considered as faulty. On the one hand, a larger training set will improve the performance. On the other hand, refraining from using strongly varying data like the current can further improve the results.

H.2. Error Identification

App. G.3 introduces a possibility to narrow down the possible causes for an anomaly. Forcing some of the dimensions in the training data to likely but constant values, e.g. -1, 0, or 1, and using the corrupted data for testing results in an identification of those dimensions as most divergent. Though, if the true value in that dimension is very close to the manually inserted value, the dimension will be located further down the list.

Solely taking the data related to a single collision and testing the error identification shows that dimensions related to the joint suffering the strongest impact are identified. Besides, only dimensions that measured the disturbance are listed as divergent while, e.g., desired values remain at the end of the list.

Although the identification of the cause remains challenging, the add-on to the RADS does support the user. Moreover, the additional effort is minimal since the evaluation only has to be performed if an alarm is issued. Unfortunately, the method is only applicable to the basic RADS since the positive model is created for the projected dimensions in the HDRADS. An additional positive model of the full data can be created if support for troubleshooting is inevitable.

I. Future Areas of Research

Although the HDRADS provides stable results and can identify errors, the possibilities for improvement are far from being exploited.

Foremost, further testing and validation of the approach is essential to confidently rely on the algorithm. The evaluation performed during the course of this thesis is based on artificially induced errors like collisions, changes in speed, and load. Typically, these errors have a strong deflection in a few related dimensions. Inducing small divergences, like they will result from wear, is difficult. A long term observation of a robot could help to show the benefit of the introduced method, especially comparing it to the currently applied fault detection mechanisms. Besides, applying established methods concurrently with the HDRADS can improve acceptance by the users, or could help to verify models of the robot.

Becoming less dependent on dimensionality reduction and being capable to deal with data remaining high-dimensional after projection can be achieved through distance metrics that inherently reduce the influence of dimensionality.

Capable of handling higher dimensionality the RADS can use more generated data. For this thesis only the first and second derivative have been applied. Identifying and testing further metrics might increase the number of perceptible errors.

For anomaly detection it is important to be able to distinguish valid from invalid data. Many projection methods, like PCA and neural networks, are intended to remove the *white space* and extensively exploit the subspace. Thus, deciding whether a data point is in or outside of the training data, respectively the valid model, becomes increasingly difficult. Switching to algorithms trying to maintain the sparsity of the data is auspicious. Minor Component Analysis (MCA) and Extreme Component Analysis (XCA) [34, 54], might tackle this problem and are recommended as a future area of investigation.

Instead of applying the introduced (HD)RADS, other approaches should be investigated and compared with regard to their performance and scalability. Density forests are a subgroup of random forests and seem to perform well in practice. Using several random trees the density of the data space is modeled [10]. Instead of first reducing the dimensionality and in a second step identifying anomalies, density forests could be capable of handling the high-dimensional input directly and classifying the data points into *positive* and *negative* data according to their likelihood in the density model.

Glossary

central limit theorem
The central limit theorem states that if iid sampling is performed long enough the distribution of the data will be Gaussian.

conjugate gradient method
The conjugate gradient method is a numerically efficient method to solve equations of the form $A\boldsymbol{x} = b$. After at most m iterations with $A \in \mathbb{R}^{m \times m}$ the exact solution is retrieved. However, since the error decreases monotonic, it is especially interesting for iterative methods.

cross-validation
In cross-validation the training set is separated into n subsets of equal size. Sequentially one subset is left out from the training data. The classifier is trained on the remaining $n-1$ subsets. Subsequently the classifier is tested against the remaining samples. Merging the percentage of correct classification the prediction accuracy for the entire set can be evaluated. The parameters that return the best overall prediction results are selected as parameters, thus avoiding overfitting.

Euclidean distance
Distance of two points $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^n$ according to the L^2-Norm, $\sqrt{\sum_{i=1}^{n}(\boldsymbol{x}-\boldsymbol{y})^2}$.

Hadamard product
For any two matrices A and B of the same dimensions $n \times m$ the Hadamard product $(A \circ B)$ (also known as entry wise product or Schur product) is defined as the matrix $C \in \mathbb{R}^{n \times m}$ with $C_{i,j} = A_{i,j}B_{i,j}$.

latent variable
A latent variable is one of the most basic influence factors for a data set. Changes in the latent variable are the cause for variance in the data.

Mahalanobis distance
The Mahalanobis distance is a distance measure that weights distances according to the covariance of an n-dimensional data set. Based on the correlations Mahalanobis distance generally is scale invariant.

Bibliography

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

[2] C. C. Aggarwal and P. S. Yu, "Outlier Detection for High Dimensional Data," in *Proceedings of the 2001 ACM SIGMOD International Conference on 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 International Journal*, vol. 34, pp. 376–385, 2007.

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

[5] B. G. Breitmeyer and J. I. Breier, "Effects of Background Color on Reaction Time to Stimuli Varying in Size and Contrast: Inferences about Human M Channels," *Vision Research*, vol. 34, pp. 1039–1045, 1994.

[6] D. Burschka, "Bewegungsplanung in der Robotik," 2009, slide set: Path Planning for Point Robots.

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

[8] H. Choset, K. M. Lunch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[9] G. N. Christenson and A. M. Winkelstein, "Visual Skills of Athletes versus Nonathletes: Development of a Sports Vision Testing Battery." *Jornal of the American Optometric Association*, vol. 59, pp. 666–675, 1988.

[10] A. Criminisi, J. Shotton, and E. Konukoglu, "Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning," Microsoft Research, Tech. Rep., 2011.

[11] D. Dasgupta and S. Forrest, "Novelty Detection in Time Series Data using Ideas from Immunology," in *Proceedings of the 5th International Conference on Intelligent Systems*, 1995.

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood for Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, pp. 1–38, 1977.

[13] Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR), "DLR MIRO: A Versatile Robot for Surgical Applications," http://www.dlr.de/rm/desktopdefault.aspx/tabid-3761/6104_read-514, 2012.

[14] W. E. Dixon, I. D. Walker, D. M. Dawson, and J. P. Hartranft, "Fault Detection for Robot Manipulators with Parametric Uncertainty: A Prediciton-Error-Based Approach," in *IEEE Transactions on Robotics and Automation*, 2000, pp. 689–699.

[15] Z. Ferdousi and A. Maeda, "Unsupervised Outlier Detection in Time Series Data," in *Proceedings of the 22nd International Conference on Data Engineering Workshops*, 2006, pp. 51–56.

[16] R. A. Fisher, *Statistical Methods For Research Workers*, ser. Cosmo study guides. Cosmo Publications, 1925.

[17] B. Fritzke, "A Self-Organizing Network that can Follow Non-Stationary Distributions," in *Artificial Neuroal Networks - ICANN'97*, A. Gerstner, Wulfram ander Germond, M. Hasler, and J.-D. Nicourd, Eds. Springer Berlin/Heidelberg, 1997, pp. 613–618.

[18] Z. Ghahramani and G. E. Hinton, "The EM Algorithm for Mixtures of Factor Analyzers," University of Toronto, Tech. Rep., 1996.

[19] 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 International Conference on Robotics and Automation*, 2000, pp. 2302–2309.

[20] U. Hagn, M. Nickl, S. Jörg, G. Passig, T. Bahls, A. Nothelfer, F. Hacker, L. Le-Tien, A. Albu-Schäffer, R. Konietschke, M. Grebenstein, R. Warpup, R. Haslinger, M. Frommberger, and G. Hirzinger, "The DLR MIRO: A Versatile Lightweight Robot for Surgical Applications," *Industrial Robot: An International Journal*, vol. 35, pp. 324–336, 2008.

[21] D. Hecht, M. Reiner, and G. Halevy, "Multimodal Virtual Environments: Response Time, Attention, and Presence," *Presence*, vol. 15, pp. 515–523, 2006.

[22] M. R. Hestenes and E. Stiefel, "Methods of Conjugate Gradients for Solving Linear Systems," *Journal of Research of the National Bureau of Standards*, vol. 49, pp. 409–436, 1952.

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

[24] D. Hsu, M. Hsu, H. Huang, and E. B. J. Montgomery, "An Algorithm for Detecting Oscillatory Behavior in Discretized Data: The Damped-Oscillator Oscillator Detector," *ArXiv e-prints*, 2007.

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

[26] A. Hyvärinen and E. Oja, "Independent Component Analysis: Algorithms and Applications," *Neural Networks*, vol. 13, pp. 411–430, 2000.

[27] Z. Ji and D. Dasgupta, "Real-Valued Negative Selection Algorithm with Variable-Sized Detectors," in *Genetic and Evolutionary Computation - GECCO'04*, K. Deb, Ed. Springer Berlin/Heidelberg, 2004, pp. 287–298.

[28] L. G. Khachiyan, "Rounding of Polytopes in the Real Number Model of Computation," *Mathematics of Operations Research*, vol. 21, pp. 307–320, 1996.

[29] D. Kresimir, M. Grgic, and S. Grgic, "Independent Comparative Study of PCA, ICA and LDA on the FERET Data Set," *International Journal of Imaging Systems and Technology*, vol. 15, pp. 252–260, 2005.

[30] KUKA Roboter GmbH, "KUKA: Presse und Downloads," http://www.automation-becomes-easy.com/de/presse-und-downloads.html, 2010.

[31] B. Kulis and M. I. Jordan, "Revisiting k-means: New Algorithms via Bayesian Nonparametrics," in *Proceedings of the 29th International Conference on Machine Learning*, 2012, pp. 513–520.

[32] N. D. Lawrence, "Dimensionality Reduction the Probabilistic Way. An Overview of Probabilistic Dimensionality Reduction," ICML Tutorial, 2008.

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

[34] F.-L. Luo and R. Unbehauen, "A Minor Subspace Analysis Algorithm," *IEEE Transactions on Neural Networks*, vol. 8, pp. 1149–1155, 1997.

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

[36] C. Marselli, D. Daudet, H. P. Amann, and F. Pellandini, "Application of Kalman Filtering to Noisereduction on Microsensor Signals," in *Proceedings of the Collloque interdisciplinaire en instrumentation*, 1998, pp. 443–450.

[37] T. Martinetz and K. Schulten, "A 'Neural-Gas' Network Learns Topologies," *Artificial Neural Networks*, vol. I, pp. 397–402, 1991.

[38] M. L. McIntyre, W. E. Dixon, D. M. Dawson, and I. D. Walker, "Fault Detection and Identification for Robot Manipulators," in *IEEE International Conference on Robotics and Automation*, 2004, pp. 4981–4986.

[39] R. Memisevic, "Gradient-based Learning of Higher-Order Image Features." in *Proceedings of the International Conference on Computer Vision*, 2011, pp. 1591–1598.

[40] K. Pearson, "On Lines and Planes of Closest Fit to Systems of Points in Space," *Philosophical Magazine*, vol. 2, pp. 559–572, 1901.

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

[42] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," in *Neurocomputing: Foundations of Research*, J. A. Anderson and E. Rosenfeld, Eds. MIT Press, 1988, pp. 673–695.

[43] A. Savitzky and M. J. E. Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures." *Analytical Chemistry*, vol. 36, pp. 1627–1639, 1964.

[44] R. Schneider and P. M. Frank, "Fuzzy Logic Based Threshold Adaption for Fault Detection in Robots," in *Proceedings of the Third IEEE Conference on Control Applications*, 1994, pp. 1127–1132.

[45] 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.

[46] J. R. Shewchuk, "An Introduciton to the Conjugate Gradient Method Without the Agonizing Pain," Carnegie Mellon University, Tech. Rep., 1994.

[47] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing.* California Technical Publishing, 1997.

[48] S. Stigler, "Fisher and the 5% Level," *CHANCE*, vol. 21, pp. 12–12, 2008.

[49] M. Stone, "Cross-Validatory Choice and Assessment of Statistical Predictions," *Journal of the Royal Statistical Society B*, 1974.

[50] Y. Tang, R. Salakhutdinov, and G. Hinton, "Deep Mixtures of Factor Analysers," in *Proceedings of the 29th International conference on Machine Learning*, 2012, pp. 505–512.

[51] L. Tarassenko, A. Nairac, N. W. Townsend, I. Buxton, and P. Cowley, "Novelty Detection for the Identification of Abnormalities," *International Journal of Systems Science*, vol. 31, pp. 1427–1439, 2000.

[52] D. M. J. Tax and R. P. W. Duin, "Support Vector Domain Description," *Pattern Recognition Letters*, vol. 20, pp. 1191–1199, 1999.

[53] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[54] M. Welling, F. V. Agakov, and C. K. I. Williams, "Extreme Components Analysis," in *Proceedings of the 16th Conference on Advances in Neural Information Processing Systems*, 2003, pp. 137–144.

[55] W. Zhao, R. Chellappa, and A. Krishnaswamy, "Discriminant Analysis of Principal Components for Face Recognition," in *Proceedings of the 3rd IEEE International Conference on Automatic Face and Gesture Recognition*, 1998, pp. 336–341.