Method for automated detection of outliers in crash simulations

David Kracker[1], Revan Dhanasekaran[1], Axel Schumacher[2], Jochen Garcke[3,4]

[1]Dr. Ing. h.c. F. Porsche AG, Body Engineering CAE Vehicle Safety, 71287 Weissach, Germany, david.kracker1@porsche.de
[2]University of Wuppertal, Faculty for Mechanical Engineering and Safety Engineering, Chair for Optimization of Mechanical Structures, 42119 Wuppertal, Germany
[3] Fraunhofer Center for Machine Learning and Fraunhofer Institute for Scientific Computing (SCAI), 53757 Sankt Augustin, Germany
[4] Institut für Numerische Simulation, Universität Bonn, 53115 Bonn, Germany

Abstract
Stricter legal requirements in crash safety lead to more complex development processes in computer-aided engineering and result in an increasing number of simulations. Both, the construction of the simulation models as well as their evaluation are costly and time-consuming. Therefore, an automated workflow is required that significantly facilitates the analysis of the results by the engineer and increases the quality of the evaluation.
In this study an automated evaluation process is proposed that detects anomalous crash behavior in a bundle of crash simulations. The individual states from the simulation are analyzed separately from each other and an outlier score is calculated using a kth-nearest-neighbor approach. Subsequently, these results are averaged into one score for each simulation. With the help of different statistical methods a threshold value is calculated, from which a simulation can be identified as an outlier. The evaluation is carried out on 5 datasets. On average, the precision and recall of the presented method are 1.0 and 0.91, respectively.

Keywords:
Crash Simulation Analysis, Outlier Detection, Machine Learning, Process Automation

1 Introduction

When developing vehicles, a large number of different load cases, covering a range of crash scenarios, are taken into account. The load cases result from legal requirements and the evaluation criteria of consumer protection organizations such as US-NCAP and Euro-NCAP [1]. It is not possible to cover all of these load cases with hardware tests of prototypes during vehicle development. For this reason, the development of crash structures has been based on the results of explicit finite element calculations for decades [2]. Tests are only used to verify the simulation models and for the final evaluation shortly before the start of production (SOP).

In the conventional, simulation-driven development process, the results of a calculation are used for new design ideas with regard to the position and arrangement, shape and dimensioning of the structural components. These ideas are integrated in Computer Aided Design (CAD) or directly in the finite element preprocessor and the new structure is calculated again. This is the process to get better and better designs over weeks and months. Nevertheless, the following properties of crash structures are hardly taken into account in this process:

- numerical and physical bifurcation points,
- non-smooth structural responses,
- mesh dependent results.

Therefore, the developed crash structures can be very unrobust. Various simulations differ in their crash behavior. Deformations, rigid body movements and failures can vary significantly. Often, a few simulations stand out as outliers with their crash behavior. If these outliers are misinterpreted or overlooked in the evaluation, this can lead to incorrectly taken measures. Extensive development loops can result and the development process is unnecessarily prolonged.

Nowadays, engineers are no longer able to evaluate all simulations for all quantities of interest on hundreds of components in detail. Standardized analysis tools support them in their work, but they are static and only evaluate certain predefined measurements. The large number of simulations and the complexity of the vehicle models require a more sophisticated evaluation of the results. Overall, there is growing need for an intelligent automated evaluation that analyzes the entire vehicle and all existing quantities of interest and provides the results to the engineer in a clear manner.

In this work, we present a method for the automated detection of anomalies in crash behavior. First an overview of the state of the art is given. Subsequently, we introduce a method for automated detection of outliers in crash simulations. In the fourth section, the method is investigated in various experiments. The last section summarizes the results and gives an outlook on further research.

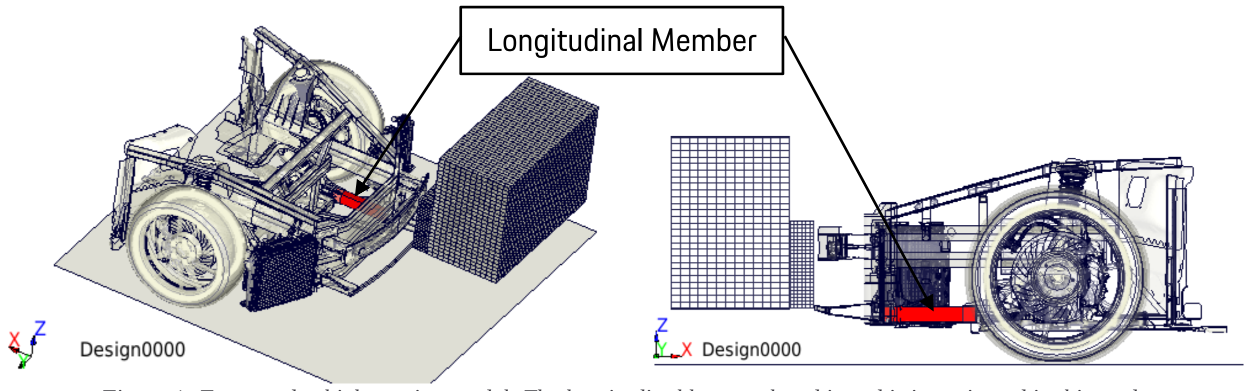

Figure 1: Front-end vehicle section model: The longitudinal beam colored in red is investigated in this study

2 State of the Art

2.1 Preparation of the Finite Element Data

Currently, different approaches for a more intelligent, comprehensive and automated evaluation of crash simulations are being investigated. All of them have the problem of the representation of the FE data in common. Different FE meshes of the different simulations make the direct processing with further algorithms difficult. In the literature, there are several approaches to prepare the data accordingly and thus make them comparable.
In [3], the geometric center of gravity is determined along the longitudinal axis of the component and the information of the surrounding elements is projected onto this axis. This procedure is suitable for profile-shaped components, but it is difficult to transfer it to planar components.
In [4] and [5], the finite elements are projected onto a spherical surface. This spherical surface is then discretized and the previously three-dimensional data are now available as a two-dimensional image. A disadvantage of this method is that a distortion of the data occurs due to the spherical projection.
In [5], the components are discretized by voxels. This preserves the three-dimensional structure. The quality of this approach depends on the discretization. For a fine discretization, it must be taken into account that this approach requires more memory than the previous approaches due to its three-dimensional structure. A coarse discretization, on the other hand, is accompanied by a greater loss of information.
In [6], one simulation is used as the reference. Its finite element mesh defines the data representation for all other simulations. The FE meshes of other simulations are mapped to the chosen reference mesh. While the full resolution of the original data is preserved, larger changes in the mesh can result in artefacts in later analysis steps if not suitably addressed, see [7], [8], [9].

2.2 Outlier detection

According to Hawkins [10], an outlier can be defined as follows:
“An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.”
Thus, outliers are not only phenomena that occur rarely, but are significantly differentiated from the other data primarily by their characteristics. The other data are referred here accordingly as inliers. In the context of crash simulations, there is no other work specifically addressing the automatic detection of outliers that we are aware of. Outlier analysis is implicitly present in data analysis using dimensionality reduction, see e.g. [5], [11], [7], [8], [9].
However, there has been long-standing research on algorithms and data representations for outlier detection in other application areas. On the one hand, since outliers can negatively affect the performance, outlier detection is an often-used step in preprocessing before machine learning models are trained [12]. On the other hand, it is important to know about existing outliers as they could be an indication of an unusual event or a defect in a system.
For example, in [13], outlier detection is used to detect unusual energy consumption in buildings. [14] analyzes abnormalities in credit card payment transactions to detect credit card fraud.

Regardless of the use case, outlier detection must distinguish between supervised, semi-supervised and unsupervised learning methods [12]. In the first two variants, prior knowledge about the data is assumed. An algorithm is trained by receiving both the data and the associated classes as information. In the context of outlier detection, the associated classes mean whether a data point is an inlier or an outlier. In unsupervised algorithms, this prior knowledge is not assumed. The algorithm must independently identify the outliers in the raw data. Since prior knowledge about inliers and outliers for hundreds of components cannot be assumed in the context of crash simulations, an unsupervised learning method is used in this work.

There are many algorithms for unsupervised outlier detection [12]. The proposed workflow is based on an outlier score, that estimates how severe the anomaly is. A simple, basic approach is using the distance to the kth nearest neighbor (KNND) [15] as the outlier score. This method depends only on a single hyperparameter k, which determines the kth neighbor used in the calculation of the outlier score. Typically, more complex models with many hyperparameters can achieve better results than simple algorithms. However, selecting these hyperparameters is a laborious process that can depend heavily on the used dataset. In particular, transferability to other data is not always feasible.

Once for each component of the simulations the outlier score has been calculated, the next step is to decide which instances are selected as outliers and are reported to the engineer. Since no prior knowledge of the underlying data can be assumed, no fixed threshold value can a-priori be defined for all components. However, there are various statistical methods for this problem.

A well-known statistical method for the identification of outliers involves the calculation of z-scores [16]. Their use is based on the assumption of normally distributed data. From each dataset, the mean and standard deviation can be calculated. Data that are more than n times the standard deviation away from the mean are marked as outliers. The choice depends on how sensitively the procedure should detect extreme values. For example, 99.7% of the normally distributed data are within three times the standard deviation (n=3). Within the 1.65 standard deviation 80% of the data lie (n=1.65). Depending on the expected proportion of outliers, the value of n can be selected from statistics tables. In the following, this method is referred to as the STD method. One disadvantage of STD is that the calculation of the threshold itself is dependent on outliers in the dataset [17].
Another well-known method, which is more robust towards outliers, stems from the visualization by box plots [18], [19]. The basis for this is the interquartile range. First, the data are sorted in ascending order by their value. The first and third quartiles are determined within which 25% and 75% of the data lie, respectively. In the next step, the middle 50% of the data is selected and the difference between the largest and smallest of these values is calculated. The result is called the interquartile range (IQR). In a boxplot, these two values mark the position of the box. Data points that are more than 1.5 times the IQR away from the third quartile of the data are marked as outliers. At this distance, the so-called whiskers are drawn in the boxplot. In the following, this method will be referred to as the IQR method. If more than 25% of the data consist of outliers the calculation of the threshold is affected.
Furthermore, the robust z-score [20], [17] & [21] is often used. Unlike quartiles, means and standard deviation, using the median of a dataset is more robust in detecting outliers. First, the median of the dataset X is calculated. Then, the median of the absolute deviation from the median is calculated and multiplied by the factor b, see equation (1). In the literature, normally distributed data is often assumed, which is why b is chosen to be 1.48 [21]. The resulting value is called the median absolute deviation:

$$\boldsymbol{MAD = b * Median(|X - Median(X)|)} \tag{1}$$

Similar to the standard deviation, the MAD is a measure of the dispersion of the data, but now with respect to the median. To calculate the threshold for outlier detection, the MAD is multiplied by the value a and added to the median. Typical values for a from the literature are 3, 2.5 and 2 [22]. All data exceeding the resulting threshold are marked as outliers. In the following, this procedure is called the MAD method.

3 Method for automated detection of outliers in crash behavior

In this chapter, a method for automated detection of outliers in crash simulations is proposed. The workflow is visualized in Figure 2. Once a new simulation has been calculated, the outlier detection is automatically started. The new simulation is compared to the existing ones in the simulation database. Note that a vehicle consists of hundreds of construction parts. In order to provide detailed information where in the car outlying crash behavior is detected, an outlier score is calculated individually for each component. Based on the computed outlier scores a threshold is calculated, which indicates whether a component shows a suspicious crash behavior or not. As a result of the procedure, the components with the most anomalous crash behavior are presented to the analyst. In the following sections, this workflow is described in detail.

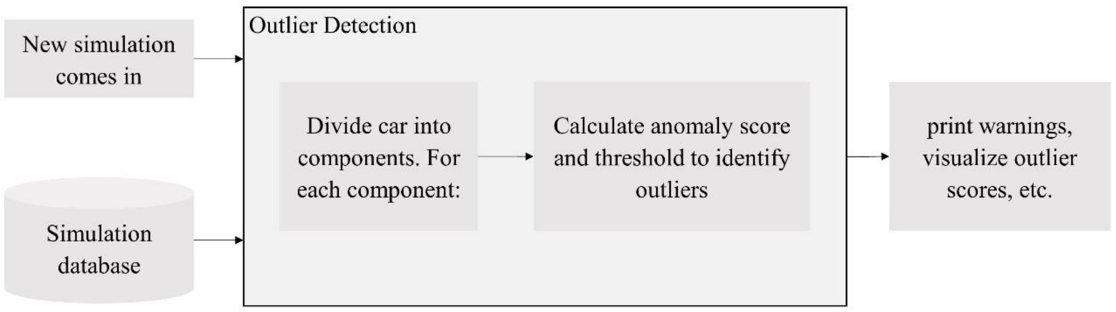
New simulation comes in
Simulation database
Outlier Detection
Divide car into components. For each component:
Calculate anomaly score and threshold to identify outliers
print warnings, visualize outlier scores, etc.

Figure 2: Procedure of the automated Outlier Detection for Crash. Each component of a new simulation is analyzed individually and compared to the existing database. The engineer gets warned if an anomalous behavior occurs in the new simulation.

3.1 Preparation of the crash simulation data

Before the FE data can be analyzed in an automated process, the meshes have to be prepared accordingly.

In the case of robustness analyses, in which parameters such as the wall thicknesses of individual components are scattered, the finite element meshes of the individual simulations are identical in each case. Here, the information of the element data can be compared directly. In the case that the finite element meshes of the individual simulations and components differ, for example due to geometric differences, the data must be preprocessed accordingly by mapping [6] or discretization [5] approaches so that the information of the elements becomes comparable. The number of elements of a component is denoted by n_{el} in the following. For each of these elements, different evaluation quantities are available, such as displacement, acceleration or plastic strain. Depending on which evaluation quantity is used, statements can be made about the deformation, failure or kinematic behavior during the analysis.

Since crash simulation is a dynamic process, the data does not only have spatial characteristics but also comes as a time series. Here, the temporal behavior has to be considered appropriately as well. The points in time where information is stored in the simulation output are usually called states.

Note that there can be situations, where simulations must be aligned in time (due to time shifts and offsets) as a preprocessing step in order to make them comparable. If, for example, the initial distance between the vehicle and the barrier differs in one simulation compared to all other simulations, but the same crash behavior is present in all variants, the one simulation will be interpreted as an outlier if no corresponding temporal correction of the signals has taken place. This would not be a desired result of the outlier detection. In this case, a cross-correlation could be used to automatically calculate the time offset and correct it accordingly, so that a fair comparison between the state-based analysis is possible and the real outliers can be found.

3.2 Calculation of the Outlier Scores

The individual steps for the calculation of a single outlier score and the decision whether a component shows a conspicuous crash behavior are summarized in Figure 3. In the following, the procedure is described in detail.

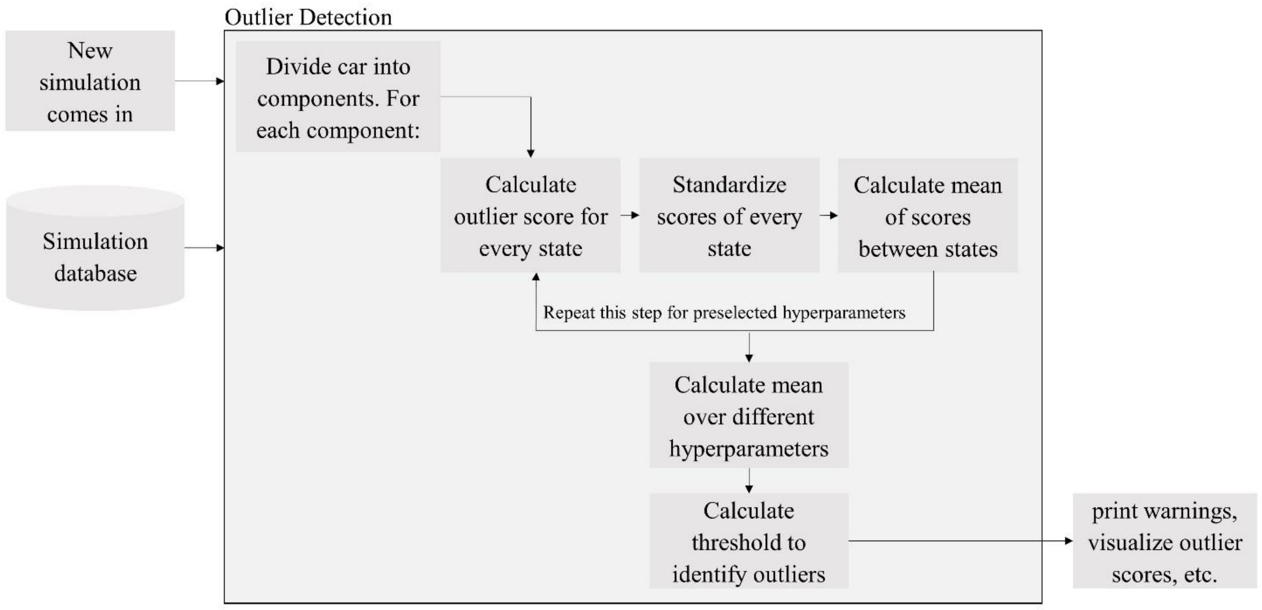
Outlier Detection
New simulation comes in
Simulation database
Divide car into components. For each component:
Calculate outlier score for every state
Standardize scores of every state
Calculate mean of scores between states
Repeat this step for preselected hyperparameters
Calculate mean over different hyperparameters
Calculate threshold to identify outliers
print warnings, visualize outlier scores, etc.

Figure 3: Procedure of the automated Outlier Detection for Crash. Outlier scores are calculated for every state. In order to get one score for the whole simulation, scores are standardized and summarized over time by mean calculation. This procedure is repeated for different hyperparameters of the outlier detection algorithm. The scores of different hyperparameters are combined to build an ensemble method and increase robustness. A threshold is used to decide whether a component shows anomalous crash behavior.

Since the focus in this report is on the approach for detecting outliers in crash simulations rather than comparing different algorithms, a simple and robust method is selected for this study. Therefore, a kth-nearest-neighbor method is used to determine the outlier score. Here, it corresponds to the chosen distance between a simulation and its kth neighbor. We calculate the distance between two simulations using the Euclidean distance of their respective data representations consisting of the individual component elements and the corresponding evaluation variable (e.g. plastic strain). The calculation of the Euclidean distance between two simulations *x* and *y* is given by

$$\boldsymbol{d(x,y) = \sqrt{((x_1 - y_1)^2 + (x_2 - y_2)^2 + .. + (x_{nel} - y_{nel})^2}} \tag{2}$$

The procedure is exemplary shown in Figure 4. State 25 of four simulations and a virtual component consisting of 3 finite elements are considered. The individual simulations are shown in the rows, while the plastic strains of the respective element of the component to be analyzed are displayed in the columns.

	Plastic Strain		
State25	el1	el2	el3
Sim1	1	3	2
Sim2	2	3	2
Sim3	1	5	2
Sim4	1	3	8

	Distance Matrix			
k=2	Sim1	Sim2	Sim3	Sim4
Sim1	0	1	2	6
Sim2	1	0	2,24	6,08
Sim3	2	2,24	0	6,33
Sim4	6	6,08	6,33	0

Figure 4: Left: plastic strains of the individual elements of four simulations exemplary for state 25. Right: Euclidean distance between the simulations. In blue the second nearest neighbors (k=2) are colored exemplarily per row and correspond to the outlier scores of the corresponding simulation. The k-nearest neighbor relation is not symmetric in general.

If the values of the elements are considered across the simulations, it is noticeable that the fourth simulation stands out as an outlier due to an increased value in the third element. The results of the distances of all simulations to each other are summarized in the distance matrix in Figure 4.
The next step is to decide which value for the hyperparameter k should be used in the analysis. In this example, k=2 is chosen as an example. The corresponding kth neighbors are highlighted in the figure. Thus, the outlier scores of the simulations for state 25 are available and it becomes obvious that the fourth simulation seems to be an outlier. This procedure is repeated for all states.

For each time step and each simulation, the outlier scores for the considered component are calculated separately and stored in the matrix S. In order to make a final evaluation of whether an entire simulation is an outlier, the evaluations from the individual states must be summarized. For this purpose, the mean value is calculated across the states. Since the distance based KNND method is used in this study, it is important that the scores of the states are scaled appropriately in each case before summation. In early states, where only small deformations and correspondingly small plastic strains are present, the distances between the individual simulations are also small, resulting in a low outlier score. In the later states where the deformations and plastic strains become larger, the distances to the nearest neighbors and as a result the corresponding scores are larger. If the scores of the individual states were averaged without prior scaling, this would lead to a larger weighting of the later states. This could cause simulations with true outlier crash behavior to be missed in the early stages of the crash. Therefore, the values must be scaled appropriately before summation [12]. In this study, the characteristic values of the states are standardized separately. The columns of matrix S contain the information about the individual states. For every column of S (denoted by S_{col}), its mean μ is subtracted and the result is divided by its standard deviation σ according to

$$\boldsymbol{Z} = \frac{\boldsymbol{S_{col}} - \boldsymbol{\mu}(\boldsymbol{S_{col}})}{\boldsymbol{\sigma}(\boldsymbol{S_{col}})} \quad . \tag{3}$$

After this step, a value is available for each simulation and component, which tells how much this simulation shows an outlier behavior (see Figure 4).
The results of the KNND method are strongly dependent on the choice of the parameter k. The ideal value of this parameter in turn depends on the nature of the data and the composition of inliers and outliers. It is only possible to determine a suitable value by having prior knowledge about the data. In general, however, there is no prior knowledge about the crash simulations at hand, so it is not known if and how many outliers are included. Accordingly, the ideal value for k cannot be determined in advance. In practice, however, the procedure should be fully automated and work without prior knowledge about the individual components.
Therefore, a so-called ensemble model is used, which increases stability of the predictions and leads to more robust results [12]. It consists of several individual models whose outlier scores are averaged. First, a selection of appropriate hyperparameters is made. Then, the procedure described up to this point is repeated for the selected values of k. For each state, models are trained, their results standardized and averaged over the states. Finally, the results of the different models with different values of the hyperparameter k are averaged.
Next, it must be decided at which value a simulation is classified as an outlier and a warning is displayed to the engineer. Since the vehicle consists of hundreds of components and outlier detection is supposed to make the evaluating engineer's job easier, it is desirable that warnings are only displayed when something out of the ordinary happens. Any misclassification distracts the engineer, costs time and reduces confidence in the system.
However, it is not possible to define a fixed threshold value, since this depends on the algorithm, the available data and the component under consideration. Instead, the earlier described statistical methods such as the median absolute deviation can be used to calculate this threshold value. The result of the outlier detection method are warnings of components that show a conspicuous crash behavior. The results can be visualized in a post-processing tool. Conspicuous components, for example, are highlighted in color so that the engineer's focus is placed on the relevant points in the vehicle.

4 Experiments

4.1 Description of the data from a longitudinal beam

To evaluate the methodology, data from a robustness study with 51 simulations were used, see [5] & [23]. A front car section model collides with a deformable barrier at 64 km/h in the Offset Deformable Barrier (ODB) load case [1]. The individual simulations differ in the scattered wall thicknesses of 36 components. This results in instabilities and highly scattered crash behavior, especially in the lower load plane. In this study, the axially loaded longitudinal beam, colored in red in Figure 1 is examined as an example. The plastic strains of the finite elements are used as the data. Since the finite element meshes of the individual simulations do not differ, the information of the 2286 finite elements can be used without a preprocessing step. Furthermore, it is not necessary to correct any time shifts between the simulations, since the distance between the vehicle and the barrier stays constant.

In order to validate the outlier detection results, each simulation for this study was classified manually. For this purpose, a Principle Component Analysis (PCA) [24] was first used to reduce the dimension of the data in order to obtain a quick overview of the different deformation behavior of the individual simulations. Analogous to [5], the temporal behavior as well as the spatial information is reduced in dimension, so that one point in the data matrix corresponds to one simulation. Figure 5 shows the results of the dimension reduction in 2D. This kind of data exploration is suitable to get a quick overview of the deformation behavior of the 51 simulations [7], [8], [9].

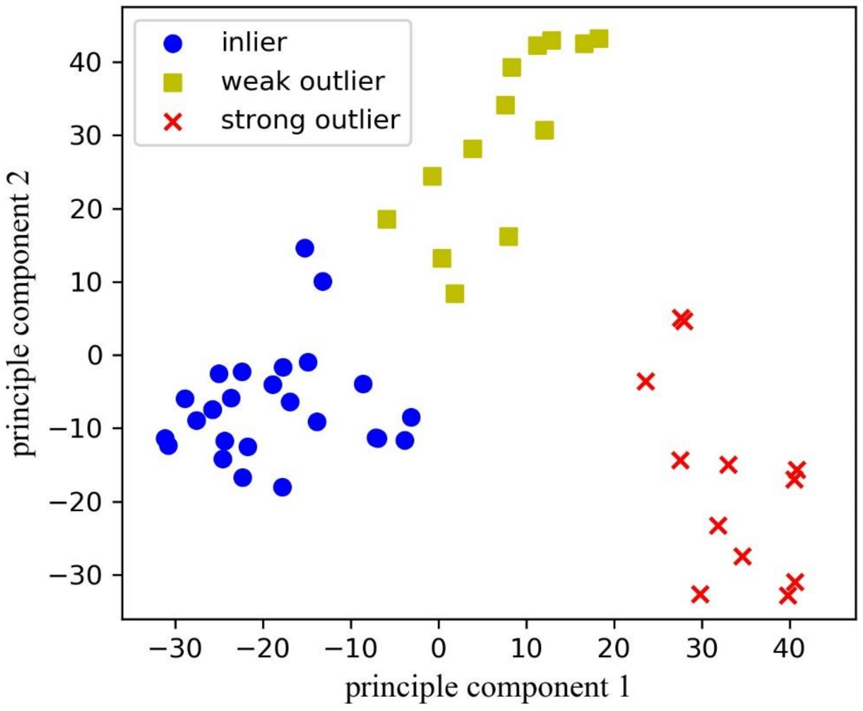

Figure 5: The results of PCA show three different deformation behaviors. Simulations in cluster 2 show folding from the front, in cluster 1 show folding from the back and in cluster 0 show a mixed behavior.

On closer inspection, three behavioral modes can be identified. In most of the simulations (26), the initial deformation starts at the front and propagates to the rear. In 12 simulations, the initial deformation starts at the rear of the component and propagates forward over time. This behavior is undesirable because it introduces instabilities into the load path and is therefore defined as a strong outlier. In the remaining 13 simulations, a mixed behavior occurs in which the component starts to deform from the front and the back simultaneously. These simulations will be referred to as weak outliers in the following. At selected time steps, the deformation behavior of these three categories is visualized in Figure 6.

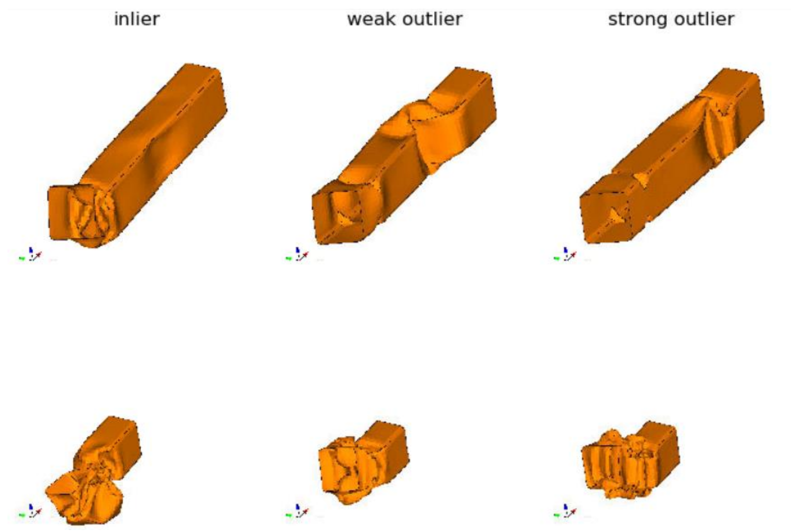

Figure 6: The Screenshots of simulations of the three different clusters are exemplarily shown. First row corresponds to state 24, second row to state 32.

In chapter 3.2 it was described that the outlier detection is automatically executed as soon as a new simulation is calculated. The database is extended step by step with new simulations, so that the context in which a new simulation is evaluated changes continuously. For the analyses in this chapter, artificially generated datasets with different amounts of outliers are used to evaluate the presented method on different difficult examples (see Table 1). The datasets correspond to artificial configurations (total number of simulations and amount of outliers), which can occur in the database during the development process. Although this procedure does not reproduce the process presented, it does allow a clear and comprehensive analysis of the outlier detection. The general validity and transferability of the results to the process explained in chapter 3.2 are not limited by this, rather the difficulty to detect outliers is increased, which makes the detection of suspicious crash behavior even more challenging in the analysis part.
Anomalies should be detected especially if they occur rarely. Therefore, dataset 1 consists of all existing inlier simulations and one strong outlier simulation. Even if the proportion of outliers increases, the algorithm should detect them reliably. Accordingly, datasets 2 and 3 consist of 26 inliers and five and 12 strong outliers, respectively. The distinction between inliers and strong outliers is also easy for humans in these datasets. It is more difficult when the weak outliers are additionally considered. Therefore, the fourth dataset was composed of all 26 inliers, all weak outliers and 5 strong outliers. Thus, this dataset poses the biggest challenge for the presented method. Since the engineer should not be disturbed by false warnings, the fifth dataset consists only of all 26 inlier simulations and does not contain any outliers. The method should be able to classify all existing simulations in this dataset as inliers.

Table 1: The 5 different datasets used for the analysis in this study are described. The amount of inliers and outliers is shown together with the computation time needed for the KNND Outlier detection.

	Inliers	**Weak Outliers**	**Strong Outliers**	**Contamination [%]**
Dataset 1 (DS1)	26	0	1	3.70
Dataset 2 (DS2)	26	0	5	16.13
Dataset 3 (DS3)	26	0	12	31.58
Dataset 4 (DS4)	26	13	5	40.91
Dataset 5 (DS5)	26	0	0	0.00

4.2 Calculation of the outlier scores

Before we examine the influence of the hyperparameters and the contamination, this chapter gives an overview of the results of the KNND procedure and the resulting outlier score, before and after standardization. For this purpose, the second data set with k=10 is used as an example. We perform the outlier detection for each state. For this the package pyod [25] with the option "method=largest" is used. This option specifies that the distance to the kth neighbor is used as the outlier score. The results are shown on the left of Figure 7.

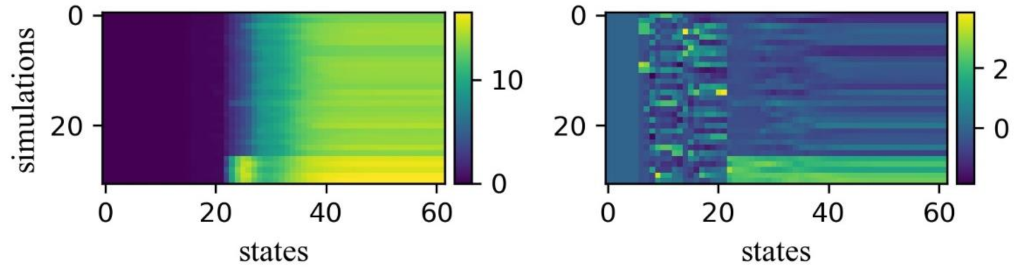

Figure 7: Visualization of the outlier scores. Each row represents a simulation. Each column corresponds to the temporal information from the states. Left: Values from KNND. Right: Column-wise standardized values from KNND.

Each row represents a simulation. The first 26 rows correspond to the 26 inlier simulations, the last 5 rows to the 5 strong outliers. The time is represented by each column. It can be seen that up to state 22, no deviations are observed in the data. Between state 22 and 28, the last 5 simulations stand out with higher scores. Here, the different crash behavior in the different simulations is clearly visible. Also, for the later states the last 5 simulations stand out from the other 26. It can be seen for each simulation that the scores of the simulations tend to increase with advancing time. This is related to the KNND algorithm, where the scores are directly proportional to the distance to the kth neighbor. The larger the deformation at the later states, the larger the distances between the simulations due to the different deformation behavior. For the early states, all simulations seem to have the same distances. This representation of the results leads to incorrect interpretations of the available data. Due to the time dependence of the plastic strains and thus of the scores, the individual states are not directly comparable with each other. Therefore, the scores should be standardized as shown in Figure 7 on the right. The plot shows that differences between the simulations already appear from the 6th state on. Only the first 6 states seem to be identical. If the simulation results are inspected, it turns out that in the first 6 states no deformation occurs in the component and therefore the simulations cannot differ yet.

In states 20 and 21, the 15th simulation clearly stands out. At this time of the crash, the longitudinal member is hit laterally by the radiator. To investigate why the 15th simulation is identified as an outlier, we compare it in detail with the 16th and 17th simulations, these simulations have low outlier score. The plastic strain results for state 21 are shown in Figure 8 for these three simulations. Note that the lateral deformation of the component caused by the radiator can be seen in all three simulations. The 15th simulation, however, shows an additional deformation in the upper area, which cannot be observed in the other two simulations. This is caused by the collision with another component at that time. For the other simulations the collision with that component occurs with a slight delay at state 22. That is the reason why the anomaly score of the 15th simulation reduces for that state, because all simulations again show the same crash behavior in that location. Furthermore, from state 22 on, the last five simulations are clearly different from the other simulations.

This example shows that even small differences in the deformation behavior can be detected by the presented method. The plot in Figure 8 is suitable to determine in which time range the anomalous behavior occurs. Thus, the presented method shows not only if and in which component, but also when an outlier behavior occurs in a simulation.

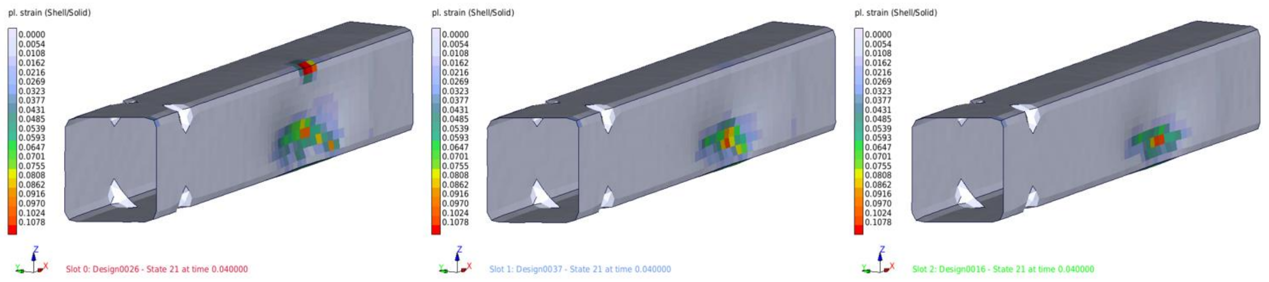

Figure 8: Longitudinal Member in state 21 with the strains variable mapped on the surface. Left: simulation 15, Middle: simulation 16, Right: simulation 17

If the engineer is interested in one outlier score summarizing the whole temporal behavior, the results of the individual states can be averaged to a single score after the standardization step explained in section 3.2. As a result, one value is obtained for each simulation, as shown in Figure 9. The 31 simulations considered are shown on the horizontal axis, while the cumulative outlier score is given on the vertical axis. In the next chapter the influence of the hyperparameters and the different datasets on the outlier scores will be considered.

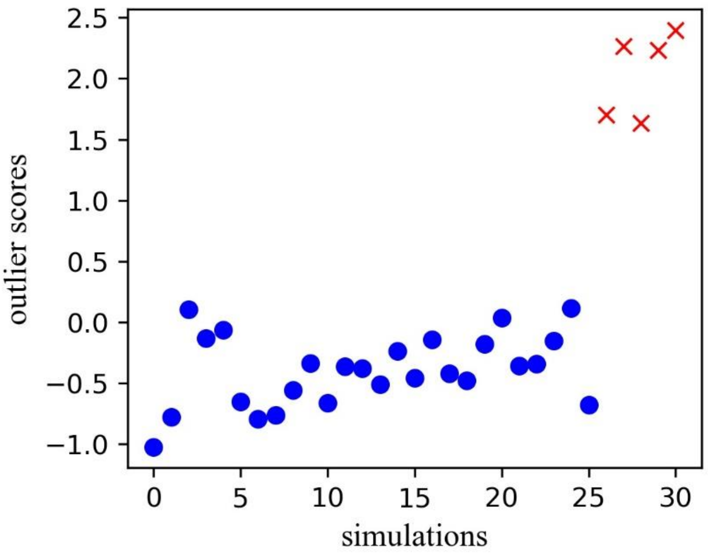

Figure 9: The Outlier scores of dataset 2 are displayed as blue points. For every simulation (horizontal axis), one score is shown on the vertical axis. The values are calculated as the mean of the time dependent outlier scores from Figure 7. The blue points correspond to inlier simulations, whereas the red crosses represent the outliers.

4.3 Hyperparameter study based on the 5 datasets

As described in chapter 3, the accuracy of the KNND algorithm in detecting anomalies depends on the choice of the hyperparameter k. Depending on how many outliers are present in the dataset, the hyperparameter k must also be chosen accordingly. Figure 10 shows the calculated scores for different values of k (2, 5, 10, 20) and the 5 datasets presented. The inliers and the outlier simulations to be detected are marked accordingly. Inliers are represented by blue dots, strong outliers by red crosses and weak outliers by yellow squares. For high accuracy in outlier detection, inliers should receive a low score and outliers a high one.
In the first data set, regardless of k, the outlier simulation clearly stands out with a significantly higher outlier score compared to the inlier simulations. In the fifth data set, in which no outliers are present, no simulation with an increased outlier score occurs, as desired.
In the data sets DS2 to DS4, for too small values of k, the outlier simulations cannot be clearly distinguished from the inliers. Especially in the third data set, the conspicuous simulations for k=2 and k=5 cannot be clearly identified. The scores of the outliers hardly stand out from those of the inliers. This is due to the fact that three clusters are formed for the three different deformation behaviors as shown in Figure 5. For outliers to be reliably detected by the KNND algorithm, k should be larger than the cluster size of the outliers. Accordingly, it can be seen from the figure that with increasing values of k, significantly better results can be obtained.

Since there is generally no prior knowledge about the data, an ensemble model is trained as described in Chapter 3 by averaging the results of the individual models with different k. The result from this approach is shown in the last row in Figure 10. Individual hyperparameters that give very poor results (e.g., k=2) are balanced by models with good hyperparameters (e.g., 20). The results show that the outliers in datasets 1-3 clearly stand out from the inliers. In the fourth dataset, it is difficult to distinguish all weak outliers from inliers. The fifth dataset doesn´t contain any outliers and no simulations stand out from the plot due to an extreme value.
Humans can make these distinctions from the plot in Figure 10. However, since the detection of outliers should be done without an intervention, this step must be automated. Statistical methods can be used to calculate a threshold value that distinguishes the outliers from the inliers. For the following investigations, the models for k=10, 15, 20, 25 are exemplarily combined and only their mean value is examined.

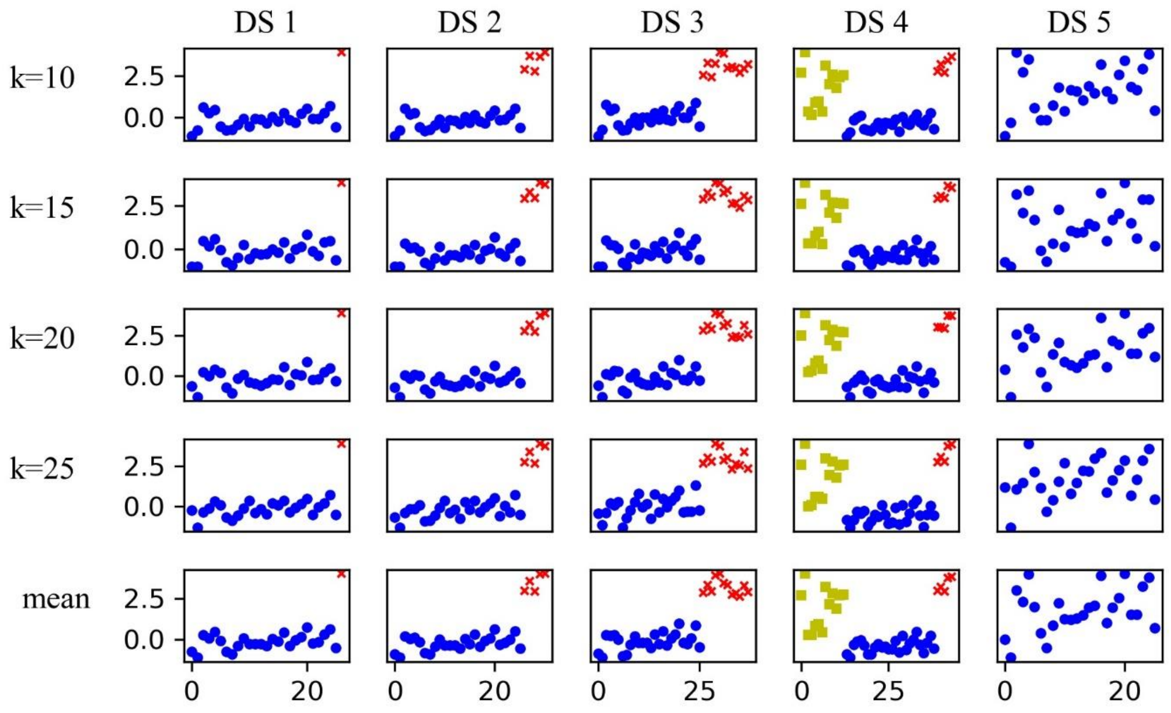

Figure 10: Influence of the parameter K and its ensemble average for the different datasets and different hyperparameters k of the KNND approach. On the horizontal axis, the single simulations of the respective dataset are displayed. The outlier scores are shown on the vertical axis. Blue points correspond to inlier simulations, red crosses to strong outliers and yellow squares to weak outliers.

4.4 Definition of the threshold

Up to now, outlier scores for each component of each simulation are available as a result of the method. In order to warn the engineer automatically in case of conspicuous crash behavior, it is necessary to calculate a threshold value and, based on this, to identify the conspicuous components automatically. As described in Section 2, this paper compares three well-known statistical methods for threshold definition. If an outlier score is above the threshold, it is an outlier. All simulations with smaller scores are classified as inliers.
All three methods require the definition of parameters that have a significant effect on the results. Their choice is application specific and strongly depends on the distribution and the outlier fraction in the data. Parameter values are used, which are common in the literature. For the parameter in the IQR method, the value 1.5 is chosen. For the method based on the standard deviation, the values 3 and 1.68 are used for the parameter n, within which 99.7 and 90% of the data of a normal distribution lie, respectively. Accordingly, these two variants are abbreviated STD99 and STD90 in the following. For the calculation of the MAD-based threshold, b is chosen to be 1.48 and a is chosen to be 2.5, as is common in the literature. This variant is abbreviated as MAD25.
The goal of calculating the threshold is to identify all outliers (high recall) and at the same time not to misclassify any inliers as outliers (high precision). Figure 11 shows the already known outlier scores of the five different datasets for the averaged KNND algorithm.

Additionally, the four different threshold values are shown in this figure, which are used for the classification into inlier and outlier. Furthermore, Table 2 shows the classification accuracies by the values Precision and Recall for the different datasets.

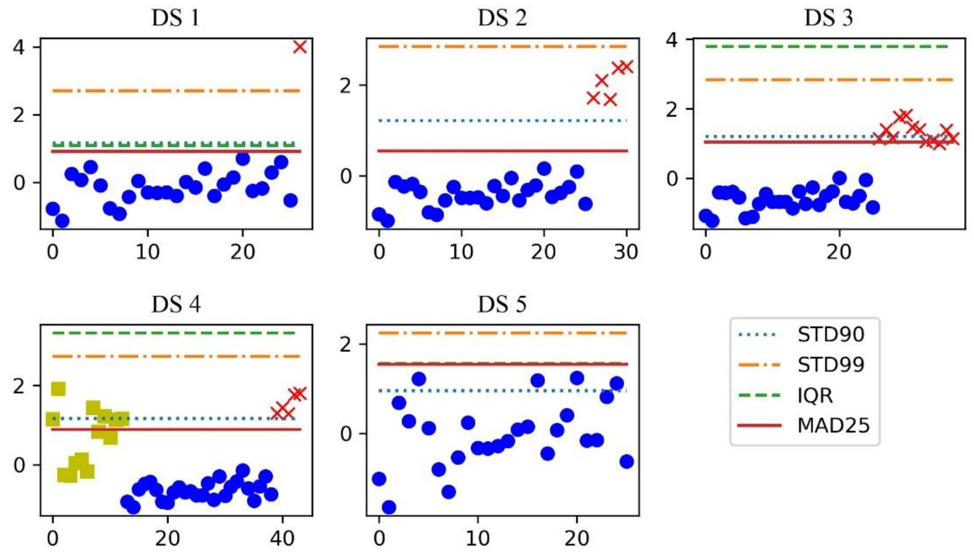

Figure 11: The outlier scores of the 5 different datasets for the respective simulations are shown. The horizontal lines correspond to the calculated thresholds for the standard deviation based methods STD99 and STD90, the IQR method and the MAD25 method.

In the first dataset, all thresholds correctly separate the outlier simulation from the rest. At the same time, there are no misclassified inliers. Accordingly, Precision and Recall are both 1 for all four methods. Also, for the second dataset, except for the standard deviation based method STD99, all others are able to detect the outliers. These two datasets are easy cases for outlier detection, since the behavior of the outlier simulations differs strongly from the inliers and anomalies occur very rarely (according to Table 1: 3.70% and 16.13%). Since the contamination in this dataset is below 25%, the thresholds of IQR and MAD25 are not influenced by outliers and therefore deliver a recall of 1. At the same time, however, it is also evident that the 3-fold standard deviation, within which 99.7% of the data of a normal distribution lie, defines the threshold too strong for the second dataset. This procedure is strongly dependent on the presence of anomalies (see section 2.2), since mean and standard deviation of the data must be determined, which are themselves influenced by anomalies. Accordingly, the recall for STD99 is 0.

Dataset three stands out from the previous two datasets due to an increased proportion of outliers (Table 1: 31.58 %). Accordingly, the 3-fold standard deviation STD99 fails to detect the anomalies here as well. The situation is similar for the IQR method. This method is also sensitive to an increasing proportion of outliers, especially from a proportion greater than 25%, as this affects the location of the third quartile used to determine the IQR. The recall for both methods is 0. The STD90 threshold still detects half of the existing outliers. Accordingly, the recall is 0.5. The MAD25 method performs best in comparison and detects 11 of 12 outlier simulations (recall = 0.92). This is because the contamination is less than 50% and therefore the calculation of the threshold is not affected by outliers. Still, one outlier is not detected by MAD25. This might be due to the fact, that the hyperparameter a=2.5, which was chosen from literature, is not optimal. A lower value reduces the threshold so that all outliers are identified.

Table 2: Representation of Precision and Recall for the different datasets and methods for threshold calculation.

	Precision				**Recall**			
	STD90	STD99	IQR	MAD25	STD90	STD99	IQR	MAD25
Dataset 1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Dataset 2	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0
Dataset 3	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.92
Dataset 4	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.61
Dataset 5	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Mean	0.8	1.0	1.0	1.0	0.8	0.4	0.6	0.91

The most difficult dataset is the fourth one. On the one hand, this has the largest contamination (40.91%) and on the other hand, it includes the weak outliers, which are challenging to detect even for humans. Accordingly, the expectation is that no ideal results for recall can be obtained. As with the previous dataset, the STD99 and IQR methods fail to detect the outliers (recall=0). Again, only STD90 and MAD25 are able to identify some outlier simulations. STD90 detects 9 out of 18 outliers, while MAD25 finds 11 out of 18 outliers. Accordingly, the recall is 0.5 and 0.61, respectively.
The fact that only just over half of the existing outliers are identified is not due to the methods used to calculate the threshold. The outlier scores from the KNND algorithm do not even allow the engineer to reliably separate the first 13 weak outliers from the following 26 inliers (see Figure 11). This is on the one hand due to the high proportion of outliers and on the other hand because the weak outliers correspond to a mixed behavior from the inlier and outlier simulations, making it even more difficult for the algorithm to distinguish between them.

In the first four datasets, no inliers have been misclassified so far, so the precision for all methods is 1 as shown in Table 2. In the last dataset, no outlier simulations are included at all. Accordingly, a good algorithm and threshold should not misclassify any simulations as outliers. While STD99 and IQR failed in most datasets, they provide good results here and do not detect any outliers. The MAD25 method also does not yield any misclassifications. Only STD90 falsely detects 4 outliers. Thus, the precision at this threshold is 0. For STD90, the hyperparameter n is smaller than for STD99, resulting in a lower threshold. For the outlier scores in the fifth data set, this threshold is too low, so inliers are falsely detected as outliers. At the same time, however, it is possible to identify more outliers in datasets 2-4 compared to STD99. This shows that the threshold calculation of the STD method is not only influenced by existing outliers, but the results are also sensitive to the hyperparameter n. Furthermore, the optimization of n faces a trade-off between high recall and high precision.

In summary, MAD25 is the most robust method in comparison to the other ones in threshold calculation. In the standard deviation based procedure, the two parameters of the dataset mean and standard deviation are themselves strongly influenced by existing outliers. The same is true for the IQR method, whose third quartile is also influenced by outliers. This means, that in cases where more than 25% of the data consist of outliers, the IQR threshold is not reliable. In contrast, the median of the dataset is not influenced by existing outliers and thus represents the most robust method for threshold determination, especially in highly contaminated datasets. Across the different datasets, according to Table 2, the mean recall is 0.91 and the Precision is 1. This means that 91% of all outliers were found and no inliers were incorrectly identified as anomalies.

5 Conclusion

In this work, a method to detect outliers in crash simulations has been presented. The approach and application in crash simulations is novel and has successfully identified outlier crash behaviors in different datasets.
Since in the standard evaluation process different simulations within the same load case category are compared, it is possible to analyze each state individually with the other simulations. The KNND algorithm was used to calculate the state-based outlier scores. Subsequently, the results were standardized and the mean was calculated to obtain a single value for the entire simulation. The work showed the dependence of the hyperparameter k in the KNND algorithm against different datasets with different proportions of outliers. While some values performed very well, others did not perform at all for outlier detection. Since, in general, no prior knowledge of the data can be assumed, that could be used to estimate a good value for k, several models were trained with appropriate values for k and finally the average was taken over the different models. This increases the quality and the robustness of the outlier detection.
Following the calculation of the outlier scores, different methods for thresholding were compared. With the help of this value, an automated identification of anomalies is possible. The MAD25 method achieved the best results. 91% of all outliers were detected on average in the 5 different datasets. With increasing contamination in the datasets, the recall is getting reduced further. At the same time, there were no simulations incorrectly identified as outliers (Precision=1). This is especially important in whole vehicle analysis, since with hundreds of components, any misclassification distracts the engineer, costs him time and makes him lose his confidence in the system.

The method presented for anomaly detection makes it possible to automatically examine all components and evaluation variables available from the simulation for conspicuous crash behavior. On the one hand, this increases the degree of automation of the evaluation. On the other hand, the completeness of the data analysis is increased, since more information can be evaluated. The results can be clearly displayed in post-processing software. Conspicuous crash behavior of individual components is highlighted in color accordingly, so that the engineer's focus is placed on the relevant points in the vehicle. This also increases the robustness of the evaluation.

It remains to be said that the hyperparameter a for the MAD method must also be adapted to the underlying task. Experience in this regard must be gathered in a broad field test across different components and load cases. In addition, further algorithms for calculating the outlier score must be compared with the results of the KNND method.
One disadvantage of the method is that the outlier scores are not normalized (e.g., in a fixed interval between 0 and 1). A continuous value would facilitate the interpretation of the results and increase the level of detail, since the user can estimate how anomalous the crash behavior is. In the future, the procedure will be extended regarding this issue.

6 Geolocation

Dr. Ing. h.c. F. Porsche AG
Porschestraße 911
71287 Weissach
Germany

7 References

[1] „euroncap," [Online]. Available: http ://www.euroncap.com/. [Zugriff am 13 02 2020].

[2] J. A. Ambrósio, M. Pereira und F. da Silva, Crashworthiness of transportation systems: structural impact and occupant protection (Vol. 332), Springer Science & Business Media., 2012.

[3] C. Diez, L. Harzheim und A. Schumacher, „Effiziente Wissensgenerierung zur Robustheitsuntersuchung von Fahrzeugstrukturen mittels Modellreduktion und Ähnlichkeitsanalyse," VDI-Berichte, 2279, 2016.

[4] T. Spruegel, T. Schröppel und S. Wartzack, „Generic approach to plausibility checks for structural mechanics with deep learning," in *Proceedings of the 21st International Conference on Engineering Design (ICED 17)*, Vol 1: Resource Sensitive Design, Design Research Applications and Case Studies, Vancouver, Canada, 21-25.08. 2017 (pp. 299-308).

[5] D. Kracker, J. Garcke, A. Schumacher und P. Schwanitz, „Automatic Analysis of Crash Simulations with Dimensionality Reduction Algorithms such as PCA and t-SNE".

[6] J. Garcke , M. Pathare und N. Prabhakaran, „Scientific Computing and Algorithms in Industrial Simulations: Projects and Products of Fraunhofer SCAI," in *ModelCompare*, Springer International Publishing, 2017, pp. 199-205.

[7] J. Garcke und R. Iza-Teran, „Machine learning approaches for repositories of numerical simulation results," in *10th European LS-DYNA Conference (Vol. 2015)*, 2015, June.

[8] B. Bohn, J. Garcke, R. Iza-Teran, B. Paptrotny, U. Pehersdorfer, U. Schepsmeier und C.-A. Thole, „Analysis of Car Crash Simulation Data with Nonlinear Machine Learning Methods.," in *Proceedings of the ICCS*, Barcelona, 2013.

[9] R. Iza-Teran, „Enabling the analysis of finite element simulation bundles.," *International Journal for Uncertainty Quantification*, 2014.

[10] D. Hawkins, Identification of Outliers, Chapman and Hall, 1980.

[11] R. Iza-Teran und J. Garcke, „A geometrical method for low-dimensional representations of simulations," *SIAM/ASA Journal on Uncertainty Quantification*, Bd. 7(2), pp. 472-496, 2019.

[12] C. C. Aggarwal, „An introduction to outlier analysis. In Outlier analysis," Springer, Cham., 2017, pp. (pp. 1-34).

[13] M. Gaur, S. Makonin, I. Bajic und A. Majumdar, „Performance evaluation of techniques for identifying abnormal energy consumption in buildings," *IEEE Access 7*, pp. 62721-62733, 2019.

[14] D. Huang, D. Mu, L. Yang und X. Cai, „CoDetect: financial fraud detection with anomaly feature detection," *IEEE Access 6*, pp. 19161-19174, 2018.

[15] S. Ramaswamy, R. Rastogi und K. Shim, „Efficient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data (pp. 427-438)*, 2000, May.

[16] E. Kreyszig, in *Advanced Engineering Mathematics*, 4th Edition ISBN 0-471-02140-7, Wiley, 1979, pp. p. 880, eq. 5.

[17] C. Leys, C. Ley, O. Klein, P. Bernard und L. Licata, „Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median.," *Journal of experimental social psychology*, Bd. 49.4, pp. 764-766, 2013.

[18] J. W. Tukey, „Exploratory Data Analysis," Addison-Wesley, Reading, MA, 1977.

[19] R. McGill, J. W. Tukey und W. A. Larsen, „Variations of box plots," *The American Statistician*, Nr. 12-16, p. 32(1), 1978.

[20] A. Leroy und P. Rousseeuw, Robust regression and outlier detection, New York: Wiley Series in Probability and Mathematical Statistics, 1987.

[21] P. J. Huber, Robust statistical procedures, New York: John Willey, 1989.

[22] J. Miller, „Reaction time analysis with outlier exclusion: Bias varies with sample size," *The quarterly journal of experimental psychology*, pp. 43(4), 907-912, 1991.

[23] N. Andricevic, *Robustheitsbewertung crashbelasteter Fahrzeugstrukturen (Doctoral dissertation, Universität)*, 2016.

[24] S. Wold, K. Esbensen und P. Geladi, „Principal component analysis," *Chemometrics and intelligent laboratory systems*, Bde. %1 von %22(1-3), pp. 37-52, 1987.

[25] Y. Zhao, Z. Nasrullah und Z. Li, „Pyod: A python toolbox for scalable outlier detection," *arXiv preprint arXiv:1901.01588*, 2019.

Aus: Submit to International Journal of Crashworthiness (tandfonline.com)

“Biographical note. Please supply a short biographical note for each author. This could be adapted from your departmental website or academic networking profile and should be relatively brief (e.g. no more than 200 words).”

David Kracker studied physics and electromobility at the University of Stuttgart. In his master thesis in cooperation with Robert Bosch GmbH, he specialized in the powertrain simulation of a hybrid electric vehicle. He is currently working on his doctorate at the University of Wuppertal in cooperation with Dr. Ing. h.c. F. Porsche AG on the topic of how crash simulations can be fully evaluated automatically using artificial intelligence in order to reduce the analysis effort of engineers, development time and costs. The focus of his work here is on methods for dimension reduction, outlier detection and automatic classification of crash behavior.

Professor Dr.-Ing. Axel Schumacher studied mechanical engineering at the University of Duisburg and RWTH Aachen. He obtained his doctorate in the field of topology optimization at the University of Siegen. Subsequent research projects together with Airbus focused on the optimization of aircraft structures. As a project manager at Adam Opel AG, he introduced mathematical optimization at numerous points in the vehicle development process. At the Hamburg University of Applied Sciences, he started his research on the integration of highly non-linear analysis in the optimization of lightweight structures. Currently, he holds the chair for Optimization of Mechanical Structures at the University of Wuppertal. At the chair, intensive research is being conducted on new structural optimization methods.

Aus: https://rd.springer.com/book/10.1007/978-3-662-60328-4#authorsandaffiliationsbook

Jochen Garcke received the Diploma degree in mathematics and the Ph.D. degree in mathematics from the Universität Bonn, in 1999 and 2004, respectively.,He was a Postdoctoral Fellow at the Australian National University, from 2004 to 2006 and a Postdoctoral Researcher, from 2006 to 2008 and a Junior Research Group Leader, from 2008 to 2011, at Technische Universität Berlin. Since 2011, he has been Professor of numerics with the Universität Bonn and a Department Head with Fraunhofer SCAI, Sankt Augustin. His research interests include machine learning, scientific computing, reinforcement learning and high-dimensional approximation.,Prof. Garcke is also a member of DMV, GAMM and SIAM. He is also a Reviewer of the IEEE Transactions on Industrial Informatics, the IEEE Transactions on Neural Networks and the IEEE Transactions on Pattern Analysis and Machine Intelligence.(Based on document published on 24 February 2020).

Aus: https://ieeexplore.ieee.org/author/37085372841