MACHINE LEARNING PROCESS TO ANALYZE BIG-DATA FROM CRASH SIMULATIONS

[1]Constantin Diez*
[1]Adam Opel AG, Germany

KEYWORDS –
Crash Simulation, Machine-Learning, Big-Data, Inference

ABSTRACT –
Machine Learning has shown to be a key technology in many disciplines like object recognition or natural language processing. In this contribution we will give an introduction to machine learning in the field of Crash Simulation and CAE. The scope will cover the topics of automated simulation results comparison by means of dimensionality reduction, as well as inference of cause and effect with decision tree learning as key technology. The paper will explain the data analytics process flow for selected examples, showing how to analyze and understand the behavior of an ensemble of 1000 simulation results quickly.

TECHNICAL PAPER -

1. INTRODUCTION

Understanding large amounts of simulation data requires a lot time and human effort. These large amounts of data most commonly result from investigations, such as Robustness Analysis or Design Space Exploration, which shall reveal the behaviour of a structure under uncertainty. In this contribution we will focus rather on design space exploration than robustness analysis, since this leaves us more freedom to choose a design later on. It is unfortunately humanly impracticable to analyse many simulations by hand. This pushes the desire for a new automated process in order to harvest the crucial information hidden in this data efficiently.

Machine Learning Methods can be seen as advanced statistical methods, enabling the analysis of large amounts of data. The algorithms try to find statistical frequent patterns within the data and provide methods to investigate or relate these patterns. Applying Machine Learning in a new discipline like CAE is unfortunately an out-of-the-box process. As a consequence it will not work out well to simply take the raw data and apply existing methods on it. For this reason we want to propose a process chain for the analysis of crash simulation results, utilizing two key technologies: Dimensionality Reduction and Rule Mining. The field of Dimensionality Reduction contains many algorithms in order to find statistical patterns, so that one may find the two or three major buckling modes of a component when comparing a large amount of simulations. Rule Mining is an inference technique to help relate these statistical patterns into cause and effect. A typical application would be to find out which designs caused specific deformation modes. This knowledge can help engineers to fix undesired behaviour of the structure quickly.

2. EXAMPLE

The example data used to describe the procedure shall be introduced here briefly. We used the bumper of a 2007 Chevrolet Silverado as example case here. The crash model itself originates from the National Highway Traffic Safety Administration (NHTSA) website [7] and covers a simplified version of a full frontal vehicle crash. We parameterized the model with 27 physical parameters, which are entirely sheet thicknesses except for the impact angle and velocity. All samples were realized within ANSA [1] by using the python scripting API. The model is shown in figure 1 (rendered in META 17.1).

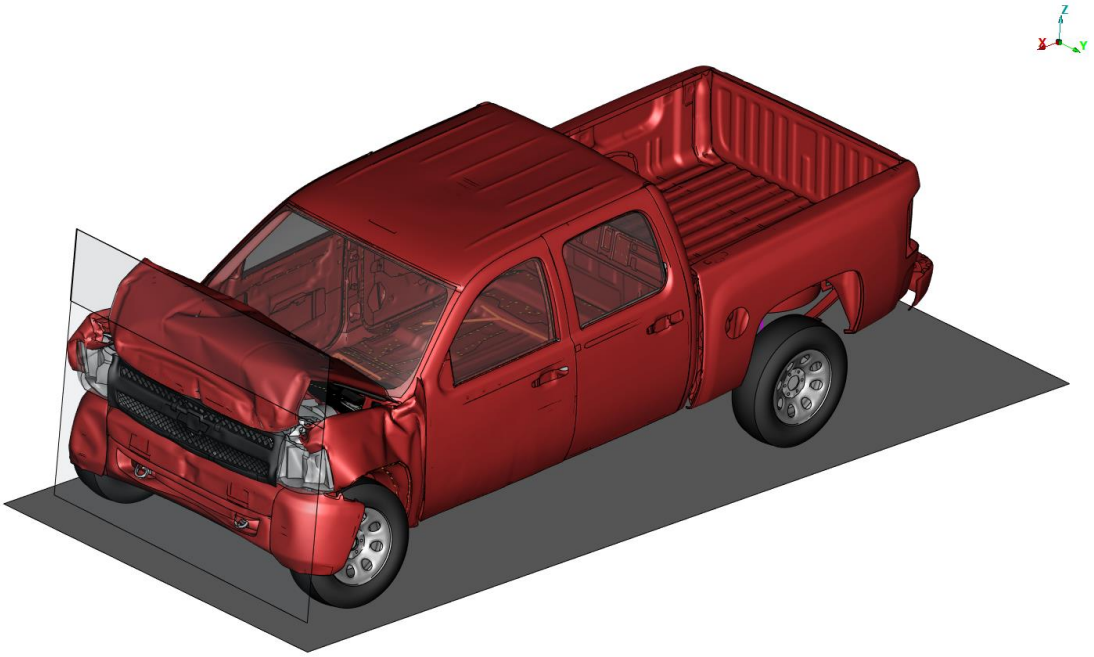

Figure 1 – In this contribution, a full frontal crash simulation of a 2007 Chevrolet Silverado will be used as example case. We ran 1000 simulations with variations in design and impact condition.

3. DIMENSIONALITY REDUCTION

The first core challenge is how to automatically compare hundreds of simulations. This challenge can be addressed with Dimensionality Reduction, which is a field of work for efficient reduction of data. The method will be used in two ways:

1. data visualization of all simulations at once
2. automatic distance computation between simulation pairs

Our geometry-based dimensionality reduction [2] converts the FEM results of a component into a distribution function of plastic strain. This is done in two steps. First a parametric Bézier regression is done through selected undeformed components. Figure 2 shows such a parametric regression through the bumper data.

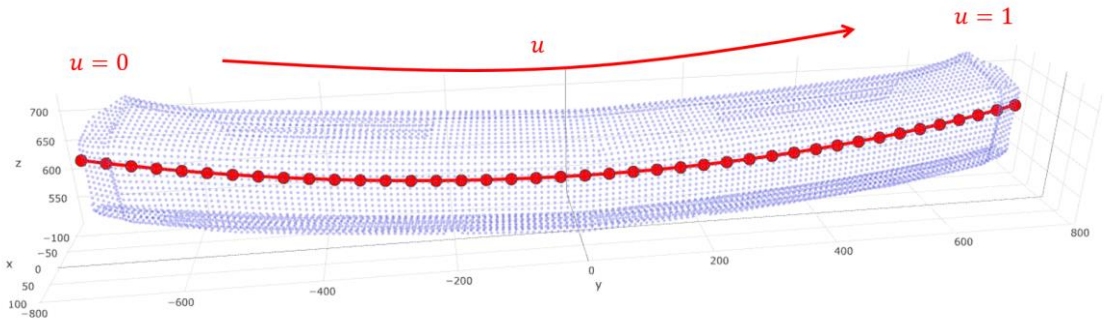

Figure 2 – The first step of dimensionality reduction is a geometric simplification of a component into a line. The line has one coordinate u, ranging from 0 to 1.

In a second step the element results are projected onto the regression and smoothed at the same time in order to compensate for different meshing. We chose the efficient plastic strain for describing large deformations well and corresponds to the energy absorption. The smoothing is done with a Gaussian distance kernel as weight [2].

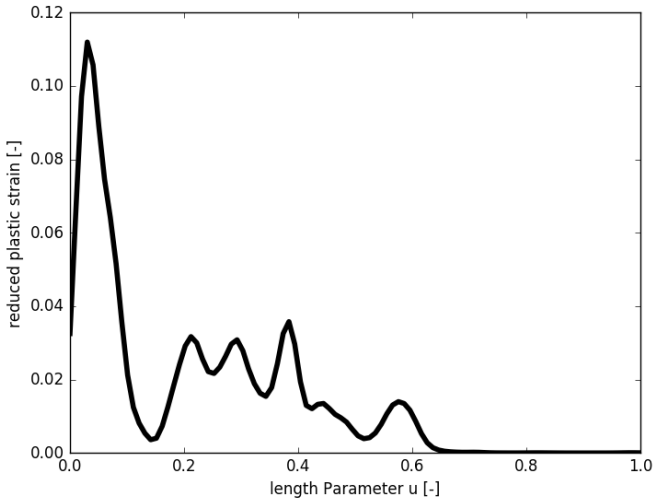

Figure 3 – After projecting and smoothing the elements plastic strain onto the simplified geometry, one gets a function of reduced plastic strain along the coordinate of the regression.

The key aspect of this dimensionality reduction is that every deformation mode has a typical distribution function of plastic strain. The distribution function of plastic strain is very cheap to save and also for all following computations the FEM-Data will not be needed anymore. The task to compare simulations has been whittled down to the task of function comparison. All distribution functions f_i can be easily compared with each other by a simple inner product in order to find out whether they are similar or different. Since the inner product is just an integral, the computation is very fast compared to full FEM-Mesh comparison

$$s_{ij} = \frac{\langle f_i | f_j \rangle}{\sqrt{\langle f_i | f_i \rangle \cdot \langle f_j | f_j \rangle}} \in [0,1], \tag{1}$$

$$\langle f_i(u) | f_j(u) \rangle = \int_{u=0}^{1} f_i(u) \cdot f_j(u) \cdot du \, . \tag{2}$$

The parameter *u* in equation 2 denotes the parametric dimension of the regression. The result of the comparison is a scalar normalized value between 0 and 1, where 0 means the two functions are totally different, whereas 1 means that the functions are identical. Comparing every simulation with each other results in a square and symmetric similarity matrix s_{ij}, which contains the similarities in between all simulations.

This dimensionality reduction has several properties which make it very practicable: First of all there is only one parameter for the entire process, which is the polynomial degree of the regression. Secondly every run can be processed entirely independent of the others, making the addition of a new run to the database very simple. Thirdly moderate topological and also mesh differences, such as missing ribs or additional holes, pose no problem anymore. It should also be mentioned that the intermediate layer (plastic distribution function) can be inspected and understood humanly, which is not common for this field of work. This white-box property makes this approach especially appealing.

4. VISUALIZATION AND CLUSTERING

The similarity matrix serves as key for two algorithms. First we want to visualize it by usage of Multidimensional Scaling [3] (MDS), which tries to reconstruct coordinates in a given space (here 3D) from distances (we use unsimilarities, which are just $d_{ij} = 1 - s_{ij}$). Doing so reveals the behaviour of all simulations relative to each other at once. An example is given in figure 4.

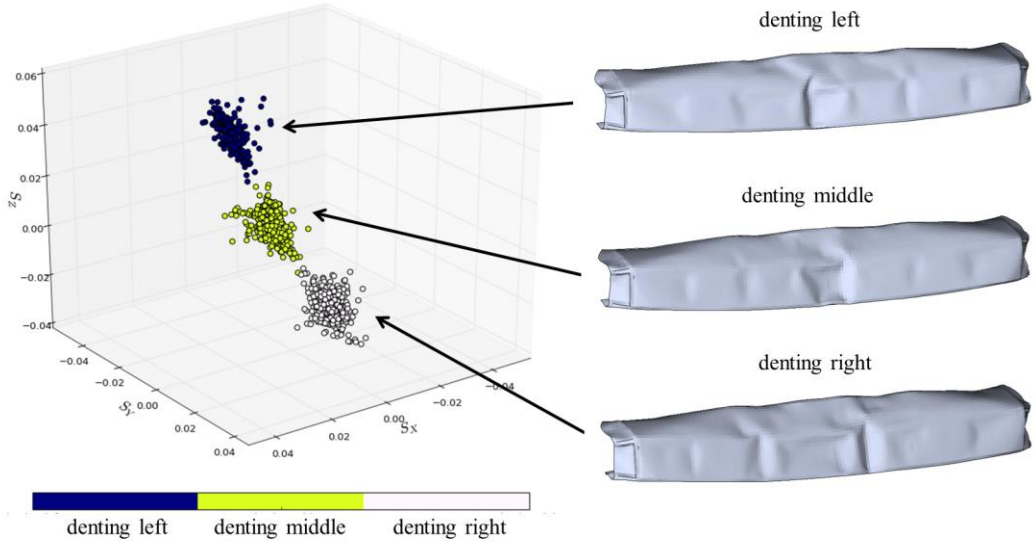

Figure 4 – Low-dimensional embedding of the bumper from 1000 crash simulations at the final timestep. The closer points are, the more similar their results have been. There are clearly three major deformation modes.

The number of visible clusters in such a plot indicates the number of major deformation modes in the dataset. Note though, that walking a path inside a clearly separable cluster will also reveal different states, but there is a transitional behaviour in between.
Determining clusters and sub-clusters can be done by hand in case of simple datasets. Since MDS is just a reconstruction and also more complicated datasets might pose to be much more difficult to separate, it is recommended to do the clustering on the distance matrix itself with a clustering algorithm. We found Agglomerative Clustering [4] to yield quite good results, if there are not too many outliers in the dataset. The runs in figure 4 are already coloured according to the clustering. The shown representative samples for every run can be visualized quickly in META [8], since our program is communicating with the Postprocessor via a network port. This technique has shown to save an enormous amount of time, since we can simply communicate the necessary scripting commands automatically. The output of this step is a clustering table, in which we save which run belongs to which simulation cluster.

5. INFERENCE

The technique from the previous section enables the detection of deformation patterns within a set of simulation. An engineer may wonder what might be causing these deformation modes. Therefore one has to associate simulation outputs and inputs, which can be done with inference. There are many inference techniques available, such as Support Vector Machines (SVM) or Artificial Neural Networks (ANN), but we found Decision Tree Learning (DT) [5] to be the best choice for the following reasons:

- Decision trees work with low sample counts quite well
- Decision trees are a white-box model, and thus easily inspectable
- For being a white-box model, they are also easy to communicate

Decision tree learning needs two inputs: the input variables of all simulation runs and a Boolean condition, which separates all runs into two groups. What the learning algorithm

does is to search in the input space for the reason which might be causing this split of the runs. This is done in such a way, that it tries to find pure design subspaces with the help of variable splits. Figure 5 illustrates a decision tree and its design space separation.

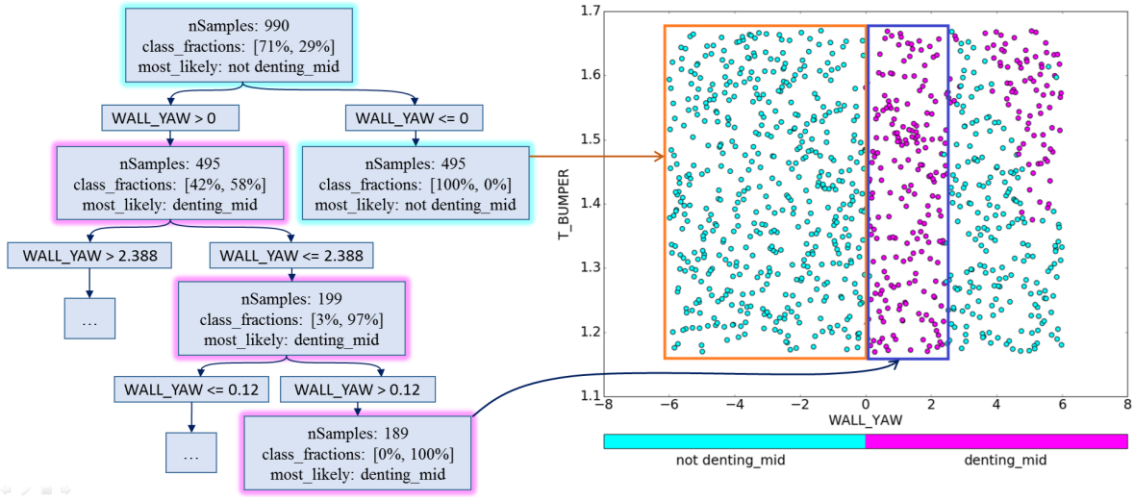

Figure 5 – Visualization of a trained decision tree. Every leaf in the tree corresponds to a design subspace, defined by the decisions leading to it.

Every leaf in a decision tree can be seen as a traditional IF-THEN rule, where the IF-part is describing the subspace with Boolean conditions (e.g. “IF 0.12 < WALL_YAW < 2.39”), whereas the THEN-part is just our deformation class name (e.g. “THEN denting_mid”). Selecting ‘good’ rules in terms of engineering is the major difficulty here and can be reformulated as a node selection problem within the decision tree. Our fully automatic rule selection is explained in detail in [6] and can be summarized as follows: As first step the algorithm ranks all rules within a decision tree in order to find the most meaningful and reliable design-subspaces. In a second step the algorithm determine how many significant rules actually exist within the dataset, so that an engineer is only confronted with the most reliable and important knowledge within the dataset. Figure 6 illustrates the resulting rules for the target “denting_mid”.

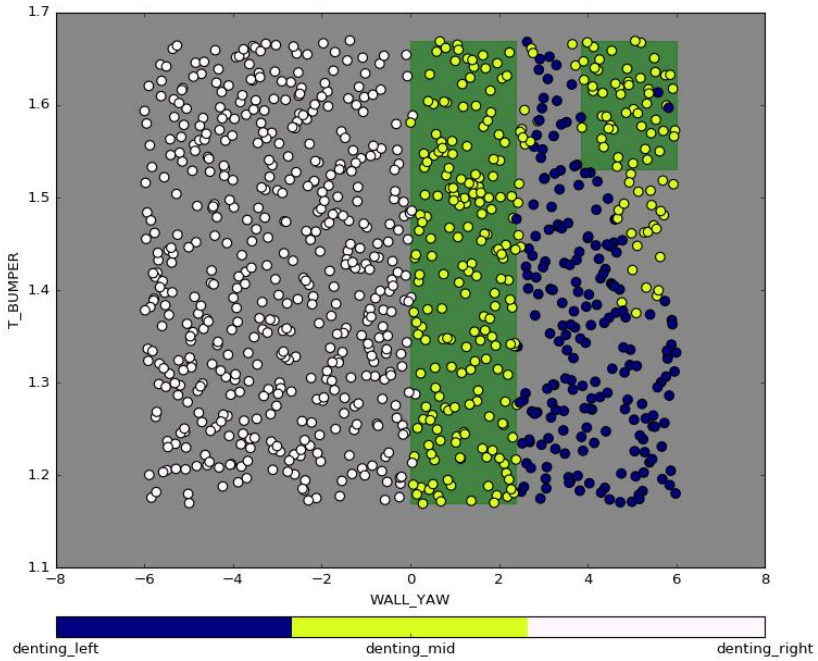

Figure 6 – The three deformation modes of the bumper in figure 4 are merely determined by the impact angle of the car (WALL_YAW) and the bumper sheet thickness (T_BUMPER). Our algorithm mines two significant rules (green area) for the target “denting_mid, isolating it well.

The two rules in figure 6 have the bounds:

1. 0 < WALL_YAW < 2.389
2. 3.86 < WALL_YAW and 1.53 < T_BUMPER

6. SUMMARY

In this article we gave a brief introduction into a machine learning process for analysis of crash simulation results. Dimensionality Reduction helps engineers to quickly identify all major deformation modes of a component in a large dataset with thousands of simulations. Our geometry-based dimensionality reduction technique has the advantage that all simulations may be processed entirely independent. Also a normalized similarity value between simulations results can be computed very easily. The following question, what actually causes certain deformation modes occurred is related to the task of finding large design-subspaces which trigger the specific deformation mode. We found Decision –Tree based Rule Mining to be an appropriate choice, since it not only matches the properties of data in the field of CAE, but also addresses todays needs of engineers.

7. OUTLOOK

A central desire for an engineer is to use as few samples as possible. In this contribution we simply chose a large amount of samples to gain certainty. A different approach with adaptive sampling might help to choose new samples in such a way, that the rules improve in terms of quality as fast as possible. This could drastically lower the amount of samples for simulation rule mining.

REFERENCES

(1) ANSA version 17.1.0 User’s Guide, BETA CAE Systems, April 2017
(2) Diez C, Harzheim L, Schumacher A, Effiziente Wissensgenerierung zur Robustheitsuntersuchung von Fahrzeugstrukturen mittels Modellreduktion und Ähnlichkeitsanalyse, VDI-Reports 2279, SIMVEC 2016, Baden-Baden Germany, 2016
(3) Borg I, Groenen PJ F, Model Multidimensional Scaling: Theory and Application, Springer Series in Statistics, 1997
(4) Johnson S C, Hierarchical clustering schemes,Psychometrika 32 (p. 241-254), 1967
(5) Criminisi A, Shotton J, Konokoglu E, Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning, Foundations and Trends in Computer Graphics and Vision 7 (p. 81-227), 2011
(6) Diez C, Kunze P, Toewe D, Wieser C, Harzheim L, Schumacher A, Big-Data based rule-finding for analysis of crash simulations, WCSMO 12, Braunschweig Germany, 2017
(7) National Highway Traffic Safety Administration (NHTSA), 2007 Chevrolet Silverado Crash Model, https://www.nhtsa.gov/crash-simulation-vehicle-models
(8) META version 17.1.0 User’s Guide, BETA CAE Systems, April 2017